Publishing

Version 2007-08-30/1
Arbeitspaket AP 1
verantwortlicher Partner: FH Worms

TextGrid

Modulare Plattform für verteilte und kooperative wissenschaftliche Textdatenverarbeitung - ein Community-Grid für die Geisteswissenschaften

Bundesministerium für Bildung und Forschung

Projekt: **TextGrid**

Teil des D-Grid Verbundes und der deutschen e-Science Initiative

BMBF Förderkennzeichen: 07TG01A-H

Laufzeit: Februar 2006 - Januar 2009

Dokumentstatus: final

Verfügbarkeit: öffentlich

Autoren:

Marc W. Küster, FH Worms

Christoph Ludwig, FH Worms

1 Einleitung

Publikationen von wissenschaftlichen Editionen erfolgen heute meist in einer oder mehreren von drei Formen: In traditioneller Weise als Print-Edition, als CD-ROM- bzw. DVD-Edition sowie zunehmend auch als Online- oder Web-Edition. Aus Sicht von TextGrid sind hierbei besonders die Print- und die Web-Publikation von Interesse, weil beide unmittelbar aus den in TextGrid gespeicherten Daten erzeugt werden sollen. Bei CD-ROM-Editionen werden die Daten zwangsläufig aus TextGrid exportiert und in ein für „TextGrid-fremde“ Programme geeignetes Format überführt. Weil TextGrid keinen Einfluss auf die Möglichkeiten oder Einschränkungen dieser Programme von dritter Seite hat, beschränkt sich dieser Report auf die Print- und die Web-Publikation.

Aufbereitung von Texten für den Druck ist seit langem eine Kernaufgabe der elektronischen Datenverarbeitung. Von jeher stand diese Tätigkeit dabei im Spannungsfeld zweier ursprünglich gänzlich unterschiedlicher Anforderungen: der Büroarbeit, deren visuelle Erwartungen seit einem Jahrhundert durch die Schreibmaschine geprägt war und nur geringe typographische Anforderungen stellte, und dem eigentlichen Satz, dessen Ziel der hochwertige Druck war und ist und sich damit in eine fünfhundertjährige Tradition der Gutenberg Galaxis [16] einreiht. Früh kam noch ein drittes Feld hinzu, das in dieser Form keine Entsprechung außerhalb der EDV hat: der vollautomatische Satz vieler strukturell gleicher Geschäftsdokumente wie Rechnungen auf der Basis großer Datenbanken.

Die Geschichte der Textverarbeitungs- und -satzsysteme lässt sich in zwei Technologieansätze aufteilen, die auch heute noch die wissenschaftliche Textdatenverarbeitung prägen:

1. Batchsysteme: Systeme, die aus einer geeignet ausgezeichneten Eingabe- automatisch eine Ausgabedatei setzen;
2. WYSIWYG-Systeme: Umgebungen, die es erlauben, einen Text interaktiv und mit sofortiger visueller Rückmeldung für den Satz aufzubereiten;

Gleichzeitig konzentrieren wir uns hier auf die für TextGrid besonders relevanten Anforderungen, insbesondere auf den Satz von kritischen Editionen inklusive mehrerer Apparate und Paralleltexten, auf Multilingualität und auf Texte mit hohem Formelanteil, natürlich keine einander ausschließenden Kriterien. Mit Blick auf eine mögliche Nutzung innerhalb TextGrids haben wir den Schwerpunkt bei der Evaluation verfügbarer Bibliotheken und Anwendungen auf Open Source Software gelegt.

Eine übergreifende Darstellung des aktuellen Forschungsstands wird dabei dadurch kompliziert, dass die Interna der meisten aktuellen Textverarbeitungsprogramme nicht publiziert werden. Programme wie Microsoft Word, WordPerfect oder OpenOffice dominieren den heutigen Büro- und zu einem nicht geringen Teil Wissenschaftsalltag, aber selbst über die Programmarchitektur eines Open-Source-Systems wie OpenOffice findet man nur wenig wissenschaftliche – oder auch nichtwissenschaftliche – Literatur, von aktuellen kommerziellen Systemen ganz zu schweigen. Notwendig muss daher diese Darstellung einen gewissen referierenden Charakter haben: aktuelle Fähigkeiten von Systemen werden mit den Anforderungen (text-)wissenschaftlichen Satzes kontrastiert.

In den Kapiteln 4 und 5 werden dem die Anforderungen an Online-Editionen gegenübergestellt und abschließend skizziert, wie eine Web-Publikation mit Software der Apache Foundation realisiert werden kann.

2 Anforderungen an das Print-Publishing

2.1 Nutzerschnittstelle

Anders als weit verbreitete Office- oder Desktop Publishing Programme (DTP) muss die Nutzeroberfläche eines Publishing-Moduls speziell die Arbeitsabläufe beim wissenschaftlichen Textsatz unterstützen. Dies bedeutet unter anderem, dass die Formatierung konsequent auf der nutzerdefinierten Auszeichnung des Textes in XML aufbaut und ein ergonomisches Arbeiten mit mehreren Apparaten, Formelsatz, komplexen Schriften, Sonderzeichen, die auch im Unicode Zeichensatz nicht enthalten sind, usw. ermöglicht.

Für technisch versierte Anwender mit Programmiererfahrung ist es üblicherweise kein Problem, ein Programm im Batch-Betrieb einzusetzen; d. h. der Anwender schreibt die Eingabedaten in einem Editor seiner Wahl, speichert diese auf dem Massenspeichermedium und übergibt anschließend diese Daten an das Satzprogramm. Das geplante Satzsystem sollte den Batch-Betrieb auch auf jeden Fall als Option vorsehen, weil er eine einfache Möglichkeit bietet, den Satz z. B. in TextGrid-Workflows einzubinden.

Die Mehrzahl der Anwender erwartet heute allerdings eine graphische Oberfläche. Dieses GUI muss dem Nutzer eine Vorschau auf die endgültige Ausgabe bieten; bei Änderungen der Eingabedaten muss diese Vorschau umgehend aktualisiert werden (WYSIWYG). Hier ist also eine enge Integration in die TextGrid Nutzeroberfläche (Eclipse RCP) erforderlich.

Ein auf die Auszeichnung von Textdaten spezialisierter XML-Editor wird ohnehin Teil der TextGrid RCP sein; für das Publishing-Modul wird darüber hinaus noch ein Style-Editor benötigt, der die Formatierung verschieden ausgezeichneter Teile der Eingabe festlegt. Ferner muss das GUI die manuelle Korrektur des Satzes ermöglichen, wobei eine strikte Trennung von diesen rein visuellen Formatierungsanweisungen und der durch die inhaltliche Auszeichnung bedingten Formatierung zu beachten ist (z. B. indem die Korrekturen ausschließlich als *standoff annotation* gespeichert werden).

2.2 Typographische Anforderungen (Print)

Der Satz wissenschaftlicher Texte für den Druck bringt hohe und teils auch sehr fachspezifische Anforderungen mit sich. Die häufigsten Anforderungen sind:

Komplexe Schriften: In vielen wissenschaftlichen Arbeiten werden Texte aus verschiedenen Sprachen kombiniert – z. B. in den Haupttext eingebettete Zitate, freigestellte längere Exzerpte oder eine in einer eigenen Spalte gesetzte Übersetzung. Abgesehen von den typographischen Gepflogenheiten (etwa bzgl eines zusätzlichen Abstandes nach einem Satzende), die abhängig von der Sprache oder noch genauer vom jeweiligen Kulturkreis sind, bedeutet dies, dass in diesen Arbeiten verschiedene Schriftsysteme vorkommen. Um mit diesen umgehen zu können, muss ein zeitgemäßes Satzprogramm eine vollständige Unterstützung des Unicode-Standards [35] bieten sowie pluridirektionalen Satz auch innerhalb eines Absatzes unterstützen.

OpenType Fonts: Um die verschiedenen Schriften und ihre diversen Schnitte in ansprechender Weise zu Papier zu bringen, bedarf es qualitativ hochwertiger Unicode-Fonts. Deshalb muss ein modernes Satzsystem OpenType Fonts verwenden können und die in diesen Fonts hinterlegten Informationen für einen optimalen Satz nutzen. Die Unterstützung aller von den Fonts definierten Schriftschnitte ist eine Selbstverständlichkeit.

Sonderzeichen außerhalb Unicode: Obwohl die Zahl der in Unicode bereits enthaltenen Zeichen enorm ist und auch immer wieder zusätzliche Zeichen in den Standard aufgenommen werden, besteht in kritischen Editionen, die z. B. mittelalterliche Handschriften oder historische

Wörterbücher transkribieren, häufig der Bedarf, zusätzliche Zeichen wiederzugeben. Unicode sieht u. a. für solche Fälle *private use areas* vor. Das Satzprogramm muss deshalb die Möglichkeit bieten, solche privaten Codepoints auf spezielle, vom Anwender bereitgestellte Fonts abzubilden und, sofern notwendig, zusätzliche Normalisierungsregeln [35] (Annex 15) zu vereinbaren.

Apparate und Kommentare: In textwissenschaftlichen und historischen Editionsprojekten ist es eher die Regel als die Ausnahme, dass mehrere Apparate – ein Erklärungsapparat, ein Variantenapparat usw. – verwaltet und gesetzt werden müssen. Grundsätzlich können die Apparate in einem eigenen Anhang, am Kapitelende, am Seitenende oder in den Marginalien gesetzt werden und die Anwender wollen die Entscheidung, wo ein Apparat erscheint, auch noch gegen Ende des Editionsprojektes revidieren können. Vor allem die beiden letzten Varianten, am Seitenende und in den Marginalien, stellen das Satzsystem vor Herausforderungen, weil dadurch mehrere Flüsse auf jeder Seite gesetzt werden müssen, deren Umbruch nicht unabhängig voneinander optimiert werden kann.

Paralleltexte / synoptischer Satz: Die Komplexität des Umbruchs steigt noch weiter beim parallelen bzw. synoptischen Satz, bei dem z. B. Übersetzungen auf gegenüberliegenden Seiten oder Varianten eines Textes in mehreren Spalten so gesetzt werden, dass korrespondierende Absätze auf gleicher Höhe beginnen.

Referenzen: In längeren Texten besteht regelmäßig die Notwendigkeit, auf andere Textstellen zu verweisen. Die entsprechenden Seitenzahlen und Kapitelnummern müssen vom Satzsystem automatisch generiert oder – abhängig vom tatsächlichen Umbruch – vielleicht auch durch einen lokalisierbaren String (etwa „auf der vorherigen Seite“) ersetzt werden. Auch Abbildungs-, Tabellen-, Diagrammnummern o. ä. müssen automatisch verwaltet werden, wobei die Zählung evtl. in jedem (Unter-) Kapitel neu begonnen werden soll und je nach Anwenderwunsch verschiedene Abbildungstypen separat oder gemeinsam gezählt werden.

Manche Referenzen verweisen direkt auf die Seite, auf der sich das betreffende Objekt befindet. Viel häufiger noch werden aber die potentiellen Verweisziele durchnummeriert: Beispielsweise würde man die erste Abbildung in einem Text typischerweise als Abb. 1 referenzieren, auf Fußnoten oder Apparate wird durch eine hochgestellte Zahl oder ein systematisch gewähltes Zeichen verwiesen. Die bevorzugten Methoden, wie Literaturreferenzen im Text formuliert und formatiert werden, sind zahllos und extrem abhängig von der jeweiligen Fachdisziplin. Jede Einschränkung, wie diese Zählung vorgenommen wird, würde die Akzeptanz des Satzsystemes schmälern. Auch die Methode, die benutzt wird, um die Nummer eines Objekts in ein geeignetes Label umzusetzen (als Dezimalzahl in arabischen Ziffern, als römische Zahl, durch die Buchstaben eines Alphabets, durch eine vorgegebene Folge von Sonderzeichen o. ä.), muss vom Anwender frei definiert werden können. Die Erstellung der Referenzlabels kann nicht unabhängig vom Satzfall erfolgen, weil beispielsweise Fußnoten häufig auf jeder Seite neu gezählt werden.

Vers- / Zeilenzählung: Bei wissenschaftlichen Editionen von literarischen Texten ist oftmals eine automatische Zählung der Zeilen oder der Verse gewünscht, um den Lesern das Referenzieren einer bestimmten Stelle zu erleichtern. Auch diese Zählung muss bei jedem neuen Stück neu begonnen werden können.

Hierarchische Register: Ein Versuch, alle Referenztypen abschließend aufzulisten, wäre ebenfalls zum Scheitern verurteilt, weil auch hier die Gepflogenheiten in hohem Maße von der jeweiligen Fachdisziplin und dem Gegenstand der Betrachtung abhängen. Das Satzsystem muss deshalb die Definition beliebiger Referenzkategorien zulassen. Um die Erstellung von verschieden granularen Inhalts-, Abbildungs-, oder Algorithmenverzeichnisse zu ermöglichen, sollten diese Kategorien hierarchisch angeordnet werden können. Auch in den Registern erwarten die Anwender die Möglichkeit, die Einträge hierarchisch zu untergliedern.

Formelsatz: Beim Satz von Formeln erwarten Anwender wenigstens die Qualität von TeX/LaTeX. Auch wenn sich in den mathematisch-naturwissenschaftlichen Fächern die TeX-Syntax für die Notation von Formeln in Textdateien durchgesetzt hat, wird man in TextGrid, das von einer XML-Auzeichnung ausgeht, das vergleichbar mächtige MathML nutzen.

Tabellen: Eine übersichtliche Präsentation von Daten erfordert häufig den Satz komplexer Tabellen. Diese Tabellen sind mitunter so groß, dass sie über mehrere Seiten verteilt oder im Querformat gesetzt werden müssen. Das Satzsystem muss deshalb in der Lage sein, Tabellen umzubrechen und die entsprechenden Kopf- und Fußzeilen einzufügen. Um sehr breite Tabellen ausgeben zu können, sollte das Satzsystem sowohl in der Lage sein, einzelne Objekte zu drehen, als auch die Seitenparameter einzelner Ausgabeseiten zu manipulieren.

Gleitobjekte: Größere Objekte wie Abbildungen sollen oder können zur Vermeidung eines ungünstigen Umbruchs regelmäßig nicht genau an der Stelle im Ausgabestrom erscheinen, an der sie in der Eingabe stehen. Statt dessen werden die Objekte z. B. an einen oberen oder unteren Seitenrand oder gar auf eine eigene Seite ohne Text verschoben.

Die existierenden Satzprogramme treffen diese Umbruchentscheidungen immer nur anhand lokal vorhandener Informationen: Es werden nur diejenigen Elemente auf der aktuellen (Doppel-)Seite ausgegeben, die unter Berücksichtigung aller Restriktionen gesetzt werden können; auf der nächsten Seite werden die übertragenen Elemente mit Priorität gesetzt. Eine Revision des Umbruchs auf bereits ausgegebenen Seiten ist nicht vorgesehen. Dadurch lässt sich häufig ein Wasserfalleffekt beobachten: Eine kleine Änderung am Anfang eines Dokuments, etwa das Verkürzen eines Absatzes um eine Zeile, bewirkt einen völlig verschiedenen Satzfall mehrere Seiten weiter hinten, weil z. B. Abbildungen auf andere Seiten rutschen.

Deshalb ist ein Satzprogramm, wünschenswert das den Umbruch aufgrund globaler Informationen festlegt und deshalb ein insgesamt besseres Ergebnis erzielen kann, indem es auf einer vorderen Seite einen geringfügig weniger ansprechenden Umbruch in Kauf nimmt, dafür aber auf den Folgeseiten den Zielvorgaben sehr viel besser gerecht wird. Die Zielfunktion, die hierbei optimiert wird, muss zwangsläufig einen Kompromiss zwischen ästhetischen und ergonomischen Ansprüchen einerseits und dem für die Optimierung notwendigen Rechenaufwand darstellen. Tatsächlich sind effiziente Algorithmen, welche die Position dieser Gleitobjekte über mehrere Seiten hinweg optimieren, nach wie vor Gegenstand der Forschung, vgl. etwa [4].

Verankerte Objekte, Kontursatz: Speziell kleinere Graphiken etc. werden häufig nicht als Gleitobjekte gesetzt, sondern relativ zu anderen Objekten. Ein als Graphik eingefügtes Initial beispielsweise muss immer am Anfang des entsprechenden Absatzes gesetzt werden. Oder ein Bild soll direkt neben dem erklärenden Text eingefügt und von diesem im Kontursatz umflossen werden. In wissenschaftlichen Texten hat der Kontursatz vermutlich eine eher nachgeordnete Bedeutung, zumindest verglichen mit dem Satz von Werbematerialien u. ä. Dennoch ist wenigstens der Kontursatz von Objekten wünschenswert, deren Umriss durch einfache geometrische Formen (regelmäßiges *n*-Eck usw.) dargestellt werden kann.

Externe Graphiken: Auch (und gerade) in wissenschaftlichen Arbeiten werden nicht alle Informationen verschriftlicht: Stemmata veranschaulichen die Überlieferungsgeschichte eines Textes, Landkarten vermitteln geographische Informationen, Manuskripte werden als Faksimile abgedruckt usw. Deshalb ist es notwendig, dass das Satzsystem externe Graphiken einbinden kann, sowohl in Bitmap- (z. B. BMP, PNG, TIFF, JPEG) als auch Vektorgraphikformaten (z. B. SVG, EPS). Auch rudimentäre Transformationen der Bilder wie Skalierung, Beschneidung, Rotation oder Überlagerung wird von den Anwendern üblicherweise erwartet.

In wissenschaftlichen Arbeiten kommt der Farbtreue beim Druck von Faksimiles oder von Fotografien eine besondere Bedeutung zu. Falls das Format der eingebundenen Graphiken keine Informationen über den zu Grunde liegenden Farbraum enthält, so muss das Satzprogramm eine externe Farbraumbeschreibung an die eigentlichen Rendering Komponente durchreichen können.

Bedingte Formatierung: Formatierungsregeln in Abhängigkeit vom Kontext, der sich durch den Satzfall ergibt. Durch solch eine Regel soll es z. B. möglich sein, bei Absätzen, die im Druck direkt unterhalb eines Gleitobjektes beginnen, automatisch auf den sonst üblichen Einzug der ersten Zeile zu verzichten.

2.3 Ausgabeformate

Das Ziel eines Satzsystems für Print-Publishing ist es selbstverständlich, das Dokument zu Papier zu bringen. Doch anders als noch zu den Zeiten, als Systeme wie TeX oder *roff entstanden, ist es dafür heute in der Regel nicht mehr notwendig, Konverter von einem internen Zwischenformat in eine hardwarespezifische Druckersprache zu schreiben.

Stattdessen hat sich in den vergangenen Jahren Adobes Portable Document Format (PDF) als Format für die Druckvorstufe durchgesetzt. Allerdings gibt es mehrere Varianten von PDF – neben den von Adobe vorgelegten Spezifikationen (seit 2004 in Version 1.6 [1]) existiert auch die Normenserie ISO 15930 (PDF/X) speziell für die Druckvorstufe und ISO 19005-1:2005 (PDF/A) für die Langzeitarchivierung.

Weil die vom Satzsystem erzeugten PDF-Dateien, zumindest im Fall von großen Editionen, auch von professionellen Verlagen und Druckereien weiterverarbeitet werden, ist es erforderlich, dass das Satzsystem zumindest optional auch PDF/X erzeugen kann (Tatsächlich gibt es auch von PDF/X nochmals verschiedene Unterversionen; PDF/X-1a:2001 [15] wird aber von den Druckereien weitgehend unterstützt.)

Heutzutage soll neben den gedruckten Exemplaren eines Werkes in aller Regel auch eine elektronische Fassung archiviert werden. Das Satzsystem sollte deshalb auch die Möglichkeit bieten, PDF/A [13] zu generieren; optimalerweise im Kompatibilitätslevel PDF/A-1a, der garantiert, dass das Dokument nicht nur auf Dauer visuell reproduziert werden kann, sondern dass auch der enthaltene Text inklusive seiner Struktur erhalten bleibt.

Elektronische Fassungen dienen nicht nur der Archivierung, sondern auch der Onlinepublikation. Für diesen Anwendungsfall werden auch die über die reine Darstellung des Druckbildes hinausgehende Merkmale von PDF benötigt: Etwa dass Verweise auch zusätzlich als navigierbarer Hyperlink genutzt werden, sowohl für Sprünge innerhalb des Dokuments als auch für Hyperlinks auf externe URLs.

In einer PDF-Datei können ferner zahlreiche Metadaten hinterlegt werden, etwa Autor, Erstellungsdatum, Dateiversion, Schlagwörter usw. Weil diese Metadaten mittlerweile von vielen Programmen und Suchmaschinen ausgewertet werden, ist es wichtig, dass diese Felder vom Satzsystem mit sinnvollen Werten belegt werden können.

Darüber hinaus bieten aktuelle Versionen von PDF (ebenso wie Mars und XPS) eine Reihe von Sicherheitsmerkmalen: Dokumente können mit einem Passwortschutz versehen und elektronisch signiert werden. Der Passwortschutz ermöglicht eine rudimentäre Rechteverwaltung: Nur diejenigen können das Dokument lesen, denen das Passwort mitgeteilt wurde. Es lässt sich aber nicht verhindern, dass jemand unbefugt das Passwort an Dritte weitergibt.

Durch die elektronische Signatur kann auf eine dem kryptographischen Stand der Technik entsprechende Weise die Authentizität und Integrität des Dokuments vom Leser verifiziert werden. Speziell beim Einsatz des Satzsystems in einer Umgebung wie TextGrid sollte diese Möglichkeit direkt genutzt werden können, um den Erzeuger einer Datei zweifelsfrei zu identifizieren.

Adobe hat die Entwicklung eines XML-basierten Nachfolgeformats für PDF mit Namen Mars angekündigt. Ob dieses Format von der Industrie und den Anwendern akzeptiert werden wird, lässt sich heute noch nicht absehen. Ähnliches gilt für ein von Microsoft vorgestelltes, mit Mars konkurrierendes Dateiformat namens XML Paper Specification (XPS).

2.4 Technische Anforderungen

Um eine möglichst hohe Nutzerakzeptanz zu erreichen, ist es notwendig, dass die TextGrid-Tools soweit möglich Abhängigkeiten von spezifischen Betriebssystemen vermeiden. Dies ist nicht nur für die RCP notwendig, die auf dem Rechner des Endanwenders läuft, sondern auch für die darüber angesprochenen Dienste. Die größte Hürde für die angestrebte Portabilität sehen wir im Bereich des Font-Managements, denn jede Plattform hat ihre eigene Konventionen und Mechanismen, wie und wo Fonts installiert werden. Bei Satzsystemen wie TeX, die erfordern, dass man zusätzliche Fonts auf proprietäre (oder gar distributionsabhängige) Weise installiert, haben erfahrungsgemäß selbst technisch versierte Anwender häufig Schwierigkeiten, diese Aufgabe zu meistern. Es ist deshalb erforderlich, dass das Satzsystem auf die mit Betriebssystemwerkzeugen installierten Fonts zugreifen kann.

Wie schon oben erwähnt ist eine vollständige Unterstützung des Unicode-Standards [35] für ein zeitgemäßes Satzsystem unabdingbar. Dies ist eine alles andere als triviale Anforderung, wie leicht an der Komplexität des APIs von Softwarebibliotheken wie den ursprünglich von IBM entwickelten International Components for Unicode (ICU) [12] abgelesen werden kann. Doch gerade in wissenschaftlichen Arbeiten, die auch historische Texte in ausgefallenen Sprachen wiedergeben, werden die Möglichkeiten von Unicode ausgereizt werden.

3 Existierende Software für das Print-Publishing

3.1 Batchsysteme

3.1.1 *roff

Seit mittlerweile über dreißig Jahren werden wissenschaftliche Arbeiten im weitesten Sinne — Bücher, Artikel, Standards, Berichts usw. aus den verschiedensten Disziplinen — mit dem Computer gesetzt, seit gut 10 davon ist es zur Norm geworden.

Als eines der ersten Systeme hat TJ-2 1963 einen kruden automatischen Satzalgorithmus implementiert, der grob den Fähigkeiten einer elektrischen Schreibmaschine entsprach, allerdings bereits den Randausgleich beherrschte. Bald darauf folgten bessere Systeme wie runoff [29] und davon inspirierte Nachbauten wie roff und dessen Nachfolger troff, nroff, groff usw, die bei AT&T im Kontext der UNIX-Entwicklung entstanden.[1] Diese begannen in den frühen 1970ern, auch anspruchsvolleren Satzanforderungen wie etwa Fußnoten, unterschiedlichen, auch proportionalen Fonts, einstellbare Seitenbreite usw. zu genügen. Im Verlauf der 1970ern wurden Zusatzwerkzeuge etwa `equ` für mathematische Formeln oder `pic` zur Erstellung von Grafiken hinzugefügt. Selbst wenn diese Programme primär zur Erstellung technischer Dokumentationen konzipiert worden waren, so ließen sich mit ihnen spätestens Ende der 1970er auch typographisch anspruchsvolle (natur-) wissenschaftliche Arbeiten publizieren.

[1] UNIX selbst wurde von AT&T intern ursprünglich als Plattform für Satz von Patentanwendungen mittels roff finanziert: Die Maschine, auf der UNIX entwickelt wurde, wurde angeschafft “specifically to support a text editing and formatting system” [28], S. 374].

3.1.2 TUSTEP

Das Satzprogramm des TÜbinger Systems von Textverarbeitungsprogrammen TUSTEP wurde bereits in den 1970er entwickelt. TUSTEP unterstützt den Textwissenschaftler bei allen Schritten der Erstellung kritischer Editionen bis hin zu deren Satz [26] – Paul Sapplers Kaufringer-Edition, die erste mit TUSTEP-Routinen gesetzte kritische Edition, erschien bereits 1972.[2] Im Verlauf der Jahrzehnte wurden so zahlreiche kritische Editionen publiziert.[3] TUSTEP produziert einen qualitativ hochwertigen Satz, auch mit mehreren Kommentaren und Apparaten; allerdings ist TUSTEP-Satz im Gegensatz zu den übrigen TUSTEP-Modulen nur unter einer kommerziellen Lizenz verfügbar. Formelsatz wird von TUSTEP allenfalls rudimentär unterstützt.

3.1.3 TeX

Das dritte und heute sicher bekannteste der noch immer eingesetzten Satzsysteme, die ihre Wurzeln in den 1970ern haben, ist TeX, das Donald E. Knuth in den späten 1970ern begann, in verschiedenen Versionen bis 1989 erweiterte und dann einfror. TeX gilt bis heute als der Maßstab insbesondere für hochqualitativen mathematischen Satz. Gleichzeitig ist eines der ersten Vorhaben, die ihren Quellcode bewusst der Öffentlichkeit zur Verfügung stellten und einzelne Algorithmen dokumentierten. Bestimmte Verfahren wie der Algorithmus zum Zeilenumbruch werden auch in modernen kommerziellen Systemen wie InDesign verwendet und gelten auch nach drei Jahrzehnten noch als *State of the Art*.

Auch TeX wurde seit den frühen 1990ern zum Satz zahlreicher kritischer Editionen genutzt. Das erste in TeX entwickelte derartige Makropaket, edmac[4] , wird heute nicht mehr aktiv weiterentwickelt. Es gibt allerdings seit 2003 mit ledmac [38] einen LaTeX-Clone von edmac, der die Entwicklung aktiv fortsetzt. Beide Pakete liefern überzeugende Druckqualität auch bei mehreren Apparaten, die jeweils unterschiedlich formatiert sein können, sind in ihrer Bedienung allerdings für nicht TeX/LaTeX-Nutzer komplex. ledmac beherrscht auch den synoptischen Druck verschiedener Versionen.

ednotes [23] ist neueren Datums und wird seit 2002 aktiv weiterentwickelt. Es wurde in München mit dem Ziel entwickelt, ein genuines LaTeX-Paket zum Satz kritischer Editionen zu bekommen, das sich auch mit anderen einschlägigen LaTeX-Paketen kombinieren lässt und somit ein hohes Maß an Flexibilität erreicht. Sowohl ledmac als auch ednotes bieten allein oder in Kombination mit anderen Paketen weitgehende Unterstützung wichtiger editorischer Aufgaben und beide wurden zum Satz einer Anzahl kritischer Editionen eingesetzt. Beide allerdings basieren auf TeX mit seinen gerade im Bereich der Mehrsprachigkeit und des Nutzerinterfaces oftmals nicht mehr zeitgemäßen Leistungen.

---

[2] Allerdings war das spätere TUSTEP zunächst einmal kein Satzsystem im engeren Sinne des Wortes, sondern ein Satz von Fortranroutinen, die der Endnutzer einbinden konnte. Erst 1978 wurde daraus ein hochgradig modulares System von Werkzeugen, die auch ohne Fortrankenntnisse benutzt werden konnten.

[3] Eine nicht vollständige Liste von Editionen findet sich unter http://www.zdv.uni-tuebingen.de/tustep/ed3.html.

[4] Eine Liste einiger mit Edmac produzierter Editionen findet sich unter http://www.homepages.ucl.ac.uk/~ucgadkw/edmac/edmac-bib.pdf

3.1.4 Kommerzielle Batchsysteme

Neben den genannten freien oder (im Fall von TUSTEP) günstigen Batchsystemen gibt es weitere aus dem kommerziellen Umfeld, die teils ebenfalls für den Satz textwissenschaftlicher Werke eingesetzt werden. Die Satzqualität von 3B2, heute unter dem Namen Arbortext Advanced Print Publisher vertrieben, genießt auch für kritische Editionen einen guten Ruf[5]. 3B2 gilt aber auch als komplex, ohne kommerzielle Unterstützung schwer nutz- und programmierbar sowie wegen seiner Lizenzkosten als für die meisten Projekte unbezahlbar.

3.1.5 XSL:FO

Mit großem zeitlichen Abstand zu den genannten Systemen wurde mit ISO/IEC 10179:1996 zunächst die *Document Style Semantics and Specification Language (DSSSL)* normiert. DSSSL, wiewohl das Produkt jahrelanger Standardisierungsarbeit, hatte keinen durchschlagenden Erfolg.[6] Ihr Nachfolger, die im W3C normierten *Extensible Stylesheet Language (XSL)* mit ihren beiden Teilen XSLT und XSL:FO wurde weitaus besser angenommen, insbesondere das auch in anderen Kontexten populäre XSLT. XSL:FO, das 2001 zum ersten Mal veröffentlicht wurde und seit Dezember 2006 in einer neuen Version 1.1 vorliegt, wurde von verschiedenen kommerziellen Firmen wie RenderX und AntennaHouse sowie von Open-Source-Vorhaben wie Apache fop und Suns xmlroff implementiert, allerdings unterstützt aktuell keines der Open-Source-Projekte den gesamten Umfang von XSL:FO 1.0, geschweige denn XSL:FO 1.1.

XSL arbeitet genau wie die anderen oben vorgestellten Systeme im Batchmodus, geht allerdings anders als diese zwangsläufig von einer XML-Datei aus und erzeugt üblicherweise Ausgabeformate wie PDF oder Postscript, seltener Zwischenformate wie rtf. Typische Anwendungsszenarien umfassen den vollautmatischen Satz von Massendokumenten (Rechnungen, ausgefüllte Formulare usw.).

XSL:FO hat sich allerdings bislang für den wissenschaftlichen Satz kaum durchgesetzt. Zum einen bietet es für sich genommen keine Unterstützung für Formeln – dies wird allerdings in einigen kommerziellen Programmen im Sinne der Norm durch die Unterstützung von MathML ergänzt –, zum andern ist es für sich genommen nur für den Satz von kritischer Editionen mit extrem einfachen Satzanforderungen geeignet., Denn trotz der in [36] hinzugekommenen *multiple flows* ist der *flow control* in XSL:FO auf die Bedürfnisse des Zeitungssatzes abgestellt; die für kritische Editionen wichtige Vernetzung mehrerer Apparate wird nicht unterstützt. Auch die Behandlung von *floats* und von beliebigen Formen, die umflossen werden können, lässt aktuell noch zu wünschen übrig. Allerdings ermöglicht [36] ausdrücklich Erweiterungen, weshalb gut vorstellbar ist, dass ein künftiges Tool für den wissenschaftlichen Textsatz ein erweitertes XSL:FO als Eingabeformat nutzt.

3.1.6 Gemeinsamkeiten und Probleme

*roff, TeX, TUSTEP und XSL:FO gemein ist ihr grundsätzlicher Batchansatz und ihre Vermischung von Inhalt und Formatierung. Der eigentliche Text ist durchsetzt mit Formatierungsanweisungen, die

---

5 vgl. z. B. http://www.hug-herstellung.de/3b2.html

6 Die einzige bekannte Implementierung einer Untermenge des Standards, James Clark's Jade, das heute als OpenJade weitergepflegt wird, transformierte SGML in verschiedene Zwischenformate wie TeX, die dann erst in einem zweiten Schritt gesetzt wurden.

sowohl bestimmte inhaltliche Aspekte wie Fußnoten als auch rein typographische Besonderheiten wie Fontwechsel oder Schriftschnitt betreffen können. Alle haben aber auch Makrosprachen entwickelt – oder wurden im Fall von XSL:FO direkt im Zusammenspiel mit einer solchen konzipiert –, die in unterschiedlichem Maß eine inhaltliche Auszeichnung und damit die Trennung von Inhalt und Form ermöglichen. Alle wurden und werden daher, wenn auch nicht im gleichen Umfang, eingesetzt, um XML-Daten zu setzen.

Gemeinsam sind allen Batchansätzen aber auch gewisse Schwierigkeiten. Sie verlangen vom Nutzer, sich (mindestens) mit je einer Makrosprache und einem Auszeichnungsformat auseinanderzusetzen. Ihr Einsatz ist zumindest für Nichttechniker komplex und ohne eine steile Lernkurve kaum zu bewältigen.

Weitgehend ungelöst ist aber vor allem auch das konzeptionelle Problem, dass der Satz hochqualitativer Texte immer manuelle Nacharbeit in Detailfragen wie Seiten- oder Zeilenfall verlangt, die dafür benötigte Information sich aber nur schwer in die Ausgangsdaten einbringen lässt, ohne die Trennung von Inhalt und Form zu opfern. *Processing instructions* lösen dieses Problem nur sehr begrenzt und *standoff annotations* sind im Zusammenspiel mit Satzsystemen weitgehend *terra incognita*.

3.1.7 Mehrsprachigkeit und Fonts

ISO/IEC 10646 / Unicode

Bis in die 1990er gab es weder eine standardisierte Methode, multilinguale Texte zu speichern, noch dezidierte Fonttechnologien dafür. Die Minderheit der Satzsysteme, die überhaupt mit komplexeren mehrsprachigen Dokumenten umgehen konnte, war auf nicht- oder kaum portable Lösungen wie *script tagging*, Zeichensatzwechsel oder *Font switching* angewiesen. Mit dem Aufkommen des *Universal Character Set* (UCS) ISO/IEC 10646 / Unicode und entsprechenden Fonttechnologien wie TrueType und OpenType änderte sich die Situation radikal. Das UCS kodiert mittlerweile über 100.000 Zeichen fast aller lebenden und vieler historischen Schriften, im Falle von Unicode samt wesentlicher Zeicheneigenschaften wie Schriftrichtung.

Selbst wenn gerade für den wissenschaftlichen Bereich wichtige Zeichen noch fehlen – die Wien Initiative[7] hat es sich z. B. zur Aufgabe gemacht, solche für wissenschaftliche Wörterbücher zu dokumentieren und in die Normierung einzubringen –, so lassen sich in allen gängigen Betriebssystemen mittlerweile die meisten Schriften dieser Welt darstellen und bearbeiten.

Fonttechnologien

Die meisten Fontformate vor der Erfindung von Postscriptfonts in den frühen 1980ern waren unstandardisierte Bitmapformate (Bitmap- oder Rasterfonts). Zeichen der Eingabedatei wurden dabei auf entsprechende Bitmaps des Fonts abgebildet, der üblicherweise nicht mehr als 256 Zeichen enthalten konnte. Klassische Postscriptfonts erlaubten es, Zeichen ihrem Aussehen nach zu beschreiben und damit ohne Qualitätsverlust beliebig zu skalieren (Outline-Fonts), aber auch Postscript kann nur maximal 256 Zeichen im Font direkt adressieren.

[7] http://wiki.cdfg.org/WienInitiative

Die 1991 von Apple eingeführten und bis heute immens populären TrueType-Fonts können bis zu 2^{16} Zeichen enthalten, die von der Anwendung direkt adressiert werden können. TrueType-Fonts lösen damit viele Probleme des multilingualen Satzes, haben aber Schwierigkeiten, mit den spezifischen Anforderungen so genannter komplexer Schriften umzugehen.[8]

Diese Probleme wurden 1996 mit dem auf dem UCS aufbauenden OpenType-Format gelöst, das seither von allen großen Fontherstellern umgesetzt und in vielen Betriebssystemen integriert wurde. Dieses Jahr wurde OpenType als ISO/IEC 14496-22:2007 „Open Font Format Specification“ auch formal normiert.

Zusätzlich zu den reinen Glyphen unterstützt OpenType, über Metadatentabellen kontextabhängig Glyphen zu ersetzen (etwa zur Erzeugung von Ligaturen: fl → fl) oder zu positionieren (z.B. für komplexe kombinierte Akzente wie im Vietnamesischen ề). Umgekehrt kann ein- und derselbe Glyph mehreren Codepositionen entsprechen.

Nicht alle Implementierungen können allerdings alle in den OpenType-Fonts integrierten Metadaten auch korrekt auswerten. Eine führende, plattformübergreifende Open-Source-Bibliothek, die dies bereits in guten Teilen leistet, ist Pango, das seinerseits auf die Freetype-Bibliothek aufsetzt. Ein neues Satzsystem wird sich dieser oder ähnlicher Bibliotheken bedienen und sie wo nötig erweitern. Keine dieser Bibliotheken implementiert aber Algorithmen für den Seiten- und Zeilenfall oder für die Behandlung von Flows.

3.1.8 Mehrsprachigkeit in existierenden Systemen

Von XSL:FO abgesehen wurden alle vorgestellten Batchwerkzeuge lange vor ISO/IEC 10646 / Unicode entwickelt und basieren in ihrer inneren technischen Anlage auf einer 8-Bit-Architektur. Entsprechend sind sie für die komplexen Anforderungen mehrsprachigen Satzes — Pluridirektionalität der Schriftrichtungen, unterschiedliche sekundäre Charakteristika der Schriftsysteme, komplexe situationsabhängige Formveränderungen von Buchstaben, Ligaturen usw. — nur schlecht gerüstet.

Für TUSTEP existieren seit 2000 Lösungen, die Unicode-Eingabedateien über einen Eingabefilter in das TUSTEP-interne Kodierungsschema überführt. Damit können prinzipiell alle Schriften, die TUSTEP überhaupt direkt unterstützt (u. a. Lateinisch, Griechisch, Kyrillisch, Arabisch, Syrisch und Hebräisch), auch über Unicode angesprochen werden.

Bereits den 1980ern gab es in der TeX-Welt Versuche, diese Beschränkungen zu überkommen, darunter Projekte wie TeX–XeT von Donald Knuth selbst für den hebräischen Satz — Rechts- neben Linksläufigkeit — und zahlreiche schriftspezifische Pakete für Sprachen dieser Welt. Diese Ansätze blieben allerdings ad-hoc-Lösungen, die (bestenfalls) bestimmte Spezialprobleme lösten und auch keinerlei allgemeinen gültigen Grundsätzen zur Datenerfassung, -speicherung und -verarbeitung genügten. Erst ab 1993 mit der Entwicklung des 16-Bit TeX-Derivats Ω — heute als bekannt — von J. Plaice und Y. Haralambous begannen sich Batch-Systeme konsequenter den Schriftsystemen der Welt zu öffnen und neben entsprechend großen Zeichenauch pluridirektionalen Satz zu ermöglichen.

Ω blieb der große Durchbruch aus verschiedenen Gründen verwehrt. Einmal war und ist die Arbeit mit Ω in der Praxis viel zu kompliziert. Omega Translation Processes (OTPs) als allgemeine State-

[8] Für eine komplette Taxonomie dieser sekundären Charakteristika s. [19], p. 55ff.

Automaten sind aber nicht nur kompliziert zu schreiben und fast nicht zu debuggen, sie sind auch, wie wir weiter unten argumentieren werden, ein fundamental verfehltes Konzept. Auch leidet Ω von jeher unter zahlreichen Fehlern, die auch von von G. Bilotta herausgebrachte korrigierte Version Aleph nicht vollständig korrigieren konnte, da sie in vielem am alten, in Web geschriebenen TeX-Code hängen, von dem auch Ω ausgeht und der den heutigen Entwicklungsstandards nicht mehr entspricht. Vor allen Dingen kann Ω keine modernen Fonts verwenden, sondern bindet über einen wiederum hochgradig komplexen, nicht-standardisierten und fehlerträchtigen Mechanismus 8-Bit-Postscript-Fonts zu virtuellen Fonts zusammen. Viele dieser Probleme beschreiben Plaice und Haralambous in [8] auch selbst, ohne das die von Ihnen angekündigte Lösung allerdings bislang zu erkennen wäre.

Längerfristig sind hier möglicherweise von dem Projekt LuaTeX[9] Fortschritte zu erwarten, in dem Aleph laut einer Ankündigung auf der Mailingliste[10] vom Juni 2007 aufgehen wird. LuaTeX wird unabhängig von Knuths Codebasis entwickelt und durch die Integration einer modernen Skriptsprache dem Anwender sehr viel mehr Flexibilität lässt. Derzeit gibt es aber noch keinen für den Produktiveinsatz geeigneten Code, weshalb hier die Entwicklung weiter beobachtet werden muss.

Einen anderen Ansatz wählt J. Kew's vom *Summer Institute for Linguistics* getragene XeTeX des , das direkt mit UTF-8-Eingabedateien und TrueType-/Opentype-Fonts arbeiten kann, dafür aber für Generierung der Ausgabedatei auf Betriebssystembibliotheken zurückgreift. Ursprünglich nur für Mac OS X und dessen *Apple Advanced Typography* entwickelt existiert XeTeX mittlerweile auch für Linux und Windows.

3.2 WYSIWYG-Textverarbeitungen

3.2.1 Office-Systeme

Sogar noch vor TeX entstanden 1974/75 mit Bravo, Gypsy und BravoX [5] die ersten interaktiven Textverarbeitungssysteme am Xerox PARC in Palo Alto, die den Begriff des *What You See Is What You Get (WYSIWYG)*-Paradigma prägten. Die erste, 1983 erschienene Version von Microsoft Word war von diesen beiden Systemen direkt inspiriert. Im Verlauf der nächsten Jahrzehnte hat sich Word dann als *de-facto*-Standard für WYSIWYG-Textverarbeitungssysteme etabliert.

Zumindest die frühen WYSIWYG-Textverarbeitungssysteme waren primär auf amerikanische Bürobedürfnisse abgestimmt. Bald spaltete sich der Markt in zwei Segmente: Produkte wie WordPerfect und MS Word, die (wie schon der Name der Office-Suiten andeutete) primär den Büromarkt hin im Blick haben und regelmäßig durch andere Tools komplettiert werden, und DTP-Produkte wie FrameMaker, QuarkXPress oder InDesign, die verschiedene Segmente des klassischen Printsektor (Buchsatz, Zeitungswesen, Werbung usw.) anvisieren.

Produkte beider Segmente wurden und werden für wissenschaftliche Publikationen einschließlich solcher mit hohem Formelanteil eingesetzt. Für die Erstellung kritischer Editionen sind Office-Programme allerdings ungeeignet, da sie nicht in der Lage sind, mehrere Apparate zu verwalten.[11]

---

[9] http://www.luatex.org/index.html

[10] http://www.ntg.nl/pipermail/aleph/2007-June/000447.html

[11] Office-Systeme können allerdings sinnvoll zur Datenerfassung für kritische Editionen genutzt werden, wie 2000 ein Pilotprojekt von G. Giacomazzi und M. Küster am Beispiel der Schleiermacher-Edition gezeigt hat, s. http://www.giacomazzi.de/tustep/word2xml2tustep.html.

Dennoch haben sie die Erwartungshaltung aktueller Nutzer (nicht nur) in den Textwissenschaften in einem derartigen Maß geprägt, dass jedes Tool, wenn es eine Chance haben möchte, allgemein akzeptiert zu werden, ihr gerecht werden muss.

3.2.2 WYSIWYG-Tools zum Satz kritischer Editionen

FrameMaker: Framemaker + SGML ist der „Klassiker“ unter den WYSIWYG-SGML-Editoren, der gerade im Bereich der technischen Dokumentation noch immer viel eingesetzt wird, vereinzelt auch in Editionen wie den *Editiones Latinae*. Gleichzeitig ist die Komplexität dieses Tools enorm. Wir können aus eigener Erfahrung Peter Flynn's Einschätzung aus einer Mail an die TEI-L-Mailingliste vom 13.4.2000 bestätigen: *„The process [of setting up Framemaker + SGML] is indeed frustrating, probably the most complex and forbidding of any SGML system I have used.“*

InDesign: InDesign entstand in den späteren 1990ern bei Adobe als Ersatz für FrameMaker und PageMaker. Als erste bekannte DTP-Lösung unterstützte sie Unicode und OpenType und gilt in dieser Beziehung bis heute als einer der Marktführer. InDesign erlaubt sehr feinjustierten multilingualen Satz bei dem jedes einzelne Zeichen mit Sprachattributen verknüpft werden kann. Im Juni 2007 erschien Version 5 der Software.

CET: Der 2000-2004 von Bernt Karasch entwickelte Critical Edition Typesetter[12] ist eine graphische Benutzerschnittstelle für TeX und das Makropaket EDMAC. Es verwendet eine selbstentwickelte Markup-Sprache für die Satzanweisungen; dieser Text wird dann von einem Preprozessor nach TeX/Edmac konvertiert. CET kann bis zu neun unabhängige Apparate verwalten und Zeilennummern sowie Verweise generieren. Der Apparat kann in bis zu drei Spalten ausgegeben werden, umfangreichere Lemmaverweise können automatisch abgekürzt werden. Fonts für die griechische, indische und hebräische Schrift werden mitgeliefert.

CTE: Das 1997-2007 von Stefan Hagel entwickelte und vertriebene Programm Classical Text Editor[13] weist eigens Module für die Verwaltung von Siglen auf und kann mehrere Apparate sowie Randtext verarbeiten. Außerdem kann das Programm mit weiteren sehr spezifischen Anforderungen, die typisch für kritische Editonen sind, umgehen, z. B. den Paralleldruck, der abschnittsweise synchronisiert wird, um etwa Urtext und die unterschiedlich lange Übersetzung wiederzugeben. Allerdings ist das Programm auf einen spezifischen Apparattypus, nämlich den lemmatisierten Apparat, zugeschnitten. Das Programm ist vor allem für den Satz von Druckeditionen gedacht; eine gleichzeitige Erstellung einer digitalen Edition ist nur mit großen Einschränkungen möglich.

4 Anforderungen an das Web-Publishing

Anders als in den 1990er Jahren mitunter vorhergesagt hat das Internet die Printmedien nicht obsolet gemacht; selbst im geschäftlichen Bereich ist die Realität vom „papierlosen Büro“ weit entfernt. Dies gilt ebenso für den Bereich der wissenschaftlichen Editionen. Printeditionen haben ihre Berechtigung und werden diese aller Voraussicht nach auch in Zukunft behalten. Denn das Medium *Web* wird anders rezipiert als ein Buch: Die wenigsten Anwender lesen gerne lange zusammenhängende Texte am Bildschirm; diese werden im Zweifel ausgedruckt, so dass sie wieder als Printmedium vorliegen.

[12] http://karas.ch/cet/

[13] http://www.oeaw.ac.at/kvk/cte/

(Wenn solche Texte online gestellt werden, ist es also immer empfehlenswert, zusätzlich zu einer HTML-Version auch eine PDF-Version bereitzustellen, die einen optimierten Satz gemäß den in den vorhergehenden Kapiteln beschriebenen Kriterien gewährleistet.)

Eine Web-Edition bietet vor allem dem Nutzer – der Begriff *Leser* ist hier zu einschränkend – Vorteile, der z. B. an kleineren Textausschnitten arbeitet, den Querverweisen innerhalb einer Edition folgt, oder mit vielen hochaufgelösten, farbigen Faksimile-Abbildungen arbeitet, deren Druck unverhältnismäßig hohe Kosten verursacht. Zugleich lassen sich Web-Editionen realisieren, die dem Nutzer ein interaktives Arbeiten ermöglichen: Der Anwender legt in gewissen Grenzen selbst fest, welche Ressourcen (also Primärtexte, Apparate, Faksimile, Fotos, Diagramme usw.) neben- oder übereinander dargestellt werden und ob bestimmte Informationen hervorgehoben oder herausgefiltert werden sollen. Eine Web-Edition kann also ganz andere Zugangsmöglichkeiten bieten als eine (aus dem gleichen Quellmaterial erstellte) Printedition. Ob der Aufwand für die Ermöglichung derartiger Interaktion gerechtfertigt ist, muss für jedes Editionsprojekt neu abgewägt werden[14].

Zugleich wird man auch an im schnelllebigen Web publizierte Editionen die aus dem wissenschaftlichen Anspruch heraus gebotenen Grundsätze nicht aufgeben. Wir betrachten in den nachstehenden Unterabschnitten einige dieser Grundsätze und der sich daraus ableitenden Konsequenzen für Web-Editionen sowie beispielhafte Anwendungen, die je nach dem Gegenstand einer Edition den Nutzern tatsächlichen Mehrwert bringen können.

4.1 Zuverlässigkeit

Ein Editor steht mit seiner Reputation für die Qualität der Textgrundlage einer Edition ein. Die Tatsache, dass eine elektronische Edition dem Anwender evtl. auch Faksimile der Handschriften oder Manuskripte zugänglich machen kann, anhand derer die Transkriptionen im Zweifel verifiziert werden können, hebt nicht die Erwartung auf, dass die Zahl der Übertragungsfehler auf ein Minimum reduziert wurde. Beispielsweise wird die nicht manuell fehlerbereinigte Ausgabe eines OCR-Programms für eine Web-Edition ebenso inakzeptabel sein wie man sie bei einer Printedition selbstverständlich zurückweisen würde.

Falls eine Edition sowohl in einer Print- wie in einer Web-Edition erscheint, müssen demnach beide Ausgaben auf der gleichen Textgrundlage aufbauen. Aus technischer Sicht bedeutet dies, dass es eine verbindliche Fassung der Texte geben muss, die vom Editor hinreichend ausgezeichnet wurde (z. B. gemäß TEI), um daraus unter Zuhilfenahme von geeigneten Stylesheets sowohl die Druckfassung als auch die fürs Web aufbereiteten Seiten automatisch zu generieren.

4.2 Zitierfähigkeit

Für das wissenschaftliche Arbeiten ist es wichtig, Texte exakt und mit einer von jedem nachvollziehbaren Stellenangabe zitieren zu können. Sofern eine Printedition existiert, wird diese bei der Stellenangabe bevorzugt, weil Bücher üblicherweise über lange Zeiträume, selbst Jahrhunderte

[14] Die CD-ROM-Edition von Gottfried Kellers Werk (Version 1.6, Beilage zu HKKA 24 [24]) ist ein schönes Beispiel, das durch derartige Features, z. B. zum Vergleich von Ausschnitten aus Manuskriptseiten, einen Mehrwert erfährt. Andererseits können bereits technisch einfache Editionen wie *Augustine: Confessions* [9] für die Forschungscommunity von Nutzen sein.

hinweg in den Bibliotheken für die Forschung zugänglich bleiben. Die Erfahrung mit Ressourcen im Web zeigt hingegen, dass deren langfristige Verfügbarkeit in der Regel nicht gesichert ist.

Die Nutzer einer Web-Edition haben deshalb den Bedarf, erkennen zu können, auf welcher Seite und in welcher Zeile der Printedition gegebene Textstellen zu finden sind. Und umgekehrt müssen sie auch in der Webedition anhand einer auf der Printedition beruhenden Stellenangabe suchen können.

Den Zeilen- und Seitenfall einer Printausgabe auch online zu imitieren, stellt aber einen Missbrauch des Mediums Web dar: Serverseitig können die zur Darstellung genutzten Schriften nur sehr eingeschränkt, die Bildschirmauflösung oder die Breite des Browserfensters überhaupt nicht vorgegeben werden. Anstelle einer Paginierung sollte eine Web-Edition diese Informationen also separat anzeigen – entweder durch serverseitig eingefügte Markierungen im oder neben dem Text, oder dynamisch aktualisiert in einem separaten Bereich des Browserfensters, z. B. bei einem Mausklick im Text. Vor allem bei statischen Markierungen werden die Anwender aus Gründen der Ergonomie die Möglichkeit wünschen, diese Information auch auszublenden.

Falls keine Printausgabe existiert bzw. bei Ressourcen, die in der Printausgabe nicht vorhanden sind, müssen Stellenangaben über URIs (typischerweise meist URLs) erfolgen. Dies setzt voraus, dass nicht nur URIs für die gesamten Dokumente, sondern auch für möglichst kleine Unterab- bzw. Ausschnitte daraus generiert werden können. Diese können im einfachsten Fall über so genannte HTML-Anker realisiert sein. Es ist sogar vorstellbar, diese Links dynamisch zu erzeugen und darin die Stellenangabe bis aufs Zeichen genau zu kodieren. Die Herausforderung besteht in jedem Fall aber darin, dass diese Links stabil bleiben müssen, auch wenn die technische Implementierung der Web-Edition evtl. aktualisiert wird.

4.3 Durchsuchbarkeit

Einer der großen Vorteile von elektronischen Dokumenten ist die Möglichkeit, darin schnell und effektiv zu suchen. Selbst eine einfache Volltextsuche ist in einer Printausgabe nicht möglich; wenn die Suchfunktion auch beliebige reguläre Ausdrücke unterstützt, so sind damit schon überraschend komplexe Recherchen möglich.

Ihren Vorteil können elektronische Editionen aber besonders dann voll zum Tragen bringen, wenn sie die Suchergebnisse untereinander, mit vom Editor erstellten Registern oder programmatisch ermittelten Zusatzinformationen (etwa Lemmatisierungen) verknüpfen.[15] Die auch in einer Printausgabe enthaltenen Register sollten selbstverständlich auch in der Web-Ausgabe verfügbar sein. Das Verhältnis zwischen dem Aufwand für die notwendige Auszeichnung und die Implementierung der Suche einerseits und dem Mehrwert der dadurch ermöglichten Recherchen andererseits hängt vom Gegenstand der jeweiligen Edition ab; der für eine komfortable und mächtige Suche bei der Erstellung einer Web-Edition aufgebrachte Aufwand wird aber in aller Regel gut angelegt sein.

4.4 Aktualität und Revisionierung

Trotz aller Sorgfalt des Herausgebers wird eine Edition praktisch nie völlig fehlerfrei sein. Bis die entdeckten Errata in die eigentliche Edition eingearbeitet werden können, dauert es mindestens bis

[15] Für eine ausführliche Diskussion der Recherchemöglichkeiten und der Grenzen der dabei genutzten Techniken vgl. [32].

zum nächsten Nachdruck, falls es einen gibt. Eine anderweitige Publikation der Errataliste, etwa im Netz, wird wegen des damit verbundenen Medienbruchs immer zweitbeste Lösung bleiben.

Eine Webedition bietet die Möglichkeit, bekannt gewordene Fehler unmittelbar zu beheben – dies ist aber zugleich auch eine erhebliche Herausforderung. Denn eine Web-Site, die nur in größeren Abständen aktualisiert wird, hinterlässt leicht den Eindruck, „aufgegeben“ zu sein; regelmäßige Pflege ist ein Qualitätsmerkmal, das aber einen nicht zu unterschätzenden Aufwand mit sich bringt. Die nach einer Änderung der Textbasis oder der Metadaten notwendigen Transformationen zur Regenerierung der vom Webserver genutzten Daten müssen deshalb weitestgehend automatisiert ablaufen.

Über die Errata hinaus kann eine Web-Edition auch den Fortschritt eines Projektes dokumentieren. Die darin bearbeiteten Textpassagen stehen der wissenschaftlichen Community somit zeitnah zur Verfügung.

Durch die Aktualisierungen ergibt sich allerdings ein Problem beim Referenzieren der Edition: Bei einem Buch ist durch Angabe der Auflage und ggf. des Druckes die Textfassung eindeutig beschrieben. Derlei Angaben existieren bei Web-Editionen zunächst einmal nicht, zumal der Anwender, der sich in einer anderen Arbeit auf einen vorliegenden Textausschnitt in der Web-Edition bezieht, nicht eine Kopie der Edition in der Bibliothek hinterlegen kann.

Um nicht die oben geforderte Zitierfähigkeit zu verletzen, ist es deshalb notwendig, dass Web-Editionen auch ältere Fassungen weiter vorhalten und auf unbestimmte Zeit zugänglich machen. Die jeweilige Revision muss deshalb in den URIs enthalten sein, die als Referenz auf Stellen in der Edition dienen.

4.5 Beispiele

Die CD-ROM-Edition der HKKA [24] enthält Scans von Gottfried Kellers Manuskriptseiten, die mit ihrer jeweiligen Transkription, dargestellt in einem separaten Fenster, verlinkt sind. Darüber hinaus kann auch als Hilfestellung beim Entziffern von Kellers Handschrift die Transkription einzelner Worte in die Manuskriptseiten eingeblendet werden. Für Anwender, die direkt mit den Quellen arbeiten wollen, ist ein weiteres Feature besonders interessant: Bis zu vier mehr oder weniger stark vergrößerte Ausschnitte aus den Manuskripten können neben den Manuskriptseiten abgelegt werden. Der direkte Vergleich unterstützt bei der Entscheidung zwischen mehreren denkbaren Lesarten der Handschrift. oder bei der Zuordnung von Schriften zu Redakteuren. Ähnliches lässt sich auch bei einer Web-Edition durch den Einsatz von JavaScript erreichen.

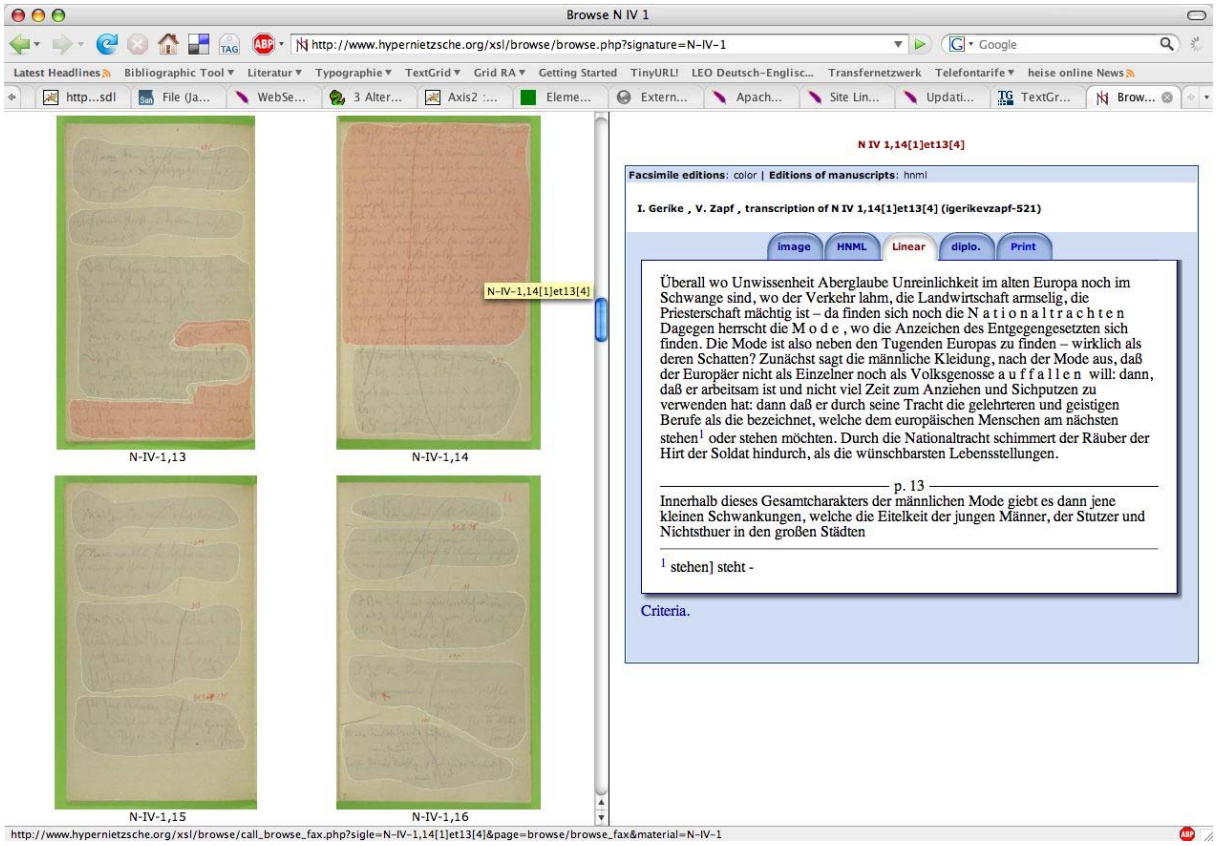

Abbildung 1 - Hypernietzsche: Manuskriptbrowser

Ein weiteres Beispiel für die Verknüpfung von Manuskripten und ihrer Transkription findet sich auf der Hypernietzsche Website[16], diesmal tatsächlich als Web-Edition. Wie in Abb. 1dargestellt, ist hier das Browserfenster zweigeteilt. In der linken Hälfte sind die Seiten aus Nietzsches Notizheft im Überblick dargestellt, wobei die jeweils zusammenhängenden Fragmente farblich abgesetzt sind. In der rechten Hälfte wird jeweils die Transkription des Fragmentes gezeigt, auf das die Maus zuletzt zeigte. Neben der linearen kann hier auch eine diplomatische Transkription, eine XML-Auszeichnung oder eine Vergrößerung des Faksimiles angezeigt werden. Andere Bereiche der Hypernietzsche Website zeigen die Entstehungsgeschichte von Textpassagen in den diversen Skizzen und Manuskripten bis hin zu den veröffentlichten Werken.

5 Software fürs Web-Publishing

Das Web Publishing ist ein Bereich, in dem einerseits sehr viel Software verfügbar ist, sowohl Open Source als auch kommerziell, so dass nur noch in den wenigsten Projekten eine grundlegende Neuentwicklung von Komponenten notwendig sein dürfte. Eine erschöpfende Marktübersicht scheint hier weder sinnvoll noch realistisch, weshalb wir uns nachfolgend auf einige typische Softwarepakete und Frameworks der Apache Foundation konzentrieren, mit denen es möglich ist, die ganze Verarbeitungskette darzustellen, beginnend bei den XML-Daten bis hin zu Content Management Systemen, mit denen die Edition in einen größeren Kontext eingebettet werden kann.

[16] http://www.hypernietzsche.org/

Andererseits ist das Web-Publishing dadurch gekennzeichnet, dass jede Edition sehr individuell gestaltet ist und spezifische Features bietet, so dass es nicht plausibel scheint, dass ein Softwarepaket, das für ein Editionsprojekt zusammengestellt wurde, *in toto* für ein weiteres Projekt übernommen werden kann. Hier wird immer ein vergleichsweise hohes Maß an Konfigurations- und Anpassungsaufwand notwendig sein. Wir erheben deshalb nicht den Anspruch, dass die von uns beschriebene Software für jedes Editionsprojekt die beste Wahl ist; sondern nur, dass sich damit die gängigsten Anforderungen realisieren lassen. Wir unterstellen ferner, dass die Daten, die dargestellt werden sollen, zumindest teilweise in TextGrid hinterlegt sind.

5.1 Anbindung an die TextGrid-Middleware

Die TextGrid-Architektur sieht bereits eine Web Service Schnittstelle zum Zugriff auf in TextGrid abgelegte Daten vor, welche Aspekte wie den physikalischen Speicherort der Daten, das Storage-Format (als gewöhnliche Datei, in einer XML-Datenbank, in einer relationalen Datenbank, ...) oder das Replikationsmanagement vor dem Anwender kapselt. Über diese Schnittstelle können veröffentlichte Daten von jedem abgerufen werden.

Über den Workflow-Editor und geeignete Adaptoren bzw. den Streaming-Editor ist es auch bereits möglich, weitgehende Formattransformationen innerhalb TextGrids durchzuführen. Auch diese Adaptoren werden über eine SOAP-basierte Web Service Schnittstelle angesteuert. Das unten beschriebene Web Development Framework kann grundsätzlich Web Services als Datenquellen nutzen; eine RESTful Web Service Schnittstelle, die es ermöglichte, die gesamte Transformation mit all ihren Parametern in einen URL zu kodieren, könnte deshalb das Abrufen von geeignet aufbereiteten Daten vereinfachen, scheint aber nicht vordringlich.

5.2 Aggregation und Caching

Falls die Edition auch Daten von außerhalb TextGrid nutzt, müssen diese mit der Ausgabe der TextGrid-Adaptoren zusammengeführt und ggf. noch weiteren Transformationen unterzogen werden. Auch kann ein Web Server nicht ohne weiteres Web Services als Datenquelle nutzen.

Für solche Zwecke steht das Apache Web Development Framework *Cocoon*[17] zur Verfügung. Es vermag Daten aus mehr oder weniger beliebigen Quellen zu aggregieren und weiteren Transformationen unterziehen (direkt in die Cocoon-Dienste hineinprogrammiert, per Web Service oder durch Aufruf geeigneter Shell-Skripte). Letzteres ist trotz Nutzung der TextGrid-Adaptoren einschlägig, weil TextGrid vorerst höchstens rudimentäre Werkzeuge zur Manipulation von Bildern (Skalierung, Beschneidung, SVG-Rendering, Konversion usw.) anbieten wird. Cocoon entscheidet anhand der übergebenen URL, welche Transformationen erforderlich sind und aus welchen Quellen die Daten zu beziehen sind.

Schließlich kann Cocoon noch eine Aufgabe übernehmen, die für die Performance sowohl von TextGrid als auch der damit erstellten Web-Editionen von entscheidender Bedeutung ist: Cocoon kann die angeforderten Daten cachen, so dass häufig angeforderte Seiten und Bilder nicht jedesmal neu von der TextGrid-Middleware angefordert werden müssen. Falls die Daten wieder vom Web Server erneut angefordert werden, kann die Anfrage ohne großen Aufwand aus dem Cache heraus bedient werden.

[17] http://cocoon.apache.org/

Weil Cocoon auch Zwischenstufen der Dokumente cachen kann, ist es möglich, die statischen Bestandteile einer Seite zu cachen, während die dynamischen Bestandteile bei Bedarf neu generiert werden.

5.3 Web-Site Layout

Um zu gefallen, benötigt eine Web-Edition wie jede andere Web-Site ein ansprechendes Layout. Dazu gehört zuallererst natürlich ein gelungenes Design (Schriften, Farben, Logos, Hintergrundbilder usw.), das aber jenseits des Aufgabenbereichs von TextGrid liegt; diesbezüglich muss jedes Editionsprojekt eine zum jeweiligen Inhalt passende Lösung finden.

Dazu kommen aber noch Fragen der Benutzerführung, der Anordnung von Menüs etc. Das Site-Layout bestimmt auch, welchen Unterbereichen der Site welche Ansichten auf Daten zugeordnet sind, d. h., welche Transformationsketten für diese Bereiche angestoßen werden müssen. Dies kann alles mit den Mitteln von Cocoon realisiert werden, aber mit Apache *Forest*[18] ist es möglich, die inhaltlichen von den gestalterischen Aspekten noch stärker zu trennen, indem Forest konsequent darauf setzt, dass einmal erstellte Templates zur Laufzeit mit Daten befüllt werden.

5.4 Portalfunktionen und CMS

Wenn ein Editionsprojekt eine eigene Community an sich binden möchte, um etwa die eigene Web-Site als Kommunikationsplattform für die an einem bestimmten Schriftsteller interessierten Forscher zu etablieren, dann muss die Web-Site neben der eigentlichen Edition weitere aktuelle Informationen, Newsfeeds und Möglichkeiten zum Austausch über Mailinglisten oder Foren bieten. Damit fallen Inhalte an, für die TextGrid nicht als Speicherort vorgesehen ist. Deren Verwaltung, getrennt von der Information, wie die Information formatiert werden soll, ist vielmehr typische Aufgabe eines Content Management Systems (CMS). Dieses CMS darf nicht in sich abgeschlossen sein, sondern muss die Möglichkeit bieten, externe Datenquellen – in diesem Fall zumindest für die Web-Edition – zu integrieren.

Ein Vertreter derartiger CMS-Software ist Apache *Lenya*[19], das wegen seiner engen Verzahnung mit Apache Cocoon und Forrest hier prädestiniert ist.

Mit dem bis hierher geschilderten Softwarestack ergibt sich folgender Ablauf, wenn ein Anwender in seinem Browser eine URL aufruft, die auf die Web-Site eines Editionsprojektes zeigt:

- Das CMS (Lenya) entscheidet anhand der URL, welche Daten es aus seinem lokalen Datenspeicher und welche es aus einer externen Quelle benötigt.
- Für die „externen“ Daten prüft Cocoon, ob sie bereits in einem Cache vorliegen und noch aktuell sind. Bei Bedarf generiert Cocoon die notwendigen Web Service Aufrufe, um die Daten aus TextGrid abzurufen und die geeigneten TextGrid-Adaptoren darauf anzuwenden.
- Cocoon cacht die Daten aus TextGrid soweit möglich und aggregiert sie mit Daten aus weiteren Quellen (etwa Forrest Templates). Außerdem führt Cocoon u. U. noch weitere Transformationen durch, um geeignete HTML-Fragmente oder Bilder zu erzeugen.
- Das CMS fügt die diversen HTML-Fragmente zusammen und liefert die Seite zusammen mit den Bildern, Cascading Stylesheets, JavaScript usw. an den Browser aus.

---

[18] http://forrest.apache.org/

[19] http://lenya.apache.org/

- Der Browser des Anwenders rendert die Seite. Mittels geeigneter JavaScript-Routinen kann der Browser auf die Eingaben des Anwenders reagieen und bei Bedarf einen neuen Request an den Web Server schicken.

Anhang A: Bibliographie

[1] Adobe Systems Incorporated. PDF Reference, fifth edition: Adobe Por table Document Format version 1.6, 2004. http://partners.adobe.com/public/developer/pdf/index_reference.html.

[2] Apted, T.; Kay, J.; Lum, A.: *Supporting Metadata Creation with an Ontology Built from an Extensible Dictionary.* In: De Bra, P.; Nejdl, W. (eds.): *Adaptive Hypermedia and Adaptive Web-Based Systems.* Proceedings of AH 2004, vol. 3137 in LNCS, S 4-13, Springer Verlag, 2004.

[3] Bradley, J.: *Pliny – a Note Manager.* http://pliny.cch.kcl.ac.uk/index.html, 2006.

[4] Brüggemann-Klein, A.; Klein, R., and Wohlfeil, S.: *On the pagination of complex documents*. In Computer Science in Perspective: Essays Dedicated to Thomas Ottmann, volume 2598 of LNCS, pages 49–68, Berlin, 2003. Springer.

[5] Burr and, B. and Kimball, R.: *"gypsy". an investigation ginn computer-assisted editing system.* http://www.bitsavers.org/pdf/xerox/alto/GypsyEvaluation, 1976.

[6] Cunnigham, H.; Bontcheva, K.; Tablan, V.; Maynard, D. et al.: *GATE – General Architecture for Text Engineering.* http://gate.ac.uk/, 2006.

[7] Electronic Facsimiles & Texts: *Edition Production & Presentation Technology (EPPT).* http://beowulf.engl.uky.edu/~eft/eppt-trial/EPPT-TrialProjects.htm, 2006.

[8] Distributed Systems Technology Centre, Resource Discovery Unit: *Reggie - The Metadata Editor.* http://metadata.net/dstc/index.html, 1998.

[9] O'Donnell, J. (Text and Commentary); Mahoney, A. (Web Edition): *The Confesions of Augustine: An Electronic Edition.* http://www.stoa.org/hippo/, 1999.

[10] Haralambous, Y. and Bella, G.: *Injecting information into atomic units of text*. In DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering, pages 134–142, New York, NY, USA, 2005. ACM Press.

[11] Hotho, A.; Jäschke, R.; Schmitz, C. und Stumme, G.: *BibSonomy: A Social Bookmark and Publication Sharing System.* In: de Moor, A.; Polovina, S. und Delugach, H.: *Proceedings of the First Conceptual Structures Tool Interoperability Workshop at the 14th International[Conference on Conceptual Structures.* S. 87-102, Aalborg Universitetsforlag, 2006.

[12] International Business Machines Corporation et al.: *International components for unicode, version 3.6*. http://icu.sourceforge.net/, Sept 2006.

[13] International Organization for Standardization: *Document management – electronic document file format for long-term preser vation – par t 1: Use of PDF 1.4 (PDF/A-1).* ISO 15009, 2005.

[14] International Organization for Standardization: *Information and documentation — The Dublin Core metadata element set* . ISO 15836:2003(E), http://www.niso.org/international/SC4/n515.pdf, 2003.

[15] International Organization for Standardization: *Graphic technology – prepress digital data exchange – use of PDF – par t 1 Complete exchange using CMYK data (PDF/X-1 and PDF/X-1a).* ISO 15930, 2001.

[16] Marshal McLuhan. The Gutenberg Galaxis: The Making of Typographic Man. Univ. of Toronto Press, Toronto, 1962

[17] Kiernan, K.; Jaromczyk, J. W.; Dekhtyar, A. und Porter, D. C. mit Hawley, K.; Bodapati, S. und Iacob, I. E.: *The ARCHway Project: Architecture for Research in Computing for Humanities through Research, Teaching, and Learning.* Literary & Linguistic Computing, 2005; 20: 69-88. Verfügbar unter http://tinyurl.com/l5h82.

[18] Kompetenzzentrums für elektronische Erschließungs- und Publikationsverfahren in den Geisteswissenschaften an der Universität Trier: *Das Wörterbuch-Netz.* http://woerterbuchnetz.de/, 2006.

[19] Marc Wilhelm Küster: *Geordnetes Weltbild. Die Tradition des alphabetischen Sor tierens von der Keilschrift bis zur EDV*. Niemeyer, Tübingen, 2007.

[20] Lamport, L.: *LaTeX: A Document Preparation System.* Addison-Wesley, 1986.

[21] Ley, M.: *Digital Bibliography & Library Project.* http://dblp.uni-trier.de/, 2006.

[22] Linguistic Data Consortium: *Simple Metadata Annotation Specification Version 6.2.* http://projects.ldc.upenn.edu/MDE/Guidelines/SimpleMDE_V6.2.pdf, 2004.

[23] Lück, U., *ednotes — critical edition typesetting with LaTeX.* TUGboat, 24(2):224–236, 2006.

[24] Morgenthaler, W. et al. (Hg.): *Historisch-Kritische Gottfried Keller-Ausgabe (HKKA).* Stroemfeld, Basel/Frankfurt a. M. sowie Verlag Neue Zürcher Zeitung, Zürich, seit 1996.

[25] Open Archives Initiative: *Open Archives Initiative.* http://www.openarchives.org/, 2006.

[26] Wilhelm Ott: *A text processing system for the preparation of critical editions.* Computers and the Humanities, 13:29–35, 1979.

[27] Plachta, B.: *Editionswissenschaft.* Reclam Verlag, 1997.

[28] Ritchie, D.M. and Thompson, Ken: *The UNIX time-sharing system*. Commun. ACM, 17(7):365– 375, 1974.

[29] Saltzer, J.: *Manuscript Typing and Editing.* Page AH.9.01. MIT Press, 2nd ed., 1965.

[30] The SEKT project: *Semantically-enabled Knowledge Technologies.* http://www.sekt-project.org/, 2006.

[31] The Text Encoding Initiative: *TEI: Yesterday's information tomorrow.* http://www.tei-c.org/, 2006.

[32] TextGrid: *Text Retrieval.* Projektbericht R 1.3, http://www.textgrid.de/fileadmin/TextGrid/reports/Report_TextRetrieval_published.pdf 2007.

[33] TextGrid: *TextGrid Szenarien*. Entwurf eines Projektberichts, verfügbar auf Anfrage, 2006.

[34] Thomson ResearchSoft: *RIS Format Specifications.* http://www.refman.com/support/risformat_intro.asp, 2001.

[35] The Unicode Consor tium: *The Unicode Standard, version 5.0.* Addison-Wesley, 2007. ISBN 0-321- 48091-0.

[36] W3C: *Extensible stylesheet language (XSL) version 1.1*. http://www.w3.org/TR/2006/REC-xsl11-20061205/, 12 2006.

[37] Wikimedia Foundation: *Wikipedia.* http://www.wikipedia.org/, 2006.

[38] Wilson, P. and Press, H.: *ledmac: A presumptuous attempt to por t EDMAC, TABMAC and EDSTANZA to LaTeX.* http://ring.atr.jp/archives/text/CTAN/macros/latex/contrib/ledmac/ledpar.pdf, 2005.

[39] Yahoo! Inc.: *del.icio.us.* http://del.icio.us/, 2006.

[40] Zhu, J.; Uren, V.; Motta, E.: *ESpotter- Adaptive Named Entity Recognition for Web Browsing.* http://kmi.open.ac.uk/people/jianhan/ESpotter/, 2004.