FAKULTÄT FÜR !NFORMATIK

Komponenten für ein next-generation collaborative Hypermedia Authoring Enviroment

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

eingereicht von

Florian Scholz
Matrikelnummer 9925374

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, 30.10.2008

(Unterschrift Verfasser) (Unterschrift Betreuer)

Danksagung

Mein Dank gilt, in md5-Reihenfolge:

Fabian Zeindl, Sonja Weber, Markus Rester, Stoiko Ivanov, Peter Purgathofer, Robert Alder, Werner Scholz, Florian Cech, Gerhard Lukawetz, Maria Scholz, Wilfried Reinthaler und Elisabeth Scholz

Kurzfassung

Die vorliegende Arbeit untersucht das Problem, Wikis und Weblogs unter Einbindung einer Overview-Map in einem System zu vereinen. Dazu wird zunächst ein kurzer Überblick über die Themen Hypertext, Wikis, Weblogs und Spatial Hypertext gegeben. Danach folgt die Vorstellung einiger Entwürfe und daraus entstandender Prototypen.

Prototypen für webbasierte Overview-Maps zeigen, diese zwar technisch praktikabel aber unter Umständen aufwendig sind und Kompromisse bezüglich der Anforderungen notwendig sind. Das browser-basierte Bearbeiten von Seiten könnte kurz davor stehen, den Sprung von Markup zu unmittelbarem Bearbeiten von Inhalten zu machen. In diesem Zusammenhang wird ein Interface für das Bearbeiten von Links entworfen, das die heute typische Textfragment-Zentrierung gegen eine Dokument-Zentrierung tauscht. Bezüglich der praktischen Vereinigung von Weblogs und Wikis werden verschiedene Varianten untersucht und besprochen.

Inhaltsverzeichnis

Kapitel 1

Einleitung und Aufbau

Aufbau

Diese Arbeit beschäftigt sich mit möglichen Entwürfen für die nächste Generation von Hypertext-Authoring-Systemen.

In Kapitel 2 geht es um die theoretischen Grundlagen sowie eine kurze Geschichte von relevanten Konzepten. Konkret beispielsweise um die Grundlagen von Hypertext, eine Besprechung von Weblog- und Wiki-Systemen, sowie von Web-basierten Authoring-Systemen im Allgemeinen.

Im darauf folgenden Kapitel 3 („Versuche und Entwürfe“) wird über die im Rahmen dieser Arbeit entwickelten Entwürfe und Prototypen berichtet.

Und schließlich wird im letzten Kapitel versucht werden, die gewonnenen Erkenntnisse zusammenzufassen.

Schreibstil

Jetzt noch ein paar Worte zum Stil dieser Arbeit.

Sand-Jensen (2007) stellt 10 Guidelines vor, um konsistent langweilige wissenschaftliche Literatur zu schreiben, mit der Motivation, dass Menschen sie brechen. Er bringt das (annektotische) Beispiel eines PhD-Studierenen, der erst den

> „[..] standard style of producing technical, boring and impersonal scientific writing [..] “

erlernen musste, und schreibt weiter:

> „ Personally, I have felt it increasingly difficult to consume the steeply growing number of hardly digestible original articles.“

Obwohl die Beispiele eher auf PhD-Arbeiten und Artikel in wissenschaftlichen Journalen abzuzielen scheinen, habe ich mich entschlossen, Teile seiner Ratschläge auf diese Arbeit anzuwenden.[1]

Einige dieser Guidelines, so muss ich zugeben, werden vermutlich von dieser Arbeit erfüllt. Beispielsweise erwies es sich als relativ schwierig, technische Termini und Abkürzungen zu vermeiden und gleichzeitig kompakt zu bleiben.

1. Im Sinne einer absichtlichen Nicht-Beachtung der Guidelines, natürlich.

Andere Guidelines („Degrade biology to statistics“) sind vermutlich hier nicht anwendbar.

Andererseits habe ich versucht Gedanken, die tendenziell Spekulation, jedenfalls aber von mir nicht stringent argumentativ beweisbar sind, der Leserin oder dem Leser nicht vorzuenthalten, speziell wenn ich sie interessant finde (Guideline 4)[2]. Sie werden allerdings von entsprechenden verbalen Warnschildern umgeben sein.

Guideline 10 empfiehlt „Quote numerous papers for self-evident statements“. Ich hoffe, dass dies in dieser Arbeit wenig passiert. Aber die Grenze zwischen „self-evident statements“ und „necessary steps of reasoning“ (siehe Guideline 6) ist manchmal schwierig zu bestimmen. Im Interesse eines besseren Lesefluss sind manche dieser Referenzen in Fußnoten[3] ausgelagert, um zu vermeiden dass

> „ [..] the meaning of whole paragraphs is veiled in the limited space between references.“

Einige Teile dieser Arbeit sind, so fürchte ich, eher frei von Originalität und Persönlichkeit[4] sowie Humor und „flowery language“[5]. Dafür will ich mich entschuldigen.

Da ich diese Ermutigung, spannend zu schreiben, recht spät in dem Prozess des Schreibens entdeckte, kann es – wie schon erwähnt – leider der Fall sein, dass manche Passagen diesem Ziel nicht so recht entsprechen. Falls sich herausstellen sollte, dass spannend zu lesende Diplomarbeiten ein wünschenswertes Ziel darstellen, könnte es nützlich sein, zukünftige Studierende früh über Mittel und Wege spannende Arbeiten zu schreiben, aufzuklären, beziehungsweise die Forschung und Traditionen in diesem Bereicht zu verstärken. Ich empfand dieses Ziel jedenfalls in hohem Grad motivierend. Das ist jedoch *sehr* außerhalb des Fokus dieser Arbeit, und daher sei dies das Schlusswort zu diesem Thema.

---

2. Sand-Jensen (2007) argumentiert, den Wert von Implikationen und Spekulationen zu erwähnen unter anderem mit dem wissenschaftlichen Mehrwert und gibt als Beispiel „This famous closing sentence [..]“ von „James Watson and Francis Crick (1953)“. Damit wir uns nicht missverstehen: Diese Arbeit hat keinen Anspruch, bahnbrechend, berühmt, oder enorm wichtig für die Nachwelt zu werden – ein Ziel, dass für eine Diplomarbeit vermutlich nicht zielführend, wenn schon nicht größenwahnsinnig wäre.
3. Ich mag Fußnoten. Manchmal vielleicht zu sehr.
4. Guideline 2: „Avoid originality and personality“
5. Guideline 8: „Suppress humor and flowery language“

Kapitel 2

Geschichte und Prinzipien

Dieses Kapitel wird einigen dieser Arbeit zugrunde liegenden Konzepten nach gehen. Hypertext und das wichtigste Merkmal von „Hyper"-Text, die Links, werden Thema sein, sowie die aktuelle Praxis des Hypertext-Authorings im World Wide Web. Danach wird ein Überblick über Weblogs und Wikis, Möglichkeiten diese zu verbinden, sowie ein kurzer Ausflug in das Konzept von Spatial Hypertext folgen.

2.1 Hypertext

2.1.1 Zum Begriff

Der Begriff „Hypertext" wird in der Literatur nicht immer ganz ident verwendet. Nelson (1987, Kapitel 0 / Seite 2), der Erfinder des Begriffes, schreibt:

> „Well, by ‚hypertext' I mean *non-sequential writing* — text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways."

Landow (2006, Seite 3) zitiert diese Stelle, um gleich darauf auszuführen:

> „*Hypertext*, as the term is used in this work, denotes text composed of blocks of text [..] and the electronic links that join them. *Hypermedia* simply extends the notion of the text in hypertext by including visual information, sound, animation, and other forms of data. [..] I do not distinguish between hypertext and hypermedia. [..] In this network, I shall use the terms *hypermedia* and *hypertext* interchangeably."

Allerdings verwendet er den Begriff nicht ganz ohne Anspruch an weitere Merkmale. So auf Seite 6, wo er schreibt:

> „A full hypertext system, unlike a book, and unlike some of the first approximations of hypertext available - HyperCard™, Guide™, and the current World Wide Web (except for blogs) - offers reader and writer the same enviroment.“

Hier scheint er zu suggerieren, dass Weblogs den Ansprüchen von „Hypertext“ genügen, das restliche World Wide Web jedoch nur als „erste Annäherung“ an Hypertext zu sehen ist.

Die meisten AutorInnen (inklusive Landow selber) scheinen jedoch das WWW meistens trotzdem als Hypertext-/Hypermedia-System zu sehen.

Für diese Arbeit ist keine scharfe Abgrenzung zwischen Hypertext und „Nicht-Hypertext“ notwendig. Es sei jedoch erwähnt, dass Hypertext auch hier „Hypermedia“ inkludiert, also sich nicht auf Text beschränkt, sondern auch z.B. Bilder beinhalten kann.

Eine umfassender Überblick über Verwendung und Definitionen des Begriffs *Hypertext* liegt leider nicht im Rahmen dieser Arbeit. Die interessierte Leserin oder der interessierte Leser findet beispielsweise bei Wardrip-Fruin (2004) einen historischen Ansatz zur Klärung des Begriffs, seine geschichtliche Verwendungen und die verschiedenen Konnotationen, die in verschiedenen Forschungsgebieten und in der Öffentlichkeit existieren.

2.1.2 Von der Memex zu Web 2.0

Memex: Fiktiver Vorläufer

Die Geschichte von Hypertext wird meist mit Vannevar Bushs Artikel „As we may think“ (Bush 1945) begonnen[1]. Er stellt darin ein fiktives Gerät namens „memex“[2] vor, das technologisch wenig mit heutigen Computern gemeinsam hat – das Speichermedium des Memex-Entwurfs sind Mikrofilme. Das Memex wäre ein persönliches Schreib-, Archiv- und Lese-Gerät für diese Mikrofilme. Es beinhaltet jedoch die Idee, die später als die Kernidee von Hypertext gesehen werden wird:

> „All this is conventional, except for the projection forward of present-day mechanisms and gadgetry. It affords an immediate step, however, to associative indexing, the basic idea of which is a provision *whereby any item may be caused at will to select immediately and automatically another.* This is the essential feature of the memex. The process of tying two items together is the important thing.“ (Bush 1945, Hervorhebung hinzugefügt)

Also: Ein Mechanismus, der automatisch und schnell zu einem Element eine spezifiziertes anderes Element aufrufen (auswählen) kann.

---

1. siehe beispielsweise (Landow 2006, Seite 9) oder (Nielsen 1990, Seite 29)
2. Ursprünglich mit kleinem „m“. Das Wort sollte in späterer Folge jedoch explizit seinen Entwurf bezeichnen (Nielsen 1990, Seite 30)

„Automatically“ und „immediately“ sind hier springende Punkte. Ein Lexikon enthält Querverweise in großer Menge, und viele weitere Texte (wie auch zum Beispiel diese Arbeit) beziehen sich auf und bezeichnen spezifische andere Texte. Vannevar Bush ging es jedoch einerseits darum, diese Querverweise schnell – auf Knopfdruck[3] – auf den Bildschirm zu bekommen und andererseits darum, eine Vielzahl assoziativer, opportunistischer Verbindungen zwischen den Texten zu erlauben und herzustellen, denn „Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing.“ und „The human mind does not work that way. It operates by association.“ (ebd.).

Der Sinn von Links

Um dies zu verdeutlichen: Tim Berners-Lee wird später und im Bezug auf das World Wide Web schreiben, wie eine Mechanisierung und Beschleunigung des Verfolgens von Referenzen einen qualitativen Unterschied erzeugen kann:

> „The research community had used links between paper documents for ages: Tables of contents, indexes, bibliographies, and reference sections are hypertext links. On the Web, however, research ideas in hypertext links *can be followed up in seconds, rather than weeks of making phone calls and waiting for deliveries in the mail.* And suddenly, scientists could escape from the sequential organisation of each paper and bibliographiy, to pick and choose a path of references that served their own interest.“ (Berners-Lee und Fischetti 2000, Hervorhebung hinzugefügt)

Der nächste Schritt in einer kurzen Geschichte von Hypertext ist Theodor Holm Nelson. Ihm wird die Prägung des Begriffs „Hypertext“ im Jahr 1965 zugeschrieben[4]. In „Literary Machines“ (Nelson 1987) legt er seine Vorstellung und Entwürfe für „Xanadu“ dar. Es ist schwierig zu beschreiben „was“ Xanadu hätte sein sollen – beispielsweise bietet Nelson selber eine Liste verschiedener Definitionen[5]. Also ist „Vision“, wie der Entwurf manchmal genannt wird[6], vermutlich ein gutes Wort um Xanadu zu beschreiben, denn der Entwurf war wegweisend, nie vollständig realisiert, und vielleicht auch unpraktikabel$^{(A)}$. Vollständige Versionierung aller Inhalte, verteilte Speicherung und natürlich flexible Verlinkung und Einbettung von Inhalten in einem globalen, verteiltem Netzwerk waren genauso Eckpfeiler dieser Vision, wie ein Entwurf für eine neue Art von

Xanadu

3. „If the user wishes to consult a certain book, he taps its code on the keyboard, and the title page of the book promptly appears before him“ (Bush 1945, Teil 6)
4. siehe Landow (2006, Seite 2), oder Nielsen (1990, Seite 33)
5. in (Nelson 1987, Kapitel 3 / Seite 2)
6. Nielsen (1990) beispielsweise schreibt: „The Xanadu *vision* has never been implemented, however, and probably never will be (at least not in the foreseeable future).“

Copyright[7] sowie ein Micropayment-System[8].

In den nächsten Jahrzehnten wurden die ersten Hypertext-Systeme konstruiert und mit ihnen experimentiert. Eine Übersicht über diese findet sich etwa bei Nielsen, der im Jahr 1990 schreibt:

> „In conclusion we can say that hypertext was conceived in 1945, born in the 1960s, and slowly nurtured in the 1970s, and finally entered the real world in the 1980s with an especially rapid growth after 1985, culminating in a fully established field during 1989. We now have several real-world systems that anybody can buy in their local computer store (or get for free bundled with their computer); we have successful conferences and a journal; and most important of all, we have many examples of actual use of hypertext for real projects.“ (Nielsen 1990, Seite 41)

Im Rückblick fällt hier auch auf, dass die Vorstellung Hypertexte in „local computer stores“ zu kaufen inzwischen vielleicht seltsam wirkt. Aber dies war offenbar die Vorstellung, wie Hypertext zu den LeserInnen kommt.

Doch kurz darauf[9] betrat ein Hypertext-System die Bühne, mit dem die meisten LeserInnen vermutlich vertraut sind und das zumindest den globalen, vernetzen Aspekt von Xanadu einlöst: das World Wide Web. Das WWW war und ist erfolgreich, obwohl – oder auch weil – es viele der angesprochen Eigenschaften für ein Hypertext-System ignoriert oder nicht implementiert. So ist wohl die Frustration und Resignation der Hypertext-ForscherInnen verständlich, über die Smith (1997) sechs Jahre später in der Keynote zur Hypertext'97 -Konferenz spricht:

WWW

> „Ignoring the hard-won knowledge of this community, the WWW has simplified the data model, ignored problems of large-scale navigation, and declared that link integrity is irrelevant. Consequently, many wish that it would go away [..]“

---

7. „To bypass some legal problems, we foresee establishing copyright convention *internal to the network* and contractually agreed upon by all participants.“ (Nelson 1987, Kapitel 2 / Seite 42)
8. „In our planned service, there is a royality on every byte transmitted. [..] Each publishing owner must consent to the standard royality — say, a thousandth of a cent per byte — and each reader contributes those few cents automatically [..]“ (Nelson 1987, Kapitel 2 / Seite 43f)
9. Das World Wide Web wurde nicht so sehr „erfunden“ sondern eher über eine Zeitspanne hinweg entwickelt – tatsächlich ist dieser Prozess nicht abgeschlossen. Ein – willkürlich gewähltes – Datum für die „Geburt“ des WWW wäre beispielsweise 1991, in dem Tim Berners-Lee die Existenz des WWW in der Newsgroup alt.hypertext ankündigte. Er selbst schreibt : „I posted a notice on several Internet newsgroups, chief among them alt.hypertext, which was for hypertext enthusiasts. Unfortunately, there was still not much a user could see unless he had a NeXT.“ (Berners-Lee und Fischetti 2000, Sete 46). Jedenfalls war das Web in diesen Jahren Größenordnungen entfernt von seiner heutigen Ausdehnung: „By January 1993 the number of known servers was increasing faster, up to about fifty.“ (Berners-Lee und Fischetti 2000, Seite 67)

Walker (2005) schreibt:

> „[..] systems like Storyspace allowed conditional links, map views and other finesses that could make early HTML seem a simplistic form of hypertext.“ (Walker 2005, Seite 50)

Es sollte erwähnt werden, dass diese Einschränkungen zum Teil durchaus beabsichtigt oder in Kauf genommen waren. So wird Tim Berners-Lee später in „Weaving the Web“ schreiben:

> „Letting go of that need for consistancy was a crucial design step that would allow the Web to scale. But it simply wasn't the way things were done.“ (Berners-Lee und Fischetti 2000, Seite 28)

Links werden später noch im Detail besprochen werden (in Kapitel 2.1.3), doch es könnte tatsächlich so sein, dass (beispielsweise) die Entscheidung „link integrity“ für „irrelevant“ zu erklären ein simples, skalierbares, dezentrales und globales Hypertext-Netzwerk erst praktikabel machen.[10]

Trotz aller dieser potentiellen Schwächen die das WWW im Vergleich zu einer hypothetischen „idealen“ Hypertext-Umgebung aufweist, ist es ein viel benutztes und noch immer wachsendes Netzwerk. Sehr wahrscheinlich ist es nützlicher, ein solches – nicht perfektes – Netzwerk zur Verfügung zu haben, als keines. Um Landow (2006, Seite 330) zu zitieren, der sich hier auf ein vorher gebrachtes Beispiel einer Interaktion mit dem World Wide Web bezieht: „[..] this hypertextual presentation of news about a current event clearly empowers the user – if by ‚empower' we mean, as I do here, ‚provides information that would be difficult if not impossible to obtain otherwise.' “

Und im Bezug auf die Verwendung und Wahrnehmung von Hypertext war die Verbreitung des WWW sicher ein qualititativer Sprung.

> „In the nineties, the advent of the web and the rapid spread of personal computers and internet connections in ordinary homes radically changed the ecosystem hypertext existed in. Hypertext, lovingly bred in captivity, was unleashed into the World Wide Web. Suddenly, anyone could publish a website and link and be linked at will.“ (Walker 2005, Seite 47)

Sicherlich ist die Evolution von Hypertext, des WWW und die Verwendung derselben noch nicht am Ende angelangt. In letzter Zeit wurde eine „neue Generation“ von Webseiten unter dem Begriff Web 2.0 beschrieben. Millard und Ross (2006) schreiben:

Web 2.0

> „Web 2.0 is the popular name of a new generation of Web applications, sites and companies that emphasis openness, community and interaction. Examples in-

10. Für eine Abwägung der Vor- und Nachteile verschiedener (technischer) Link-Architekturen siehe Davis (1998).

> clude technologies such as Blogs and Wikis, and sites such as Flickr. “

und weiter:

> „The most recent generation of Web applications and Web sites have been considered by some to be fundamentally different from the ones found on the early Web, these have been grouped together under the term Web 2.0, and while the name is arguably misleading (implying a designed version and a discrete evolution) the concepts beneath it provide a valuable insight into the way in which the Web has evolved. “

In der Tat ist „Web 2.0“ ein komplexer und gleichzeitig vielgenutzer Begriff. Tim O’Reilly, einer der Erfinder des Begriffs, schreibt später:

> „In the year and a half since, the term ‚Web 2.0‘ has clearly taken hold, with more than 9.5 million citations in Google. But there’s still a huge amount of disagreement about just what Web 2.0 means, with some people decrying it as a meaningless marketing buzzword, and others accepting it as the new conventional wisdom.“ (O’Reilly 2005)

Er versucht hier zu erklären was „Web 2.0“ aus seiner Sicht, bzw. nach der Intention der ErfinderInnen bedeutet, und es scheint als ob hier der Begriff eher den Fokus von Eigenschaften von (oder Ratschlägen an) Firmen besitzt, beispielsweise in den Schlussworten:

> „Let’s close, therefore, by summarizing what we believe to be the core competencies of Web 2.0 companies:
>
> - Services, not packaged software, with cost-effective scalability
> - Control over unique, hard-to-recreate data sources that get richer as more people use them
> - Trusting users as co-developers
> - Harnessing collective intelligence
> - Leveraging the long tail through customer self-service
> - Software above the level of a single device
> - Lightweight user interfaces, development models, AND business models
>
> The next time a company claims that it’s ‚Web 2.0,‘ test their features against the list above.“ (ebd.)

Budd (2005) formuliert einen anderen Weg, und schreibt „Web 2.0 is a Buzzword“, um hinzuzufügen, dass „Buzzwords“ durchaus nützlich sein können, um eine komplizierte und nebulöse Menge von Konzepten zu definieren und zu kommunizieren.

Er scheint hier die zentralen Eigenschaften einer Web 2.0-Anwendung in „Open Data“[(B)], einer „Architecture of Participation“

sowie einer „Rich User Experience“ zu sehen.

Millard und Ross (2006) (von denen die auch die eingangs erwähnte Beschreibung von Web 2.0 stammt) vergleichen einige Web 2.0-Seiten als „next generation tools“ mit den „aspirations of the early Hypertext pioneers“ und kommen zu dem Ergebnis, dass

> „[...] we set out to show whether Web 2.0 fulfils the aspirations of the original hypertext pioneers, and the community that took up their work. However, it seems that the relationship between Web 2.0 and those original visions is more complex than this: many of the aspirations of the hypertext community have been fulfilled in Web 2.0, but as a collection of diverse applications, interoperating on top of a common Web platform (rather than as one engineered hypertext system). Some aspirations are unsupported because they seem to be unnecessary for a given domain – so for example, Wiki pages are versioned, but Blogs are not. This indicates that versioning is chosen carefully, to avoid a user overhead if it is not really needed. Other aspirations, such as typed, n-ary links, are hardly supported at all. “

Die Sichtweise das mit Web 2.0 Features und Eigenschaften in das WWW Einzug halten, die in älteren Hypertext-Entwürfen enthalten waren, aber zwischendurch „verloren gingen“, ist für diese Arbeit recht ermutigend, da sie teilweise versucht Features früherer Hypertext-Systeme auf ihre Anwendbarkeit bzw. technische Machbarkeit im heutigen World Wide Web zu überprüfen. Dies sind vor allem die Overview-Maps bzw. Spatial Hypertext (Kapitel 2.6 bzw. große Teile von Kapitel 3), aber auch Details, wie ein Dokumentzentriertes Link-Erstellen (Kapitel 3.9.1).

2.1.3 Links

„Link“ ist ein Begriff der kurz etwas nähere Aufmerksamkeit verdient, da er in den meisten Hypertext-Definitionen und -Beschreibungen zentral ist. Bernstein (2006, Seite 109) schreibt:

> „Links are the distinctive feature of hypertext. Ever since the World Wide Web became ubiquitous, nearly everyone who has thought about them seems to have jumped to the conclusion that they intuitively understand links and their use.“

In der Tat scheint die Verbreitung des WWW inzwischen bewirkt zu haben, dass (von der Erfahrung im WWW-geformte) Erwartungshaltungen für die Funktionen und sogar das Aussehen von Links existieren. Nielsen (2004) beschreibt beispielsweise Guidelines wie Links auszusehen und sich zu verhalten haben um als solche erkannt und verstanden zu werden. Seine Empfehlungen für eine Best-Practice enthalten unter anderem empfohlene Typographie (u.A. unterstrichen) und Farbe (am besten Blau).

Aber in vielen entworfenen oder real existierenden Hypertext-Systemen existieren auch andere Arten von (bzw. Sichtweisen auf) „Links“.

Link-Verankerung

Zunächst beinhalten diese nicht einmal das Element, das heute scheinbar als konstituierend gesehen wird, nämlich den Link-Anker. Beispielsweise bei Landow (2006, Seiten 14–15), der in seiner Einführung zu Links erst als dritte Variante von Links die Verankerung eines Links im Text beschreibt. Den Begriff „lexia“ verwendet er im Sinn von „Textblöcken“.(C)

> „Linking a string—that is, word or phrase—to an entire lexia, the third form of linking, has three advantages. First, it permits simple means orienting readers by allowing a basic rhetoric of departure [...]. When readers see a link attached to a phrase, such as ‚Arminianism‘ or ‚Derrida‘, they have a pretty good idea that such a link will take them to information related in some obvious way to those names. Second, because string-to-lexia linking thus provides a simple means of helping readers navigate through information space, it permits longer lexias. Furthermore, since one can choose to leave the lexia at different points, one can comfortable read through longer texts. Third, this linking mode also encourages different kinds of annotation an linking, since the ability to attach links to different phrases, portions of images, and the like allows the author to indicate different kinds of link destinations. One can, for example, use icons or phrases to indicate that the readers can go to, say, another text lexia, one containing an illustration, bibliographical information, definitions, opposing arguments, and so forth.“

Er bringt hier starke Argumente, die für die Verankerung von Links im Text, doch zunächst ist hier interessant, dass diese Möglichkeit[11] einer „rhetoric of departure“ zunächst nicht selbstverständlich ist. Denn die beiden Varianten, die er davor bespricht, geben nur vor, dass zwei lexia verlinkt werden (Uni- bzw Bi-direktional).

Im Zeitalter des WWW taucht hier die Frage auf, wie genau diese Links bedient werden sollen, wenn nicht durch Klicken auf den Text. Die Antwort wäre vermutlich, dass das Hypertext-Lese-Programm diese Links extern zum Text/Lexia anbietet. Erinnern wir uns kurz an die Memex. In der Beschreibung von Vannevar Bush, beispielweise, würden Links durch Drücken eines von mehreren speziellen Link-Buttons aktiviert, gekennzeichnet durch ein „code word“ in einem code space (siehe Bush 1945, Teil 7).

---

11. Heute ist diese Variante, nämlich dass die Aktivierung eines Link durch einen Klick auf einen im (Hyper-)Text vorhandenen speziell markierten Text, ein Bild oder einen andersweitig markierten Bereich passiert, oft gar nicht mehr erklärungswürdig.

Um zu unterscheiden, ob ein Link bzw. das Interface des Links extern oder intern zum „Dokument“ angezeigt wird, müssten wir allerdings zuerst die Grenzen des Dokuments festlegen, und dies ist nicht immer einfach oder möglich(D). Sind beispielsweise Links die (räumlich) neben dem Haupt-Textteil einer Webseite stehen – aber innerhalb des Browser-Fensters – außerhalb des „Textes“?[12]

Speichermodelle

Es handelt sich bis zu diesem Punkt nur um eine Interface-Frage, aber sie interagiert mit den Daten-Modellen der Hypertext-Systeme, und auch hier stellt sich die Frage, wie (bzw. wo) Links gespeichert werden sollen. Davis (1998) bespricht drei verschiedene Modelle und ihre Vor- und Nachteile. In Modell 1 („Embedded Link Model“) werden sowohl Anker als auch Ziel direkt im „Dokument“ (oder „Lexia“, „Text“, „Seite“) gespeichert. In Modell 2 („External Links with Named Endpoints“) werden Ziel und Endpunkte im Dokument beschrieben aber erst durch eine externe Link-Datenbank miteinander verknüpft. Und schließlich kommen in Modell 3 („External Link and Reference Model“) alle Bestandteile von Links aus dieser externen Datenbank.

Das WWW beruht auf Variante 1, die laut Davis (1998) den Vorteil von Einfachheit und guter Skalierbarkeit aufweist. Nachteile seien, dass Links prinzipiell nur uni-direktional sein können[13], und die Analyse der von den Links gebildeten Netzwerke schwieriger wird. „Dangling Links“[14] sind ein großes Problem, da es – wenn Seiten verschoben werden – meist nicht möglich ist, herauszufinden welche anderen Seiten auf eine verschobene Seite zeigen. Und falls dies doch möglich ist, besteht immer noch das Problem, dass vermutlich die Berechtigungen fehlen würden alle diese verlinkenden Seiten zu ändern.[15]

Da sich scheinbar das WWW als *das* Hypertext-System durchgesetzt hat, könnten wir es hier theoretisch dabei belassen und mit dem Status Quo arbeiten. Wenn nicht im Prinzip jede Webseite, jedes Wiki und jedes Blog-System, das auf seinen Seiten Links auf sich selbst

---

12. Tendenziell legen WWW-Browser zumindest eine scharfe Grenze, indem sie den Bereich festlegen in dem Inhalte einer Webseite angezeigt werden. Aber auch hier wird die Trennlinie zunehmend unscharf. RSS-Feeds beispielsweise sind syntaktisch, wenn auch nicht immer inhaltlich, ein „Link“ auf eine „externe“ Ressource, und dieser Link wird in vielen Browsern außerhalb des „Seitenbereichs“ angezeigt.

13. In einem gewissen Sinn allerdings sind alle Links im WWW bi-direktional. „[..] the return function provided by most browsers creates the effect of a bidirectional link“ (Landow 2006). Andererseits bedeutet dies hier nur, dass eine Rückkehr zum vorher besuchten Knoten möglich ist, zugegebenermaßen allerdings ein wichtiges Feature.
In einer anderen Sichtweise sind bidirektionale Links natürlich „möglich“, indem uni-direktionale Links in beide Richtungen gesetzt werden.

14. Damit sind Links gemeint, die nicht mehr funktionieren, weil das Ziel verschwunden ist. Exakt: „The dangling link problem occurs when a link anchor no longer refers to a valid file.“ (Davis 1998, Seite 210)

15. In der Realität des globalen Hypertexts namens WWW taucht auch noch ein grundsätzlicheres Problem auf: Nicht alle Referenzen („Link“ wäre hier vielleicht das falsche Wort) auf Webseiten befinden sich überhaupt in Computersystemen. URLs auf Plakaten, Büchern, Aufklebern oder Radiosendungen lassen sich nachträglich kaum ändern und bleiben doch unter Umständen lange bestehen.

setzt, wieder ein Hypertext-(Sub-)System wäre, das die Regeln – für sich – neu schreiben kann. Ein solches System kann für die von ihm verwalteten Seiten die Verantwortung übernehmen und alle Links auf eine Seite, die im System vorhanden sind, aktualisieren, wenn diese Seite im System verschoben oder umbenannt wird. Auch kann es herausfinden, welche seiner eigenen Seiten auf eine bestimmte andere Seite verlinken, oder dass auf eine Seite gar nicht gelinkt wird. Mediawiki bietet beispielsweise eine Seite mit einer Liste von Links auf „Seiten, auf die von keiner anderen Seite verlinkt wird“ („Verwaiste Seiten“)[16].

Interne und externe Links

Wikis im besonderen, aber auch Weblogs und andere Content-Management-Systeme, können also eine Unterscheidung in interne und externe Links treffen. Wobei „intern“ hier bedeutet, dass der Link von einer vom System verwalteten Seite auf eine andere vom System verwaltete Seite zeigt. „Extern“ sind alle anderen, von Seiten im System ausgehenden Links.

Alternativ dazu bezeichnet Bernstein (2006, Seite 124) mit „internal link“ Links die technisch im Text gespeichert werden (wie oben als „Modell 1“ besprochen) und als „external link“ Links, die extern zum Text gespeichert werden (entsprechend „Modell 3“).

Typed Links

Eine weitere angedachte Anreicherung von Links wären Link-Typen („link types“). In einer solchen Konzeption würden Links klassifiziert je nachdem ob das verlinkende Fragment inhaltlich dem Link-Ziel zustimmt oder nicht zustimmt, sogar welche Unterarten von Kritik geäußert werden, oder ob es ergänzt, kommentiert, und so weiter.

Eine strenge Formalisierung hat sich dabei jedoch scheinbar als eher wenig zielführend herausgestellt. Bernstein (2006, Seite 113) schreibt unter dem Titel „Formality Considered Harmful: The Rejection of Link Types“:

> „Experience with typed links soon led researchers – most notably Trigg himself – to conclude that users did not want them and would not use them. If the vocabulary of link types is too small, then users cannot find a link type that expresses what they wish to say. As the vocabulary grows, however, choosing the correct link type becomes more and more difficult.“

Hier besteht also ein Problem wenn das System im vorhinein zu definieren versucht welche möglichen Bedeutungen bzw. Beschreibungen ein Link haben kann. Eine strenge Typologie wäre vielleicht für (maschinelle) Analysen des Hypertexts praktisch. Andererseits müssen nach derzeitigen Stand immer noch Menschen diese Beschreibungen zuordnen. Landow (2006, Seite 20), obwohl grundsätzlich „typed links“ nicht unbedingt abgeneigt, schreibt :

---

16. Die Begriffe ändern hier teilweise ihre Bedeutung recht rasch. Gemeint sind nur interne Links, und Seiten wie die „Seite“ der „Verwaisten Seiten“ sind ausgenommen. „Link“ meint hier etwas spezielleres als ein allgemeiner WWW-Link.

> „A greater danger for authors exist in systems that prescribe the kind of links possible. [..] The very first version of Intermedia used by faculty developers and students differentiated between annotation and commentary links, but since one person's annotation turned out to be another's commentary, no one lobbied for retaining this feature, [..]"

Das „title"-Attribute von HTML könnte im Prinzip als „Typed Link" verstanden werden, wenn wir also eine Klassifizierung durch Freiform-Strings ersetzen. Dieses Attribut versieht Elemente mit einer zusätzlichen Beschreibung die üblicherweise erscheinen, wenn der Mauszeiger eine längere Zeit über dem Element gehalten wird. Dies scheint jedoch in der Praxis selten eingesetzt zu werden.

Hypertext without Links?

Eine weitere Perspektive sind dynamisch erzeugte Links die sogar die Frage aufwerfen ob Hypertext ohne Links möglich ist.

> „Most writing about hypertext from Bush and Nelson to the present assumes that someone, author or reader functioning as author, creates an electronic link, a so-called hard link. Recently, workers in the field, particularly the University of Southhampton's Microcosm development group, have posed the question, ‚Can one have hypertext „without links"?' — that is, without the by-now-traditional assumption that links have to take the form of always-existing electronic connections between anchors. This approach takes the position that the reader's actions can create on-demand links." (Landow 2006, Seite 20)

Eine Form die on-demand links annehmen könnten wären eventuell die von gewissen Browser-Plugins bereitgestellte Möglichkeit, markierte Wörter oder Phrasen in Wikipedia oder in einem Wörterbuch nachzuschlagen.[17] Das Interface zum Aktivieren des „Links" ist allerdings nicht mehr nur ein Klick, und so ist fraglich ob viele BenutzerInnen dies als „Link" betrachten würden.

In einer solchen Perspektive werden auch noch andere Phänomene zu Hypertext, die vielleicht nicht sofort als Hypertext beschrieben werden würden.

Als Beispiel soll hier das Lesen einer PDF-Datei in Apples „Preview"-Software herhalten, wo das Lesen des Textes unter Umständen sehr hypertextuell wirkt (auch bei Abwesenheit vorgefertigter Links). Wie Zhang (2006) anmerkt kann es wichtig sein

> „[..] to differentiate between *reading hypertext* and *reading in a hypertextual manner* (hyperreading). [..] hyperreading can take place in the absence of hypertext." (Hervorhebung im Original)

---

17. Z.b die „Wikipedia Lookup Extension", siehe https://addons.mozilla.org/en-US/firefox/addon/744 (abgerufen am 18. August 2008).

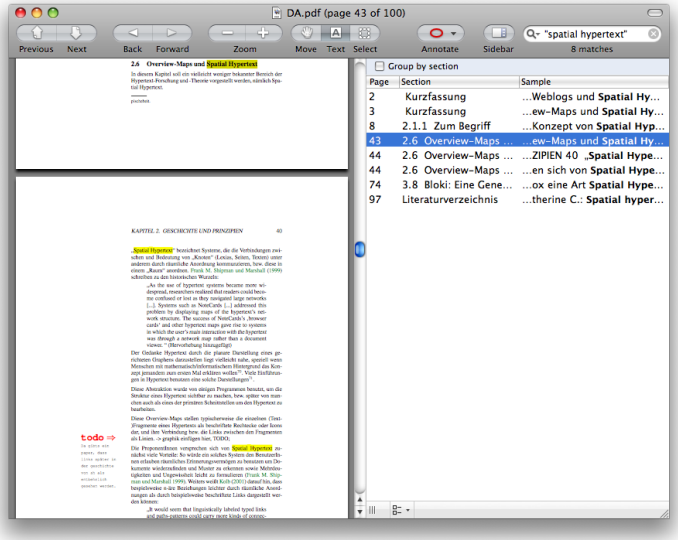


Abbildung 2.1: Die Suchergebnisse in Apples Preview (als Beispiel) können als on-demand-links gelesen oder zumindest als solche benutzt werden.

Hier ist der Punkt vielleicht, dass das Interface „hyperreading“ auf verschiedene Arten unterstützt. Inhaltsverzeichnisse als Links sind ein Aspekt davon, aber auch die Suchergebnisse, wenn sie als on-demand erzeugte Links auf Stellen im Text gelesen werden (siehe Abbildung 2.1).

Andere dynamisch erzeugte Links werden ohne explizite Anforderung der BenutzerInnen erstellt. Die Software „VoodooPad“[18] beispielsweise setzt automatisch Links von allen Wörtern oder Phrasen, die dem Titel einer bestehenden Seite entsprechen, auf die jeweilige Seite.

n-äre Links

Links mit mehreren Endpunkten, also mehr als einem Ziel sind ebenfalls ein Konzept, „klassische“ Links zu erweitern. Dies wird oft als n-ärer Link bezeichnet. Allerdings macht es hier scheinbar einen Unterschied, diese Frage von einer technischen Seite her zu betrachten oder die Frage in Begriffen der Interface-Praxis zu stellen. In der ersten Perspektive bietet das WWW zunächst diese Möglichkeit nicht und andererseits wird sie, so implementierbar, nicht verwendet.

> „There is no support for typed n-ary links except in the research systems. These are possible to implement on the Web in a number of ways (including XLink) and the lack of support probably indicates that they are not needed in the interface layer.“ (Millard und Ross

18. http://flyingmeat.com/voodoopad/, abgerufen am 19. August 2008.

2006, Seite 29)

Nielsen (1990) schreibt:

> „There are several possibilities for dealing with having a single anchor connected to several destinations. The two simplest options are either to show a menu of the links or to go to all the destinations at the same time. Intermedia uses the menu option [..]“

So gesehen (und die Option alle Ziele zu öffnen außer acht lassend[19]) könnte ein n-ärer Link als Link auf mehrere weitere Links verstanden werden.

Tags

Spannend für uns ist, dass in dieser Perspektive sogenannte „Tags“ als n-äre Links verstanden werden könnten. „Tags“ sind (in diesem Sinn[20]) Text-Strings, mit denen eine Ressource annotiert wird.

> „In recent years, *tagging systems* have become increasingly popular. These systems enable users to add keywords (i.e., ‚tags‘) to Internet resources (e.g., web pages, images, videos) without relying on a controlled vocabulary.“ (Marlow, Naaman, Boyd und Davis 2006)

Tags und Tagging-Systeme sind in mehrere Hinsicht interessant. Hier sei jedoch nur kurz die Perspektive angeführt, dass das Zuordnen eines Tags zu einer Ressource in vielen Tagging-Interfaces auch impliziert, einen Link von dieser Ressource auf alle Seiten mit dem selben Tag zu setzen.[21]

Der hier vorgestellte Begriff von Link ist, dies sei abschließend gesagt, nicht sehr scharf. Es gibt viele Praxen und Phänomene, wo die Zuschreibung „Link“ diskutiert werden kann oder nicht auf den ersten Blick offensichtlich ist.

2.1.4 Interface-Praxen des Web-Authoring

Der erste „Web-Browser“ war ein kombinierter Browser/Editor.[22] Die meisten der darauf folgenden Browser waren vor allem zum Lesen von Webseiten geeignet. Dies war nicht ganz „im Sinne des Erfinders“. Wie Tim Berners-Lee schreibt:

> „I never intended HTML source code (the stuff with the angle brackets) to be seen by users. A browser/editor would let a user simply view or edit the language of a page of hypertext, as if he were using a word processor. The idea of asking people to write the angle

19. Eine Webseite durch einen simplen Klick eine beliebige Zahl neuer Browserfenster öffnen zu lassen, war eine Möglichkeit im WWW. Neuerdings besitzen allerdings die meisten Browser sogenannte Pop-Up-Blocker. Dieses Feature war möglicherweise einfach zu leicht zu missbrauchen.
20. In der HTML-Terminologie bedeutet „Tags“ etwas anderes.
21. Über einen Zwischenschritt, sei es nun ein Popup-Menü oder eine eigene Webseite. Natürlich ist es auch explizit möglich einen solchen Link über die URL zu setzen.
22. siehe (Berners-Lee und Fischetti 2000, Seite 29)

> brackets by hand was to me, and I assumed to many, as unacceptable as asking one to prepare a Microsoft Word document by writing out its binary coded format. “ (Berners-Lee und Fischetti 2000, Seite 42)

Das heißt die Idee war zu Beginn, dass die Browser-Software sowohl das Bearbeiten von Webseiten direkt im Browser als auch das Zurückspeichern auf den Server erlauben sollte, Access Control inklusive. Die POST, PUT und DELETE Anweisungen im HTTP-Protokoll[23] zeugen auch von dieser Konzeption.

Doch es sollte anders kommen. Schon der nächste Browser konnte Webseiten nur mehr lesen[24], und die meisten Browser seitdem sind hauptsächlich für das „Lesen“ von Webseiten konzipiert. So stellt sich die Frage wie Webseiten geschrieben oder erzeugt werden.

Texteditor + FTP-Client

Der Inhalt von Webseiten wird typischerweise in der HyperText Markup Language (HTML) an den Browser geliefert. HTML ist ein für Menschen (relativ) les- und schreibbares Format. Daher besteht die „natürliche“ Möglichkeit, „direkt“ HTML-Dateien zu schreiben (ein beliebiger Texteditor kann verwendet werden). Die so erzeugten Dateien werden dann mittels anderer Hilfsprogramme, beispielsweise FTP-Clients, auf einen Server gestellt.[25]

Viele der frühen Webseiten entstanden vermutlich auf diese Weise. Andere wurden automatisch aus Datenbanken generiert[26].

Nun erlauben Wikis und Weblogs ihren BenutzerInnen Web-Seiten zu bearbeiten ohne den Browser zu verlassen. Wie wird dies bewerkstelligt? Im folgenden werden einige Varianten diskutiert.

Diesen Systemen ist dabei tendenziell ein Prinzip gemeinsam: Der HTML-Code, der für eine URL zurückgeliefert wird, wird von der Software am Server aus generischen und URL-spezifischen Teilen erzeugt. Im Gegensatz dazu kann ein Webserver auch einen Teil der URL einfach als Pfadangabe interpretieren und die entsprechende Datei in einer Dateihierarchie zurückliefern.

Variante 1: HTML in Textareas

Variante Eins funktioniert, indem die Webseite eine HTML-Textarea generiert in die HTML-Code geschrieben (bzw. bearbeitet) wird. Dieser HTML-Code wird an den Server geschickt, der den Code dann in die Webseite einbindet.

Ein Problem dieser Variante (aber nicht nur dieser) ist Security. Beliebigen fremden HTML-Code in die eigene Webseite einzubinden, erzeugt eine große Menge an Möglichkeiten für Missbrauch. Einerseits die absichtliche Ausnutzung von Sicherheitslücken, anderseits

---

23. (Fielding, Gettys, Mogul, Frystyk, Masinter, Leach und Berners-Lee 1999)
24. (Berners-Lee und Fischetti 2000, Seite 32)
25. Später wurden auch Programme entwickelt die den „Editor“-Teil eines kombinierten Editor/Browsers übernehmen wollen. „Adobe Dreamweaver“ oder „Microsoft FrontPage“ sind (bzw. waren) Beispiele für solche Programme.
26. Tatsächlich beschreibt Tim Berners-Lee das die erste „Showcase“-Anwendung für das WWW ein Web-Interface für das CERN-Telefonbuch war. (Berners-Lee und Fischetti 2000, Seite 32)

auch unabsichtlich, da der HTML-Code mit der einbindenen Seite interagiert und so das Layout stören kann, im einfachsten Fall durch nicht geschlossene HTML-Tags.

Daher müssen diverse HTML-Tags und HTML-Tag-Attribute ausgefiltert werden und bestimmte Zeichen uminterpretieret werden. Dies ist eine nicht ganz triviale Angelegenheit, da es leicht passieren kann, dass ein Angreifer oder eine Angreiferin eine bisher nicht berücksichtige Möglichkeit findet, schädlichen Code auf der Webseite unterzubringen. Da die Möglichkeiten, d.h. die implementierten Effekte von HTML erstens browserspezifisch sind und zweitens im Laufe der Zeit größer werden, ist dies außerdem ein Problemfeld, das nicht als abgeschlossen angesehen werden kann – außer, und auch dies nur eventuell, mit sehr starken Einschränkungen, z.B. mit einer Whitelist von erlaubten harmlosen Tags und dem Wegfiltern aller Attribute.

Dadurch wird aber im Prinzip nicht mehr HTML geschrieben, sondern ein HTML-Subset. Unterschiedliche Seiten implementieren verschiedene Subsets. Am Ende sehen wir Symptome für Verwirrung, beispielsweise in Abbildung 2.2. Ein einfaches Formular als Kommentar-Möglichkeit hat eine Anzahl an Erläuterungen welche Möglichkeiten bestehen und welche nicht. Sechs Zeilen, und doch eine Kurzfassung, wie der Link am Ende zu verstehen gibt.

Auf Seite der BenutzerInnen sind die Hauptprobleme hier also, dass zum einen ein Wissen über HTML bestehen muss, und zum zweiten Kenntniss des spezifischen HTML-Dialekts der Seite.[27] Weiters ist diese Methode Hypertext zu bearbeiten, von der Unmittelbarkeit weit entfernt davon, was die BenutzerInnen bei – beispielsweise – Textverarbeitungen zu ertragen bereit wären, wovon auch das eingangs erwähnte Zitat von Tim Bernes-Lee zeugt.

Variante 2: Andere Markup-Sprachen

Die nächste Variante versucht einige Schwächen des Schreibens in HTML zu korrigieren und benutzt eigens für diesen Zweck gedachte und erfundene Markup-Sprachen[28]. Diese Sprachen orientieren sich oft an Konventionen, die entstanden sind, um in nichtformatierbaren Texten Hervorhebungen und Überschriften zu kommunizieren, z.B. in (Nicht-HTML-)E-Mails.

Damit versuchen sie, besser für Menschen lesbar zu sein – äquivalentes HTML ist oft weniger lesbar – und erzeugen eine zusätzliche Abstraktions-Schicht, die vor dem Einschleusen von Schad-Code schützen kann.

Diese beiden Varianten treten oft auch gemischt auf. Also etwa gefilterter HTML-Code, in dem zusätzlich Zeilenumbrüche im Code

27. Für BenutzerInnen in der Rolle von AutorInnen. Für nur lesende BenutzerInnen ergeben sich unter Umständen daraus abgeleitete Probleme durch „Fehler“ der AutorInnen, beispielsweise, dass anstatt der gewünschten Formatierung HTML-Tags stehen. Also „fett“ statt „**fett**“.

28. Der Einfachheit halber soll in Folge „Markup“ „Nicht-HTML-Markup“ bezeichnen. Die Abkürzung HTML bedeutet eigentlich HyperText *Markup* Language.

Abbildung 2.2: Ein in „freier Wildbahn“ gefundenes Beispiel für eine Erklärung der verwendeten Syntax unter einem Kommentar-Feld. Die Menge an Hinweisen und der „Weitere Informtationen“-Link lassen vermuten, dass dieses Interface auch nach Einschätzung der UrheberInnen Erklärung benötigt, vor allem in Anbetracht der Tatsache, dass diese Konventionen auf verschiedenen Seiten unterschiedlich sind. Die Mischung der Sprachen kann verschiedene Ursachen haben, vielleicht eine unvollständige Lokalisierung der zugrunde liegenden Software.

auch in Zeilenumbruch-Tags übersetzt werden[29], oder Markup in dem auch bestimmte HTML-Tags verwendet werden können.

Das Problem, dass viele verschiedene „Dialekte“ existieren, ist hier noch etwas ausgeprägter als bei der ersten Variante. Für den gleichen Effekt wird in verschiedenen Markup-Sprachen oft unterschiedliche Syntax verwendet. Die Probleme von Markup-Sprachen entsprechen also in etwa den Problemen, HTML direkt zu bearbeiten, unter Umständen etwas abgemildert durch leichter les- und schreibbaren „Quellcode“.

Exkurs: Preview

An dieser Stelle soll ein Spezifikum vieler dieser Systeme erwähnt werden: Preview. Aus dem geschriebenen Markup oder HTML wird in einem Zwischenschritt eine Voransicht (Preview) generiert, aber noch nicht „gespeichert“. Dies dient der Überprüfung und Kontrolle, ob der geschriebene Code in etwa das gewünschte Ergebniss erzeugt.

Entscheidend ist dies vor allem in Systemen, die dem Akt des „Speicherns“ besondere Bedeutung einräumen. In Weblog-Systemen beispielsweise ist ein neu geschriebener Eintrag sofort ziemlich prominent. Halbfertige Einträge können daher zumindest Missverständnisse auslösen. In Wikis ist dies ähnlich, da dort oft Listen mit den letzten Änderungen publiziert werden. Diese dienen unter anderem der Kontrolle durch andere.[30] Eine Unzahl kleiner Änderungen würde diese Listen „verschmutzen“ und schwieriger lesbar machen. Also dient die Voransicht auch dazu. dem Akt des „Speicherns“ größere Bedeutung zuzuweisen (mehr Aufwand) und dazu größere Kontrolle über diesen Schritt zu haben – die Voransicht gibt Feedback über das voraussichtliche Ergebnis, ohne dass ein „Speicher“-Schritt notwendig wäre.

So gesehen hilft in der Praxis ein Voransichts-Mechanismus oft, die Probleme der beiden ersten Varianten zu kompensieren. Markup und HTML-Tags werden ausprobiert und oft genug zeigt die Voransicht, dass nicht das gewünschte Ergebnis erzielt wurde. Dann beginnt unter Umständen die Suche nach Hinweisen und Dokumentation zum von der Seite geforderten Markup.

Für ein Beispiel der Einbindung einer Voransicht in den Workflow des Bearbeitens einer Seite siehe Abbildung 2.3.

Rich-Text-Edit

Die dritte Variante, die hier besprochen werden soll, nähert sich wieder ein bisschen den ursprünglichen Visionen für das WWW. Sie beginnt in dem Moment, in dem Browser wieder beginnen, die Möglichkeit, formatierten Text zu bearbeiten, zur Verfügung zu stellen. Dies ist *im Prinzip* schon möglich, da dies eine Erfordernis für die praktische Verwendung von HTML-Mails war und als solche von

29. Ein Beispiel hierfür wäre etwa twoday, siehe Kapitel 2.2.2
30. Ohne diese Listen der letzten Änderungen wären böswillige Änderungen an einer wenig besuchten Unterseite nur wahrzunehmen, wenn alle diese Seiten regelmäßig überprüft würden.

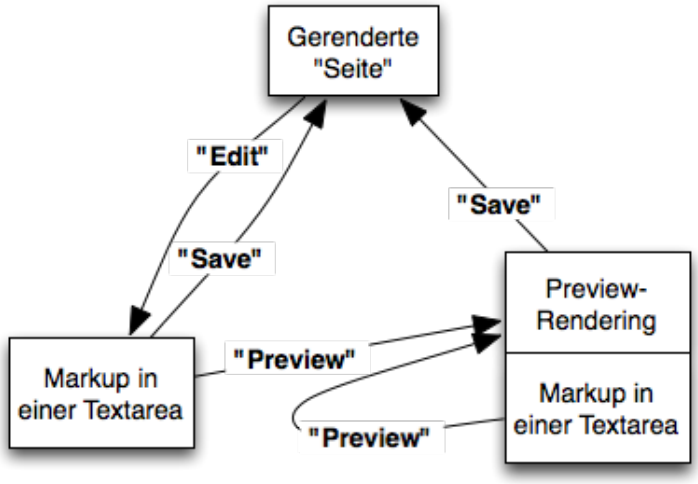


Abbildung 2.3: Der Fluß zwischen Edit-, View-, und Preview-Modi in Mediawiki (siehe auch Kapitel 2.3.2). Der Preview-Modus (rechts unten) beinhaltet sowohl das Preview-Rendering als auch weiter unten auf der Seite wieder eine Textarea mit dem Markup-Code.

Microsoft und Netscape-Browsern implementiert wurde – leider allerdings sehr unterschiedlich.

Ein Blick über die Welt der Webbrowser hinaus zeigt, wie erwähnt, dass es meist keine großen (prinzipiellen) Unterschiede in der Präsentation eines Dokuments gibt je nachdem ob dieses bearbeitet oder angezeigt werden soll. Manchmal wird dieser Anspruch mit „What You See Is What You Get“ (WYSIWYG) beschrieben.

In Folge ermöglicht dieses Prinzip die Aufhebung der Unterscheidung von verschiedenen Modi für Lesen und Schreiben. Viele (Desktop-)Programme besitzen und benötigen also weder einen „Edit“-Button noch ein „Preview“[31].

Es existieren nun Bibliotheken, die die erwähnten Browser-Features nutzen, um ein solches Verhalten auch für das WWW zur Verfügung zu stellen. Diese nennen sich meist „Rich-Text-Editoren“, und es ist vermutlich kein Zufall, dass sie sich selten als „WYSIWYG-Editoren“ bezeichnen.[32]

Denn die „Rich-Text-Editoren“ können meist das prinzipielle Versprechen von „What You See Is What You Get“ kaum einhalten – dass das während des Bearbeitens angezeigte Aussehen dem späteren exakt entsprechen wird.

---

31. Die Ausnahme hier ist, dass viele Programme eine Druck-Voransicht anbieten.
32. Ein anderer Grund könnte natürlich sein, dass es sich um ein handlicheres Vokabel handelt.

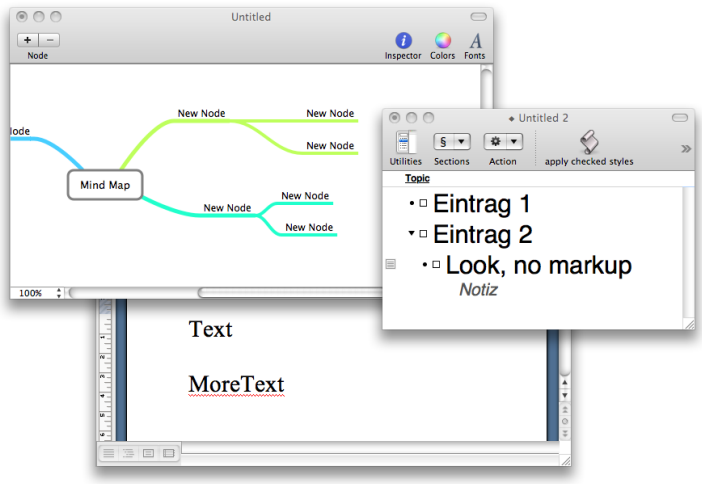


Abbildung 2.4: Viele Programme kommen inzwischen ohne Markup, „Edit“-Buttons und Preview aus. Hier drei Beispiele.

Das hat zum Teil historische bzw. prinzipielle Gründe: HTML ist nicht von vornherein dafür ausgelegt worden, ein bestimmtes definiertes Aussehen zu erzeugen. Teile der ursprünglichen Tags wollen „logische“, inhaltliche Beschreibungen sein. Beispiele wären die HTML-Tags für „Überschrift“ (h1-h6). Hier ist dem Browser prinzipiell freigestellt, wie diese dargestellt werden. Cascading Style Sheets (CSS) wollen dieses Problem lösen, aber in der Praxis ist es (für nicht-triviale Seiten) kein einfach zu lösendes Problem, eine Seite so zu gestalten, dass sie in allen[33] Browsern ident gerendert wird[34], auch wenn es hier scheinbar Fortschritte gibt.

Bestehende Rich-Text-Editoren kämpfen jedoch auch noch mit anderen Problemen. So hatten lange Zeit verschiedene Browser unterschiedliche Methoden, diese Funktionalität zur Verfügung zu stellen (wenn überhaupt). „Browserübergreifend“ verlangt also nach Kompromissen, die meist so aussehen, dass der bearbeitbare Bereich technisch gesehen in ein eigenes Unter-Dokument (iframe) ausgelagert wird.

Praktisch bedeutet dies, dass den BenutzerInnen ein spezifizierter rechteckiger Bereich zur Verfügung gestellt wird, in dem bearbeitet werden kann. Ein Problem bei diesem Ansatz: Ist der Inhalt zu groß werden innerhalb dieses Bereichs Scrollbalken erzeugt. Ist der

33. Gemeint ist hier: Allen graphischen Browsern mit relevanten Marktanteilen. Textbasierte Browser wie Lynx (beispielsweise) miteinzubeziehen, wäre hier ein eigenes und tendenziell unlösbares Problem, dies ist jedoch hier nicht gemeint.
34. Exakt gesprochen gibt es natürlich Wege: Beispielsweise könnte jede Seite ein große vorgerenderte Graphik sein. Für diese Auswege ist aber meist auch ein Preis zu zahlen indem andere Nachteile in Kauf genommen werden müssen.

Inhalt deutlich kleiner nimmt dieser Bereich trotzdem die spezifizierten Fläche ein.

Ein anderes Problem ist, dass es erstaunlich nicht-trivial ist, Stylesheets zu erzeugen, die für ein HTML-Fragment sowohl in einen komplexen Seiten-Layout als auch alleine in einem iFrame das selbe Aussehen erzeugen. Zwar wäre dies vermutlich theoretisch automatisierbar, aber keiner der getesteten Rich-Text-Editoren versuchte dies durchzuführen, wohl auch in Anbetracht der Tatsache, dass (wie schon erwähnt) Stylesheets, die browserübergreifend idente Ergebnisse erzeugen, schon für sich genommen eine komplexe Angelegenheit sind.

Praktisch gesehen beschränken sich diese Rich-Text-Editoren daher auf einfachere Probleme. Beispielsweise darauf, dass „bold“ gestalteter Text schon im Editor bold ist, oder dass Überschriften markanter als einfacher Text sind.

Was sie *nicht* versuchen sicherzustellen: dass die gleiche Fontgröße (oder das gleiche Font) verwendet wird; dass Zeilenumbrüche an den selben Stellen im Text stattfinden; Einrückungen die gleiche Länge haben, etc.

Weiters besitzen viele Rich-Text-Editoren ein Fallback und ermöglichen zwischen der „Rich-Text-Anzeige“ und einer nicht-gerenderten Anzeige des erzeugten HTML-Codes umzuschalten. Dieses Fallback hat sich für mich in der Praxis real existierender Rich-Text-Editoren oft als sehr notwendig herausgestellt.

In gewisser Weise ist dies vielleicht ein Eingeständnis des Scheiterns, wenn das Ziel sein soll, dass HTML-Kenntnisse eben *nicht* notwendig sind um im WWW zu schreiben. Andererseits muss zugestanden werden, dass das Problem tatsächlich ein komplexes ist, und das Detail-Verhalten der verschiedenen Browser in diesen Edit-Modi wenig spezifiziert und daher auch manchmal unterschiedlich ist.

Und in einer anderen Sichtweise wäre es auch möglich einen HTML-Code-View in ein Feature zu verwandeln. So besaß die Textverarbeitungs-Software WordPerfect ein „Reveal Code“-Feature, dass es erlaubte eine quasi symbolisierte Repräsentation des internen Markups zu bearbeiten, dass angeblich durchaus geschätzt wurde[35]. Ein relevanter Unterschied zum Bearbeiten von HTML-Code in Stil eines Texteditors ist jedoch, dass die „Tags“ in Word-

35. Auf http://en.wikipedia.org/wiki/WordPerfect steht zum jetzigen Zeitpunkt (14. August 2008) *„The Reveal Codes feature is a second editing screen that can be toggled open and closed at the bottom of the main editing screen. Text is displayed in Reveal Codes interspersed with tags and the occasional objects, with the tags and objects represented by named tokens. The scheme makes it far easier to untangle coding messes than with styles-based word processors, and object tokens can be clicked with a pointing device to directly open the configuration editor for the particular object type, e.g. clicking on a style token brings up the style editor with the particular style type displayed. WordPerfect users forced to change word processors by employers frequently complain on WordPerfect online forums that they are lost without Reveal Codes.“*, ohne dass weitere Quellen genannt werden.

Perfect nicht bearbeitbarer Text waren sondern ein nur als ganzes manipulierbares Objekt. Das Problem ungültiger Syntax stellte sich daher nicht in der Form wie es sich beim Freihand-Bearbeiten von HTML stellt.

Es soll auch noch erwähnt werden, dass – da diese Rich-Text-Editoren üblicherweise HTML erzeugen – die Software am Server natürlich immer noch potentiell schädlichen Code herausfiltern muss. Viele unabsichtliche Fehler durch nicht geschlossene Tags können jedoch vermieden werden.

Eine letzte Variante ist momentan noch nicht verbreitet, aber (inzwischen) theoretisch möglich. Die letzte Browsergeneration unterstützt durchgängig eine Technologie mit der *Teile* eines Dokuments im Browser zur Bearbeitung freigegeben werden können. Zumindest das Ziel, dass der Text im gleichen Browser während des Bearbeitens ident, bzw. fast ident[36] aussieht, ist damit – relativ – leicht zu verwirklichen. In Kapitel 3.9 wird über disbezügliche Versuche berichtet werden.

In dieser Arbeit wird der Term „Rich-Text-Editor“ verwendet, um real existierende Hilfsprogramme wie oben besprochen zu bezeichnen. Obwohl etwas sperrig, wird der Begriff WYSIWYG-Editor verwendet für Programme, die versuchen (beziehungsweise erreichen), dass auch das Styling des Textes während des Bearbeitens imitiert wird – mit den Ausnahmen, dass das Verhalten, aber nicht das Aussehen von Links unterschiedlich sein darf und der weiteren Einschränkung, dass die Unterscheidung in einen Edit- und View-Modus nicht völlig aufgehoben wird, wie dies in diversen Desktop-Programmen der Fall ist.

Diese letzte Einschränkung hat (auch) den Hintergrund, dass Selbstbeobachtung und informelle Beobachtung im BekanntInnenkreis ergaben, dass es zu den Lese-Gesten in Web gehören kann, Teile des Textes, der gelesen wird, wiederholt zu selektieren. Kurze Selbstversuche ergaben, dass diese Geste mit dem Vorhandensein von bearbeitbarem Text insofern kollidiert, als dass der Text dann während des „Lesens“ manchmal unabsichtlich verändert wird indem eine solche Selektion verschoben wird. Weiters ist die Unterscheidung momentan notwendig, da Links während des Bearbeitens ihre Link-Funktion verlieren.

Zuletzt sei noch die Beobachtung angemerkt, dass die meisten dieser Rich-Text-Editoren eine bestimmte Sichtweise auf das Hypertext-spezifische, nämlich den Link, aufweisen. Ein Link wird tendenziell als eine Eigenschaft eines Text-Strings gesehen, so wie er über HTML erzeugt wird, und fast in Analogie zu Font-Eigenschaften wie „bold“ oder „italic“. Dies wird auch später noch Thema sein.

Die Erfahrung des Autors mit WordPerfect deckt sich jedenfalls in gewisser Weise mit dieser Aussage.

36. In den getesteten Browsern (Firefox 3.0, Opera 9.50 Beta, Safari 3.1.2) sind die Darstellungen in den „Edit“- und „View“-Modi in einem konkreten Browser jeweils pixelgleich. Aber das Verhalten von Links, beispielsweise, ist unterschiedlich.

Hier sei noch auf Abbildung 3.11 in Kapitel 3.5 verwiesen, die den Link-Erstellungs-Prozess in Wordpress zeigt. In Kapitel 3.9 wird ein Entwurf für ein Interface vorgestellt werden, das versucht, das Bearbeiten von Links von einem anderen Ausgangspunkt aus zu gestalten.

2.2 Weblogs

Es ist einfach auf eine Webseite zu zeigen und zu sagen „Das ist ein Weblog“. Aber exakt zu beschreiben was ein Weblog „ist“, ist schon komplizierter.

Was folgt ist daher eine kurze theoretische Beschreibung von Weblogs (oder kurz „Blogs“) und später ein Blick auf zwei Beispiele von Server-Software-Systemen für das Erstellen und Managen eines Weblogs.

2.2.1 Geschichte: Definitionen und Zeitrahmen

Rebecca Blood beginnt einen Überblick über die geschichtliche Entwicklung von Weblogs mit den Worten:

> „Weblogs have become so ubiquitous that for many of us the term is synonymous with ‚personal Web site,‘ though many commercial sites now incorporate them, too. For others, they are ‚sites made with blogging software,‘ which seems obvious, except that some of us still update ours by hand. In either case, the form is familiar: frequently updated, reverse-chronological entries on a single Web page.“ (Blood 2004)

Der Begriff selber sei 1997 eingeführt worden, und eine wichtige Entwicklung für Blogs sei 1999 mit der ersten weitverbreiteten Software-Plattform speziell für Weblogs passiert, die laut ihr durch die Gestaltung des Interface für die Erstellung von Einträgen die weitere Entwicklung von Weblogs geprägt hätte.

So seien frühe Weblogs vor allem um Links zu anderen Seiten zentriert gewesen, während das Interface von „Blogger“ nicht mehr den einen – zentralen – Link suggerierte sondern einfach „[..] a single form box field into which bloggers typed whatever they wanted.“ (ebd.) anbot.

> „So, with the overwhelming adoption of Blogger, and without an interface that emphasized links as the central element of the form, the blog-style Weblog was born. Much controversy ensued in the original Weblog community: These are diaries, not Weblogs; Weblogs are about links. “ (ebd.)

Die „ursprüngliche“ Form wird von Rebecca Blood als „filter-style weblog“ bezeichnet. Der Begriff bezieht sich vermutlich auf die intendierte Funktion dieser Weblogs, nämlich aus der Menge der im Netz verfügbaren Seiten besonders „relevante“ auszufiltern:

> „Some of us directed attention to notable but overlooked news stories; others provided professional information or links to the weird and wonderful Web, which we combed and filtered for our readers.“ (ebd.)

Die Eigenschaften „frequently updated“ und „reverse-chronological“ werden gerne verwendet um Weblogs zu beschreiben oder zu definieren, so auch bei Herring, Scheidt, Bonus und Wright (2004), die schreiben:

> „ Weblogs (blogs), defined here as frequently modified web pages in which dated entries are listed in reverse chronological sequence, are becoming an increasingly popular form of communication on the World Wide Web.“

Hybrid-Form

In ihrer Analyse von Weblogs kommen sie zu dem Schluss, dass Weblogs als Hybrid-Form zwischen Standard-Webseiten[37] und Computer-Mediated-Communication (also beispielsweise E-Mail oder Chat) angesiedelt werden können. Weblogs hätten das Potential die Unterscheidung zwischen den beiden „Genres“ aufzuheben. Und dieses „transformative potential“ sei auch bedingt durch technische Verbesserungen:

> „ Ultimately, we believe that blogs have the potential to change the way we think about the Web and about CMC, by rendering obsolete any hard-and-fast distinction between the two. At the root of this transformative potential are two technical enhancements provided by weblog software, neither of them revolutionary in itself. By enabling faster and easier content modification that does not require knowledge of HTML, blogs can be used by almost anyone, and be responsive to people's daily needs. Second, by enabling readers to post comments, blog software makes Web pages truly interactive, even if that interactive potential has yet to be fully exploited.“

Blogosphere

Manchmal wird die Gesamtheit aller Weblogs als ein eigenes System, ein spezieller Raum gesehen. Der Begriff der „Blogosphere“ hat auch schon Eingang in die wissenschaftliche Literatur gefunden[38] und wird dort beispielsweise als „the totality of blog-related Web sites“ [39] oder auch als „the sum of all blog entries recorded to date“ [40] definiert wird.

Unabhängig davon, ob die Blogosphere tatsächliche ein spezielles Kollektiv bzw. einen abgegrenzten Raum in einem relevanten Sinn

---

37. „[..] static, single author, multimedia HTML documents that are the standard means of communication on the World Wide Web.“ (Herring u. a. 2004)
38. Eine Suche nach „blogosphere“ auf `portal.acm.org` beispielsweise ergibt zum momentanen Zeitpunkt (22. August 2008) 185 Ergebnisse. Viele dieser Arbeiten scheinen sich mit der Analyse von oder Extraktion von Daten aus der „Blogosphere“ oder Untermengen davon zu beschäftigen.
39. in Chi, Tseng und Tatemura (2006)
40. in Zhou, Truran, Brailsford, Ashman und Pourabdollah (2008)

darstellt (wie die Benutzung des Begriffs manchmal nahelegen würde), eignen sich Mengen von Blogs unter Umständen recht gut dafür, analysiert zu werden. Kumar, Novak, Raghavan und Tomkins (2003)[41] argumentieren übrigens durchaus überzeugend, dass beide Punkte zutreffen, aber dies ist nicht direkt Teil dieser Arbeit.

Allerdings erstreckt sich damit der Blick aber nicht mehr nur auf den einzelnen Blog, sondern auch auf die Verbindungen zwischen ihnen, größere Mengen an Weblogs und das gebildete Netzwerk, und hier gibt es außer dem Ease-of-use und der Möglichkeit zu kommentieren hinaus noch zwei andere Errungenschaften im Kontext von Weblogs die erwähnenswert sind.

Linkbacks

Viele Blogs besitzen die Fähigkeit, eine Seite, die das Ziel eines Links ist, von der Existenz des Links zu benachrichtigen. Dies wird „Linkback“, „Trackback“ oder „Pingback“ genannt[42] – im weiteren wird als generische Bezeichnung „Linkback“ verwendet. Die Ziel-Seite kann – muss aber nicht – einen Link zurück auf die Ausgangs-Seite setzen.

In Hypertext-Begriffen ermöglicht dies das Setzen eines quasi-bidirektionalen Links, wenn dies von beiden Seiten gewünscht ist.

Linkback-Mechanismen helfen unter anderem dabei, Gespräche zwischen verschiedenen Weblogs in Gang zu bringen oder zu halten, denn sie benachrichtigen die AutorInnen von Weblogs über eine Reaktion auf einen ihrer Einträge, von denen sie sonst eventuell nicht erfahren hätten.

RSS-Feeds

Zweitens ermöglichen Schnittstellen wie RSS-Feeds[43] die mechanische Verarbeitung von größeren Mengen an Updates. RSS-Reader erlauben, eine große Menge an Weblogs zu verfolgen ohne sie regelmäßig manuell im Browser auf Änderungen überprüfen zu müssen. Andere Anwendungen wie „Yahoo! Pipes“ erlauben es, RSS-Feeds zusammenzufassen, zu filtern und wieder auszugeben. Beides wäre ohne diese Schnittstellen deutlich schwieriger zu bewerkstelligen.

Vor diesem Hintergrund ist auch die (in Kapitel 2.1.1 erwähnte) Begeisterung von Landow für Weblogs leichter verständlich, der ja Weblogs näher an einem „full hypertext system“ zu sehen scheint

41. Sie verwenden allerdings statt Blogosphere den Begriff „Blogspace“
42. „Trackback“ wird manchmal als generische Bezeichnung verwendet. Technisch gesehen sind Pingback und Trackback verschiedene Protokolle und zum Teil spiegelt sich das auch im User-Interface wieder, da das Trackback-Protokoll zu diesem Zeitpunkt nur einen eher unhandlichen Auto-Discovery Mechanismus besitzt und die Software somit manchmal darauf angewiesen ist, dass gewisse Informationen manuell von den AutorInnen beigesteuert werden. Siehe z.B. http://www.tamba2.org.uk/wordpress/ping/ (abgerufen am 27. März 2008), für eine Erläuterung der Unterschiede zwischen Pink- und Trackback in Wordpress. Für Spezifikationen siehe http://hixie.ch/specs/pingback/pingback (Pingback) sowie http://www.sixapart.com/pronet/docs/trackback_spec (Trackback, beide abgerufen am 27. März 2008)
43. Der Einfachheit halber wird in dieser Arbeit die ganze Klasse an Schnittstellen und APIs die einen ähnlichen Zweck dienen als „RSS“ zusammengefasst werden. Dies beinhaltet also die einzelnen RSS-Variaten und auch das „Atom“-Format.

als den Rest des „current World Wide Web“. (Landow 2006, Seite 6)

2.2.2 Beispiele für Blog-Systeme: Wordpress und Twoday

In diesem Kapitel sollen kurz zwei Beispiele für Weblog-Systeme vorgestellt und einige ihrer Merkmale besprochen werden. Eines der besprochenen Systeme ist twoday[44], ein kommerzieller Weblog-Service, der für die TU Wien Weblogs für Lehrende und Studierende bereitstellt[45]. Wordpress[46] ist im Unterschied zu twoday(E) ein Open-Source Weblog-System unter der GPL Lizenz.

Your mileage may vary. ..

Eine Warnung sei noch hinzugefügt: Die beschriebenen Eigenschaften dieser Systeme – insbesondere von Wordpress – sind mit Vorsicht zu genießen. Der Autor hat sie in dieser Form in zumindest *einer* Installation vorgefunden (bzw. nicht vorgefunden). Die Open-Source-Natur von Wordpress sowie die Verfügbarkeit von Plugins machen es jedoch schwierig bis unmöglich, für alle oder auch nur viele Installationen von Wordpress zu sprechen. Das „Interface“, wenn wir so sagen wollen, für die Aktivierung oder Konfiguration mancher Features ist es nämlich, mit einem Texteditor den Quellcode des Programms zu verändern, so dass die Grenzen zwischen Konfigurieren und Um-Programmieren hier unscharf sind.[47]

Markup / Editor

Für Hintergrund siehe auch Kapitel 2.1.4

Beide Systeme verwenden im Großen und Ganzen HTML als Markup-Sprache wobei twoday zusätzliche macro-tags, etwa für das einfachere Einbinden von in twoday hochgeladenen Bildern zur Verfügung stellt, und außerdem Zeilenumbrüche im Code in Zeilenumbrüche im Text verwandelt[48]. Weiters gibt es Buttons, die Markup im Textfeld erzeugen (Abbildung 2.5).

Wordpress hingegen stellt einen Rich-Text-Editor zur Verfügung, in dem der Text formatiert bearbeitet werden kann, jedoch das endgültige „Theme“ des Weblogs noch nicht angewandt wird. Zusätzlich kann der HTML-Code des Textes angesehen und bearbeitet werden. Dieses Interface ist dann dem Interface von Twoday ähnlich (siehe Abbildungen 2.6 und 2.7). Beide Systeme bieten die Möglichkeit eine Preview-Ansicht zu erzeugen, allerdings ist diese Funktion bei twoday etwas versteckter und erfordert es, den Artikel zuerst „offline“ zu „speichern“.

Kommentare

Die Kommentare verwenden jeweils ein ähnliches Markup wie die Einträge. Ein sichtbarer Unterschied beim Erstellen von Kommentaren und Blog-Einträgen ist bei beiden Systemen, dass die Button-Leisten beim Schreiben von Kommentaren nicht angezeigt werden.

---

44. http://twoday.net/
45. http://twoday.tuwien.ac.at/
46. http://wordpress.org/
47. Zum Beispiel die Konfiguration von RSS-Feeds. Siehe http://codex.wordpress.org/WordPress_Feeds, abgerufen am 23. August 2008
48. In HTML werden Zeilenumbrüche im Code eigentlich ignoriert. Streng genommen handelt es sich bei twodays Markup-Sprache eigentlich weder um HTML noch um ein Subset. Aber im Prinzip wird HTML-Syntax verwendet.

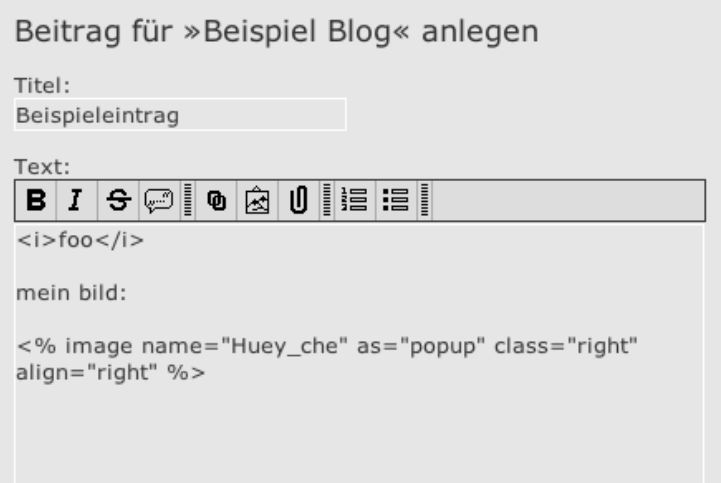


Abbildung 2.5: Textformatierung in Twoday: Das HTML-Markup kann entweder manuell oder durch Anklicken der Buttons erzeugt werden. Der Button für die Funktion „Bild einfügen“ erzeugt ein Popup-Fenster mit einem Wizard bzw. Gallery-Viewer

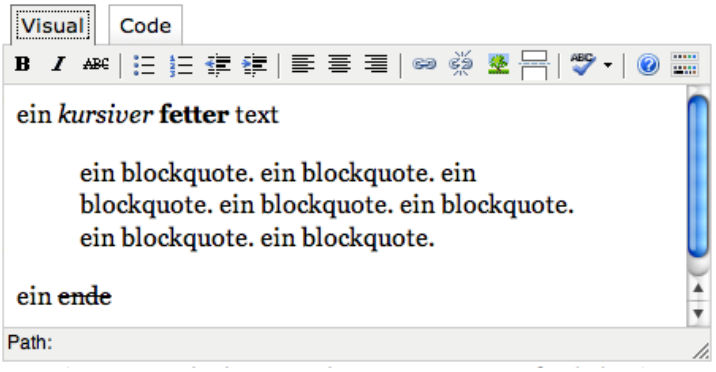


Abbildung 2.6: „Visual“-View in Wordpress. Das Aussehen des Textes wird später noch durch das gewählte Theme beeinflusst.

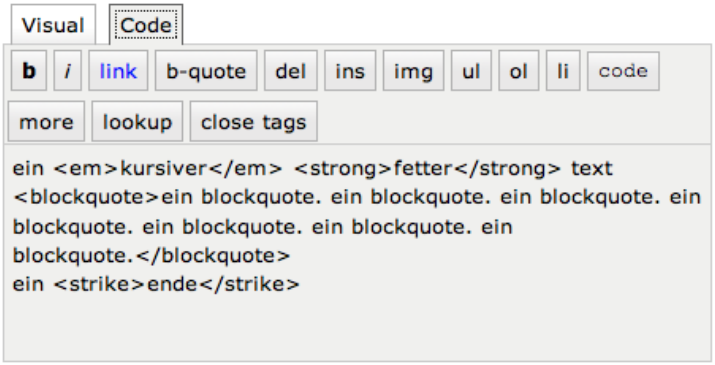


Abbildung 2.7: Code-View in Wordpress

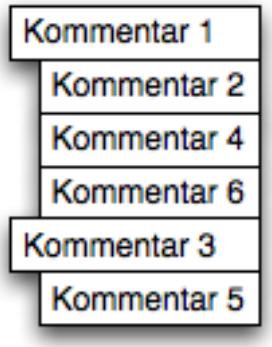


Abbildung 2.8: Kommentar-Anzeige twoday-Style.

In Wordpress werden Kommentare einfach der Reihe nach aufgelistet (älteste oben). Twoday bietet die Möglichkeit auf einzelne Kommentare zu antworten, wobei Antworten eingerückt werden, allerdings nur eine Ebene weit. (siehe Abbildung 2.8).

Linkbacks

Wordpress unterscheidet dabei in der Darstellung nicht notwendigerweise zwischen Track- bzw. Pingbacks und Kommentaren (siehe Abbildung 2.9), während twoday sie getrennt auflistet. Das Interface für Outbound Trackbacks ist sowohl bei Wordpress als auch bei twoday ein Textfeld, in dem die „Trackback-Urls“ eingegeben werden können. Wordpress versucht zusätzlich alle im Text verlinkten Seiten automatisch mittels Pingback zu benachrichtigen (sofern dies nicht in den Optionen abgeschalten ist).

Beide Systeme können Einträge und Kommentare des Weblogs importieren und exportieren, wobei Wordpress die größere Vielfalt an Import-Möglichkeiten zu bieten scheint, und gleichzeitig demonstriert, dass eine große Anzahl an Dateiformaten für Weblog-Archivierung bzw. Im- und Export existieren.

RSS-Feeds

Sowohl Wordpress als auch twoday bieten RSS-Feeds an. In Word-

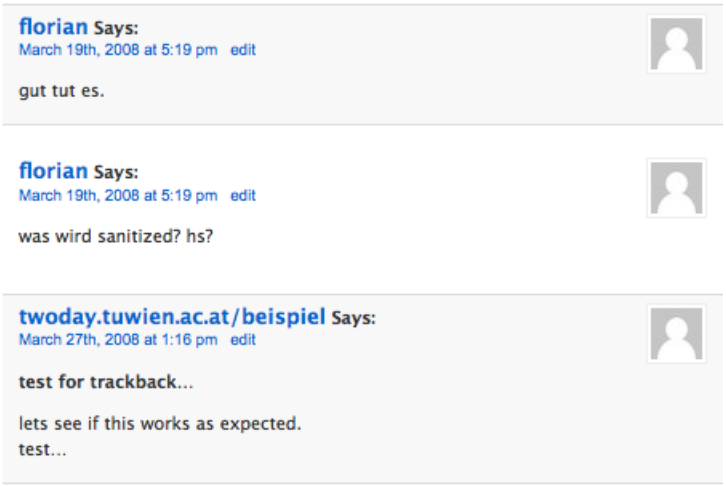


Abbildung 2.9: Kommentare und eingehende Links werden in Wordpress oft ident dargestellt. Die zwei oberen Einträge sind Kommentare, der letzte Eintrag ist ein Trackback. Statt einem Namen hat der Eintrag des Trackbacks eine URL, aber ansonsten sind sie hier identisch.

press beinhaltet der RSS-Feed die Einträge eines Weblogs[49], während twoday auch die Option bietet, Einträge *und* Kommentare über RSS zu publizieren.

Kategorien und Tags

Eine Unterschied zwischen den Systemen besteht bei der Zuordnung von Einträgen zu „Kategorien“. In twoday gehört ein Eintrag zu *einer* bestimmten Kategorie oder gar keiner. In Wordpress hingegen können Einträge beliebig vielen Kategorien zugeordnet werden. Gleichzeitig bietet es auch „Tags“, wobei der Unterschied zwischen Tags und Kategorien hauptsächlich historische Ursachen hat[50] und somit scheinbar verschiedene Interfaces bedingt.

Beide Systeme können durch vorgefertigte Themes in Aussehen und Layout angepasst werden.

2.3 Wikis

Definition ...

„Wiki“ ist eine Bezeichnung für eine Art von Hypertext-Authoring-System, zumeist Web-basiert. Leuf und Cunningham (2005) schreiben:

49. Wie jedes andere Feature in Wordpress ist natürlich auch dies „hackable“. Siehe z.B. http://codex.wordpress.org/Customizing_Feeds (abgerufen am 26. März 2008)
50. siehe http://faq.wordpress.com/2007/09/21/the-difference-between-tags-and-categories/ (abgerufen am 26. März 2008)

> „The WikiWikiWeb server concept, most often called simply ‚a wiki', originated with Ward Cunnigham. A wiki is a freely expandable collection of interlinked Web ‚pages', a *hypertext system* for storing and modifying information – a *database*, where each page is easily editable by any user with a forms-capable Web browser client." (Seite 14; Hervorhebungen im Original)

Die Pointe ist hier „easily editable by any user". Es ist einfach, vom Erfolg von Wikis überrascht zu sein. Désilets, Paquet und Vinson (2005) schreiben:

... und Philosophie

> „From the point of view of innovation in the process, philosophy and sociology of collaborative web authoring, wiki introduced a new way of thinking that favours:
>
> - Content over Form
> - Open Editing over Security and Control
> - Free Form Content over Structured Content
> - Incremental Growth over Upfront Design
>
> While this may at first *look like a recipe for disaster* except for small and obscure web sites, it turns out to be a reasonable strategy with attributes that enable a wide variety of applications." (Hervorhebung hinzugefügt)

Inzwischen gibt es viele verschiedene Wiki-Systeme. Wenn wir die oben erwähnte Definition von Leuf und Cunningham (2005) betrachten, sind die Anforderungen an eine Wiki-System zunächst nicht sehr groß, und bestehen im Kern nur aus der Forderung, dass es sich um eine Webseite handelt und alle Inhalte im Browser bearbeitbar sind. „Any user" könnte die Abwesenheit von Access Control meinen und tut dies wohl auch oft, allerdings sicher nicht immer.

Wikipedia ...

Eine Seite, die vermutlich beigetragen hat, den Bekanntheitsgrad von Wikis zu erhöhen, ist die Wikipedia[51]. Nicht nur wissenschaftliche Arbeiten sind über dieses Wiki geschrieben worden[52] sondern auch Berichte in den Medien, etwa Zeitungen.[53]

... ist vielleicht kein typisches Beispiel.

Allerdings ist es wichtig, bei „Wiki" nicht nur und vor allem an Wikipedia zu denken. Die Größenordnung der Menge von Inhalten und BenutzerInnen der Wikipedia übertrifft sicherlich die der meisten

---

51. http://www.wikipedia.org (zuletzt abgerufen am 6. Oktober 2008)
52. Ein Beispiel: (Zhang 2006)
53. Es ist hier leicht Beispiele zu finden, und schwierig eines herauszugreifen. http://news.google.com etwa lieferte am 7. Oktober 2008 für den Suchbegriff „wikipedia" 6,825 Ergebnisse für den letzten Monat und 54,400 Ergebnisse für eine Suche über „All dates". Die Zahlen sind vermutlich etwas willkürlich und verändern sich wenn die Einstellungen der Suche verändert werden. Sie zeigen aber zumindest, dass Ergebnisse existieren. Es sei auch noch erwähnt dass ich einen Vergleich mit einem weniger bekannten Wiki anstellen wollte, allerdings ist mir kein anderes Wiki eingefallen, dass in der Nähe der medialen Aufmerksamkeit kommt, die Wikipedia geschenkt wird.

existierenden Wikis. Dies macht die Wikipedia zu einem vermutlich eher untypischen Fall.

2.3.1 Eigenschaften und potentielle Schwächen von Wikis

Das Ziel hier ist es, Wege zu suchen, auf denen Wikis verbessert werden könnten. Insofern ist es natürlich spannend, sich nicht nur darauf zu konzentrieren was Wikis potentiell gut können, sondern auch zu fragen, wo den Schwächen von Wikis liegen könnten.

Désilets u. a. (2005) beschreiben (wie schon auf Seite 31 zitiert) in ihrer Studie über die Usability von Wikis die Schwerpunkte der „Wiki-Philosophie“ mit den Punkten:

- „Content over Form“
- „Open Editing over Security and Control“
- „Free Form Content over Structured Content“
- „Incremental Growth over Upfront Design“

Diese Formulierung lädt dazu ein, potentielle Schwächen von Wikis in den Bereichen, die angeblich den Anti-Focus darstellen, zu suchen, was ich im Folgenden mit den ersten drei Punkten („Form“, „Security and Control“ sowie „Structured Content“) versuchen möchte.

Schwäche: Form?

Dem Autor ist unbekannt, ob Studien über die Ästhetik von Wikis existieren. Aufgrund der Vielzahl von Wikis und Wiki-Systemen wäre es vermutlich schwierig festzustellen, ob Wikis *im Allgemeinen* weniger „schön“ sind als andere Webseiten. Des Weiteren ist „schön“ vielleicht eine nicht ganz triviale Metrik.

Allerdings sollte hier festgehalten werden, dass – subjektiv empfundene – Mängel von Wikis in diesem Bereich des Aussehens durchaus zur Motivation dieser Arbeit beigetragen haben. Und es wirkt sicherlich so, als wäre ansprechendes Aussehen kein Focus in Wiki-Technologie und -Philosophie bzw. sogar ein Aufwand der vermieden werden sollte. Beispielsweise schreibt Ron Jeffries, in erwähntem Vorwort von Leuf und Cunningham (2005):

> „Wiki technology is simple—it looks almost crude. You'll see many ways to make it better, to make the pages prettier, to give it more structure. Other than putting in just enough security—if you do even that—please hold off.“

Der Verdacht liegt hier also nahe, dass ein minimalistisches Design in manchen Wikis vielleicht durchaus Absicht sein könnte.

Verschiedene Untersuchungen legen nahe, dass Ästhetik und Usability (form und function, sozusagen) nicht unabhängig von einander existieren. Der Zusammenhang geht vielleicht sogar so weit, dass Menschen eventuell mit als hübsch empfundenen Gegenständen und Schnittstellen tatsächlich besser umgehen können, wie dies Norman (2004) argumentiert, da ein angenehmes Aussehen die BenutzerInnen in einer Geisteshaltung unterstützt würden, in der leichteres oder verspielteres Problemlösen möglich sei.

Auf jeden Fall scheint die empfundene Ästhetik stark mit der empfundenen Usability zu korrelieren (Tractinsky, Katz und Ikar 2000). Allerdings sind diese Effekte sehr wahrscheinlich unter anderem auch beeinflusst von dem Hintergrund der Personen, dem Kontext und der gestellten Aufgabe (Hartmann, Sutcliffe und Angeli 2007).

Diese Erkenntnisse öffnen die Tür für einige eher spekulative Gedanken, die ich hier anführen will.

Erstens die Theorie, dass Wikis in den Augen ihrer DesignerInnen und ProgrammiererInnen unter Umständen durchaus so aussehen wie gewünscht. Bei der Gestaltung einer Webseite gibt es unzählige Entscheidungen die getroffen werden können und sogar getroffen werden müssen – sogar die Entscheidung, etwa nur logisches Markup zu verwenden und keinerlei CSS zu verwenden (was im übrigen kein mir bekanntes Wiki-System macht). Arrangement der Teilelemente ist ebenfalls ein gestalterisches Moment, dass schwierig (oder unmöglich) zu vermeiden ist.

Das Aussehen von Wikis wäre dann nicht ein Zufall der auf mangelnder Aufmerksamkeit beruht, sondern z.B. eine bewusste oder unbewusste Entscheidung für einen Retro-Look. Die Kehrseite wäre dann, dass diese Design-Entscheidungen bei anderen Zielgruppen weniger gut ankommen könnten und in diesem Fall eventuell breitere Akzeptanz von Wiki-Systemen behindern würden.

Die zweite Spekulation ist, dass ein Design, das mit vorherrschenden Ästhetik-Konventionen bricht, für ein Wiki-System den Vorteil bringen könnte, dass für potentielle AutorInnen die Hemmschwelle sinken könnte, Inhalte beizusteuern. Anders gesagt: Wenn es schon ein bisschen kaputt oder ungepflegt aussieht, sinkt bei manchen vielleicht die Angst, Sachen kaputt zu machen. Und Menschen, die gern positive Veränderungen durchführen wollen, sind vermutlich die Zielgruppe, die Wiki-Systeme ansprechen wollen.

Umgekehrt wäre die Konstruktion einer gewissen Nicht-Seriösität dann hinderlich, wenn sie LeserInnen nur deswegen davon abhält, die Informationen auf der Seite für relevant oder richtig zu halten.

Eine letzte wilde Spekulation: Vielleicht wäre die Überraschung und die Aufregung über die Qualität der Inhalte von Wikipedia weniger groß, wenn die Seite gängige visuelle Techniken einsetzen würde um Professionalität zu suggerieren. Vielleicht aber auch nicht.

In jeden Fall ist es vermutlich nicht zielführend, das Aussehen von Webseiten komplett außer Acht zu lassen, wenn die Benutzbarkeit und Attraktivität verbessert werden soll.

Security and Control

Unter „Security and Control“ können verschiedene Aspekte gemeint sein. Drei verschiedene Aspekte von Kontrolle will ich kurz behandeln.

Zum Ersten kann Kontrolle im Sinn von „access control“ gemeint sein. Also in Code eingeschriebene Verbote, Dinge zu tun (z.B. auf Ressourcen schreibend oder lesend zuzugreifen) sowie

Mechanismen, diese Verbote aufzuheben (klassischerweise eine Accountname/Passwort-Kombination).

Die klassische Wiki-Idee sieht hier tatsächlich vor, auf Verbote weitgehend zu verzichten. Ein Account-Mechanismus ist im Gegensatz dazu meist vorgesehen, vermutlich um die Möglichkeit zu haben, TeilnehmerInnen mit einer Identität auszustatten, aber auch weil auf Verbote eben nur weitgehend aber nicht komplett verzichtet wird.

Kontrolle und Formung der Inhalte passiert nach diesem Konzept potentiell im Nachhinein und funktioniert über die Versionierung von Seiten, also die Möglichkeit Änderungen am Inhalt nachzuvollziehen und Vandalismus oder ungewollte Inhalte wieder löschen, bzw. leicht rückgängig machen zu können.

Das wäre tatsächlich ein Unterschied zum Ansatz, Inhalte zu formen, indem versucht wird, die Personengruppe zu definieren, die diese Inhalte vermutlich generieren kann und soll. Die Vorstellung, Lese-Zugriff einzuschränken, ist damit jedoch tendenziell abwesend, und somit erfordert es manchmal externe Mittel, um eine Art von Privacy herzustellen.

Dieser Ansatz erfordert, dass die Änderungen in Wikis tatsächlich verfolgt werden, dass es (genügend) Menschen und Arbeitszeit gibt diese Änderungen zu erkennen und rückgängig zu machen. Ein unbeaufsichtigtes und ansonsten nicht abgesichertes Wiki lauft in Gefahr unbeabsichtigt Spam oder andere unerwünschte Inhalte zu beherbergen. Auch stoßt er an seine Grenzen, wenn die Ressourcen der „SpammerInnen“ erheblich größer sind als die der „richtigen“ BenutzerInnen. Unter Anführungszeichen, da in dieser Situation sich natürlich die Frage stellt wer denn nun definieren darf was erwünschte und unerwünschte Inhalte sind. In jeden Fall ist es vermutlich nicht zielführend zu erwarten, dass ein Wiki, einmal aufgesetzt und mit den „richtigen“ Inhalten gefüllt, für alle Zeiten den eigenen Ansprüchen genügen wird.

Die Menschen (oder Institutionen), die die Rechner kontrollieren auf denen die Software läuft, sind natürlich auch in der Lage zusätzliche Einschränkungen zu implementieren, auch wenn dies von manchen als Abkehr vom der Wiki-Philosophie empfunden wird. In Wikipedia, beispielsweise, wurde eine spezielle Art bestimmte Seiten zu sperren namens „Semi-Protection“ eingeführt, die nur relativ neue und anonyme BenutzerInnen vom Bearbeiten einer Seite abhält.[54]

Offene und geschlossene Wikis

Erwähnenswert ist zu diesem Thema auch noch die Perspektive, dass zwei verschiedene und typische Modi für Wikis existieren: Die des zum gesamten Internet hin offenen Wikis; und die des „privaten“, also durch Access-Controls für die meisten Menschen gesperrten Wikis.

---

54. siehe http://en.wikipedia.org/wiki/Wikipedia:Semi-protection_policy#Semi-protection, bzw. ein Zitat gefunden unter http://slashdot.org/article.pl?sid=05/12/24/0843220: „Does this mark the end of the free encyclopedia that anyone can edit?“, beide abgerufen am 20. Februar 2008

Letzteres beispielsweise für eine kleine Projektgruppe oder in einem Intranet.[55] Vandalismus-Probleme existieren in einem solchen Szenario weniger, dafür die Anforderung den Zugriff auf eine endliche Gruppe einzuschränken.

Kontrolle über das „Werk“

Soweit zur technischen Betrachtungsweise. Auf der Ebene der individuellen Autorin (oder des Autors) fällt – insbesondere im Vergleich von Wikis mit Weblogs – auf, dass diese in einem Wiki relativ wenig „Kontrolle“ hat, sowohl über „ihren“ oder „seinen“ Beitrag als auch über den Kontext in dem der Beitrag steht. Ein geschriebenes Fragment kann von anderen in jeder Form verändert werden, es kann in einen Kontext gestellt werden die der oder dem „ursprünglichen“ Autorin oder Autor gar nicht mehr recht ist.

Konventionen vs. Free Form

Zum zum dritten fällt auf, dass in Wikis sehr viel, und zum Teil recht explizit, soziale Kontrolle ausgeübt wird. Iorio und Zacchiroli (2006) merken an, dass sich in Wikis Konventionen entwickeln die Inhalte erfüllen sollen. Unabhängig davon wie diese eingeschätzt werden – in ihrem Artikel nennen sie diese „best practices“[56] – handelt es sich doch um ein Regelset, dessen Einhaltung zum Teil überwacht und notfalls erzwungen werden muss.

Soziale Kontrolle vs. Anonymität

Betrachten wir kurz das Aushängeschild von Wikis, die Wikipedia.[57] Hier existiert eine Große Menge[58] an Konventionen, inhaltlicher und formeller Natur. Und zum Teil wird ihre Einhaltung erzeugt, indem auf Anonymität weitgehend verzichtet wird. IP-Adressen werden standardmäßig mitgeschrieben und unter Umständen machinell ausgewertet.[59] BenutzerInnen, die sich keinen Account anlegen wollen haben weniger Rechte – im Sinne von access rights – als angemeldete BenutzerInnen, und junge Accounts haben weniger Rechte als alte.

Die IP-Adressen nicht angemeldeter (schreibender) BenutzerInnen werden zusätzlich relativ offen und prominent dargestellt[60]. Wollen BenutzerInnen nicht ihre IP-Adressen mit den Änderungen im Wi-

---

55. Siehe (Ebersbach, Glaser und Heigl 2006, Seite 11): „Generally, we differentiate between two application options with wikis: They can be used as tools in a closed work group, or they can be directed at potentially everybody over the WWW.“
56. Wie oben erwähnt ist in einem offenen Wiki wahrscheinlich eine Art informeller Konsens über das Ziel und Intention des Wikis notwendig – beziehungsweise in diesem Zeitalter des organisiertem Spams eine Abgrenzung zu ebendiesem.
57. Dies ist vermutlich ein Extrembeispiel, und vermutlich nur bedingt aussagekräftig für die größere Menge an vergleichsweise kleineren und weniger besuchten Wikis. Allerdings hat der Autor in seinen eigenen Erfahrungen mit Wikis ähnliche Tendenzen auch in kleineren Projekten erlebt.
58. Der Autor hat nicht versucht sich hier einen wirklichen Überblick zu verschaffen, ist aber beeindruckt. Die Seite `http://en.wikipedia.org/wiki/Category:Wikipedia_official_policy`, beispielsweise, listete am 28. August 2008 53 Seiten und 6 Unterkategorien auf. Die schiere Menge an Text und miteinander interagierenden Regeln und Guidelines stehen auf den ersten Blick in einem spannenden Widerspruch zu „Free Form Content“ wenn schon nicht zu „Open Editing“.
59. Beispielsweise von `http://wikiscanner.virgil.gr/` (abgerufen am 4. Oktober 2008)
60. In den Listen der letzten Änderungen, global und lokal für eine Seite.

ki verknüpft haben, müssen sie sich einen Account anlegen. Womit ihre Identität in Zukunft u.U. auch bei Änderungen der IP-Adresse nachvollzogen werden kann.

Als informelle Übung hat der Autor versucht sich über das TOR-Netzwerk[61] anonym einen Wikipedia-Account anzulegen, was in 9 von 10 Fällen scheiterte, da die TOR-Exit-Nodes meist für das Anlegen neuer Accounts gesperrt waren.

Diese Maßnahmen sind durchaus verständlich vor dem Hintergrund, dass das vermutlich bekannteste Wiki-System einer große Menge an Attacken ausgesetzt ist. Trotzdem handelt es sich um Kontroll- und Sicherheitsmaßnahmen, hier mit dem Nebeneffekt, dass es relativ schwierig ist anonym oder pseudonym Inhalte in diesen Systemen zu schreiben.

Zusammenfassend ergibt sich also ein uneinheitliches Bild von Kontrolle in Wikis (für einen eher diffusen Begriff von Kontrolle). In der Sicht des Autors scheinen Wikis jedoch weniger durch die Abwesenheit von Kontrolle als vielmehr durch die Anwesenheit vieler verschiedener und subtilerer sozialer und technische Kontroll-Mechanismen charakterisiert zu sein. Und mit dem Fokus auf das gemeinsame Werk scheinen die Kontroll-Möglichkeiten des Individuums nicht sehr ausgeprägt zu sein.

Möglichkeit für Struktur

Was strukturierte Inhalte betrifft, besteht vermutlich ein Bedarf an *Möglichkeiten* Wiki-Seiten stärker zu strukturieren. Haake, Lukosch und Schümmer (2005) argumentieren diesen Bedarf[62] und schlagen als Abhilfe Wiki-Templates vor, ein System um Bestandteile und Struktur von Wiki-Seiten vorzugeben. Iorio und Zacchiroli (2006) empfehlen „Light Constraints“ um den BenutzerInnen die Möglichkeit zu geben, den Seiteninhalt gegen formalisierte Bedingungen zu verifizieren, und ergänzen das Argument damit, dass viele Wikis informelle oder explizite „best practices“ einsetzen, die die Freiheit der Form des Inhaltes in der Praxis einschränken.

2.3.2 Das Mediawiki-System

Als ein Beispiel für eine Wiki-Software soll hier MediaWiki[63] betrachtet werden. Einerseits, weil das zur Zeit wohl bekannteste Beispiel für Wikis, die WikiPedia [64] auf der dieser Software beruht, beziehungsweise MediaWiki ursprünglich für das WikiPedia-Projekt entwickelt wurde[65]. Zum anderen, weil sich MediaWiki darüber

---

61. http://www.torproject.org/ (abgerufen am 4. Oktober 2008)
62. „Examples where the need for structured representation of content in wikis becomes obvious include (1) more and more complex wiki syntaxes, e.g., as means to express text-boxes as in the WikiPedia [..], (2) provision of means to edit only parts of a wiki-page [..], or (3) the development of special purpose wikis such as the XPSwiki [..], offering built-in page-types with different text input boxes and specialized input fields to support the XP-Planning game.“ (Haake u. a. 2005, Seite 41)
63. http://www.mediawiki.org/ (zuletzt überprüft am 7. September 2008)
64. http://www.wikipedia.org/ (zuletzt überprüft am 7. September 2008)
65. „Originally developed to serve the needs of the free content Wikipedia encyclopedia, today it has also been deployed by companies as an internal know-

hinaus Verbreitung hat, und die Mehrzahl kleiner, privater Wikis in Umfeld des Autors auf Mediawiki beruhen, und somit eine gewisse Vertrautheit der Anwendung sowie leichter Zugang zu nicht-öffentlichen Anwendungs-Fällen möglich ist.

Diskussionen in Mediawiki

Die Mediawiki-Software stellt zu jeder („Artikel“-)Seite eine zusätzliche „Diskussions“-Seite zur Verfügung. Auf diesen findet oft ein Austausch der am Schreiben der Artikel-Seite beteiligten Menschen statt, auch können Kommentare hinterlassen werden, die auf Fehler, Auslassungen, etc der Hauptseite hinweisen.

Allerdings ist diese „Diskussions“-Seite nur eine weitere Wiki-Seite, d.h. die Gesprächsteilnehmer simulieren (zum Teil, nicht immer!) über spezielle Wiki-Syntax das treaded-view-Aussehen wie es zum Beispiel in Mail-Clients, Usenet-Readern oder auch Webforen vorkommt, also Einrückungen und Darunterschreiben bei Antworten und die Information wer und wann den darüberstehenden Text geschrieben hat.

Das hat natürlich den Vorteil, dass die TeilnehmerInnen, sofern sie die Wiki-Syntax kennen und anwenden können, keine neue Markup-Sprache lernen müssen. Auf der anderen Seite setzt es eine Hemmschwelle für neue BenutzerInnen: Die gesamte Apparatur der Markup-Sprache, der Benutzung der Bearbeitungs-Instrumente muss bekannt sein, bevor auch nur ein Kommentar hinterlassen werden kann.

Und noch ein Problem kann auftreten: da jeder Text von jedem und jeder verändert werden kann, ist es für TeilnehmerInnen durchaus möglich die Beiträge anderer zu bearbeiten oder zu löschen. Solche Tätigkeiten, oder auch nur der Vorwurf derselben kann einer hitzigen Diskussionen eine neue Ebene der Verwirrung hinzufügen. [66]

Die Markup-Sprache von Mediawiki ist inzwischen recht komplex und durchaus mächtig. Allerdings kann der Markup-Code schnell überwältigend wirken. Als Beispiel sind in den Abbildungen 2.10 und 2.11 Screenshots der Wikipedia-Seite zu „Wiki“ zu sehen, sowohl im der Ansicht auf den Code als auch gerendert. Zumindest der Autor würde nicht spontan „leicht leserlich“ zum Markup-Code für diese Seite assoziieren.

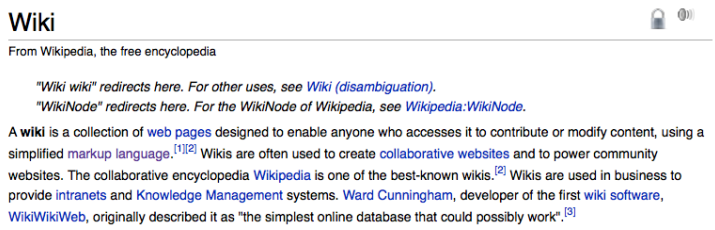
Es sei auch ergänzt, dass die Markup-Sprache von Mediawiki vermutlich bereits den Rahmen einer reinen Formatierungs-Sprache gesprengt hat, nachdem unter anderem Funktionen für Transclusion[67] – also die Einbettung anderer Seiten in eine Seite – sowie logische

---

ledge management solution, and as a content management system.“ (von http://en.wikipedia.org/wiki/Mediawiki, zuletzt abgerufen am 7. September 2008)

66. Der informellen Beobachtungen des Autors zufolge kann es in anderen Fällen eine freundliche Restrukturierung der Gesprächsbeiträge für die Diskussion vielleicht durchaus nützlich sein. Sie hinterlässt trotzdem unter Umständen Verwirrung bei den TeilnehmerInnen.

67. siehe z.B. http://en.wikipedia.org/wiki/Wikipedia:Transclusion (zuletzt abgerufen am 7. September 2008)



Wiki

From Wikipedia, the free encyclopedia

"Wiki wiki" redirects here. For other uses, see Wiki (disambiguation).
"WikiNode" redirects here. For the WikiNode of Wikipedia, see Wikipedia:WikiNode.

A **wiki** is a collection of web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified markup language.[1][2] Wikis are often used to create collaborative websites and to power community websites. The collaborative encyclopedia Wikipedia is one of the best-known wikis.[2] Wikis are used in business to provide intranets and Knowledge Management systems. Ward Cunningham, developer of the first wiki software, WikiWikiWeb, originally described it as "the simplest online database that could possibly work".[3]

Abbildung 2.10: Screenshot/Auszug aus der Wikipedia-Seite zu „Wiki“. Siehe auch Abbildung 2.11

```
{{pp-semiprotected|small=yes}}
{{Redirect|Wiki wiki|other uses|Wiki (disambiguation)}}
{{selfref|"WikiNode" redirects here. For the WikiNode of Wikipedia, see
[[Wikipedia:WikiNode]].}}
A '''wiki''' is a collection of [[web page]]s designed to enable anyone who
accesses it to contribute or modify content, using a simplified [[markup language]].
<ref>[http://dictionary.oed.com/cgi/entry/50293088 wiki, n. ] [[Oxford English
Dictionary]] (draft entry, March 2007) Requires Paid Subscription</ref><ref
name="Britannica">{{cite encyclopedia|title=wiki|encyclopedia=[[Encyclopædia
Britannica]]|volume=1|publisher=[[Encyclopædia Britannica,
Inc.]]|date=2007|location=[[London]]|url=http://www.britannica.com/eb/article-
9404276/wiki|accessdate=2008-04-10}}</ref> Wikis are often used to create
[[collaboration|collaborative]] [[website]]s and to power community websites. The
collaborative encyclopedia [[Wikipedia]] is one of the best-known wikis.<ref
name="Britannica"/> Wikis are used in business to provide [[intranet]]s and
[[Knowledge Management]] systems. [[Ward Cunningham]], developer of the first
[[wiki software]], [[WikiWikiWeb]], originally described it as "the simplest online
database that could possibly work".<ref>{{cite web|url=http://www.wiki.org
/wiki.cgi?WhatIsWiki|title=What is a Wiki|accessdate=2008-04-10|publisher=
[[WikiWikiWeb]]|author=[[Ward Cunningham|Cunningham, Ward]]|date=2002-06-27}}</ref>
```

Abbildung 2.11: Screenshot/Auszug der Edit-Ansicht auf die Wikipedia-Seite zu „Wiki“. Siehe auch Abbildung 2.10. Beide Screenshots wurden am 25. August 2008 erstellt.

Operatoren und Kontroll-Fluß-Elemente[68] existieren.

2.4 Wikis vs. Weblogs - ein Vergleich

In diesem Kapitel will ich auf das vorher gesagte zurückgreifen und Weblogs und Wikis vergleichen. Zunächst scheinen diese Systeme Ähnlichkeiten aufzuweisen. Beispielsweise sind sie in etwa um die gleiche Zeit entstanden.

Sowohl Weblogs als auch Wikis sind Web-basierte Hypertext-Authoring-Systeme[69], und beide haben mit ihrer individuellem Entwurf für Ausdruck und Kollaboration Enthusiasmus – um nicht zu sagen Hype – ausgelöst. Beide Entwürfe können und werden auch in die grobe (und nicht ganz klar abgegrenzte) Kategorie von „Web 2.0“ eingeordnet werden, wie beispielsweise von Millard und Ross (2006).

Bei einer näheren Betrachtung zeigen sich jedoch auch deutliche Unterschiede. In mancher Hinsicht sind Weblogs und Wikis sogar

---

68. siehe z.b. http://meta.wikimedia.org/wiki/ParserFunctions (zuletzt abgerufen am 7. September 2008)
69. Was zumindest zu Beginn dieser Arbeit den Autor zur Annahme verführt hat, dass sie beide nur verschiedene Seiten der gleichen Münze wären.

entgegengesetzte Punkte auf einer Skala.

Beispielsweise bei der zu erwartenden Konstanz der Inhalte eines Lexias. Von einer Wiki-Seite wird erwartet, dass sie in der Zukunft unter Umständen häufig geändert werden wird, auch von vielen verschiedenen Personen.[70] In einem Weblog ist dagegen die Erwartung, dass sich ein einzelner Beitrag nicht ändern wird, bzw. dass der Autor oder die Autorin dies explizit anmerkt, wenn es doch einmal passiert.

Ein einzelnes Wiki dient typischerweise und in der Intention der Kollaboration einer Menge theoretisch (technisch) gleichberechtigter TeilnehmerInnen. In einem Weblog ist dagegen typischerweise eine einzelne Person[71] privilegiert, diese kann Beiträge schreiben und den Weblog verwalten und gestalten. Die Teilnahme aller anderen beschränkt sich auf Kommentieren, bzw. verlinken (was, wie besprochen, mit Linkbacks eine ähnliche Form annehmen kann).

Eine Metapher für den Unterschied wäre vielleicht: Während Wikis einen großen von allen gemeinsam kontrollierten Raum darstellen wollen, erzeugen Weblogs individuell kontrollierte Orte, die sich miteinander ad-hoc verbinden können.

Ein weiteres Modell für die Unterschiede wäre, dass Weblog-Einträge mehr *Nachrichten* wie einer E-Mail oder einem Foreneintrag ähnlich sind als Standard-Webseiten, wie Herring u. a. (2004) bemerken.

Wie eine E-Mail, ein Post in einer Newsgroup oder einem Webforum besitzt ein Weblog-Eintrag beispielsweise eine Autorin oder einen Autor. Weiters gibt es ein „Subject“, eine Überschrift und vor allem einen expliziten Zeitpunkt an dem der Eintrag geschrieben wurde. Ein Weblog-Eintrag kann – im Gegensatz zu einer Mail – nachträglich geändert werden, ähnlich einem Foren-Eintrag. Aber wie bei einem Foren-Eintrag ist dies tendenziell unüblich, und höflicherweise begleitet von einer expliziten Anmerkung.

2.5 Blokis

Wenn eine Person oder Personengruppe sowohl ein Wiki, als auch einen Weblog einsetzt, kann es passieren, dass der Wunsch entsteht die beiden Systeme zu vereinen. Der Grund dafür kann recht einfach sein. Beispielsweise kann der Wunsch nach einen einheitlichen Aussehen vorhanden sein, oder der Wunsch nicht zwei verschiedene BenutzerInnen-Schnittstellen lernen zu müssen.

---

70. Ausnahmen exisiteren. Beispielsweise wenn Sitzungs-Protokolle in einem Wiki gespeichert werden (eine Anwendung die dem Autor begegnet ist). Aber auch hier ist vorstellbar, dass später noch Änderungen an der Formatierung oder der Verlinkung vorgenommen werden.

71. Oder auch eine kleine Gruppe mehrerer Personen. Der Punkt ist hier, dass die a-priori Annahme dass ein Weblog mit einer Person korreliert, im großen und ganzen möglich ist. Kumar u. a. (2003), beispielsweise, gehen von einer solchen Annahme aus.

Das Aussehen ist dabei das geringere Problem. Mit ein bißchen[72] Aufwand ist es möglich ein Wiki und einen Blog einander im Layout anzupassen. Die Wordpress-Homepage kann hier als Beispiel dienen.[73]

Dies bedeutet jedoch auch weiterhin zwei verschiedene Interfaces bedienen zu müssen. Eine Variante, die zumindest dieses Problem löst, ist es ein Wiki-System als Weblog zu verwenden, oder eine Weblog-Software als ein Wiki. Für ersteres kann vielleicht „Martin Fowler's Bliki" als Beispiel dienen. Es sei jedoch erwähnt, dass eines der prominenteren Kriterien für ein Wiki, die Idee dass viele verschiedene Leute die Text bearbeiten, fallengelassen wird.[74] In der anderen Richtung existiert Blog-Software die es erlaubt „Seiten" außerhalb des Weblog-Eintrags-Schemas zu schreiben. Hier kann wiederrum Wordpress als Beispiel dienen. Auch hier ist allerdings ein Problem, dass die Software nicht so leicht auf eine „OpenEditing"-Philosophie adaptiert werden kann.

Wenn wir „Bloggen" vor allem als eine Tätigkeit sehen wollen, und Wikis als (per WWW) schreibbare Orte mit sozialen Konventionen, kann im Prinzip jede Content-Management-Software, die flexibel genug ist, beide Konzepte abbilden. Ob dies dann auch bequem ist, und andere Eigenschaften von Weblogs und Wikis gut abbilden kann, ist natürlich eine andere Frage.

Für Versuche Weblogs und Wikis zu vereinen sind verschiedene Bezeichnungen denkbar.[75] Hier ist die Wahl auf „Bloki" gefallen. Diese Bezeichnung erlaubt auch, den Buchstaben „O" als für „Overview-Map" stehend zu deklarieren, bei Systemen mit einer solchen.

2.6 Overview-Maps und Spatial Hypertext

In diesem Kapitel soll ein – vielleicht weniger bekannter –Bereich der Hypertext-Forschung und -Theorie vorgestellt werden, nämlich Spatial Hypertext.

„Spatial Hypertext" bezeichnet Systeme, die die Verbindungen zwischen und Bedeutung von „Knoten" (Lexias, Seiten, Texten) unter anderem durch räumliche Anordnung kommunzieren, bzw. diese in einem „Raum" anordnen. Frank M. Shipman und Marshall (1999) schreiben zu den historischen Wurzeln:

---

72. oder ein bißchen mehr
73. http://wordpress.org/, abgerufen am 6. Oktober 2008. Die „Docs"-Sektion scheint auf den ersten Blick auf MediaWiki zu basieren, und die „Blog"-Sektion (so ist zu hoffen) auf Wordpress selber.
74. Siehe http://martinfowler.com/bliki/WhatIsaBliki.html: „Some have questioned the term 'bliki', since wikis allow anyone to edit while this is just for me." (abgerufen am 7. Oktober 2008).
75. http://en.wikipedia.org/wiki/Bliki listet (am 7. Oktober 2008) zum Beispiel die Möglichkeiten „bliki", „wikiLog", „wog", „wikiWeblog", „wikiblog" und „bloki" auf.

> „As the use of hypertext systems became more widespread, researchers realized that readers could become confused or lost as they navigated large networks [...]. Systems such as NoteCards [...] addressed this problem by displaying maps of the hypertext's network structure. The success of NoteCards's ‚browser cards' and other hypertext maps gave rise to systems in which *the user's main interaction with the hypertext was through a network map* rather than a document viewer. " (Hervorhebung hinzugefügt)

Der Gedanke Hypertext durch die zweidimensionale Darstellung eines gerichteten Graphens darzustellen liegt vielleicht nahe, speziell wenn Menschen mit mathematisch/informatischem Hintergrund das Konzept jemandem zum ersten Mal erklären wollen[76]. Viele Einführungen in Hypertext benutzen solche Darstellungen[77].

Diese Abstraktion wurde von einigen Programmen benutzt, um die Struktur eines Hypertext sichtbar zu machen, bzw. später von manchen auch als eine der primären Schnittstellen um den Hypertext zu bearbeiten.

Overview-Maps stellen typischerweise die einzelnen Fragmente eines Hypertexts (genannt Seiten, Lexia, Knoten, ...) als beschriftete Rechtecke oder Icons dar, und Links zwischen den Fragmenten als Linien. Abbildung 2.12 zeigt ein Beispiel.

Die ProponentInnen versprechen sich von Spatial Hypertext zunächst viele Vorteile: So würde ein solches System den BenutzerInnen erlauben räumliches Erinnerungsvermögen zu benutzen um Dokumente wiederzufinden und Muster zu erkennen, sowie Mehrdeutigkeiten und Ungewissheit leicht zu formulieren (Frank M. Shipman und Marshall 1999). Weiters weißt Kolb (2001) darauf hin, dass bespielsweise n-äre Beziehungen leichter durch räumliche Anordnungen, als durch beispielsweise beschriftete Links dargestellt werden können:

> „It would seem that linguistically labeled typed links and paths-patterns could carry more kinds of connections than spatial arrangements, if link types could be conveniently indicated. This may be so, but spatial arrangements can accommodate n-ary relations and gradations of meaning relations that are difficult to put into link type labels."

Eine spannende Perspektive ergab sich nämlich in diesen Versuchen: Marshall und Rogers (1992) beobachteten, dass die BenutzerInnen weniger als erwartet Links verwendeten um Beziehungen zwischen

---

76. Heute könnte eine Einführung in Hypertext natürlich auch daraus bestehen das WWW beziehungsweise einen Link zu demonstrieren.
77. Siehe beispielsweise (Nelson 1987, z.B. 0/2), (Nielsen 1990, Seite 1). Die Diagramme von Landow (2006, Seite 16f) um Link-Typen zu illustrieren weisen ähnliche Eigenschaften auf.

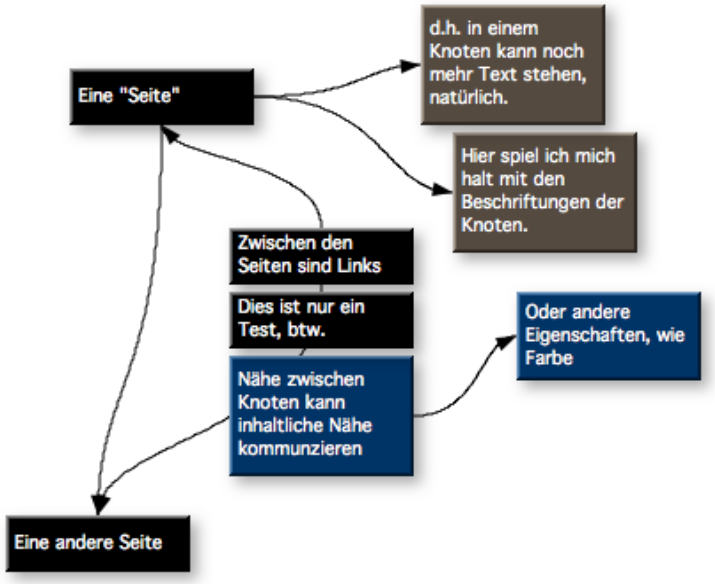


Abbildung 2.12: Ein mit Tinderbox erstelltes Beispiel für eine Overview-Map.

Objekten zu beschreiben, sondern stattdessen räumliche Anordnungen dafür verwendeten. Frank M. Shipman und Marshall (1999) schreiben später in Bezugnahme auf dieses Paper:

> „Users avoided the explicit linking mechanisms in favor of the more implicit expression of relationships through spatial proximity and visual attributes“

Es sei aber angemerkt, dass es sich in diesem konkreten Fall scheinbar um ein recht expressives System typisierter Links handelte, und somit die Möglichkeit besteht, dass die BenutzerInnen auch nur diese konkrete Art von Links nicht so annahmen wie erwartet.

Einige Arbeiten widmeten sich auch dem Versuch diese informell angedeuten Zusammenhänge maschinell zu interpretieren[78]. Dies ist allerdings nicht Thema dieser Arbeit.

Jedenfalls eröffnet dies eine weitere Perspektive auf die Möglichkeit eines Hypertextes ohne Links, wie dies etwa Landow (2006, Seite 20) anspricht (siehe auch Kapitel 2.1.3).

Overview-Maps im WWW

Die Overview Maps, Hilfsmittel früherer Hypertexte, sind jedenfalls im WWW erstaunlich wenig präsent. Nielsen (2005) schlägt vor, dass es Zeit sein könnte Overview-Maps für das Web zu implementieren:

> „Since the first hypertext systems in the 1960s, many more features have been invented and several turned

78. Beispielsweise in Frank M. Shipman, Marshall und Moran (1995)

> out to be useful. Maybe it's time to implement some of these features for the Web. [...]
> In several studies of pre-Web hypertexts, having an overview map of the information space improved users' performance between 12% and 41%. Knowing where you are, where you've been, and where you can go is a significant help in navigating online information.
> *Navigation menus* and *site maps* are two common approximations of overview maps, but neither provides the full set of features that users need. "

Die mangelnde Präsenz von Overview-Maps könnte natürlich daran liegen, dass mit zunehmender Erfahrung mit dem Medium diese Hilfsmittel obsolet wurden, oder dass die Methode sich (in der Praxis) generell nicht bewährt hat, oder überflüssig war. Eine andere Erklärung wäre, dass die Technik des WWW noch nicht die geeigneten Hilfsmittel bietet (oder bot), um Overview Maps praktisch zu ermöglichen. Auch andere in frühen Hypertext-Systemen angedachten Techniken konnten sich erst in jüngerer Zeit in der Praxis verbreiten[79].

Die technische Machbarkeit von Overview Maps in Webbrowsern wird später in Kapitel 3 dieser Arbeit noch ein größeres Thema sein.

79. Die Versionierung aller Inhalte beispielsweise ist zentraler Bestandteil von Nelsons Xanadu Entwurf (Nelson 1987), hat aber erst mit dem Aufkommen von Wikis größere (öffentliche) Verbreitung gefunden.

Kapitel 3

Versuche und Entwürfe

Um das Potential für ein kombiniertes Weblog-/Wiki-System auszuloten wurden eine Reihe von Entwürfen und Prototypen erstellt. Diese sollen im Folgenden vorgestellt und besprochen werden – in etwa in der zeitlichen Abfolge in der sie entstanden sind, da sie sowohl in praktischer Hinsicht und auch in ihren Intentionen oft aufeinander aufbauen.

3.1 Erste Entwürfe

Zuerst wurde die Frage gestellt wie und in welcher Form sich Overview-Maps in und mit Weblogs und Wikis integrieren lassen. Beispielsweise: Welche (visuelle) Gestalt und Aufbau hat ein „Weblog“ auf einer solchen fiktiven Overview-Map?

Infinite Canvas statt Verschachtelten Räumen

Geleitet waren die Entwürfe außerdem von einigen Voraussetzungen bzw. Design-Entscheidungen. Die Overview-Map sollte nach dem Prinzip eines Infinite Canvas funktionieren (also potentiell unbeschränkten Platz bieten). Damit sollte vermieden werden Hierarchien von Map-Spaces zu erzeugen. Inhalte auf der Map sollten direkt bearbeitbar (insbesondere verschiebbar) sein, und auf der Overview-Map sollte ihre Verlinkung untereinander durch Linien dargestellt werden.

Eine Annahme war, dass die BenutzerInnen / AutorInnen die Repräsentationen des Hypertexts auf der Overview-Map, also ihr Aussehen, gestalten wollen und dass das Aussehen der Map als wichtiges Merkmal gesehen wird, basierend auf den Ergebnissen von Pohl und Purgathofer (2004), die Gruppen von Studierenden mithilfe eines Authoring Tools, dass Overview-Maps beinhaltete, Hypertexte entwerfen ließen:

> „[..] shows, on the other hand, that the visual appearance of their overview maps is very important for the students. A relevant part of their activities is made up of moving nodes around on the overview map (the links automatically move with the nodes).“ (ebdt., Seite 95)

Die Idee war also, dass die AutorInnen die Objekte auf der Overview-Map so weit wie möglich selber platzieren und anordnen.

Weblog-Posts waren eine mögliche Ausnahme von dieser Regel. Das Erstellen eines neuen Eintrags in einem Weblog sollte so einfach wie möglich sein. Daher wardie Idee zu vermeiden, dass es notwendig und/oder aufwendig ist einen Eintrag auf der Overview-Map zu positionieren.

Weblogs auf Overview-Maps

Die Idee Weblog-Einträge auf der Overview-Map überhaupt nicht darzustellen[1] wurde zunächst verworfen als klar wurde, dass die Einträge eines Weblogs durchaus untereinander verlinken[2], und dass es interessant sein könnte diese Information graphisch darzustellen. In jedem Fall sind konkrete Weblog-Einträge oft auch das Ziel externer Links – daher auch Permalinks und Linkbacks – bzw. mit Metainformation[3] versehene Entitäten, im Unterschied etwa zu einem einfachen Absatz in einem Text. So scheint es plausibel, dass sie auf einer Overview-Map in irgendeiner Form sichtbar sein sollten.

Ein Entwurf aus dieser Phase (der später verworfen wurde) sah vor, dass Weblog-Posts (i.e. Einzeleinträge) spiralformig um einen zentralen Knoten angeordnet werden. Abbildung 3.1 zeigt eine dieser frühen Skizzen. Die „Gestalt“ eines Weblogs wäre also die einer Spirale. Ein Weblog-Eintrag wäre ein „Node“ auf der Map, der aber nicht frei-positionierbar ist. Hingegen wären Größe und Dichte der Spirale unter Umständen von den BenutzerInnen veränderbar. Einträge würden, wenn sie älter werden[4], in Richtung Mitte der Spirale wandern, wo die Spirale dichter und die Knoten kleiner würden. Die Mitte würde eine Repräsentation eines „Archiv“ bilden. Einzelbeiträge wären in dieser Ansicht nicht mehr gut zu unterscheiden und wären schwierig (oder gar nicht) auszuwählen, aber ihre Anzahl wäre ungefähr abschätzbar. Für den Zugriff auf diese Beiträge würde es einen zusätzlichen Mechanismus brauchen.

Weblog-Spiralen

Abbildung 3.2 zeigt vier Versuche einer solchen Visualisierung die später unternommen wurde. Hierfür wurde der freedom-to-tinker-Weblog[5] auf seine interne Verlinkung untersucht.

Auch wenn diese Darstellung später verworfen wurde, soll hier kurz der Weg, der zu diesem Entwurf geführt hat, besprochen werden um zu verdeutlichen welche Probleme, Constraints und Anforderungen eingeflossen sind.

Verworfen aber lehrreich

Wie erwähnt wurde es für notwendig erachtet, dass Weblog-Einträge vom System positioniert werden, damit der Workflow von „Weblog-

1. In diesem Fall würde nur der Weblog als ganzes auf der Overview-Map dargestellt werden
2. überlicherweise wenn sich der Autor oder die Autorin auf einen früher geschriebenen Eintrag bezieht.
3. Datum, AutorIn, Tags und Kategorien beispielsweise
4. eigentlich: wenn neue Einträge geschrieben werden
5. http://www.freedom-to-tinker.com (zuletzt abgerufen am 17.9.2008). Die Wahl viel auf diesen Weblog unter anderem deshalb weil er eine relativ hohe Zahl ein Einträgen aufweist (>1000).

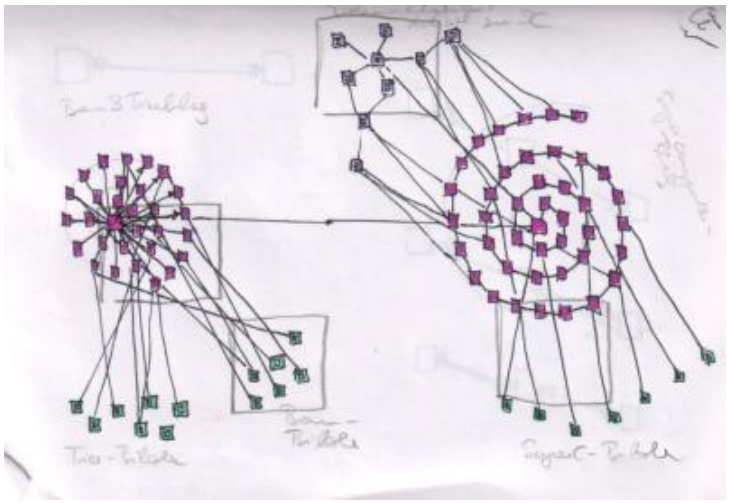


Abbildung 3.1: Frühe Skizze für spiral-förmige Weblogs auf einer Overview-Map. Die zwei Weblog-Spiralen sind durch Links mit Clustern von Knoten verbunden. In der rechten Spirale sind die chronologisch aufeinanderfolgende Einzel-Einträge durch Links verbunden, während in der linken Spirale ein zentraler Knoten auf die Einzel-Einträge linkt.

Eintrag erstellen“ so einfach wie möglich ist. Also in erster Näherung:

- Auf eine Weblog-Seite gehen.
- „Neuer Eintrag“ klicken
- Text schreiben
- einen „Absenden“-Button klicken

Daher muss entweder das System die Position des neuen Eintrags auf der Overview-Map zumindest vorläufig zuweisen, oder aber Einträge werden nicht dargestellt (siehe oben).[6]

Wie erwähnt entsteht Bedeutung in einen „Spatial Hypertext“-System auch durch die räumliche Beziehung der Objekte untereinander. Wenn Einträge nachträglich automatisch verschoben werden ist dies daher ein Eingriff in diese Bedeutung. Wenn sie schon vom System verschoben werden müssen, sollte daher wenigstens die Bewegung im Raum gering gehalten werden.

Ein Teilproblem ist daher beispielsweise, dass Formen der Weblog-Darstellung, die im Vergleich zu den anderen Map-Inhalten größer werden, eventuell in benachbarte Bereiche mit bestehenden Inhalten

6. Eine dritte Variante wäre es Einträge zunächst nicht darzustellen, aber später von der Benutzerin oder dem Benutzer positionieren lassen. Dies würde jedoch dazu führen, dass manche Einträge auf der Map dargestellt sind und andere nicht, und erfordert daher vermutlich ein mentales Modell, in dem die Einträge in einen zusätzlichen „Warte-Raum“ sind

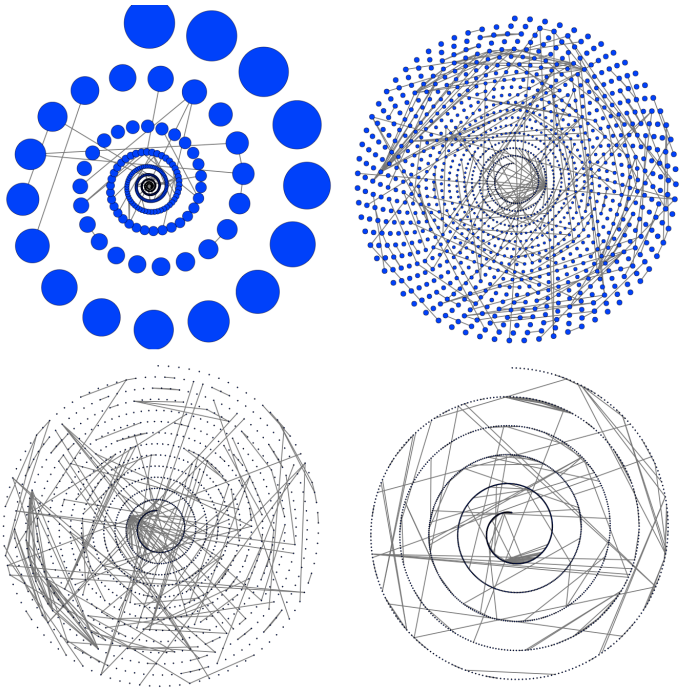


Abbildung 3.2: Vier Varianten einer Blog-Spirale. Als Datenmaterial diente der Weblog „freedom-to-tinker“. Wie vielleicht erkennbar ist, gibt es viele Parameter eine solche Spirale zu zeichnen. Anstatt a-priori ein für alle Zwecke ideales Parameter-Set zu suchen wäre es vielleicht sinnvoll, dass die BenutzerInnen dieses manipulieren und ihren Vorlieben anpassen können.

wachsen würden. Dann wären Umordnungen auf der Gesamt-Map erforderlich (oder ein Durcheinander die Folge).

Weiters sollte die Zugehörigkeit eines Eintrags zu einem Weblog kommunziert werden. Der Weblog braucht daher in irgendeiner Form eine Gestalt, die ihn als Weblog kennzeichnet. Gleichzeitig könnte diese Gestalt oder Teile von ihr, auch eine Anlaufstelle sein, an der die BenutzerInnen mit dem Gesamt-Weblog interagieren können. Also beispielsweise neue Weblog-Einträge (von der Overview-Map aus) anlegen können.

Aus all dem folgt also die Forderung, dass die Form eines Weblogs kompakt ist und kompakt bleibt, wenn neue Einträge hinzukommen.

Ein Weblog bewirbt prototypisch vor allem die neuesten Einträge. Daher sollten diese auch auf einer Overview-Map prominenter dargestellt werden.

Die besprochene Form einer begrenzten Spirale, die nach innen hin dichter wird, würde viele dieser Eigenschaften erfüllen. Sie würde

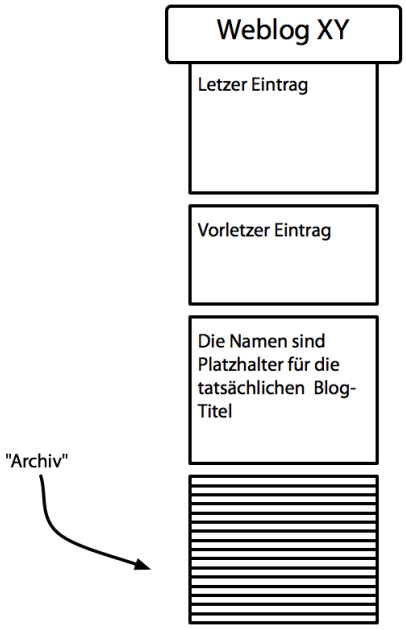


Abbildung 3.3: Skizze der finalen Darstellung eines Weblogs auf der Overview-Map. Das Aussehen auf der Overview-Map ist dabei dem typsichen Aussehen eines Weblogs im Browser nachempfunden.

nur einen beschränken, definierten und kompakten Raum ausfüllen. Die letzen Einträge wären sichtbarer. Durch die Spiral-Form und das Schrumpfen der inneren Knoten würde sie sich klar von anderen Inhalten unterscheiden.

Letztlich wurde jedoch eine andere Form favorisiert (Siehe Abbildung 3.3). Diese bietet vermutlich den Vorteil einer schnelleren Wiedererkennbarkeit für BenutzerInnen, die mit Weblogs vertraut sind.

Diese Form hält sich eng an das vorhandene und gewohnte „Aussehen“ von Weblogs, also die graphische Form die ein Weblog im Browser üblicherweise besitzt. Damit wäre vermutlich ein Wiedererkennen ihrer Funktion gegeben. Das Problem des Platzbedarf löst diese Form in dem Einträge in ein „Archiv“ wandern, in dem sie nur wenig Platz einnehmen.

Mini-Weblogs

Ein Anforderung, die diese Form *nicht* erfüllt, ist es die Intra-Weblog-Links durch Linien darstellen zu können, denn diese würden (so sie überhaupt gezeichnet werden) immer verdeckt werden. Als möglicher Ersatz wurde angedacht, dass die von und zu einem Eintrag linkenden Einträge hervorgehoben werden, wenn sie ausgewählt werden oder/und der Mauszeiger über ihnen ist – ähnlich den „Blog continuum sparklines“ in Wroblewski und Wood (2005), allerdings weniger statisch.

Ein anderes Thema war der Zusammenhang zwischen Overview-

Map und den restlichen Elementen des Systems.

Eine Idee, die in diesen Entwürfen entstanden ist, sollte im weiteren Verlauf der Experimente immer wieder auftauchen: Eine „Mini-Map“, eine verkleinerte und kleinere Darstellung der Overview-Map (siehe Abbildung 3.4). Durch die Präsenz der Overview-Map auf den einzelnen Seiten des Weblogs/Wikis könnte eine solche Mini-Map helfen, diese Seiten in Bezug auf die Overview-Map zu verorten.

Mini-Maps

Auf der Mini-Map wäre der aktuell betrachtete Knoten zunächst im Mittelpunkt, aber sie wäre mehr als nur eine Darstellung der „Umgebung“. Ein Klick auf einen Knoten würde zu den zugehörigen Inhalten führen. Hier ergäbe sich dann der Effekt, dass Seiten, die nahe zur aktuell betrachteten Seite sind, einfach und bequem angewählt werden können. Sie wären implizit verbunden, gewissermaßen „verlinkt“, ohne dass notwendigerweise ein (Text)-Link bestehen müsste[7].

Weiters wäre ein Zusammenspiel von Mini-Map und Text denkbar. Beispielweise könnte die Overview-Map beim „hovern“ über einem Text-Link die Map-Repräsentation des Link-Zieles hervorheben, oder, falls das Ziel nicht sichtbar ist, die „Richtung“ in der dieses liegt anzeigen, beispielsweise durch Hervorheben der entsprechenden Link-Linie.

Am Schluss dieser ersten Design- und Brainstorming-Phase wurde klar, dass große Unklarheit über die technische Realisierbarkeit besteht. Einerseits war unbekannt ob Möglichkeiten existierten diese Entwürfe auch nur ansatzweise umzusetzen und ob - falls diese vorhanden sind - diese Technologien angemessen skalieren und funktionieren würden[8].

Umgekehrt war es denkbar, dass sich Annahmen über technologische Einschränkungen der aktuellen Web-Technologien in die Entwürfe eingeschlichen hatten, die so nicht stimmten. In der Design-Phase wurde versucht zunächst technische Einschränkungen außer Acht zu lassen, aber natürlich finden solche Entwürfe nie in einem Kontext-Vakuum statt.

Also war es Ziel der nächsten Phasen die technischen Möglichkeiten zu testen.

3.2 Erste Overview-Map

Das Ziel dieser ersten Prototypen war es zunächst zu untersuchen inwieweit die bisherigen Vorstellungen von Overview-Maps mit da-

7. Je nachdem wo die Grenzen des aktuellen Dokuments gesehen werden, d.h. ob die von der Overview-Map zur Verfügung gestellte Navigation zum Dokument gehört ist der Link explizit oder implizit.

8. Diese Entwürfe und Experimente wurden im Winter 2005 durchgeführt. Zu dieser Zeit war beispielsweise gerade Firefox 1.5 fertiggestellt und veröffentlicht worden.

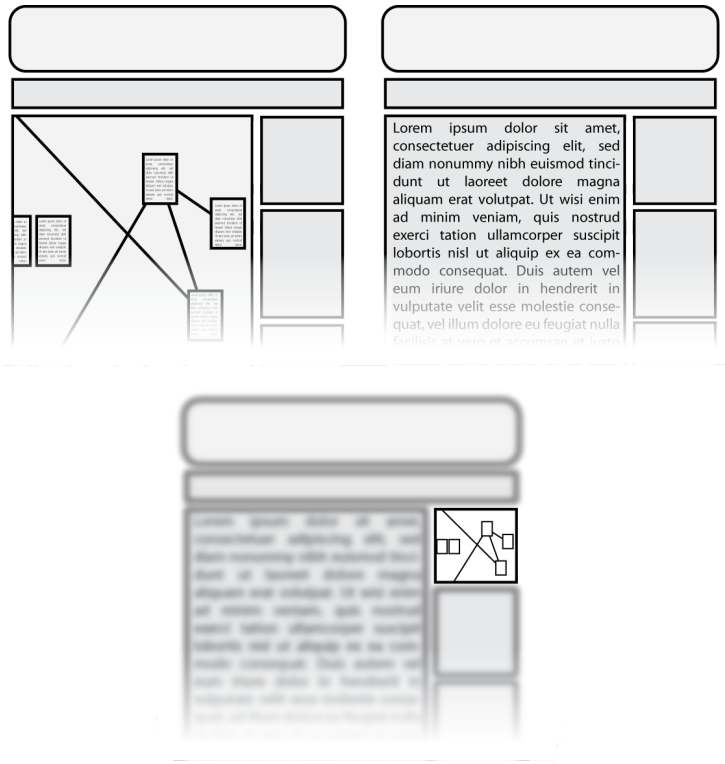


Abbildung 3.4: Skizze einer Mini-Map. In den oberen Bildern zeigt die Webseite die Overview-Map bzw. den Inhalt eines Knotens an. In einer Ansicht mit Mini-Map können beide Blicke auf den Inhalt verbunden werden.

maligen Web-Technologien realisierbar wären – beziehungsweise eine Vorstellung von der Größenordnung des Problems zu gewinnen.

Erste Untersuchung der technischen Möglichkeiten

Eine andere Frage, die im Zuge dieser Versuche auftauchte, war inwieweit von dem Prinzip abgegangen werden kann, dass Webseiten im Browser bearbeiten bedeutet Markup beziehungsweise HTML-Code zu bearbeiten. Also inwieweit das Bearbeiten in Richtung WYSIWYG gehen kann. Auch bezüglich dieser Frage sollten Technologien und Methoden untersucht und getestet werden.

Bezüglich der Overview-Map erschien als primäres Problem die Darstellung der „Link-Linien“, also der Linien, die Knoten verbinden und Links darstellen sollen. Das Problem besteht hier hauptsächlich darin, dass HTML nicht direkt diagonale Linien darstellen kann – und es gleichzeitig wünschenswert wäre diese Linien schnell und im Brower upzudaten.

Aufgrund des Wunsches die Overview-Map direkt bearbeiten zu können, schien die Variante mit vorgenerierten Karten zu arbeiten

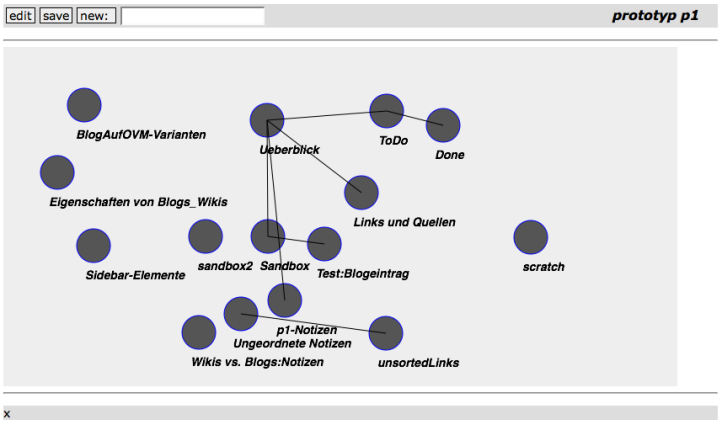


Abbildung 3.5: Der erste im Zuge dieser Arbeit erstellte Prototyp einer Overview-Map. Kreise und Links wurden mit SVG dargestellt.

suboptimal, da dies Möglichkeiten die Overview-Maps direkt und bequem im Browser zu bearbeiten einzuschränken schien[9].Also die Herausforderung: Wenn ich ein Objekt auf der Overview-Map verschiebe sollen Links sofort die neuen Positionen übernehmen.

Die „JavaScript Vector Graphics Library“ von Walter Zorn – die die notwendigen Formen mittels vieler positionierter HTML-div-Elemente zeichnet – wurde in Erwägung gezogen, schied allerdings rasch aus Performance-Gründen aus[10].

Mit dem Erscheinen von Firefox 1.5 gab es zumindest einen Browser, der auf allen großen Plattformen[11] SVG prinzipiell darstellen konnte. SVG ist der vom World Wide Web Consortium entworfene Standard für Vektorgraphik im WWW.[12] SVG in Firefox schien also eine logische Plattform für erste Versuche. Abbildung 3.5 zeigt einen Screenshot eines dieser Prototypen.

In diesem Prototyp wurden Seiten durch Kreise dargestellt, deren Position schon auf der Overview-Map direkt manipuliert werden konnte. Die Link-Linien passten sich sofort an. Die Overview-Map konnte bearbeitet, aber nicht bewegt oder gezoomt werden, war also kein „Infinite Canvas“.

---

9. Bequeme Navigation in Kartenmaterial in Form von Bildern war zu diesem Zeitpunkt durch Google Maps (http://maps.google.com) zumindest grundsätzlich demonstriert.
10. http://www.walterzorn.com/jsgraphics/jsgraphics_e.htm (abgerufen zuletzt am 8. Mai 2008). Dort steht bezüglich Performance: „Making a browser create DIV elements is of course much slower than just colouring pixels [...]. Even this JavaScript Vectorgraphics Library can’t escape from this fundamental restriction - it just tries to squeeze out the maximum of what’s possible. It’s recommended to refrain from creating shapes that extend more than 600 - 1000 pixels in both dimensions.“
11. Soll hier Windows, OS X, Linux bedeuten.
12. siehe http://www.w3.org/Graphics/SVG/ (abgerufen am 8. Mai 2008)

Bezüglich des Bearbeitens von Seiten wurde zunächst versucht, das Aussehen des Markup-Bearbeitungs-Modus dem Aussehen der gerenderten Seite so weit wie möglich anzupassen, also vor allem Ausdehnung und Aussehen des Form-Elements, Art und Größe der verwendeten Fonts und Art des Markups. Später wurde mit einem WYSIWYG-Editor auf Basis der Firefox-Midas-APIs[13] experimentiert.

Diese ersten Versuche zeigten, dass ein WYSIWYG-artiger Editor subjektiv höchst erstrebenswert schien, und Overview-Maps zunächst prinzipiell machbar erschienen.

Sie zeigten auch, dass Schwierigkeiten im Detail liegen, und der Aufwand für derartige Zugänge schwer abschätzbar war. Beispielweise war es notwendig dafür zu sorgen, dass im „Edit-Mode“ der Overview-Map die Kreise ihre Link-Funktion verlieren, während sie ansonsten auf die jeweils repräsentierte Seite verlinkten. Es zeigte sich auch, dass diese erste Generation von SVG-Unterstützung in Firefox noch vielerlei Bugs und Eigenheiten hatte. Einige Features von SVG, deren Einsatz spannend gewesen wäre, standen unter Firefox 1.5 noch nicht zur Verfügung.

Ähnlich beim Editor: Obwohl prinzipiell als großer Fortschritt im Vergleich zu Markup-Edit empfunden, tauchte hier zum ersten Mal ein Problem auf, dass unter anderem auch in den Kapiteln 2.1.4, 3.3 und 3.9 behandelt wird.

Das Aussehen während des Bearbeitens wurde dadurch gestört, dass das Prinzip der Midas-API – nämlich einen ganzen iFrame bearbeitbar zu machen – es erfordert die Größe des Bereichs in dem der Text bearbeitet wird dem Browser explizit mitzuteilen. Während der Text länger wurde kam ein Punkt, an dem er länger wurde als der iFrame-Bereich. Dann beanspruchten Scrollbars innerhalb des iFrames Platz. Somit stimmten Zeilenumbrüche und Layout im Vergleich zur späteren Anzeige nicht mehr überein.

3.3 WYSIWYG mit Midas (Midas-Resize)

Siehe auch Kapitel 2.1.4

Der nächste Prototyp behandelte explizit nur das Thema WYSIWYG-Bearbeiten. Ziel war es zu versuchen einen Midas-iFrame dynamisch an die richtige Größe anzupassen. Damit würden einerseits keine Scrollbalken das Layout verfälschen. Andererseits würde damit Content, der unter dem zu bearbeitenden Bereich (beispielsweise Kommentare) angezeigt werden soll schon während des Bearbeitens korrekt platziert sein.

Das technische Problem war hier vor allem, dass mit dem Aktivieren des Midas-Modus für einen (i-)Frame, Javascript in diesem Frame deaktiviert wird. Die Suche nach Beispielen ergab jedoch zu-

13. siehe http://www.mozilla.org/editor/midas-spec.htm (abgerufen am 19.4.2008)

mindest eine Anwendung, die sofort auf Änderungen in diesem Frame reagiert[14]. Reverse-Engineering dieses Codes ergab den Trick die Event-Handler aus dem, den iFrame umgebenden, Dokument zu setzen. Auf diese Weise konnte ein bearbeitbarer iFrame, der sich dynamisch in der Größe anpasst, realisiert werden, zumindest in Firefox. Bei dem Versuch dieses Prinzip auch für Safari umzusetzen gab es allerdings Probleme: Die vom Browser zur Verfügung gestellte Werte über die aktuelle Höhe des Textes hatten eine drastisch andere Bedeutung.

Es wurde klar, dass der Versuch ein solches System Browserübergreifend zu gestalten eventuell recht aufwendig sein könnte.

Aus heutiger Sicht sind diese Verfahren vermutlich nahezu obsolet. Aktuelle Versionen von Safari, Opera, Firefox und Internet Explorer unterstützen den „contentEditable"-Edit-Mode für HTML-Elemente.

Kapitel 3.9 behandelt einen Prototypen der mit dieser technologischen Basis, also dem contentEditable-Attribut, arbeitet.

3.4 Overview-Map Prototypen - Untersuchung der Machbarkeit

Die erste erstellte Overview-Map (Kapitel 3.2) war noch recht einfach. Im Gegensatz dazu sollte eine „echte" Overview-Map (gemäß den Entwürfen) zumindest eine unbeschränkte Fläche zur Verfügung stellen, der Ausschnitt der Overview-Map der gerade betrachtet wird sollte verschiebbar und vermutlich zoombar sein.

Infinite Canvas

Ziel war es also eine solche Overview-Map zu konstruieren. Im Idealfall sollte sie demonstrieren, dass eine benutzbare Overview-Map im Bereich des technisch möglichen liegt.

Einschränkungen wurden im Hinblick auf Cross-Browser-Unterstützung getroffen: Zielplattform war hauptsächlich Firefox. Die Einschränkung passierte einerseits, weil nicht klar war, dass Internet Explorer (zu dieser Zeit seit mehr als vier Jahren auf Version 6) überhaupt solche Entwürfe unterstützen könnte – bzw. wann, wie und ob er das je würde. Nachdem Internet Explorer ausschied hätte also der Versuch mehr als einen Browser zu „unterstützen" mehr Aufwand bedeutet, aber nicht bewirken können, dass die erstellten Prototypen wirklich universell verwendbar wären, bzw. diese Verwendbarkeit demonstriert worden wäre[15].

Firefox hatte neben der Unterstützung für mehrere Betriebssysteme den Vorteil, dass relativ gute Dokumentation und Entwicklungs-Werkzeuge frei verfügbar waren.

---

14. http://codeeditor.mozdev.org/v0.1a/ (zuletzt getestet am 20.4.2008)
15. Mir ist keine Untersuchung bekannt die für diesen Zeitpunkt (bzw. auch spätere Zeitpunkte) dem Internet Explorer nicht einen (global gesehen) dominanten „Marktanteil" ausweist.

3.4.1 Seiten-Thumbnails

Thumbnails von Seiten

Im Laufe der bisherigen Versuche war die Idee entstanden Seiten[16] nicht nur durch beschriftete Rechtecke, sondern auch mittels verkleinerter Screenshots ihres Inhalts darzustellen (im weiteren „Thumbnails“ genannt). Wenn dies vernünftig möglich wäre sollte dies in die Versuche miteinfließen, daher wurde bald eine Möglichkeit gesucht, diese Thumbnails zu erstellen.

Leider wurde damals keine leichtgewichtige Technologie gefunden, die dies ermöglicht hätte. Es wurde eine relativ komplexe (und leider recht instabile) Methode entwickelt, um die Thumbnails halbautomatisch zu erzeugen. Für eine Anwendung außerhalb experimenteller Prototypen war sie vermutlich unbrauchbar. Doch sie lieferte die benötigten Thumbnails, und so konnte zumindest ein Eindruck von dem Effekt ihres Einsatzes gewonnen werden.

Die Methode, die Beispiel-Thumbnails zu erstellen war wie folgt:

- Es wurde in einem virtuellen X-Server (Xvfb) Firefox mit einer vorbestimmten Breite und einer Größe, die die maximale Größe einer Seite übertreffen sollte, gestartet. Ebenfalls mitgegeben wurde dem Browser die Anweisung eine spezielle Version der Seite zu laden.
- Diese Version wurde vom Server speziell für diesen Zweck generiert und unterschied sich von der normalen Darstellung dadurch, dass sie sämtliche Navigations- und Nicht-Inhalts-Elemente wegließ sowie durch Javascript-Anweisungen, die den Browser noch einmal auf die richtige Breite und eine Größe skalierte, die sich aus der Größe des Inhalts und der Größe der Navigationselemente im Browser ergab.[17]
- An dieser Stelle wurde eine fixe Zeit gewartet um dem Browser Zeit zu geben die Seite zu laden, zu interpretieren und anzuzeigen. Danach wurde ein Screenshot des Browserfensters erstellt.
- Schlussendlich wurden aus dem Screenshot die Browser-Elemente herausgeschnitten und das Ergebnisse in verschiedene Auflösungen konvertiert. Der virtuelle X-Server und die Firefox-Instanz wurden wieder beendet.

Wie sich der Leser oder die Leserin an diesem Punkt vielleicht denken wird, war dieser Prozess eine eher fragile und vor allem sehr aufwendige Angelegenheit. Optimierungen wurden angedacht, beispielsweise ein Wiederverwenden der Browser-Instanzen. Eine Rückmeldung durch den Browser, dass das Laden und Rendern der Seite fertig war hätte die Lösung zuverlässiger gemacht. Doch diese Ansätze brachten ihre eigenen Probleme mit sich.

---

16. hier in der Bedeutung von „Etwas, das als Knoten auf der Map repräsentiert wird“.
17. Würde der Inhalt nicht in das Browserfenster passen, würde die Scroll-Leiste des Browsers Platz einnehmen, und die Inhalts-Größe anders zurückgeben werden. Daher die Vorgehensweise der zweimaligen Größenanpassung.

Der sauberste Ansatz erschien damals sich direkt der Rendering-Engine eines Browsers zu bedienen (ohne den Gesamtbrowser).

Schlussendlich wurde jedoch beschlossen, dass die vorliegende Lösung gut genug war, um mit den erstellten Thumbnails an anderen Problemen weiterzuarbeiten.

Die Erstellung dieser Thumbnails ist heute kein unüberwindbares Hindernis mehr. Eine kurze Suche liefert heute einige Programme die automatisch Screenshots von Webseiten erzeugen. Manche davon sind vielleicht dazu geeignet, von einem Webserver gestartet und benutzt zu werden. Vom Autor getestet wurde beispielsweise „webkit2png“[18]. Laut dieser Seite existierte dieses Tool zum Zeitpunkt dieser Versuche schon. Leider wurde es aber damals nicht gefunden. Allerdings kann auch dieses Programm (in dieser Form) noch nicht direkt von einem Web-Server gestartet zu werden da es immer noch einen Window Server benötigt.

In Kapitel 3.9 wird ein Prototyp vorgestellt der das erwähnte Programm einsetzt, um Thumbnails von Seiten zu erzeugen.

3.4.2 SVG-Map

Bezüglich der Overview-Map wurde als nächstes versucht, einen „Infinite Canvas“ auf SVG-Basis zu implementieren. Das funktionierte im Prinzip. Eine SVG-Graphik konnte mit der Maus verschoben, verkleinert und vergrößert werden. Es zeigte sich aber schnell, dass Performance ein Problem sein könnte. Die gefühlte Geschwindigkeit schien nicht nur sehr stark von der Größe der mit SVG gefüllten Fläche abzuhängen sondern auch von der Anzahl der Elemente. Eine browser-füllende Overview-Map mit genügend Elementen um alle Seiten eines Wikis oder eines Weblogs repräsentieren zu können erschien damit eher unplausibel.

3.4.3 Canvas-Maps

Daher wurde als nächstes eine von Apple in Webkit[19] bzw Safari eingeführte Webtechnologie getestet: Das Canvas-Tag[20], das im Gegensatz zu SVG nicht die Graphik beschreibt sondern erlaubt diese (prozedural) zu zeichnen.

Das Canvas-Tag wurde auch von Firefox unterstützt und somit bestand die Chance, dass darauf aufbauende Prototypen sowohl in Firefox als auch in Safari funktionieren würden. Sie wurden daher mit einem Blick auf beide Browser geschrieben.

---

18. (http://www.paulhammond.org/webkit2png/, abgerufen am 28.04.2008)
19. http://webkit.org
20. Das Canvas-Tag (oder auch „canvas-element“) ist momentan Teil des Entwurfes für HTML 5. Siehe http://www.w3.org/TR/html5/ (abgerufen am 17. September 2008).

Abbildung 3.6: Screenshots von drei verschiedenen Entwicklungsstufen der rein canvas-basierten Overview-Map-Prototypen in Safari (in chronologischer Ordnung)

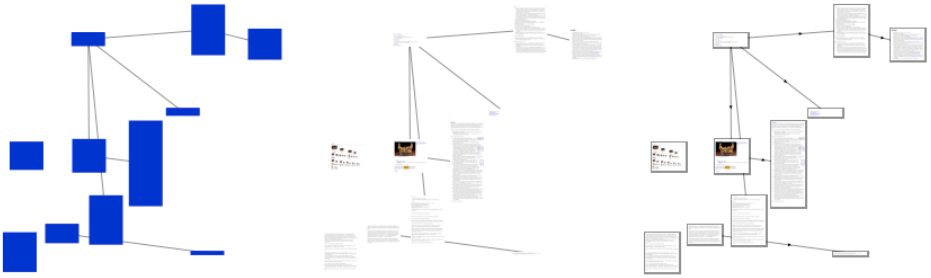


Abbildung 3.7: Äquivalente Screenshots zu Abbildung 3.6, diesmal in Firefox.

Die Abbildungen 3.6 und 3.7 zeigen Screenshots der Overview-Maps dieser Prototypen im Laufe ihrer Evolution, jeweils in Safari und Firefox.

In diesen Prototypen konnte die Overview-Map bewegt, verkleinert und vergrößert werden[21].

Evaluation der rein canvas-basierten Prototypen zeigte zwei Dinge. Zum einen wurde klar, dass das Canvas-Tag für Firefox zunächst ähnlich schlecht wie SVG funktionierte. Während nämlich Safari auch auf schwachen Rechnern relativ gute Performance lieferte war die Darstellung in Firefox (insbesondere unter Linux, aber in geringerem Ausmaß auch unter Windows und OS X) so langsam, dass die Interaktionen mit der Overview-Map nicht mehr wie gewünscht funktionierten. Das heißt, dass sich das „Ziehen“ der Karte nicht mehr wie Ziehen „anfühlte“, da ein Refresh der Karte mehrere Sekunden dauerte.

Informelle Untersuchungen mit verschiedenen Größen der

21. In Firefox konnte der Zoom auch per Mausrand verändert werden

Overview-Maps[22] ergab, dass die Performance in Firefox sehr stark von dieser Größe, dass heißt, der Anzahl der für das Canvas-Tag verwendeten Pixel abzuhängen schien. Während bespielsweise eine kleine (300x300 Pixel) Overview-Map in Firefox relativ gut zu funktionieren schien, war eine Overview-Map, die das Browser-Fenster füllte, im großen und ganzen unbrauchbar.

Anders in Safari, hier suggerierten die (leider nur dort direkt unterstützten) Drop-Shadows und die Flüssigkeit der Bewegung sogar eine gewisse „Professionalität“ des Prototypens.

3.4.4 „Mixmaps“

In Anbetracht der Performance-Probleme von Firefox mit den ersten canvas-basierten Prototypen wurde ein weiterer Versuch unternommen einen in diesem Browser verwendbaren Overview-Map-Prototypen zu konstruieren.

Der erste Ansatz für eine Verbesserung der Perfomance sollte sein, die Seiten-Miniaturen/Symbole nicht über SVG und/oder das Canvas-Tag sondern über ganz normale HTML-Elemente und nur die Link-Pfeile über ein Canvas-Tag zu „zeichnen“.

Dahinter steckte die Idee, dass nur die Link-Linien schräge Elemente benötigen, und durch diese Mischung die Canvas-Ebene während Drag-Operationen[23] ausgeblendet bzw. abgeschalten werden könnte. Im schlechtesten Fall würden der Canvas-Ebene in dieser Zeit zumindest nicht neu gezeichnet werden. Zusätzlich wurde gehofft, dass eine versteckte oder entfernte[24] Canvas-Fläche den Browser nicht mehr im beobachteten Ausmaß beinträchtigen würde.

Zu überprüfen war ob die Position und Positionierung der verbleibenden Elemente[25] für eine Benutzerin oder einen Benutzer potentiell genügend Information beinhaltet, um die Overview-Map angenehm manipulieren zu können. Beziehungsweise ob sich das Gefühl einstellen würde, dass in beiden Modi dieselbe Overview-Map manipuliert wird.

Phase 1

Die Implementation passierte dann in grob zwei Phasen. In der ersten wurde ein Prototyp nach dem oben skizzierten Prinzip geschrieben. Dieser Prototyp stellte mehr Interaktionen mit der Overview-Map als die vorigen Prototypen zur Verfügung: Es konnte der Map-Ausschnitt verschoben, verkleinert und vergrößert werden; Knoten auf der Overview-Map konnten verschoben werden und temporäre Links zwischen Knoten gelegt werden[26]. Für die Darstellung der

22. mit Größe ist hier die Anzahl der verwendeten Pixel im Browser gemeint
23. Beispielweise das Verschieben eines Symbols auf der Overview-Map oder das Verschieben des sichtbaren Ausschnittes der Overview-Map.
24. gemeint hier: aus dem DOM entfernt
25. Die einzigen verbleibenden Elemente waren in diesem Fall die Rechtecke die die einzelnen Seiten repräsentieren. Diese „Nodes“ müssen aber nicht die einzigen rechteckigen Elemente auf einer Overview-Map sein.
26. „temporär“ heißt, dass diese Links nur im der aktuellen Browser-Fenster erstell wurden und daher beispielsweise bei einem Reload der Seite verschwanden.

Knoten wurden die Seiten-Thumbnails [27] benutzt

Weiters wurden Funktionen eingebaut diese Overview-Map mit Zufalls-Konfigurationen in verschiedenen Größen zu füllen um die Performance besser einschätzen zu können – Im Normalfall benutzten dieser und die bisherigen Prototypen die gleichen Datenbank wie der Prototyp aus Kapitel 3.2.

Das Ausblenden des Canvas während der Drag-Operationen bewirkte tatsächlich, dass sich auch in Firefox eine gewisse Unmittelbarkeit der Bewegung einstellte. Und die während dieser Operation dargestellten Elemente – also die Knoten-Symbole – reichten meinem Gefühl nach aus um die „Gestalt“ der Overview-Map zu vermitteln.

Allerdings zeigten Tests mit den Zufalls-Datensätzen, dass die Performance für große Maps nicht ausreichend war. Karten mit 1000 Knoten und etwa 500 Links beispielsweise brauchten unter Umständen circa eine Minute um überhaupt angezeigt zu werden. Das Verschieben des Kartenausschnittes war dementsprechend anstrengend. Das Verschieben eines einzelnen Knotens hingegen war dagegen sehr flüssig. Dies legte die Vermutung nahe, dass es an diesem Punkt vor allem die Menge an zu positionierenden Knoten war die den Browser beanspruchten.

Phase 2: Optimierungen

Daher wurde in der zweiten Phase versucht durch Optimierungen möglichst viel an Geschwindigkeit zu gewinnen. Mehrere Ansätze hierfür wurden in Erwägung gezogen:

1. In der bisherigen Implementierung wurde bei einer Verschiebung des Kartenausschnittes jeder „Knoten“ einzeln auf die neue Position (in Bezug auf das Browserfenster) gesetzt. Vielleicht würde es schneller gehen, wenn alle Knoten relativ zu einen Eltern-Element positioniert wären und nur dieses Eltern-Element neu positioniert werden müsste.
2. Der bisherige Code führte Neu-Positionierungen von Knoten (bei Verschieben oder Zoomen des Kartenausschnittes) für alle vorhandenen Knoten durch. Knoten die dabei auf jeden Fall nicht sichtbar wären können hingegen „ignoriert“ werden[28].
3. Änderungen am DOM, also an der Struktur der Webseite, sind im Vergleich zu reinen Rechen-Operationen in Javascript relativ „teuer“. Wenn alle Daten zur Darstellung der Overview-Map in einer Parallel-Datenstruktur verfügbar wären und
   - nur aus dieser Datenstruktur gelesen würde sowie
   - nur bei tatsächlichen Änderungen in den DOM geschrieben würden

   ergäben sich vermutlich auch Geschwindigkeits-Vorteile.

---

27. siehe Kapitel 3.4.1.
28. Die Sache ist in diesem Fall etwas komplexer, weil die Knoten ja nicht jedes mal neu erzeugt wurden, sondern ihre Position umgeschrieben wurde.

Zunächst wurde der vorhandene Code refaktorisiert, um die Basis für diese Experimente zu schaffen. Dann wurden die beschriebenen Optimierungen in den Code eingebaut.

Alle diese Optimierungen brachten spürbare Geschwindigkeitsvorteile. Bei Punkt 2 wurden zwei Varianten getestet: In einer wurden nicht sichtbare Elemente über die CSS-„display"-Eigenschaft „unsichtbar" geschalten, in der anderen wurde die Elemente ganz aus dem DOM entfernt. Die zweite Variante stellte sich dabei als schneller heraus.

Nach Optimierungen: 1000 Knoten-Overview-Maps scheinen möglich

Zusammenfassend lässt sich über die Ergebnisse dieser zweiten Phase also sagen: Mittelgroße Overview-Maps scheinen mit heutigen Mitteln und Rechnern theoretisch machbar zu sein. Der optimierte Prototyp kann (auf heute verfügbaren Rechnern, beziehungsweise Browsern[29]) eine Overview-Map mit 1000 Knoten und etwa 500 Links durchaus flüssig darstellen, wenn der am meisten verbreitete Browser – Microsoft Internet Explorer – außer acht gelassen wird.

Es existieren für den Internet Explorer Ansätze das Canvas-Tag zur Verfügung zu stellen oder zu emulieren. Angesichts des Aufwandes den es vermutlich bedeuten würde den (alten) Code des Prototypens – der über weite Strecken nur auf Firefox bzw. Safari ausgereichtet war – kompatibel zu Internet Explorer zu gestalten wurde jedoch davon abgesehen diese Ansätze zu testen.

3.4.5 Comet-Map

Als nächstes Ziel wurde ins Auge gefasst, alle Änderungen an der Overview-Map sofort allen Browser-Instanzen, in denen die Overview-Map betrachtet wird, zur Verfügung zu stellen. Die Browser sollten die Änderungen auch sofort darstellen.

Dazu zunächst ein paar Worte zur Motivation:

Exkurs zum gleichzeitigen Bearbeiten der Map

Die Overview-Map stellt für das System wie es bisher entworfen wurde ein ziemlich wichtiges und globales Objekt dar. Ihre Verwendung als System-Komponente begründet sich gerade aus der Idee, dass sie gerne und viel bearbeitet werden wird und soll. In einem Wiki-artigem System mit vielen BenutzerInnen muss es daher früher oder später zu einer Situation kommen in der mehrere Menschen gleichzeitig die Overview-Map bearbeiten wollen.

Wenn jetzt Person A aber Änderungen durchführt und Person B auf der Basis der alten Version der Overview-Map ebenfalls Änderungen vornimmt kommt es potentiell zu einem Durcheinander: Die Änderungen von B sind eventuell obsolet oder passen nicht zu den Änderungen die A durchgeführt hat. In einem Fall in dem die beiden nicht darüber informiert werden, dass diese Situation vorliegt und dass der Zustand der Overview-Map, der ihnen präsentiert wird,

29. Es wurden Firefox 3.0 beta 5, Safari 3.1.1 und Opera 9.27 auf OS X 10.5. getestet. Der verwendete Rechner war ein MacBook Pro mit 2.33 GHz Intel Core 2 Duo mit 2 GB RAM.

nicht mit der Realität des Systems übereinstimmt, oder dass sie die Map nicht auf Basis einer „aktuellen“ Version verändert haben.

Das Problem des Schreibens auf gemeinsame Ressourcen ist kein nur hier auftretendes. Mögliche Ansätze sind:

- Das Problem zu ignorieren. Die letzte Schreib-Operation setzt den Inhalt, auch wenn ihre Ausgangsbasis veraltet ist. Ein Ansatz der auch in Wikis grundsätzlich möglich wäre.
- Die Situation vermeiden, dass zwei Personen gleichzeitig Änderungen vornehmen. Beispielsweise in dem die Seite nach der ersten Anfrage sie zu bearbeiten für andere gesperrt wird.
- Die Situation im Nachhinein erkennen und der Person helfen den potentiellen Konflikt zu lösen. Beispielsweise in dem ein Hinweis angezeigt wird und/oder die zwischenzeitlich vorgenommenen Änderungen angezeigt werden. Mediawiki verfolgt beispielsweise diesen Ansatz.

Spezielle Herausforderungen der Overview-Map

Im Fall der Overview-Map ergeben sich jedoch einige Unterschiede.

Erstens lädt die Overview-Map dazu ein kleine Änderungen an Teilen der Overview-Map durchzuführen. Also etwa einen bestimmten Knoten zu verschieben. In dem Interface-Entwurf für die Overview-Map werden solche Änderungen als Einzel-Ereignisse betrachtet und sofort durchgeführt werden[30]. Es gibt also weniger ein explizites „Save“- bzw. „Edit“-Ereigniss, sondern sehr viele kleine, während die Benutzerin oder der Benutzer versucht ein gefälliges Aussehen der Overview-Map zu erzielen.

Zweitens lädt die Overview-Map zum „Spielen“ ein. Also verschiedene Positionierungen auszuprobieren und zu testen. In einen solchen Szenario ist damit zu rechnen, dass die meisten Änderungen gleich wieder überschrieben werden.

Und drittens ist die Overview-Map als ganzes ein globales Objekt. Während sich das Bearbeiten von verschiedenen Einzelseiten oft nicht gegenseitig behindert, ist es unmöglich ein einzelnes Element der Overview-Map zu verändern ohne die Gestalt der Overview-Map (zumindest minimal) zu verändern. Es ist natürlich möglich, dass bestimmte Änderungen die an einer Stelle vorgenommen werden inhaltlich und praktisch nicht mit Änderungen an einer anderen, entfernten Stelle kollidieren – aber dies festzustellen dürfte zumindest für ein Computerprogramm schwierig sein, zumal die Elemente der Overview-Map ihre Bedeutung nicht aus ihrer absoluten Position sondern erst durch ihre Relation untereinander gewinnen.

Damit wurde die einfachste Lösung unattraktiv. Nämlich die Overview-Map für alle Änderungen durch andere BenutzerInnen zu sperren, wenn ein Mensch in den Edit-Modus der Overview-Map eintritt und sie wieder freizugeben wenn der Edit-Modus verlassen

30. Im Gegensatz dazu könnte ein Interface auch vorsehen, dass viele kleine Änderungen durchgeführt werden, und dann der neue „Gesamt“-Zustand der Overview-Map gespeichert wird.

Locking?

wird. Allerdings gibt dies einem einzelnen offenen gebliebenem und vergessenem Browser-Fenster die Macht alle anderen vom Bearbeiten der Map abzuhalten. Also würde das Locking einen Time-Out brauchen. In jedem Fall ist es schwierig davon auszugehen, dass diese Vorgehensweise gut auf größere Overview-Maps skalieren würde.

Potientielle weitere Probleme?

Des weiteren sollte erwähnt werden, dass das Problem der gleichzeitigen Änderungen vermutlich nicht das einzige Problem ist das beim kollaborativem Erstellen und Bearbeiten einer Overview-Map auftritt. Beispielsweise ist es nicht schwer vorzustellen, dass „Edit-Wars“ auf einer Overview-Map passieren könnten und es wäre zu überprüfen inwieweit Methoden bestehender Wikis für eine Overview-Map adaptiert werden können.

Comet

In diesem Prototypen sollte eine andere Lösung für dieses Problem untersucht werden: Nämlich (wie eingangs erwähnt) um ein System, dass eine Änderung an der Overview-Map sofort in die Ansicht aller die Overview-Map betrachtenden Browser integrieren könnte. Die Inspiration hierfür kam von einem Artikel von Alex Russell, der den Namen „Comet“ (in Analogie zu „Ajax“) als Namen für eine Klasse von Methoden und Technologien die dies ermöglichen würden vorschlägt:

> „New services like Jot Live and Meebo are built with a style of data transmission that is neither traditional nor Ajax. Their brand of low-latency data transfer to the browser is unique, and it is becoming ever-more common. Lacking a better term, I've taken to calling this style of event-driven, server-push data streaming „Comet“. It doesn't stand for anything, and I'm not sure that it should. There is much confusion about how these techniques work, and so using pre-existing definitions and names is as likely to get as much wrong as it would get right.“ (Russell (2006))

Wie funktionieren diese Methoden:

> „Fundamentally, they all use long-lived HTTP connections to reduce the latency with which messages are passed to the server. In essence, they do not poll the server occasionally. Instead the server has an open line of communication with which it can push data to the client.“ (ebdt.)

Für diesen Prototypen sollte das Ergebniss also sein, dass jede kleine Änderung an der Overview-Map – beispielweise das Verschieben eines Knotens – sofort in allen[31] aktuellen Overview-Map-Ansichten dargestellt werden sollte. Damit würden sich die Overview-Maps synchronisieren. Die BenutzerInnen hätten immer den aktuellsten Stand der Overview-Map zur Verfügung. Sie würden sich auch ge-

31. „Alle“ ist ein hoher Anspruch, der sicher nicht garantiert werden kann. Es sei dazugesagt, dass „alle“ hier im Gegensatz zu „keine“ gelesen werden sollte bzw. einen Ideal-Fall darstellt. Lynx oder IE-Instanzen sind z.b. ausgenommen.

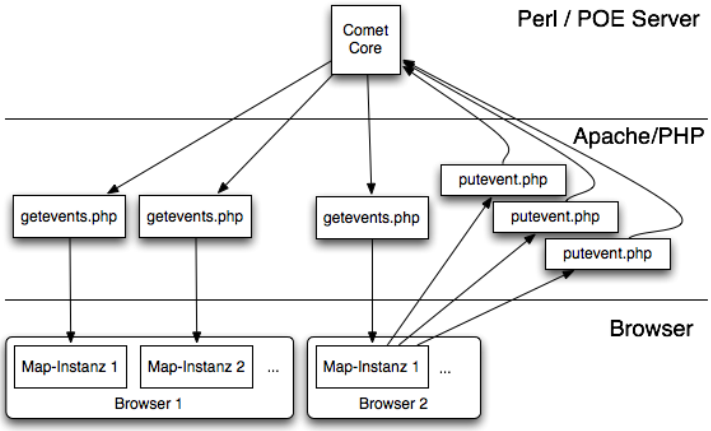


Abbildung 3.8: Übersicht über den Comet-Versuchsaufbau. Die Pfeile zeigen welchen Weg ein „Event“ nimmt. Die putevent-Prozesse sind kurzlebig und geben die Daten einfach an den CometCore-Server weiter. Die getevent-Prozesse sind dagegen langlebig: Eine konkrete Map-Instanz verbindet sich einmal zu dieser Adresse und bekommt dann von dort alle ankommenden Daten des CometCore-Servers durchgereicht. In der Graphik ruft nur die „Map-Instanz 1“ in „Browser 2“ putevent.php auf. Dies dient nur der besseren Übersichtlichkeit, auch andere Map-Instanzen tun dies.

genseitig beim Bearbeiten der Overview-Map zusehen können.[32]

Implementierung auf Basis des letzten Mixmap-Prototypen

Für die Implementierung wurde der aktuellste Prototyp (siehe Kapitel 3.4.4) herangezogen. Frameworks die diese Art der Client-Server-Kommunikation zu unterstützen behaupteten schienen für unsere Zwecke etwas schwergewichtig zu sein. Daher wurde ein relativ simpler Versuchsaufbau realisiert. Eine Übersicht davon ist in Abbildung 3.8 zu sehen.

Aufbau und Architektur des Prototypens

Der Kern des Aufbaus ist ein in Perl[33] geschriebener Server (genannt „CometCore“), der auf einem bestimmten Port Verbindungen entgegennimmt. Seine einzige Funktion ist es alle Daten die über eine solche Verbindung geschickt werden an alle anderen[34] Verbindungen auszugegeben. Die Kommunikation mit dem Browser wird über php-Skripts abgewickelt. Deren Aufgabe ist darauf beschränkt

32. Ob dies immer notwendigerweise gewünscht ist sei dahingestellt. Es ist jedenfalls potentiell ein Werkzeug für Kollaboration.
33. Mithilfe der POE-Library.
34. Tatsächlich wurden die Daten an *alle* Verbindungen weitergereicht, also auch an die Map-Instanz die die Änderung ausgelöst hat. Ansonsten hätte eine Zuordnung von put-request und der getevents-Instanz stattfinden müssen. Dank des Cachings im der Map-Instanz (siehe Kapitel 3.4.4) hatte dies jedoch nur minimalen Zusatzaufwand zur Folge.

Requests entgegenzunehmen und an den CometCore-Server weiterzureichen[35]. „putevent“ baut also eine Verbindung zu CometCore auf, interpretiert die Daten des Browser-Requests, formuliert eine Javascript-Anweisung für die Zielbrowser, reicht diese weiter und beendet sich. „getevents“ hingegen, wenn es vom Browser aufgerufen wird, baut eine Verbindung zu CometCore auf und reicht dessen Ausgabe an den Browser weiter. Es beendet sich nicht von selber und kann also (theoretisch) ewig laufen.

Im Browser wird ein unsichtbarer iFrame erzeugt und dieses angewiesen die URL von getevents aufzurufen. In diesem iFrame treffen nach und nach die erzeugten Event-Anweisungen ein und werden jeweils sofort ausgeführt. Soll selber ein Event erzeugt werden wird putevent mit den passenden Parametern aufgerufen.

Abbildung 3.9 zeigt den fertigen Prototypen in vier Browser-Fenster gleichzeitig.

Erscheint prinzipiell wünschenswert.

Die (wenigen) Testperson die den Prototyp ausprobierten waren im Prinzip sehr angetan von diesem Stil der Overview-Map-Kollaboration. Da jeder Zwischenschritt des Verschiebens von Knotens durch anderen Personen angezeigt wurde, schien (in Abwesenheit anderer Mittel) Kommunikation über Gesten zu passieren.

So gesehen wäre es also sehr spannend erschienen, diesen Weg weiter zu untersuchen.

Das Problem der Browser Connection Limits

Leider zeigten sich jedoch technische und praktische Einschränkungen. Die in diesem Fall verwendete Architektur erfordert für jedes Browser-Fenster eine permanente Verbindung zum Webserver. Skalierbarkeit des Servers außer acht lassend treten hier Einschränkungen der Browser zutage die ein Limit für die gleichzeitig offenen Verbindungen zum Server setzen.

Dies ist ein tendenziell erwünschtes Verhalten und bespielsweise in RFC 2616 (Fielding u. a. 1999) definiert, auch wenn in der Praxis manchmal andere Werte verwendet werden:

> „Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy.“

Nach dem Erreichen des Verbindungslimits trat in einem Browser also der Effekt auf, dass keinerlei Anfragen an den Server mehr geschickt wurden, bis bestehende Fenster geschlossen wurden. Das war einerseits ein unbefriedigender Failure-Mode und andererseits eine unbefriedigende Einschränkung wenn angenommen wird, dass zumindest einige BenutzerInnen gern mehrere Fenster einer Seite öffnen[36].

---

35. Dieser zusätzliche Umweg befreit den CometCore-Server von der Aufgabe HTTP-Requests zu verstehen und HTTP-Antworten geben zu müssen.
36. Informelle Beobachtungen des Autors an sich selbst und anderen deuten dar-

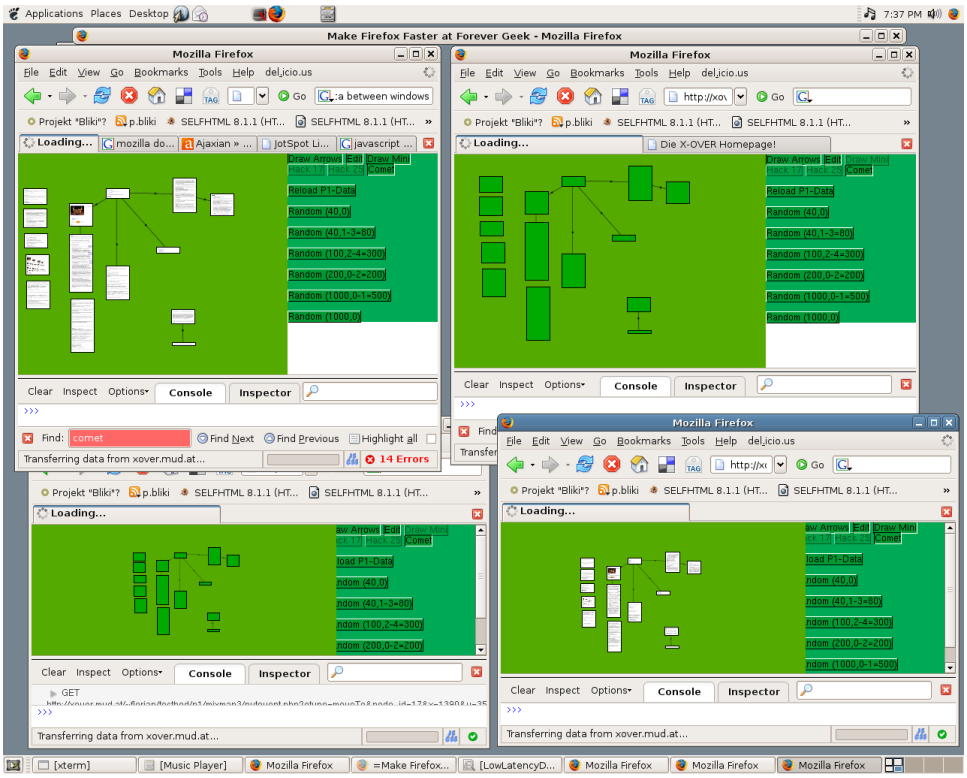


Abbildung 3.9: Screenshot des Comet-Prototypen in vier Browser-Fenstern. Jede Verschiebung eines Knotens in einem Fenster wurde sofort (d.h. ziemlich rasch) in den anderen Fenstern angezeigt. Dabei war es egal ob diese Fenster zu einem oder mehreren Browser gehörten bzw auf einem oder mehreren Rechnern liefen, von Browserlimits abgesehen. In zwei der Map-Instanzen ist die Darstellung von Seiten-Thumbnails ausgeschalten, dies hatte jedoch ansonsten keine Auswirkungen.

Leider schien zunächst keine einfache Möglichkeit gegeben, dass verschiedene Fenster (im gleichen Browser) außer über einen Server miteinander kommunizieren – tatsächlich wird vom Browser in den meisten Fällen versucht dies aus Sicherheitsgründen explizit zu verhindern.

Long Polling?

Aus heutiger Sicht wäre „Long polling“ eventuell ein möglicher Ausweg. Bei diesem Verfahren würden XMLHTTP-Requests geöffnet, aber erst beim Vorliegen neuer Daten vom Server beantwortet. Diese Verbindungen bleiben nicht permanent offen, sondern werden immer wieder neu aufgebaut. Dies beseitigt leider nicht den Umweg über den Server. Es könnte aber bedeuten, dass der Browser die Verbindungswünsche der gleichzeitig offene Overview-Map Instanzen scheduled und nacheinander abarbeitet[37]. Eine Voraussetzung wäre allerdings, dass die Anzahl auf Daten wartender XMLHTTP-Request entweder immer kleiner ist als das Verbindungslimit oder diese nach einiger Zeit abgebrochen werden, da sonst ein ewiger Wartezustand auftreten könnte.

Nicht weiterverfolgt, aber potentiell spannend

Zu diesem Zeitpunkt war jedenfalls nicht klar wie eine solche Lösung funktionieren könnte. Daher wurden die entsprechende Versuche vorläufig abgebrochen. Es besteht jedoch die Vermutung, dass diese Methode recht gut funktionieren würde, wenn die technischen Schwierigkeiten und Hindernisse überwunden würden.

Ob dieses Paradigma für das kollaborative Bearbeiten einer Overview-Map auf der BenutzerInnen-Seite gut skaliert ist eine andere Frage. Die Vorstellung eine (Infinite Canvas) Overview-Map für ein Wiki in der Größe von Wikipedia bei die Änderungen aller Benutzer von allen Browsern in quasi-Echtzeit dargestellt werden sollen klingt zunächst unplausibel.

Heute existieren jedenfalls eine Anzahl von Frameworks für Comet. Eventuell kann das Problem mit ihnen gelöst werden. Dies wurde jedoch nicht mehr untersucht.

In weitere Folge wurden eine Reihe von Prototypen konstruiert deren Ziel es war – grob gesprochen – die Integration der verschiedenen Bestandteile zu untersuchen und zu designen.

3.5 Overview-Map als Teil eines Wiki-Interfaces („p2“)

Die Ziele für diesen Prototypen waren: Es sollte ein einigermaßen funktionierendes Wiki mit WYSIWYG-Editor und Overview-Map-

---

auf hin, dass solche BenutzerInnen existierten, gerade in Wikis. Es ist vermutlich ein verbreitetes Phänomen. Bright (2008) beispielsweise spricht von der Nützlichkeit eines Tab-Grouping-Features in Zusammenhang mit „epic Wikipedia sessions“ und die Graphik in Munroe (2007) deutet an dass nach mehreren Stunden Wikipedia-Benutzung typischerweise mehrere Browser-Ansichten offen sind.

37. Welches Verfahren dafür verwendet wird würde vom Browser abhängen. Informelle Beobachtungen an Firefox 2.0 lassen ein Round-Robin-Prinzip (eventuell mit Priorisierung aktuell angezeigter Fenster) vermuten.

Abbildung 3.10: Ein Screenshot des p2-Prototypen am Ende der Entwicklung

Anbindung entstehen.

Abbildung 3.10 zeigt einen Screenshot des p2-Prototypen am Ende der Entwicklung. Auf der rechten Seite wurde der Inhalt einer Seite angezeigt auf der rechten Navigations- und Bedien-Elemente. Nach dem Klicken auf den Edit-Button konnte nach kurzer Zeit (allerdings ohne Seiten-Reload) der Inhalt der Seite bearbeitet werden. Dabei gab sich die Seite Mühe möglichst exakt gleich auszuschauen wie das spätere „Ergebniss“. Ein „Save“-Button existierte in früheren Versionen, wurde aber zwischenzeitlich beseitigt. Gespeichert wurde automatisch beim Verlassen des „Edit“-Modus bzw. beim Verlassen der Seite.

Frühere Versionen der Seite konnten mit den „<“- und „>“-Buttons durchgeblättert werden. Dabei war kein Reload der Seite notwendig, was es ermöglichte durch schnelles Springen zwischen den Versionen visuell die Unterschiede zwischen ihnen zu identifizieren. Oder anders: da der Rest der Seite sich beim Blättern nicht veränderte oder auch nur „flickerte“, konnten durch die „Bewegung“ während des Umschaltens Unterschiede identifiziert werden.

Eine Mini-Map (siehe Kapitel 3.1), eine Tag-Cloud, eine Liste mit Seiten die (intern) auf die aktuelle Seite verlinken und ein Suchfeld (das ohne Reload sofort Suchergebnisse lädt) ermöglichen Navigation im Wiki. Ein RSS-Feed mit den letzten Änderungen im System wurde bereitgestellt.

Zur Implementation des Systems wurden serverseitig PHP und MySQL auf einem Linux-System eingesetzt. Im Browser wurden die Prototype-Bibliothek[38] und das Dojo-Framework[39] verwendet. Letzteres stellte vor allem den Editor zur Verfügung. Die Mini-Map wurde nicht neu implementiert, sondern es wurde die letzte Version der „mixmap“ eingesetzt.

Bearbeiten von Links

Eine gewisse Einschränkung gab es beim Bearbeiten von Links. Nach der Entdeckung, dass es möglich ist[40] Links zu kopieren und in den WYSIWYG-Bereich im Browser einzufügen bzw. auch Links per Drag and Drop im WYSIWYG-Bereich einzufügen wurde damit experimentiert dies als die einzige Art Links einzufügen zur Verfügung zu stellen. Ein typischer Workflow um einen Link von einer internen Seiten auf eine andere interne Seite zu setzen sah in etwa so aus:

1. Den gewünschten Link auf der aktuellen Seite finden bzw. erzeugen; typischerweise indem ein Suchbegriff in das Suchfeld eingegeben wird, aber auch der Bereich „Pages linking here“ stellt Links zur Verfügung.
2. Den Link in den Text (an die gewünschte Stelle) ziehen. Exakt: Den Link anklicken, aber die Maus gedrückt lassen. Dann den Mauszeiger an die gewünschte Stelle bewegen und die Maus loslassen.
3. Eventuell den Linktext verändern. Also in den Link-Text klicken, und dann entweder neuen Text eingeben oder bestehenden löschen. Dabei ist zu beachten, dass immer ein Link-Text vorhanden sein muss, ansonsten wird der Link gelöscht.

Wenn dies die einzige Möglichkeit ist Links zu erstellen, gibt es keine direkte Möglichkeit aus dem System heraus Links auf externe Seiten zu setzen. Stattdessen muss eine (Web-)Seite mit dem gewünschten Link gefunden werden und von dort kopiert werden[41]. In manchen Fällen erforderte dies manuell eine HTML-Seite mit dem gewünschen Link anzulegen, was für die Praxis eines Systems, das einfaches Hypertext-Authoring bieten will eher nicht akzeptabel ist.

Desweiteren dreht dieser Workflow den „typischen“ Workflow des Link-Erstellens in Rich-Text-Editoren um. Abbildung 3.11 zeigt etwa die Erstellung eines Links in einer aktuellen Wordpress-Version (Juni 2008). Hierbei wird *zuerst* der als Link-Anker dienende Text selektiert, dann ein „Link bearbeiten“-Button geklickt und erst dann wird die URL spezifiziert.

Es fehlt explizite Link-Erstellung

Der Prototyp zeigte, dass Link-Erstellung per Drag and Drop eine reizvolle aber vermutlich nicht ausreichende Möglichkeit ist. Selbst wenn wir ignorieren, dass die Browser die Details was und wie bei

38. http://www.prototypejs.org/
39. http://dojotoolkit.org/
40. Das Kopieren bzw. „Droppen“ vorhandener Links wurde damals in Safari und Firefox ausprobiert. In Firefox ging beides, in Safari letzteres nicht.
41. In Firefox etwa indem der Text des Links – möglicherweise mit Zusatztext rund um den Link – markiert und kopiert wird

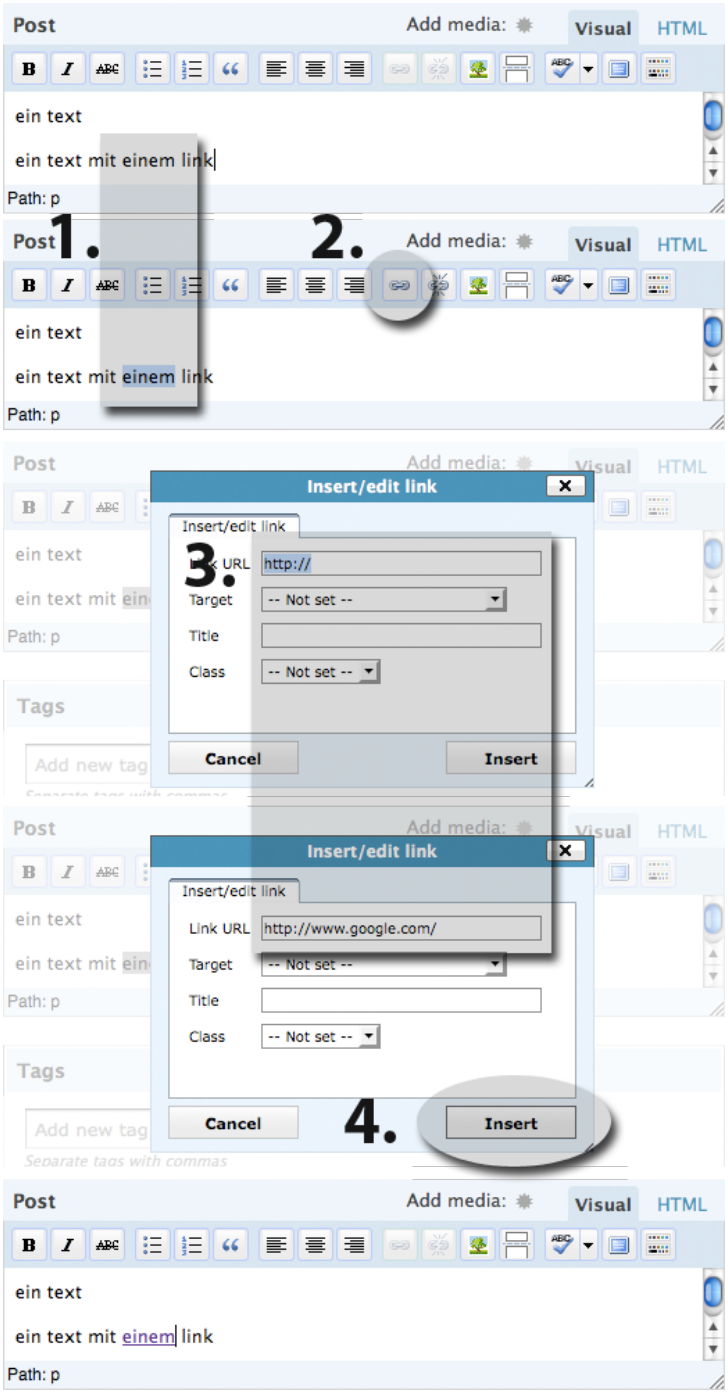


Abbildung 3.11: Erstellung eines Links im Rich-Text-Editor von Wordpress: Zuerst wird der als Link-Anker dienende Text ausgewählt (1); dann wird auf das Link-Erstellen-Werkzeug geklickt (2); daraufhin kann in einem Mini-Fenster die URL eingeben werden (3), diese wird unter Umständen auch erst an diesem Punkt gesucht bzw. kopiert; am Schluss wird auf den „Insert“-Button geklickt (4).

einer Drop-Operation eingefügt unterschiedlich zu implementieren scheinen. Auf diese Weise können nämlich nur vorhandene Links dupliziert werden. Außerdem ist das nachträgliche Bearbeiten des Link-Textes eher mühsam, wenn keine einfach Kontrolle darüber besteht ob ein Text-Teil jetzt Bestandteil des Links ist oder nicht.

In Kapitel 3.9.1 wird eine vollständige Alternative zu herkömmlicher Link-Erstellung entworfen werden.

Versions-Browsing alleine reicht vermutlich nicht.

Bezüglich der Darstellung von Versionen wurde das vorliegende Verfahren eher als mühsam empfunden. Das mag zum Teil an der recht kleinen Dimension der Bedienelemente gelegen haben. Allerdings ist dies vermutlich nicht der einzige Grund.

Andere Systeme wie Mediawiki machen die Unterschiede zwischen Versionen explizit sichtbar. Auch wenn diese Systeme nicht immer tatsächlich die Änderungen so beschreiben wie dies ein Mensch vielleicht tun würde, so ersparen sie doch in vielen Fällen die Arbeit die Unterschiede identifizieren zu müssen. Das in diesem Prototypen notwendige „visuelle Diff“ erfordert dies jedoch jedesmal von der Benutzerin oder dem Benutzer.

Des weiteren stößt das manuelle Identifizieren von Unterschieden an seine Grenzen wenn mehrere kleine Unterschiede vorliegen, bzw. zuerst im Text vorkommenden Unterschiede den Textfluss und das Aussehen des Rest-Textes verändern. In diesen Fällen wird es recht mühsam die Unterschiede zu identifizieren.

Andererseits gibt es einen schnellen Überblick über die Gesamtseite was in manchen Fällen nützlich sein könnte – beispielsweise bei der Identifzierung von Spam.

Als einzige Möglichkeit Versionen zu begutachten ist es jedoch vermutlich nicht ausreichend. Es zeigte sich auch, dass der Einsatz vorhandener Frameworks und Hilfs-Bibliotheken eine zweischneidige Sache ist. Einerseits ermöglicht dies schnell mit fertigen Features zu experimentieren und sie nicht mühsam selber entwickeln zu müssen. Andererseits besteht die Gefahr (bzw. das Dilemma) das Interface-Design der Bibliotheken zu übernehmen, oder (falls überhaupt möglich) das Design anpassen und dazu die Bibliotheken erst recht relativ umständlich umschreiben zu müssen.

Fertige Frameworks implizieren fremde Interface-Entscheidungen

Im konkreten Fall, beispielsweise, bestand das Interface-Paradigma des Dojo-Editors daraus, dass die Formatierungs- und Bedien-Elemente des Editors am oberen Rand des Editors platziert wurden, und dort Platz wegnahmen. Diese Anordnung findet sich in den meisten Rich-Text-Editoren im WWW. Verbunden mit einer beschränken Größe des Textfeldes gibt sie auch durchaus Sinn. In unserem Fall wäre die bevorzugte Variante jedoch eine Platzierung neben dem Textfeld gewesen. Da es sich als schwierig herausstellte das Framework den Wünschen anzupassen, wurde die einzige vom Framework angebotene Variante gewählt, die nicht damit kollidierte das Aussehen des Editors und des Texts möglichst ident zu gestal-

ten. Leider war dies das vollständige Ausschalten aller Buttons für die Bearbeitung des Textes.

Insgesamt waren die Ergebnisse aber durchaus ermutigend. Zwar wurden keine formellen Tests durchgeführt, jedoch zeigten sich die Personen die das System ausprobierten durchaus angetan von der Möglichkeit den Text direkt zu bearbeiten.

Die Overview- bzw. Mini-Map waren jedoch abgetrennt vom restlichen System und tauchten in der Bedienung des Systems so gut wie nicht auf. Daher wurde beschlossen einen Prototypen zum Thema „Interaktion der Mini-Map mit dem Umfeld“ zu erstellen.

3.6 Maps-Interaktionen („p3“)

Die bisherigen Map-Prototypen demonstrierten grob die technische Machbarkeit einer Overview-Map. In dieser Phase sollte anhand eines neuen Prototypen die praktische Benutzbarkeit der Overview-Map verbessert werden.

AJAX Seiten Wechsel

Die erste Änderung am Verhalten des Sytems war zunächst einen Reload der Gesamtseite beim Verfolgen eines internen Links zu vermeiden. Für diesen Zweck wurde das Schema der internen URLs von „?<id>“ auf „#<id>“ geändert und entsprechende Logik geschrieben die den Seiteninhalt mit AJAX änderte[42]. Darauf aufbauend wurde bei einem Wechsel auf eine Seite der entsprechende Knoten auf der Mini-Map zentriert, durch sanftes Scrollen an die entsprechende Position. Insgesamt ergab sich dadurch ein spannender Effekt bei der Benützung des Back-Buttons: Der zurückgelegte (virtuelle) „Pfad“ zwischen den Seiten wurde rückwärts zurückgelegt.

Obwohl dies eine spannende Verdeutlichung der Position und des „Weges“ im System war, wurde die automatische Zentrierung des aktuellen Knotens letztendlich wieder fallengelassen, da sie insgesamt als unangenehm empfunden wurde. Eventuell liegt dies daran, dass im explorativen Lese-Modus eine Seite oft nur genug Aufmerksamkeit bekommt um zu entscheiden ob der Inhalt interessant ist (bzw. häufig dass der Inhalt eben nicht interessant ist). Die Mini-Map erzwingt jedoch jedesmal einen Perspektiven-Wechsel, d.h. eine neue Ansicht auf den Infinite Canvas.

Bezugspunkte wie „Links oben war ich schon“ verlieren rasch Nützlichkeit und Sinn wenn „Links oben“ zu schnell seine Bedeutung wechselt.

Eine Variante wäre die Knoten nicht zu zentrieren sondern den Viewport nur soweit zu bewegen, dass der Knoten sichtbar ist. Dies wurde jedoch in diesem Prototypen nicht getestet.

---

42. Der Teil einer URL nach dem #-Zeichen wird Fragment Identifier genannt und wird nicht vom Browser an der Server geschickt (siehe beispielweise Berners-Lee, Fielding und Masinter (2005)). Änderungen an diesem Teil bewirken keine Anforderungen an den Server. Javascript kann jedoch diesen Teil auslesen und darauf reagieren.

Mini-Map Suche

Ein anderer Versuch wurde unternommen eine Suche nach Seiten auf der Mini-Map zu realisieren. Der in einem Textfeld eingegebene Text wurde in allen Seiten gesucht und die Seiten mit Treffern wurden visuell markiert[43]. Die Nützlichkeit dieser Lösung war jedoch in diesem Fall unklar. Hauptsächliches Problem schien, dass es sich weniger um eine Darstellung von Suchergebnissen handelt, also nur die Menge der Seiten die das Kritierium erfüllen, sondern mehr um einen „Filter“. Praktisch gesehen waren in Ansichten die die Map-Objekte groß anzeigten nicht notwendigerweise alle Treffer sichtbar und in Ansichten die die Map von weit „oben“ zeigten gingen die einzelnen Treffer in der Menge aller Objekte unter.

Aufhebung der Edit-/View-Unterscheidung

In bisherigen Prototypen waren Interaktionen mit der Map in einen „Edit“- und einen „View“-Modus geteilt, zwischen denen gewechselt werden musste. Im Edit-Mode konnte die Map bearbeitet werden aber nicht zu den Seiten-Inhalten gewechselt werden. Im View-Modus konnten Seiten ausgewählt werden, aber nicht ihre Position verändert. Hier war ein Ziel die Unterscheidung in zwei verschiedene Modi aufzuheben, da die Einschränkungen beide Modi als hinderlich empfunden wurden.

Dazu wurde es (auf der technischen Seite) zunächst für notwendig erachtet, zwischen einem einfachen Klick in die Map und einem Klick, bei dem die Maus bewegt wird während die Maustaste gedrückt ist, zu unterscheiden. Zusätzlich wurde unterscheiden

- ob in die Map oder auf einen Knoten geklickt wurde
- ob die Shift-Taste gedrückt wurde und
- ob eine „Auswahl“ vorhanden ist

Letzeres wurde eingeführt um mehrere Knoten auf einmal verschieben zu können. Dies wurde nämlich als ein für die Praxis sehr relevantes Feature gesehen. Insgesamt sahen die möglichen Interaktionen mit der Map wie folgt aus:

---

43. Eine erste Variante versuchte durch verschiedene Graustufen Anzahl und Art der Treffer zu kommunzieren. In diesem Fall wurde dies als eher schwierig zu lesen wargenommen. Nichtsdestotrotz exisitert in dieser Frage ein breites Spektrum möglicher Lösungen bezüglich Art und Inhalt der Markierung von Treffern.

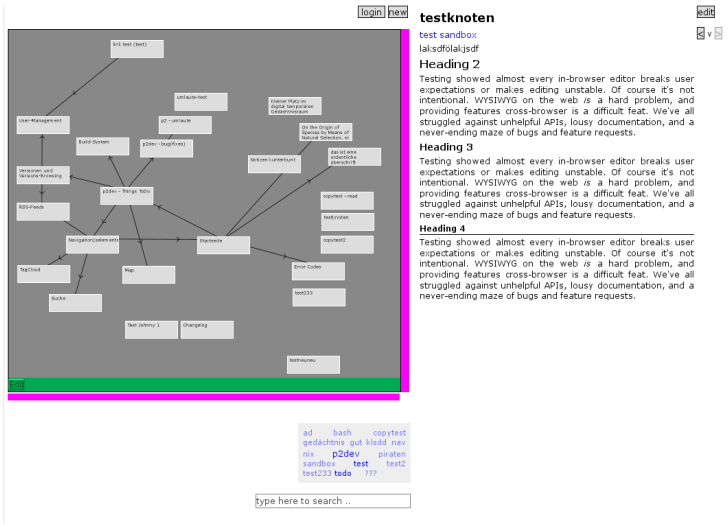


Abbildung 3.12: Ein Screenshot des „p3“-Prototyps

	Startpunkt in Map	Startpunkt in Knoten
Click	Focus auf Map-Punkt oder, falls Knoten selektiert sind: Selektion entfernen	Anzeigen des Knotens
Click mit Move	Map-Drag	Knoten verschieben. Falls der entsprechende Knoten in der Auswahl ist: alle ausgewählten Knoten verschieben
Shift-Click	Auswahl entfernen	Knoten zur Auswahl hinzufügen/entfernen
Shift-Move	Knoten auswählen mit Rahmen	Link anlegen

Kurzzeitig wurde bei einem Klick auf einen Knoten auch dieser Knoten zentriert, dies wurde jedoch später fallengelassen (wie oben beschreiben).

Abbildung 3.12 zeigt einen Screenshot des Prototypens in Zuge der Entwicklung. Die pinken Balken dienten dabei als Handle um die Mini-Map in der Größe anpassen zu können.

Zuletzt wurden noch Versuche mit der Darstellung von Links als gebogene, runde Linien durchgeführt. Es wurde aber klar, dass Zufalls-Daten ohne echten Inhalt die Interaktionen mit und das Aussehen von einem „echten“ System nur unzureichend simulieren können. So war unklar ob die Link-Strukturen unserer Beispiel-Maps potentiellen „echten“ Strukturen überhaupt ähnlich sind. Dies ist vielleicht ein Thema, das leichter behandelbar wäre, wenn ein solches System

schon in der Praxis benutzt werden würde.

3.7 Bloki: Tag-Aggregator

Eine Frage die zu diesem Zeitpunkt noch wenig in den Prototypen abgehandelt wurde, war die Integration von Weblogs und Wikis in einem System. In diesem Prototypen sollte dies versucht werden, basierend auf den Annahmen, dass

- Weblogs einfach eine invers-chronologische Darstellung zusammengehöriger Einträge und ein dazugehörender RSS-Feed sind
- sowohl Wiki-Seiten als auch Weblog-Einträge eigentlich nur ein HTML-Fragment (eventuell erzeugt aus Markup) sind

Der Entwurf sah vor, dass eine Seite mehrere Tags haben kann (bzw. Kommentare und andere Metadaten) und jeder Tag einen RSS-Feed erzeugt. Ein Tag erzeugt eine invers-chronologische, an Weblogs orientiere Auflistung von Seiten. So gesehen ist jeder Tag ein potentieller Weblog.

Ein „Weblog“ würde in diesem System dadurch erzeugt, dass ein neues Tag benutzt wird. Jede weitere Benutzung des Tags würde einen Eintrag zu diesem „Weblog“ hinzufügen. Obwohl das System die verschiedenen Tags nicht unterscheidet oder manche speziell behandelt wurde angenommen, dass in der Praxis ein bestimmtes „blog“-Tag durch soziale Konvention und Praxis zu einen Weblog (als Ort einer bestimmten Kommunikation) werden würde und andere Seiten und Tags eher wiki-artig benutzt werden würden.

In jedem Fall wurde ein Prototyp nach diesem Entwurf erzeugt. Die Prototyp besaß auch noch die Zusatz-Navigations-Hilfe zu einem bestimmten Tag A (wenn dieses angewählt war) die Menge aller Tags aller Seiten mit Tag A anzeigte. Gewissermaßen „Untertags“, allerdings *nicht* in der Bedeutung dass diese „Untertags“ nur auf Seiten mit Tag A benutzt würden. Mit diesem Werkzeug konnten auch Kombinationen von Tags ausgewählt und angezeigt werden.

Implementierung geschah mit Ruby on Rails[44] und Zuhilfenahme diverser Zusatz-Module. Für das Bearbeiten von Seiten beispielweise wurde eine Standard-Edit-Save-Formular-Konstruktion gewählt, und ein Markup-Modul integriert. Eine der wenigen Stellen an der nicht auf vorgefertigte Module zurückgegriffen werden konnte waren die Berechnungen über Tags, beispielsweise das Bestimmen der Menge aller Tags aller Seiten mit bestimmten Tags oder das Filtern von Seiten nach mehreren Tags. Da das Ziel schnelle Evaluation der „Tag = RSS-Feed = Blog“-Idee war, wurde auf den Einsatz von Interfaces die AJAX oder andere Client-seitige Logik erfordern würde weitgehend verzichtet.

---

44. http://www.rubyonrails.org/

Potentielle Probleme

Obwohl der Prototyp prinzipiell zufriedenstellend funktionierte (bzw. den gestellten Anforderungen genügte) überwiegt in der Analyse Skepsis, allerdings ist nicht ganz klar woher diese kommt. Eine Vermutung ist, dass die Grundannahmen fehlerhaft sind oder zumindest lückenhaft sind und der Prototyp somit die Unterschiede zwischen Weblogs und Wikis nicht gut abbilden kann.

Beispielsweise ist die typische Menge der „BesitzerInnen“ eines Weblogs nur eine Person groß, während Wikis zumindest theoretisch möglichst viele BenutzerInnen zulassen wollen. Auch ist die Annahme, dass Wiki-Seiten und Weblog-Einträge im Prinzip ident sind, vermutlich falsch, sobald die rein technische Ebene (HTML-Fragment, adressierbar) verlassen wird. Ein für diesen Entwurf potientiell relevanter Unterschied ist beispielsweise, dass die Sichtbarkeit und Prominenz einer Seite in Wikis weitgehend manuell kontrolliert wird (über die gesetzten Links) während Navigation und Sichtbarmachung in Weblogs eher von automatischen Prozessen kontrolliert wird. Desweiteren laden Tags dazu ein, sie spontan und nicht notwendigerweise konsistent zu verwenden. Indem die Tags zum Hauptmechanismus der Navigation erklärt werden, hat jede Änderung große Auswirkungen und dadurch geht unter Umständen die Möglichkeit verloren, sie ohne viel Nachdenken zu verwenden.

3.8 Bloki: Eine Generalisierung, Tinderbox-inspiriert („jbox“)

Aus der Erkenntnis, dass Weblog-Einträge und Wiki-Seiten verschiedene Anforderungen etwa an Darstellung, Zugriffsrechten, Sichtbarkeit haben entstand der Wunsch diese Änderungen modellieren zu können. Dazu sollte ein Prototyp geschrieben werden, der in der Lage sein sollte diese Unterschiede aus einem generalisiertem Framework zu erzeugen.

Tinderbox

Der Entwurf dieses Prototyps war stark beeinflusst von der Software „Tinderbox“ von Mark Bernstein. Tinderbox[45] ist ein Programm für Mac OS X. Mark Bernstein selbst beschreibt Tinderbox in der Einführung von „The Tinderbox Way“ als ein „personal content management assistant“ (Bernstein 2006). Vor allem zwei Eigenschaften machen Tinderbox als potentielle Inspiration für unsere Experimente spannend:

Spatial Hypertext

Erstens stellt Tinderbox eine Art Spatial Hypertext Ansicht auf die in einem Dokument vorhanden Inhalte zur Verfügung. Diese Eigenschaft macht Tinderbox auch immer wieder zum Thema bzw. zu einem Bespiel in der Literatur zum Thema Hypertext bzw. Spatial Hypertext, beispielsweise in Atzenbeck und Nürnberg (2005) oder m. c. schraefel (2007), um nur einige herauszugreifen und natürlich in Arbeiten von Mark Bernstein selber. Abbildung 2.12 zeigt einen Screenshot der Map-Ansicht von Tinderbox.

45. http://www.eastgate.com/Tinderbox/

Konzepte von Tinderbox

Zweitens ist das Framework, dass die Software zur Verfügung stellt, scheinbar flexibel genug um diverse Anwendungen wie Weblogs, (persönliche) Wikis[46], Präsentationen oder GTD-Methodik abzudecken[47].

Die Art und Weise wie Tinderbox diese Funktionalität und Flexibilität zur Verfügung stellt läuft hauptsächlich über zwei Mechanismen ab:

Einerseits besitzt jeder Knoten („Note“ in Tinderbox-Begriffen) ein große Anzahl an „Attributen“, die vorhandene Funktionalität abbilden (Titel, Position, Farbe, Erstelldatum etc), Funkionalität aktivieren können (bespielsweise Templates für Export-Funktionen) oder auch von den BenutzerInnen hinzugefügt werden können um beispielsweise strukturiert Informationen zu verwalten.

Diese Attribute können von einem anderen Knoten „geerbt“ werden. Alle nicht explizit gesetzten Attribute werden von einem anderen Knoten gelesen, wenn das Prototyp-Attribut gesetzt ist und auf diesen Knoten zeigt. Dies funktioniert auch transitiv.

Weitere Hilfsmittel umfassen beispielsweise „Agenten“, die Listen von Such-Ergebnissen generieren, und eine Skript-Sprache, die auf bestimmte Ereignisse wie das Erstellen oder Hinzufügen von Knoten reagieren kann.

Dieses System wirkte auf den ersten Blick simpel und doch mächtig genug um zu versuchen auf einer solchen Basis ein System zu erstellen, das Wikis-Seiten und Weblog-Einträge aus allgemeinen Elementen ableitet. Im Unterschied zu Tinderbox sollte unser Prototyp jedoch nicht einfach das Framework zur Verfügung stellen, sondern das Interface soweit vereinfachen, dass es wie vorhandene Weblog- oder Wiki-Systeme benutzt werden könnte und nicht erfordern würde sich in das Framework einzuarbeiten.

Implementiert wurde der Prototyp wieder in Ruby on Rails. Wie schon in Kapitel 3.5 angesprochen erforderte die Benutzung eines fertigen Frameworks im Laufe der Entwicklung immer wieder Kompromisse zwischen dem angestrebten Interface und den zur Verfügung gestellen Elementen. Um nicht selbst ein Layout entwerfen zu müssen, dass die ästhetischen Ansprüche an einen Weblog erfüllt, wurde ein Wordpress-Theme als Vorbild für das Aussehen der Seite adaptiert.

Abbildung 3.13 zeigt einen Screenshot der Startseite des Systems gegen Ende der Entwicklung.

Das Design des Systems sah vor, dass Unterschiede im Verhalten

46. Das Konzept des Wikis das nur für eine Person ist scheint manchmal etwas seltsam unter der Prämisse, dass Wikis durch eine „JedeR kann bearbeiten“-Philosophie definiert werden. Der Begriff scheint jedoch gebräuchlich zu sein für Notiz-Anwendungen, die Anleihen bei Wikis nehmen.

47. Der Autor hat nicht alle diese Anwendung getestet. Die Erfahrungen mit Tinderbox lassen diese Ansprüche aber durchaus plausibel erscheinen. Siehe beispielsweise http://www.eastgate.com/Tinderbox/Exchange.html.

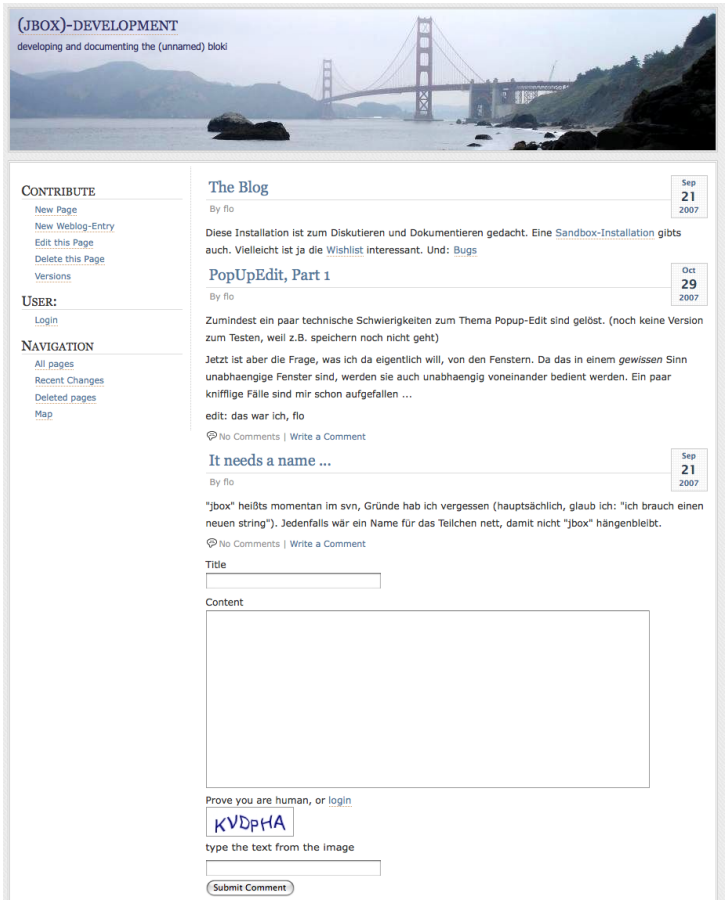


Abbildung 3.13: Einstiegseite des jbox-Prototypen

und Aussehen eines Knotens über Attribute gesteuert werden. Diese Attribute wären später theoretisch für die BenutzerInnen einzeln bearbeitbar (in der Praxis aber nur durch direktes Schreiben in die Datenbank). Ein Knoten würde die Attribute aber typischerweise von einem andern Knoten „erben“. Knoten die ein „is_prototype“-Attribut gesetzt haben stellen die Auswahl-Möglichkeiten bei neuen Knoten dar. Dies ist analog der Funktion von Tinderbox.

Anders als in Tinderbox jedoch versucht dieses System die möglichen Prototypen als *die* möglichen Optionen darzustellen, und dies auch relativ prominent, während die (im Hintergrund verfügbare) Komplexität der Attribute und Optionen nur bei Bedarf dargestellt wird. Um einen Startpunkt für die BenutzerInnen herzustellen war vorgesehen, dass beim Installieren/Anlegen eines solchen Systems Standard-Prototypen bereitgestellt werden, also beispielsweise eine Vorlage für einen Weblog-Eintrag. In Abbildung 3.14 ist zu sehen wie dies ausgesehen hat. „New page“ ist die leere Default-Seite danach werden mit „New <xy>“ alle Knoten mit dem „is_prototype“-

Abbildung 3.14: Neue Seiten im jbox-Prototypen

Attribut als Vorlagen aufgelistet.

Ein Nebeneffekt dieser Modellierung ist, dass für jeden modellierten Typ von Seite ein Knoten existiert, der nur den Zweck hat Vorlage zu sein. Dass heißt das Interface zum Ändern der Einstellungen aller Weblog-Einträge wäre eine Änderung an der Weblog-Vorlage. Die Vorlage existiert jedoch im gleichen Raum wie die restlichen Seiten.

Für die Funktionalität eines Weblogs fehlt noch eine Möglichkeit Weblog-Einträge zu sammeln. Dafür gab es spezielle Knoten die andere Knoten nach vorgegebenen Kriterien sammelten und sortierten. Hier war das Kriterium, dass ein Knoten seine Eigenschaften von der Weblog-Eintrags-Vorlage erbt. Sortiert wurde wie in einem Weblog invers chronologisch.

Der Weblog war selbst ein Knoten und hatte als solcher einen Text und die Möglichkeit Kommentare zu ihm (also nicht zu einen Eintrag, sondern zum Weblog) zu verfassen. Dies ist ein Unterschied zu Weblog-Systemen wie Wordpress oder twoday.

Instant-Preview

Mit dem konkreten Interface, mit dem Einzel-Seiten erstellt und bearbeitet wurden, wurde auch hier wieder etwas experimentiert. Die ursprüngliche Version benutzte Standard-Rails-Elemente und besaß eine simple Edit-View-Logik, in der Markup in einer HTML-Textarea bearbeitet wurde (analog zu unzähligen vorhandenen Systemen). Später wurde versucht nach dem Vorbild von jottit[48] ein System zu implementieren in dem das geschriebene Markup sofort während des Schreibens in einem anderen Bildschirm-Bereich gerendert wird. Obwohl theoretisch ein Fortschritt im Vergleich zu einem Preview-System, dass einen Zusatz-Schritt erfordert, war dieses System für uns leider nicht ganz praktikabel.

Problem: Screen-Real-Estate

Das Problem ist hier im Prinzip der begrenzte Screen-Real-Estate (bzw. ist ein Problem des Seiten-Layouts). Jottit benutzt ein geteiltes Browser-Fenster. Hier wäre hingegen die Idee gewesen das Preview des generierten HTML an der exakt gleichen Stelle zu gene-

48. http://www.jottit.com, abgerufen am 4. Juli 2008

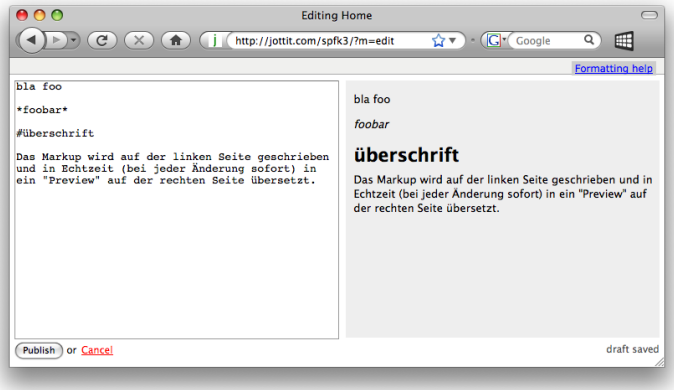


Abbildung 3.15: Screenshot des Edit-Screens von jottit. Auf der linken Seite wird Markup geschrieben, auf der rechten bei jeder Änderung ein Preview erzeugt.

rieren, an dem es später stehen wird. Das führt dazu, dass entweder im Lese-Layout Platz für das Edit-Interface reserviert werden muss, oder dass das Edit-Interface Teile des normalen Interfaces überlagert.

In Jottit selber stellt sich dieses Problem so nicht, da es weniger Anforderungen stellt. Das Preview entspricht nicht ganz exakt dem Endergebnis. Und während des Bearbeitens wird der gesamte Platz im Fenster ausschließlich für ebendiesen Zweck verwendet. Für einen Screenshot dieses Layouts siehe Abbildung 3.15.

In diesem Fall war aber der Wunsch vorhanden, ein exaktes, dem Endergebniss entsprechendes Preview zu erzeugen und damit auch eine Einbindung in das Gesamt-Layout der Seite, so wie sie typischerweise betrachtet werden würde.

Pop-Ups, leider auch keine perfekte Lösung

Eventuell ein sinnvoller Ausweg wäre hier, den überdeckten Platz von den BenutzerInnen verschiebbar zu machen. Beispielsweise durch ein Javascript-verwaltetes Fenster-im-Fenster oder aber dadurch die Fensterverwaltung gleich an den Browser bzw. Window-Manger zu delegieren und mit Pop-Ups zu arbeiten. Letzteres wurde versucht, allerdings sind (aus guten Gründen) das Verhalten des Browsers bezüglich Pop-Ups nicht vollständig von den EntwicklerInnen einer Webseite kontrollierbar, und so ist es beispielsweise möglich das Haupt-Fenster und das Pop-Up unabhängig voneinander zu navigieren was in Verbindung mit dem Phänomen, dass die Pop-Ups in diesem Versuchen für mehrere Ursprungs-Fenster alle das gleiche Pop-Up-Fenster verwenden[49] zu *sehr* merkwürdigen Effekten führt, die für eine Benutzerin oder einen Benutzer vermutlich nur durchschaubar sind wenn sie oder er die dahinterstehenden (und eher technischen) Mechanismen kennt.

49. Dies war das zu beobachtende Verhalten in Firefox und Safari

Der Client rendert das Markup?

Eine anderere (technische) Schwierigkeit in diesem Szenario ist, dass die Logik im Server, die aus Markup HTML generiert, im Client dupliziert werden müsste – es sei denn jedes Preview wird über den Server generiert, aber dies belastet den Server und hat ein relativ großes Delay.

Für ein perfektes – im Client erzeugtes – Preview wäre es außerdem notwendig andere, normalerweise serverseitig vorhandene, Information zu duplizieren und zu synchronisieren, etwa das Vorhanden- oder Nicht-Vorhandensein bestimmter Seiten um das passende Aussehen von internen Wiki-Links zu generieren. Natürlich nicht unmöglich, aber vermutlich etwas aufwendig.

Aufwand der Implementierung

Es stellte sich heraus, dass der Aufwand ein solches System zu implementieren ein bißchen höher war als erwartet. Einerseits war es mühsam viele kleine Unterschiede zu parametrisieren. Weiters scheint es, dass ein Datenmodell in dem Objekte

1. viele verschiedene (und zur Laufzeit erweiterbare) Attribute haben können (aber nicht müssen)
2. diese von beliebigen anderen Objekten „erben“ kann und
3. die Objekte selber modellieren sollen wie sie dargestellt werden und sich verhalten sollen (im Interface und im Zusammenspiel mit anderen)

nicht von vornherein gut in ein MVC-Framework passt, das „Objekte“ typischerweise als Datenbank-Zeilen sieht. Es ist natürlich möglich, aber die entsprechenden Mechanismen werden nicht zur Verfügung gestellt sondern müssen selbst geschrieben werden.

Eine andere spannende Erkenntnis war, dass eine große Vielzahl von Markup-Sprachen für die Verwendung in Weblogs, Wikis und ähnlichem existiert, samt fertiger Bibliotheken für den Einsatz in diversen Programmier-Sprachen und sogar eigener Versuche diese Vielfalt ein wenig zu standardisieren[50]. Das System besaß sogar die Möglichkeit für verschiedene Seiten verschiedene Markup-Systeme zu verwenden. Eine Anwendung hierfür könnte beispielsweise sein Markup-Text aus anderen Wiki- und Weblog-Systemen importieren zu können. Andererseits erzeugt dies natürlich erst recht Unsicherheit welches Markup für eine bestimmte Seite zu verwenden ist, und BenutzerInnen müssen unter Umständen sogar in *einem* Wiki-System zwischen verschiedenen Markup-Sprachen hin- und herschalten.

Zu den erfreulichen Ergebnissen gehört, dass eine einfache Overview-Map, die das Problem der Darstellung von Links ignoriert, erstaunlich leicht zu implementieren war.

---

50. http://www.wikicreole.org/, abgerufen am 4. Juli 2008.

3.9 Bloki: Getrennte Systeme und Current-Generation WHYSIWYG („moki“)

Motivation: contentEditable

Dieser letzte Prototyp entstand einige Zeit nach den anderen. In der Tat war der hauptsächliche Anlass ihn zu entwickeln die Verfügbarkeit des „contentEditable“-Mechanismus in nunmehr allen größeren Browsern, d.h. in Internet Explorer, Firefox, Opera und Safari. In Anbetracht dieser Entwicklung stellte sich die Frage, ob es dadurch vielleicht möglich wäre ein WHYSIWYG-ähnliches Bearbeiten unkompliziert und browserübergreifend zu implementieren.[51] Des weiteren sollte die automatische Erstellung von Seiten-Thumbnails wieder inkludiert werden, wenn dies einfach möglich wäre, nachdem zwischenzeitlich Programme gefunden wurden, die behaupten diesen Prozess automatisieren zu können.[52]

Ein zweiter wichtiger Aspekt war das Ziel in diesem Prototypen die Wiki- sowie die Weblog-Funktionen getrennt zu sehen, zu modellieren und zu entwerfen. Das heißt: nicht eine Übermenge der Funktionen modellieren und dann zu spezialisieren sondern (auf einer technischen Ebene) einzeln ihre Eigenschaften zu modellieren. Die Annahme war, dass dies erstens leichter wäre und zweitens weniger Kompromisse im Design des Interfaces erfordern würde. Diese Annahme entstand aus den Erfahrungen mit dem „jbox“-Prototypen (Kapitel 3.8).

Link-Interface

Ein dritter wichtiger Aspekt ergab sich im Laufe der Entwicklung des Prototypen: Es wurde versucht ein Interface für die Erstellung und Verankerung von Links zu entwerfen, dass, anders als die meisten Rich-Text-Editoren und Markup-Systeme, Links als Merkmal einer Seite oder eines Eintrags sieht, und weniger als weitere Eigenschaft von Wörtern und Wort-Ketten.

Implizite „Überschriften“

Eine Design-Entscheidung im Zusammenhang mit dem Bearbeiten war es kein explizites „Titel“- oder „Überschrifts“-Feld vorzusehen. Stattdessen nimmt das System die erste vorhandene „H1“-Überschrift als Beschreibungs-String, wenn ein solcher gebraucht wird um die Seite zu beschreiben, beispielsweise bei einer Suche. Wenn keine solche Überschrift vorhanden ist wird die Seite als „Untitled Page“ bezeichnet.

Einige Aspekte des Prototypens wurden vom Design der Wiki-Server-Software von Apple[53] nachempfunden. Die Buttons für das Bearbeiten, Erstellen und Löschen von Seiten sowie ihre Anordnung wurden beispielsweise großteils eins-zu-eins übernommen (siehe Abbildung 3.16). Auch von Konzept her ist diese Software hier ein relevantes Beispiel – sie versucht ebenfalls Wikis und Weblogs in einer Oberfläche zu vereinen.

---

51. Ein Thema, das auch in den Kapiteln 2.1.4, 3.3 und 3.8 behandelt wird.
52. Siehe auch Kapitel 3.4.1
53. Ein Teil der Leopard Server Software. Siehe http://www.apple.com/server/macosx/features/wikis.html (zuletzt abgerufen am 24. Juli 2008)

Abbildung 3.16: Buttons für das Bearbeiten, Erstellen und Löschen von Seiten im Moki-Prototyp (links) und der Apple Wiki-Software (rechts; eines von vielen Themes.)

Live-Search

Einige Aufmerksamkeit wurde dem Design des Suchfeldes geschenkt. Das Suchfeld bietet die Möglichkeit einer „Live-Search“, das heißt, dass der Suchstring mit Ajax an den Server geschickt wird und mit Javascript angezeigt wird, ohne dass ein Reload der Seite notwendig wäre. In Abbildung 3.17 ist das Ergebnis einer solchen Suche zu sehen. Die Suchanfrage startet wenn nach einer Eingabe in das Suchfeld eine festgelegte Zeitspanne lang keine weitere Eingabe erfolgt ist (eine Sekunde in diesem Fall). Zu den speziellen Eigenschaften dieser Implementation einer Live-Suche gehört, dass einerseits die aktuelle besuchte Seite, so sie in den Suchergebnissen auftaucht, markiert wird (Schrift ist bold) sowie andererseits, dass der Suchstring im Browser gespeichert wird und nach dem Wechsel zu einer anderen Seite wieder aktiv ist. Dies ermöglicht es eine Menge an Suchergebnissen schrittweise durchzusehen ohne die Suche immer wieder anwerfen zu müssen oder neue Fenster dafür zu verwenden.

Ein zunächst fehlendes Merkmal der Suche, ein „Spinner“, der anzeigt dass die Suchabfrage gerade bearbeitet wird, wurde hinzugefügt, nachdem sein Fehlen als deutliches Manko empfunden wurde. Dies traf auch auf das Fehlen einer Möglichkeit das Suchfeld schnell zu löschen zu, aber dies wurde hier nicht implementiert.

Seiten-Thumbnails

Der Prototyp erzeugte Thumbnails von jeder neuen Version einer Seite. Diese wurden einerseits für die Overview-Map verwendet (siehe Abbildung 3.18, ein früherer Map-Prototyp wurde hierfür angepasst). Diese Overview-Map hatte hier allerdings nur den Aspekt eines Zusatz-Features und nicht eines zentralen Bestandteils. Sie war also nicht in die „normalen“ Ansichen des Systems eingebunden.

Die Thumbnails wurden weiters für das Navigieren in der Seiten-Versions-Geschichte verwendet (siehe Abbildung 3.19). Wie schon bei dem p2-Protoypen (Kapitel 3.5) erwähnt ist das schnelle Schalten zwischen verschiedenen Versionen eine spannende Möglichkeit Unterschiede zu erkennen, aber vielleicht nicht explizit genug für manche Anwendungsfälle. Die Möglichkeit Thumbnails zu klicken ist aber doch – meiner subjektiven Einschätzung nach – angenehmer als das Vor-/Zurück-Interface des p2-Prototypen.

Die Suchfunktion verwendete ebenfalls Seiten-Thumbnails. Wenn der Mauszeiger über einem Suchergebniss war wurde der dazu pas-

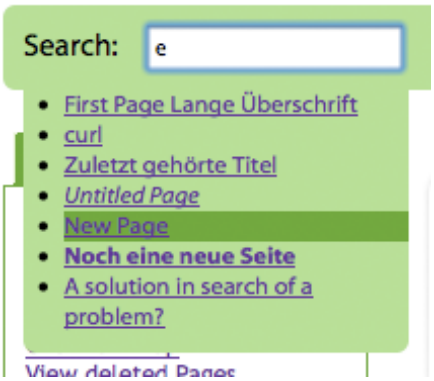


Abbildung 3.17: Screenshot der im Moki-Prototypen implementierten Live-Suche. Der Link über dem der Mauszeiger ist ist dunkelgrün hinterlegt. Der fett geschriebene Link ist die aktuelle Seite. „Untitled Page“ ist eine Seite ohne Überschrift und kursiv geschrieben um sie von einer Seite mit der Überschrift „Untitled Page“ zu unterscheiden. Möglicherweise ist hier zuviel Information als Text-Eigenschaft encodiert.

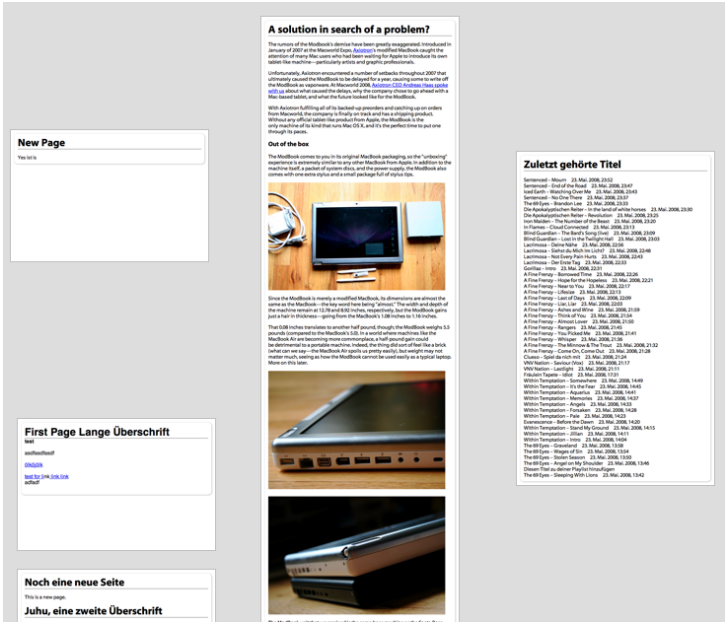


Abbildung 3.18: Overview-Map in Moki mit Thumbnails der Seiten.

Abbildung 3.19: Versionsbrowsing in Moki erfolgt über Thumbnails der Revisionen.

sende Thumbnail eingeblendet.

Implementierung

Implementiert wurde der Prototyp in django[54].Dies einerseits, weil beabsichtigt war das „webkit2png"-Skript zur Erstellung der Thumbnails zu benutzten, und dieses in Python geschrieben ist, aber vor allem weil die Hoffnung bestand, dass dieses Framework ein bißchen besser als Ruby on Rails für die Implementierung experimenteller Interfaces geeignet sein würde. Als Javascript-Framework wurde jQuery[55] verwendet. Diese Kombination war gut geeignet für diesen Prototypen und ist unter Umständen für solche Versuche weiter zu empfehlen.

Die getrennte Modellierung von Weblogs und Wikis erfüllte die Erwartungen an leichtere und „kompromisslosere" Implementierung. Es ist meine Vermutung, dass dies der einfachere Weg ist ein kombiniertes Wiki-/Weblog-System zu erstellen. Allerdings lauern hier andere Fallen: So existieren im Prototyp an vielen Stellen sehr ähnliche, aber doch nicht ganz idente Code-Stellen, die im Laufe der Zeit auseinander drifteten. Für einen Prototyps ist dies vermutlich in Ordnung, da es sich ja tendenziell um Wegwerf-Code handelt bzw. als solcher behandelt wurde. Das Ergebnis war jedenfalls, dass das Be-

54. Ein Framework in Python, Motto: „The Web framework for perfectionists with deadlines." Siehe http://www.djangoproject.com/ (zuletzt abgerufen am 24. Juli 2008)
55. http://jquery.com/ (zuletzt abgerufen am 24. Juli 2008)

arbeiten und Bedienen von Weblog-Einträgen und Wiki-Seiten sich im Laufe der Entwicklung unähnlicher wurden. Für die Wartung eines Produktiv-Systems ist dies jedoch vermutlich suboptimal.

Nahezu WYSIWYG

Die Verwendung des contentEditable-Elements erbrachte gute Ergebnisse: Zumindest in ein und demselben Browser war die Darstellung des Textes während des Bearbeitens großteils ident mit der „normalen“ Darstellung, ohne dass spezielle Stylesheets oder Hacks verwendet werden mussten.[56] Ausnahmen waren ein Rahmen, der während des Bearbeitens vom Browser rund um die contentEditable-Area gelegt wurde, sowie das Verhalten der Links, die während des Bearbeitens nicht mehr verlinkten, sondern als normaler Text zählten und somit in Links geklickt werden konnte um sie (d.h. den Anker-Text) zu bearbeiten. Eine Unterscheidung zwischen Edit- und View-Modi war damit leider weiterhin notwendig. Dies sind jedoch Unterschiede, die durchaus einen Zweck erfüllen.

contentEditable-Problem mit der Selektion

Ein technisches Problem ergab sich dadurch, dass die zur Verfügung gestellten Text-Formatierungs-Schnittstellen auf die aktuelle Selektion, beziehungsweise die aktuelle Cursor-Position, wirken. Aus der Sicht des Browsers ist allerdings die ganze Seite ein „Dokument“ und ein Klick auf eine andere Stelle in diesem Dokument löscht die vorhandene Selektion. Dieses Problem ist insofern contentEditable-spezifisch, als dass typische Rich-Text-Editoren die bearbeitbare Region in ein iFrame legen, womit sie aus Browser-Sicht tendenziell ein eigenes „Dokument“ ist.

Dieses Problem ist vielleicht dadurch lösbar, dass vor einer Interaktion mit den Text-Werkzeugen die aktuelle Selektion gespeichert wird. Es wurde versucht eine solche Lösung in diesem Prototypen zu implementieren, allerdings ohne großen Erfolg.

Die Textwerkzeuge in diesem Prototypen reagierten daher auf das „MouseDown“-Event und brachen dieses ab, so dass es für den Browser nicht zu einem „Click“-Event kam. Nichtsdestotrotz ist es sicherlich eine Schwäche des Prototypens, dass unvorsichtige Klicks knapp neben die Text-Werkzeuge die Selektion löschen und diese wiederholt werden muss. Desweiteren werden damit Drop-Down-Menüs – beispielsweise für die Auswahl eines Absatz-Stils – tendenziell recht schwierig zu implementieren. Alle Text-Werkzeuge in diesem Prototypen waren daher einfache Buttons.

Eine Möglichkeit dieses Problem zu umgehen wäre es vielleicht die Text-Werkzeuge in einem iFrame anzusiedeln. Versuche in diese Richtung wurden jedoch nicht ernsthaft unternommen, da das Verhalten der verschiedenen Browser in Bezug auf Auswahl und iFrames seltsame Unterschiede an den Tag legte.

Implementation der Thumbnail-Erstellung

Eine enge Integration der Thumbnail-Erstellung direkt in die Weblog- und Wiki-Software wurde angestrebt, getestet und wieder verworfen. Das Erstellen eines Thumbnails geht zwar mit dem er-

56. Wie beispielsweise in Kapitel 3.3 besprochen.

wähnten Python-Script viel schneller und zuverlässiger als die früheren Experimente. Allerdings immer noch zu lang um den Prozess in die Abarbeitung eines HTTP-Request einzubinden. Außerdem stellte sich heraus, dass auch mit der Verwendung von Webkit als Library weiterhin eine Verbindung zu einem Windowmanager erforderlich war.

Im Prototyp passierte die Thumbnail Erstellung also über einen zusätzlichen Prozess der eine „Queue“ mit Revisionen die Thumbnails benötigen abarbeitete. Dieser Prozess musste von einem in der graphischen Oberfläche eingeloggten Account gestartet werden. Der Einfachheit halber wurde außerdem das webkit2png-Script nicht umgeschrieben sondern von diesem Prozess mit passenden Parametern aufgerufen.

Insgesamt war der Aufwand für die Erstellung dieses Prototypens geringer als erwartet. Dies, und die Tatsache dass der Editor auch ein Interface zur Erstellung von Links benötigte, führte dazu auch noch mit diesem Interface zu experimentieren.

3.9.1 Link-Erstellung

Typische Rich-Text-Editoren verwenden für die Erstellung von Links ein Interface, das diese Links als Eigenschaft eines Text-Strings versteht (auch das technisch zugrundeliegende HTML modelliert dies so). In Abbildung 3.11 beispielsweise ist das diesbezügliche Interface von Wordpress zu sehen, dass schon relativ ausgefeilt ist. Das Prinzip ist es – in diesem und in vielen andern Fällen – zuerst den Text zu markieren, der verlinken soll (im weiteren „Link-Anker“ genannt). Danach wird der Link spezifiziert. Die Lösung über einen Button und ein Dialogfenster[57] ist häufig zu finden, aber auch andere Methoden sind denkbar, beispielsweise in der erwähnten Weblog-/Wiki-Software von Apple (siehe Abbildung 3.20).

Ein sich (zumindest dem Autor) häufig stellendes Problem in diesem Workflow entsteht, wenn vor oder während dem Schreiben der Wunsch besteht einen Link zu setzen.

Das Ziel des Links ist oft bekannt oder zumindest definitiv, daher ist die URL ein Datum das relativ fix ist. Wie der Link im Text verankert werden soll, ist während des Schreibens des Textes vielleicht noch unbekannt bzw. wird sich während des Schreibens und Bearbeitens des Textes noch ändern.

Der Workflow verlangt von der Autorin oder dem Autor jedoch zunächst den Link-Anker zu definieren und danach die URL zu spezifizieren. Spannenderweise kann die URL später noch geändert werden (was bei Tipp- bzw. Copy&Paste-Fehlern nützlich ist), jedoch nicht (oder nur in beschränkten Ausmaß) die Verankerung.

---

57. Das Dialogfenster kann vom Browser zur Verfügung gestellt sein oder wie im Fall von Wordpress dynamisch mit Javascript generiert.

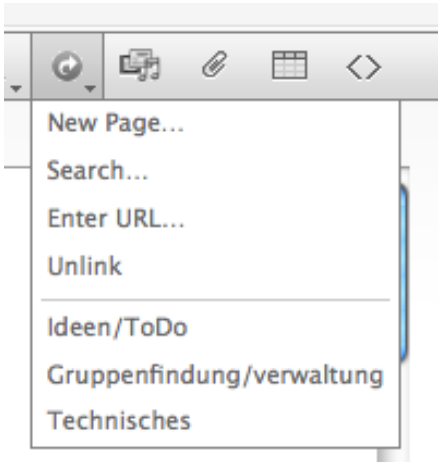


Abbildung 3.20: Menü zur Ver- und Entlinkung in der Weblog-/Wiki-Software von Apple. Hier werden einige interne Seiten gleich als Optionen geboten (unterer Teil des Menüs). Die Option „New Page“ erstellt eine neue Seite und verlinkt auf diese. Dies ist dem Workflow aus diversen Wikis ähnlich wo der Erstellung einer neuen Seite oft mit der Erstellung eines „Links“ auf ebendiese beginnt. Wenn kein Text markiert ist fügt das System den Titel der Seite oder die URL in die Seite ein.

My personal problem

Persönlich gehe ich mit diesem (meinem) Problem um indem ich alle Seiten auf die ich verlinken will in anderen Browserfenstern geöffnet lasse und häufiger als notwendig die URL kopiere. In der Tat scheine ich mich noch immer nicht von dem Impuls befreit zu haben die URL als erstes zu kopieren wenn ich das Link-Ziel im Browser habe, obwohl ich oft genug noch keine Gelegenheit habe diese URL dem Authoring-Tool zu geben, dann andere Teile (wie Titel oder einen Auszug) aus dem Link-Ziel kopiere, einfüge, und später zurückkehre um die URL ein zweites und hoffentlich letztes Mal zu kopieren. Wenn ich mich entscheide den Link-Anker zu verändern ist es unter Umständen notwendig die Kopieroperation nochmals zu wiederholen.

Mir persönlich ist dies (wie schon erwähnt) sehr lästig. Dieses Konzept der Link-Erstellung verlangt implizit extern eine Liste mit den später zu setzenden Links zu führen, wenn die Verankerung noch unklar ist. Eine Situation die beispielsweise entsteht wenn eine ansprechende Verankerung als wichtig erachtet wird beziehungsweise der Wunsch besteht verschiedene Varianten auszuprobieren.

Ein Link- und Dokument-zentriertes Link-Interface

Um diesen „personal itch“ zu behandeln wurde in diesem Prototyp ein Link-Erstellungs-Interface entworfen, dass versucht Links nicht als Text-Eigenschaft sondern als Attribut der Seite oder des Eintrags zu behandeln. Dafür wird der typische Workflow umgedreht. Nach

Abbildung 3.21: Screenshot des „Moki"-Prototypen. Der Eintrag hier wird gerade bearbeitet und hat drei Links, von denen einer zweimal im Text verankert ist. Die unverankerten Links sind unter „More Links" angeführt, und in der Liste links mit einem Fragezeichen versehen.

einem Klick auf einen Button fragt das System zunächst nach einer URL (und optional nach einer Beschreibung). Der Link ist zunächst „unverankert", hat also noch keinen Link-Anker im Haupttext.

Da ein Link ohne Link-Anker aber eher unpraktisch ist und auch um hier tatsächlich eine reale Alternative zu verankerten Links zu bieten, wird eine Darstellung verwendet, die einer real exisiterenden Weblog-Praxis[58] nachempfunden ist: Sie werden nach dem Ende bzw. am Ende des Eintrags nach einer Zwischenüberschrift „More Links" angezeigt. Als Anker-Text wird entweder die Beschreibung oder, falls diese leer ist, die URL verwendet.

Diese Anzeige ist nicht nur temporär während des Bearbeitens sondern eine tatsächliche Alternative bzw. Vorschau auf das Endergebnis.

Wenn der Link im Haupt-Text verankert werden soll kann zu jeder Zeit während des Bearbeitens ein Text-Fragment markiert werden und der entsprechende „Verankern"-Button des Links in einer getrennt geführen Liste aller Links geklickt werden. Ebenso ist es möglich den Text wieder zu „entlinken", über einen „Unlink"-Button. Ein Link der gerade keinen Anker hat wird wie erwähnt unter „More Links" aufgeführt und umgekehrt aus dieser Liste entfernt falls er einen (oder mehrere) Anker im Haupt-Text hat.

Abbildung 3.21 zeigt einen Screenshot dieses Interfaces.

Dieses Interface ist vermutlich noch nicht ausgereift[59]. Die Liste aller Links beispielsweise wirkt relativ redundant, obwohl sie hier die

---

58. Ein Beispiel von `http://lifehacker.com`: `http://lifehacker.com/5054510/battle-of-the-linux-distros` (abgerufen am 4. Oktober 2008, und aufgrund der – relativen – Kürze der URL ausgewählt.)
59. Gemeint ist der konzeptuelle Aufbau des Interfaces. Die Ästhetik ist, einem Pro-

wichtige Funktion erfüllt Überblick über alle Links zu geben, die URLs umkompliziert anzuzeigen und Links zu verankern. Kommunikation der Zuordnung eines Links aus dieser Liste zu einem Link-Anker passiert in diesem Prototyp nur in eine Richtung, nämlich über die Liste: Wenn der Mauszeiger über einem Link in der Liste ist wird der Hintergrund aller Link-Anker des Links verändert.

Dennoch, der Ansatz dieses Interface ist –meiner Meinung nach – eine verfolgenswerte Perspektive. Zugegebenermaßen ist das Gefühl eines einzelnen Menschens vielleicht eine nicht ausreichende Metrik für die Qualität eines Interfaces, aber es behandelt, wie eingangs erwähnt, ein real auftretendes Problem: Dass Link-Ziele feststehen, bevor überhaupt Text geschrieben wurde, der als Link-Anker dienen kann; dass das Authoring-System diese Information entgegennehmen könnte und sie für den oder die BenutzerIn verwalten und anzeigen könnte; dass dies aber in gängigen Systemen nicht möglich ist, und dass Änderungen am Link-Anker eher kompliziert sind, und solche Änderungen oft eine Neueingabe der URL erforderlich machen.

Schönere Anker?

Spekulation an diesem Punkt: Ein Interface, dass es ermöglicht Link-Anker nachträglich und leicht zu verändern, würde, zumindest mehr als die verbreiteten Systeme, zu einem spielerischen Umgang mit der „rethorics of departure“ einladen. Ich vermute, dass die „... findest du hier“-Rhetorik[60] eine Notlösung ist, eine unelegante aber einfache Lösung für das Problem einen geeigneten Anker zu finden.

Das entworfene Interface wurde hauptsächlich vom Autor getestet, der es als ziemlich angenehm empfand – was vermutlich nicht ganz verwunderlich ist, da es im Prinzip von und für ihn geschrieben wurde. Was der Prototyp jedoch zeigt ist, dass ein solches Interface möglich und sogar praktisch machbar ist, auch wenn, beispielsweise, die nachträgliche Bearbeitung von Links und die Verankerung „unverankerter“ Links ein Problem darstellt, dass ein textfragmentzentriertes Link-Interface nicht in diesem Ausmaß lösen muss.

3.9.2 Schlussbemerkungen zu „moki“

Was könnte aus dem „Moki“-Prototypen gelernt werden?

Zunächst, dass es tatsächlich weniger aufwendig ist ein Weblog- und ein Wiki-System konzeptuell getrennt aber in einem Framework zu realisieren, als ein generalisiertes System zu konstruieren, dass in einer sinnvollen Weise Weblogs und Wikis (und vielleicht Dinge dazwischen) abbilden kann.

Dies war eine Überraschung und hätte vielleicht keine sein müssen. Im Nachhinein betrachtet ist letzteres der Versuch ein spezielleres, aber doch sehr flexibles Framework-im-Framework zu konstruieren.

---

totyp vielleicht angemessen, in diesem Fall eher ein schneller Hack.
60. mit „hier“ als Link-Anker

Dazu kommt, dass ein solches System „Objekte“ besitzt die in hohem Ausmaß damit beschäftigt sind ihr eigenes Aussehen zu beschreiben, was gelegentlich mit der MVC-Logik der eingesetzten Web-Frameworks kollidierte.

Jedenfalls war das Moki-System vergleichsweise schneller und früher in der Lage Weblogs und Wiki-Seiten sinnvoll abzubilden. Ein strenger Vergleich ist jedoch nicht möglich, da die Prototypen zu anderen Zeiten und mit anderen technologischen Basen erstellt wurden.

Erfreulich war, dass die Technologie der heutigen Webbrowser soweit zu sein scheint, dass erstaunlich ansprechende WHYSIWYG-artige Editoren in relativ kurzer Zeit entwickelt werden können. Ob und wie schnell diese in die im Netz eingesetzen Software-Systeme Einzug halten werden, wird die Zukunft zeigen.

Die Erstellung von Thumbnails bleibt leider ein nicht ganz triviales Problem, allerdings ein durchaus lösbares. Beispielsweise setzt „delicious“[61] inzwischen Thumbnails routinemäßig ein.

---

61. http://delicious.com/ (abgerufen am 6. September 2008)

Kapitel 4

Zusammenfassung

Diese Arbeit beschreibt, nach einer (recht gerafften) Geschichte von Hypertext und einiger zugrundeliegender Prinzipien (Kapitel 2), Versuche die unternommen wurden mit dem Ziel ein kombiniertes und ansprechendes Weblog-/Wiki-System zu entwerfen (Kapitel 3).

Die Versuche und Entwürfe umspannen einen Zeitraum von mehreren Jahren. Entsprechend entwickelte sich der Stand der Technik weiter. Probleme die unlösbar, oder nur aufwendig lösbar erschienen, waren zu einem späteren Zeitpunkt relativ einfach lösbar.[1].

Overview-Map

Begonnen wurde mit der Überlegung ein solches System an einer Overview-Map auszurichten. Entsprechend waren viele frühe Prototypen daraus ausgerichtet, die Machbarkeit einer solchen Overview-Map mit real existierenden Browser-Technologien zu überprüfen. Je nachdem wie „machbar“ definiert wird, schwankt meine Einschätzung zwischen „fast machbar“ und „tatsächlich machbar“. Das Haupt-Problem ist, dass kein wirklich praktisches Konzept für das massiv kollaborative Bearbeiten einer solchen Map greifbar scheint.

Bearbeiten von Seiten

Ein zweiter Fokus der Prototypen war das Bearbeiten von Seiten-Inhalten. Hier ist das Ergebnis, dass bessere Interfaces – vor allem im Sinn eines möglichst kleinen Unterschieds in der Anzeige während und nach dem Bearbeitens – grundsätzlich möglich sind und dass der Aufwand sie zu schreiben und zu gestalten zunehmend kleiner wird. Das Hauptproblem scheint hier zu sein, dass die aktuell meist-benutzte Browser-Software im Vergleich mit den meisten anderen Systemen, in Bezug auf Konformität mit aktuellen Standards, eher hinterher hinkt. Aber auch dies könnte sich unter Umständen in der Zukunft ändern.

Link-Edit-Interface

Ein für mich in diesem Zusammenhang spannender Versuch war es ein Interface für das Hinzufügen und Verankern von Links zu gestalten, dass sich nicht mehr (wie es technisch naheliegend ist) an dem Konzept von „Link als Text-Eigenschaft“ ausrichtet. Ich persönlich finde diesen Entwurf den im WWW verbreiteten Interfaces

1. Beispielsweise das Problem des direkten Bearbeitens, siehe Kapitel 3.3 vs. Kapitel 3.9

überlegen, aber ob dies ein Konzept ist dass Verbreitung findet bleibt abzuwarten.

Bloki

Drittens war es Thema, Weblogs und Wikis in einen Software-System zu vereinen. Hier sind die Ergebnisse eher unschlüssig. Die Anforderungen an Wikis und Weblogs sind erstaunlich verschieden und der Versuch die Unterschiede zu generalisieren eher aufwendig. Für Anwendungsfälle in denen sowohl Weblogs als ein Wikis wünschenswert wären, wäre es natürlich praktisch ein System zu haben das beide Anforderungen gut abdecken kann. Der einfachere Weg ist hier aber vermutlich nicht zu generalisieren, sondern beide Funktionen explizit abzudecken. Vielleicht wird es in Zukunft Toolkits (wie die Rich-Text-Editoren) geben die es ermöglichen den „Look“ aber vor allem das „Feel“ ähnlich zu gestalten. Vielleicht ist es aber auch nur eine Frage der Zeit, bis solche Systeme breiter verfügbar sind.

Anmerkungen

(A), (2.1.2): Es liegt nicht im Bereich dieser Arbeit auf die Politik von Hypertext einzugehen, trotzdem will ich anmerken, dass es im Licht der Konflikte der jüngeren Vergangenheit um Copyright und Filesharing unwahrscheinlich wirkt, dass Inhalte nur von Copyright-InhaberInnen oder dazu autorisierten Personen in ein globales und verteiltes Hypertext-Netzwerk eingespeist werden.

Der Xanadu-Entwurf sieht nicht vor, dass Inhalte entfernt werden können, als Vorbedingung für das zuverlässige Funktionieren von Links. Ob die RechteinhaberInnen diese Einschränkung akzeptieren würden ist fraglich. Zusätzlich zu anderen Anforderungen müsste dieses „ideale“ Hypertext-System also auch zensur-resistent sein.

(B), (2.1.2): Budd (2005) charakterisiert „Open Data“ folgendermaßen:

- *Open Data Formats*
- *No data lock-in or walled gardens*
- *User created data*
- *User owns their own data*
- *Ability to use data outside the confines of the application*
- *Data used across devices*

Andererseits weißt O'Reilly (2005) darauf hin, dass es im Interesse von (Web 2.0-)Firmen liegt, bestimmte Klassen von „*core data*: location, identity, calendaring of public events, product identifiers and namespaces.“ zu „besitzen“, und sagt vorher:

> „ Much as the rise of proprietary software led to the Free Software movement, we expect the rise of proprietary databases to result in a Free Data movement within the next decade.“

und dass die Frage „Who owns the data?“ von hoher Wichtigkeit werde. Umgekehrt weißt Felten (2008) darauf hin, dass „ownership“ und „property“ möglicherweise unpassende und wenig hilfreiche Begriffe in dieser Debatte sind.

(C), (2.1.3): Was genau Landow mit „Lexia“ gemeint ist bleibt jedoch etwas unklar.

> „*Hypertext*, as the term is used in this work, denotes text composed of blocks of text — what Bartes terms a *lexia* — and the electronic links that join them.“. (Landow 2006, Seite 3, Hervorhebung im Original)

Was Lexia von andern Textfragmenten unterscheidet ist nicht unbedingt die Möglichkeit, auf ein Lexia zu linken, da auch die Möglichkeit nur auf bestimmte Teile von Lexias zu linken angesprochen wird. Meine Vermutung ist, dass das Wort in etwa in der Bedeutung von „Texteinheit, die als ein Ganzes (am Bildschrim) präsentiert wird“ verwendet wird.

(D), (2.1.3): Schwierigkeiten in Hypertext Grenzen festzulegen, kämen vielleicht nicht ganz unerwartet. Landow (2006) schreibt: „Hypertext redefines not only beginnings and endings of text but also its borders—its sides, as it were“ (Seite 113), und weiter: „Hypertext both on the Internet and in its read-write forms thus creates an open, open-bordered text, a text that cannot shut out other texts and therefore embodies the Derridean text that blurs ‚all those boundaries that form the running border of what used to be called a text, of what we once thought this word could identify, i.e., the supposed end and beginning of a work, the unity of the corpus, the title, the margins, the signatures, the referential realm outside the frame, and so forth.‘“ (Seite 114)

(E), (2.2.2): Auch twoday könnte theoretisch als Open-Source bezeichnet werden, denn 2005 wurde Quellcode für twoday unter einer BSD-Lizenz veröffentlicht (Siehe beispielsweise http://twoday.tuwien.ac.at/info/stories/15923/). Allerdings scheint kein Open-Source-Entwicklungsprozess stattgefunden zu haben. Die „Latest Version“ laut http://twoday.org/wiki/ ist immer noch „1.0.0 beta“ (März 2008). Es scheint korrekter twoday als ein Closed-Source-Produkt zu bezeichnen, insbesondere da mit „twoday“ in dieser Arbeit hauptsächlich das von der Firma Knallgrau für die TU Wien bereitgestellte System gemeint ist.

Abbildungsverzeichnis

Literaturverzeichnis

Atzenbeck und Nürnberg 2005
ATZENBECK, Claus ; NÜRNBERG, Peter J.: Constraints in spatial structures. In: *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM Press, 2005, S. 63–65. – ISBN 1-59593-168-6. DOI: 10.1145/1083356.1083368 *(referenziert auf p. 74)*

Berners-Lee u. a. 2005
BERNERS-LEE, T. ; FIELDING, R. ; MASINTER, L.: *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard). Januar 2005. – URL http://www.ietf.org/rfc/rfc3986.txt. – Zugriffsdatum: 2008.09.11 *(referenziert auf p. 70)*

Berners-Lee und Fischetti 2000
BERNERS-LEE, Tim ; FISCHETTI, Mark: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. First Paperback Edition. HarperColins Publishers Inc., New York, NY, 2000 *(referenziert auf p. 5, 6, 7, 15, and 16)*

Bernstein 2006
BERNSTEIN, Mark: *The Tinderbox Way*. Eastgate Systems, Inc, Watertown MA, USA, 2006. – ISBN 1-8845-1146-5 *(referenziert auf p. 9, 12, and 74)*

Blood 2004
BLOOD, Rebecca: How blogging software reshapes the online community. In: *Communications of the ACM* 47 (2004), Nr. 12, S. 53–55. – ISSN 0001-0782. DOI: 10.1145/1035134.1035165 *(referenziert auf p. 24)*

Bright 2008
BRIGHT, Peter: *IE8 Beta 2 shows Microsoft is serious about playing catch-up*. August 2008. – URL http://arstechnica.com/news.ars/post/20080828-ie8-beta-2-shows-microsoft-is-serious-about-playing-catch-up.html. – Zugriffsdatum: 7. September 2008 *(referenziert auf p. 65)*

Budd 2005
BUDD, Andy: *What is Web 2.0?* 2005. – URL http://www.andybudd.com/presentations/dcontruct05/. – Zugriffsdatum: 2008.09.11 *(referenziert auf p. 8 and 92)*

Bush 1945
BUSH, Vannevar: As We May Think. In: *The Atlantic Monthly* (1945), Juli. – URL http://www.theatlantic.com/doc/194507/bush. – Zugriffsdatum: 2008.09.11 *(referenziert auf p. 4, 5, and 10)*

Chi u. a. 2006
CHI, Yun ; TSENG, Belle L. ; TATEMURA, Junichi: Eigen-trend: trend analysis in the blogosphere based on singular value decompositions. In: *CIKM '06: Proceedings*

of the 15th ACM international conference on Information and knowledge management. New York, NY, USA : ACM, 2006, S. 68–77. – ISBN 1-59593-433-2. DOI: 10.1145/1183614.1183628 *(referenziert auf p. 25)*

Davis 1998

DAVIS, Hugh C.: Referential integrity of links in open hypermedia systems. In: *HYPERTEXT '98: Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space—structure in hypermedia systems*. New York, NY, USA : ACM, 1998, S. 207–216. – ISBN 0-89791-972-6. DOI: 10.1145/276627.276650 *(referenziert auf p. 7 and 11)*

Désilets u. a. 2005

DÉSILETS, Alain ; PAQUET, Sébastien ; VINSON, Norman G.: Are wikis usable? In: *WikiSym '05: Proceedings of the 2005 international symposium on Wikis*. New York, NY, USA : ACM, 2005, S. 3–15. – ISBN 1-59593-111-2. DOI: 10.1145/1104973.1104974 *(referenziert auf p. 31 and 32)*

Ebersbach u. a. 2006

EBERSBACH, Anja ; GLASER, Markus ; HEIGL, Richard: *Wiki: Web Collaboration*. Springer-Verlag Berlin Heidelberg, 2006. – ISBN 3-540-25995-3 *(referenziert auf p. 35)*

Felten 2008

FELTEN, Ed: *Scoble/Facebook Incident: It's Not About Data Ownership*. Jan. 2008. – URL `http://www.freedom-to-tinker.com/?p=1246`. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 92)*

Fielding u. a. 1999

FIELDING, R. ; GETTYS, J. ; MOGUL, J. ; FRYSTYK, H. ; MASINTER, L. ; LEACH, P. ; BERNERS-LEE, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Juni 1999 (Request for Comments). – URL `http://www.ietf.org/rfc/rfc2616.txt`. – Updated by RFC 2817 *(referenziert auf p. 16 and 63)*

Frank M. Shipman und Marshall 1999

FRANK M. SHIPMAN, III ; MARSHALL, Catherine C.: Spatial hypertext: an alternative to navigational and semantic links. In: *ACM Computing Surveys* 31 (1999), Nr. 4es. – ISSN 0360-0300. DOI: 10.1145/345966.346001 *(referenziert auf p. 40, 41, and 42)*

Frank M. Shipman u. a. 1995

FRANK M. SHIPMAN, III ; MARSHALL, Catherine C. ; MORAN, Thomas P.: Finding and using implicit structure in human-organized spatial layouts of information. In: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995, S. 346–353. – ISBN 0-201-84705-1. DOI: 10.1145/223904.223949 *(referenziert auf p. 42)*

Haake u. a. 2005

HAAKE, Anja ; LUKOSCH, Stephan ; SCHÜMMER, Till: Wiki-templates: adding structure support to wikis on demand. In: *WikiSym '05: Proceedings of the 2005 international symposium on Wikis*. New York, NY, USA : ACM, 2005, S. 41–51. – ISBN 1-59593-111-2. DOI: 10.1145/1104973.1104978 *(referenziert auf p. 36)*

Hartmann u. a. 2007

HARTMANN, Jan ; SUTCLIFFE, Alistair ; ANGELI, Antonella D.: Investigating attractiveness in web user interfaces. In: *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2007, S. 387–396. – ISBN 978-1-59593-593-9. DOI: 10.1145/1240624.1240687 *(referenziert auf p. 33)*

Herring u. a. 2004
HERRING, Susan C. ; SCHEIDT, Lois A. ; BONUS, Sabrina ; WRIGHT, Elijah: Bridging the Gap: A Genre Analysis of Weblogs. In: *HICSS '04: Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0-7695-2056-1. DOI: 10.1109/HICSS.2004.1265271 *(referenziert auf p. 25 and 39)*

Iorio und Zacchiroli 2006
IORIO, Angelo D. ; ZACCHIROLI, Stefano: Constrained Wiki: an Oxymoron? In: *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*. New York, NY, USA : ACM, 2006, S. 89–98. – ISBN 1-59593-413-8. DOI: 10.1145/1149453.1149471 *(referenziert auf p. 35 and 36)*

Kolb 2001
KOLB, David: *Places and Spaces: Adjacency Effects*. First Workshop on Spatial Hypertext. 2001. – URL `http://www.csdl.tamu.edu/~shipman/SpatialHypertext/SH1/kolb.pdf`. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 41)*

Kumar u. a. 2003
KUMAR, Ravi ; NOVAK, Jasmine ; RAGHAVAN, Prabhakar ; TOMKINS, Andrew: On the bursty evolution of blogspace. In: *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA : ACM, 2003, S. 568–576. – ISBN 1-58113-680-3. DOI: 10.1145/775152.775233 *(referenziert auf p. 26 and 39)*

Landow 2006
LANDOW, George P.: *Hypertext 3.0: Critical Theory and New Media in an Era of Globalization*. The John Hopkins University Press, Baltimore, 2006. – ISBN 0-8018-8257-5 *(referenziert auf p. 3, 4, 5, 7, 10, 11, 12, 13, 27, 41, 42, 92, and 93)*

Leuf und Cunningham 2005
LEUF, Bo ; CUNNINGHAM, Ward: *The Wiki Way: Quick Collaboration on the Web*. 5. Auflage. Addision-Wesley, Boston, Mass., USA, 2005 *(referenziert auf p. 30, 31, and 32)*

Marlow u. a. 2006
MARLOW, Cameron ; NAAMAN, Mor ; BOYD, Danah ; DAVIS, Marc: HT06, tagging paper, taxonomy, Flickr, academic article, to read. In: *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2006, S. 31–40. – ISBN 1-59593-417-0. DOI: 10.1145/1149941.1149949 *(referenziert auf p. 15)*

Marshall und Rogers 1992
MARSHALL, Catherine C. ; ROGERS, Russell A.: Two years before the mist: experiences with Aquanet. In: *ECHT '92: Proceedings of the ACM conference on Hypertext*. New York, NY, USA : ACM, 1992, S. 53–62. – ISBN 0-89791-547-X. DOI: 10.1145/168466.168490 *(referenziert auf p. 41)*

Millard und Ross 2006
MILLARD, David E. ; ROSS, Martin: Web 2.0: hypertext by any other name? In: *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*. New York, NY, USA : ACM Press, 2006, S. 27–30. – ISBN 1-59593-417-0. DOI: 10.1145/1149941.1149947 *(referenziert auf p. 7, 9, 14, and 38)*

Munroe 2007
MUNROE, Randall: *The Problem with Wikipedia*. Jan. 2007. – URL `http://xkcd.com/214/`. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 65)*

Nelson 1987

NELSON, Theodor H.: *Literary Machines.* Edition 87.1. South Bend, IN : The Distributors, 1987. – ISBN 0-89347-055-4 *(referenziert auf p. 3, 5, 6, 41, and 43)*

Nielsen 1990
NIELSEN, Jakob: *Hypertext and Hypermedia.* Academic Press, Inc., 1990. – ISBN 0-12-518410-7 *(referenziert auf p. 4, 5, 6, 15, and 41)*

Nielsen 2004
NIELSEN, Jakob: *Guidelines for Visualizing Links.* May 2004. – URL http://www.useit.com/alertbox/20040510.html. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 9)*

Nielsen 2005
NIELSEN, Jakob: *Reviving Advanced Hypertext.* Jan. 2005. – URL http://www.useit.com/alertbox/20050103.html. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 42)*

Norman 2004
NORMAN, Donald A.: *Emotional Design - Why we love (or hate) everyday things.* Basic Books, New York, 2004. – ISBN 0-465-05135-9 *(referenziert auf p. 32)*

O'Reilly 2005
O'REILLY, Tim: *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.* September 2005. – URL http://oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 8 and 92)*

Pohl und Purgathofer 2004
POHL, Margit ; PURGATHOFER, Peter: Hypertext Writing Profiles and Visualisation. In: *Computer and the Humanities* Volume 38 (2004), February, Nr. Number 1, S. 83–105. DOI: 10.1023/B:CHUM.0000009226.51168.47 *(referenziert auf p. 44)*

Russell 2006
RUSSELL, Alex: *Comet: Low Latency Data for the Browser.* März 2006. – URL http://alex.dojotoolkit.org/?p=545. – Zugriffsdatum: 2008.05.09 *(referenziert auf p. 61)*

Sand-Jensen 2007
SAND-JENSEN, Kaj: How to write consistently boring scientific literature. In: *Oikos* 116 (2007), May, Nr. 5, S. 723–727. – ISSN 0030-1299. DOI: 10.1111/j.2007.0030-1299.15674.x *(referenziert auf p. 1 and 2)*

m. c. schraefel 2007
SCHRAEFEL m. c.: What is an analogue for the semantic web and why is having one important? In: *SIGWEB Newsl.* 2007 (2007), Nr. Winter, S. 6. – ISSN 1931-1745. DOI: 10.1145/1324960.1324966 *(referenziert auf p. 74)*

Smith 1997
SMITH, John B.: The king is dead; long live the king (keynote). In: BERNSTEIN, Mark (Hrsg.) ; ØSTERBYE, Kasper (Hrsg.) ; CARR, Leslie (Hrsg.): *HYPERTEXT '97: Proceedings of the eighth ACM conference on Hypertext.* New York, NY, USA : ACM Press, 1997, S. 240. – ISBN 0-89791-866-5. DOI: 10.1145/267437.270924 *(referenziert auf p. 6)*

Tractinsky u. a. 2000
TRACTINSKY, N. ; KATZ, A. ; IKAR, D.: What is beautiful is usable. In: *Interacting with Computers* 13 (2000), December, Nr. 2, S. 127–145. DOI: 10.1016/S0953-5438(00)00031-X *(referenziert auf p. 33)*

Walker 2005
WALKER, Jill: Feral hypertext: when hypertext literature escapes control. In: *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hy-*

permedia. New York, NY, USA : ACM, 2005, S. 46–53. – ISBN 1-59593-168-6. DOI: 10.1145/1083356.1083366 *(referenziert auf p. 7)*

Wardrip-Fruin 2004

WARDRIP-FRUIN, Noah: What Hypertext Is. In: *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2004, S. 126–127. – ISBN 1-58113-848-2. DOI: 10.1145/1012807.1012844 *(referenziert auf p. 4)*

Wroblewski und Wood 2005

WROBLEWSKI, Luke ; WOOD, Jed: *Blog Interface Design 2.0*. 2005. – URL `http://www.lukew.com/resources/articles/blogs2.asp`. – Zugriffsdatum: 2008.09.12 *(referenziert auf p. 48)*

Zhang 2006

ZHANG, Yuejiao: Wiki means more: hyperreading in Wikipedia. In: *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2006, S. 23–26. – ISBN 1-59593-417-0. DOI: 10.1145/1149941.1149946 *(referenziert auf p. 13 and 31)*

Zhou u. a. 2008

ZHOU, Dong ; TRURAN, Mark ; BRAILSFORD, Tim ; ASHMAN, Helen ; POURABDOLLAH, Amir: Llama-b: automatic hyperlink authoring in the blogosphere. In: *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2008, S. 133–138. – ISBN 978-1-59593-985-2. DOI: 10.1145/1379092.1379119 *(referenziert auf p. 25)*