Projekt FORAUS

Anforderung an ein innovatives Dokumentenverarbeitungssystem

1. Projekt-Semester

WS92/93

UNIVERSITÄT
BREMEN

Fachbereich Informatik

Verfasser:
Tarik Ali, Jens-Martin Brinkmann, Guido Frick, Andree Hähnel, Frank Hettling, Jan Hiller, Andreas Hinken, Kai Hofmann, Michael Jäschke, Gunnar Kavemann, Trinh Le, Van Son Le, Lu Lei, Xiaoxia Lu, Achim Mahnke, Bernd Rattey, Christian Schaefer, Rainer Schmidtke, Volker Schmidtke, Kai Siegele, Quangchau Vo, Jens Warnken

Projektbetreuung:
Prof. Dr. Bernd Krieg-Brückner, Michael Fröhlich, Mattias Werner

Inhaltsverzeichnis

1. Einleitung

Jan Hiller, Andreas Hinken

An der Universität Bremen gibt es im Studiengang Informatik das Modell des Projektstudiums. Dabei soll über vier Semester ein Themengebiet nach wissenschaftlichen Maßstäben vertieft bearbeitet werden.

Seit Wintersemester 92/93 existiert das Projekt „FORAUS“ (Formatierung von Dokumenten mit verschiedenen Sichten). Bei der Projektarbeit spielen neben Entwurf und Implementierung auch Aspekte wie Teamarbeit und das Mitwirken am Aufbau eines realitätsnahen, modularen Systems eine wesentliche Rolle. Als Ergebnis soll eine, wissenschaftlichen Ansprüchen genügende, Textverarbeitung entstehen, die sich von bereits vorhandenen Produkten u.a. durch ihre Multiview Umgebung unterscheidet.

Nach genauerer Auseinandersetzung mit dem Themenbereich „Textverarbeitung“ haben sich die im folgenden genauer beschriebenen Kapitel herauskristallisiert.

Grundlegend besteht jeder Text aus Zeichenfolgen, die in ihrer Gesamtheit das Erscheinungsbild des Textes beeinflussen. Die wichtigsten Regeln zur Typographie und zur Formatierung sind im ersten Kapitel zusammengefaßt. Neben der graphischen bzw. optischen Sicht ist ein Text ebenfalls durch eine logische Struktur charakterisiert. Diese wird teilweise auch nach außen hin sichtbar, z.B. durch die Gliederung in Überschriften oder Absätze. Es sollte neben einer wirklichkeitsgetreuen Darstellung eines Dokumentes, in der typographische Kriterien berücksichtigt werden, auch die logische Struktur visualisiert werden. Auf diese Gesichtspunkte wird im Abschnitt „Logische Struktur“ eingegangen. Im Kapitel „Ausgabe und Datenaustausch“ wird über den Weg von der internen Struktur zur realitätsnahen Abbildung und umgekehrt nachgedacht. Außerdem werden verschiedene Ausgabestrategien und Exportformate auf ihre Tauglichkeit und Verwendbarkeit geprüft.

Bei der Festlegung des Funktionsumfanges der geplanten Software erfolgt eine thematische Unterteilung in zwei Kategorien. Die Standardfunktionalität hat dabei mehr den Charakter eines „Pflichtenheftes“ zur Vollständigkeitsprüfung, die erweiterte Funktionalität stellt eine Sammlung von Eigenschaften dar, die die Textverarbeitung des Projektes FORAUS von anderen positiv abgrenzen soll.

Den heutigen Anforderungen an die benutzergerechte Programmgestaltung, unter Berücksichtigung psychologischer Aspekte, widmet sich das Kapitel „Softwareergonomie“. Dazu werden die allgemeinen Prinzipien von zwei bereits auf dem Markt befindlichen Arbeitsumgebungen dargestellt.

Abschließend werden in der Auswertung einer Umfrage zur Nutzung von Textverarbeitungssystemen die erweiterten Funktionalitäten hinsichtlich ihrer Aktzeptanz überprüft.

Das vorliegende Dokument ist das Ergebnis des ersten Projektsemesters und soll als Wissensbasis zum Thema Textverarbeitung und Softwareergonomie dienen.

2. Typographie und Formatierer

Tarik Ali, Andree Hähnel, Xiaoxia Lu

2.1. Mindestanforderungen an die Typographie

Als typographische Voraussetzung für ein System zur Dokumentenerstellung benötigt man:

- verschiedene Schriftarten:

  Mindestens die Schriften Times (mit Serifen) und Helvetica (ohne Serifen)

  Proportionalschrift muß möglich sein - wünschenswert wäre die manuelle Einstellungsmöglichkeit von Laufweite und Buchstabenbreite
- verschiedene Schriftgrößen zwischen 5pt (Punkt) und 72pt (Punkt) müssen möglich sein
- verschiedene Schriftschnitte: mindestens normal, fett, kursiv und unterstrichen; wünschenswert wäre zusätzlich durchgestrichen und negativ.

2.2. Lesbarkeitskriterien

Die wichtigste Aufgabe bei der Gestaltung eines Dokuments besteht darin, es leicht lesbar zu machen. Dabei soll die Gestaltung möglichst gute Voraussetzungen dafür schaffen, daß der Leser das Dokument inhaltlich verstehen und es in der vom Autor intendierten Form benutzen kann. Ziel der typographischen Gestaltung ist es deshalb nicht in erster Linie, ein ästhetisch befriedigendes Kunstwerk zu schaffen, sondern den Gebrauch des Schriftstücks zu unterstützen. Wenn also eine Schrift in einem Dokument die Aufmerksamkeit auf sich zieht, anstatt sie auf den Inhalt und die Funktion des Textes zu lenken, so hat sie ihre Aufgabe schon verfehlt. Überspitzt ausgedrückt ist es deshalb eine gute Strategie, bei der Wahl der Schriften die Irritation des Lesers zu vermeiden.

Mögliche Irritationen sind:

1. Schlechtes (falsches) Schriftdesign. Kein Ausgleich von optischen Effekten.

Abbildung 1. Optische Effekte

Optischer Effekt: Das Quadrat wirkt größer als der Kreis, obwohl beide die gleiche Höhe und Breite haben. Ausgleich: Kreis etwas größer machen.

2. Ungleichmäßiger Weißraum zwischen den Buchstaben.
   Beispiel
   Unterschied zwischen Proportionalschrift und `Schreibmaschinenschrift`

3. Wahl einer Schrift, die vom Charakter her nicht zum Inhalt paßt (siehe Abbildung 2).

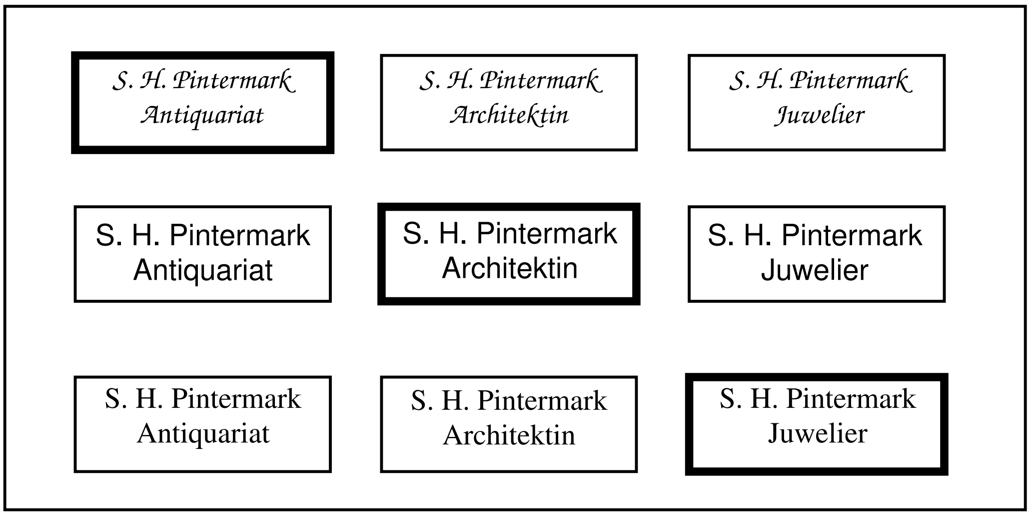

Abbildung 2. Wirkung von Schriftarten

Schild 1: Antiquariat Zapf Chancery = klassisch-blumige Schrift, die auf die Liebe zu wertvollen Büchern deutet

Schild 2: Architektin Avant Garde = geometrisch-klar, vermittelt den Eindruck von Kreativität und Innovationsfreude

Schild 3: Juwelier Palatino = verbindet Klarheit mit Eleganz

4. Buchstabenformen unterscheiden sich zu wenig von einander:
   Beispiel 1
   c und o   das c darf nicht zu weit geschlossen sein.

   Beispiel 2
   Schriften ohne Serifen unterscheiden sich nicht ausreichend.
   1,l,I   (Times, Schrift mit Serifen)
   1,l,I   (Helvetica, Schrift ohne Serifen)
   Man erkennt hier deutlich, daß sich in der Schrift Helvetica das kleine „L“ nicht ausreichend vom großen „I“ unterscheidet.

5. Mischen von Schriften aus verschiedenen Familien im Fließtext

6. Lückenreißende Buchstaben

   Beispiel 1

   VAK ohne Unterschneidung (Kerning)

   VAK mit Unterschneidung (Kerning)

   Beispiel 2

   Trinkflasche ohne Ligatur

   Trinkflasche mit Ligatur

   aber Auflage ohne Ligatur (keine Ligatur bei Wortgrenzen in zusammengesetzten Wörtern)

   Weitere Ligaturen sind: fi, ff, ffl.

   (übrigens: ß und & sind auch Ligaturen, und zwar entstanden aus ss und et=lat. und)

   Beispiel 3

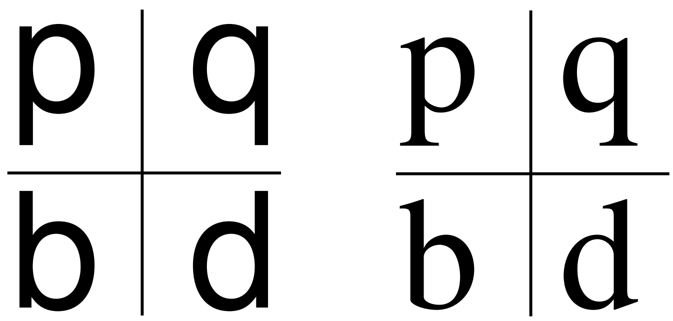

Abbildung 3. p q und b d

   gespiegelte Buchstaben sich nicht ausreichend voneinander zu unterscheiden (siehe Abbildung 3)

7. Schlechte Schriftqualität aus technischen GründenAnsätze für den Formatierer

Wir benötigen nun für den Umbruch von Zeilen, Absätzen und Seiten einen Formatierer. Dieser soll eingegebenen Fließtext anhand von Designregeln bearbeiten. Diese Designregeln sind schon im Formatierer vorhanden. Es soll aber auch die Möglichkeit bestehen, nachträglich neue Regeln einzugeben.

8. Serifenlose Schriften:

a,g und q in Palatino

a,g und q in Avant Garde

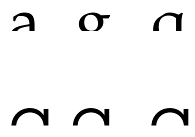

Abbildung 4. Schriftkonturen

Die oberen Konturen eines Wortes bieten dem Leser in der Regel besseren Aufschluß über das Wort, als die unteren Konturen. Problem: serifenlose Schriften differenzieren hier (obere Kontur) nicht stark genug. Dieses wird an Hand von Abbildung 4 deutlich.

2.2.1. Absätze und Zeilenumbruch

Ein Formatierer für den Absatz müßte mindestens folgende Funktionen leisten können:

- Ausrichtung: links-, rechtsbündig, zentriert, Blocksatz
- Einzug: vertikaler Freiraum am Anfang eines Absatzes (frei einstellbar, Abhängig vom Schriftgrad)
- Zeilenbreite: Beliebige Zeilenbreite muß möglich sein. Optimal: 50-70 Zeichen/Zeile (Abhängig vom Schriftgrad)
- Durchschuß: Freier Platz zwischen den Zeilen muß frei einstellbar sein. Das Verhältnis zwischen Durchschuß und Wortabstand muß harmonisch sein. Wenn der Wortabstand größer wird, als der Abstand zwischen den Zeilen, geht die horizontale Ausrichtung am Zeilenrand verloren. Stattdessen entsteht der Eindruck von vertikalen Spalten.
- geschützte Leerzeichen: Leerzeichen zwischen Wörtern auszeichnen, um eine Trennung zu vermeiden. Beispiel: Hans~Joachim von~Strotens

2.2.2. Ansätze zum Algorithmus für den Zeilenumbruch

Der Vorgang des Zeilenumbruchs besteht im Wesentlichen aus zwei Phasen.

1. Das Festlegen der Zeilengrenze
2. Das Ausschließen der Zeilen auf ein vorgegebenes Maß (dehnen und stauchen)

Dazu kann man folgendermaßen vorgehen: Eine Zeile wird solange mit Wörtern aufgefüllt, bis das kritische Wort erreicht ist, durch das die Zeile zu voll werden würde. Hierbei geht man von der normalen/idealen Größe der Wortzwischenräume aus. Nun gibt es mehrere Optionen, weiter zu verfahren:

- Man kann die Zeile so weit zusammenstauchen, daß das kritische Wort noch in die Zeile paßt. Dabei darf ein vorgegebener minimaler Wortzwischenraum nicht unterschritten werden.

- Man kann die Zeile ohne das kritische Wort so weit dehnen, daß ein entsprechend vorgegebener maximaler Wortzwischenraum nicht überschritten wird.
- Man kann das kritische Wort trennen, so daß einer der ersten beiden Fälle für ein Anfangsstück des kritischen Wortes zutrifft.

Eine Auswertung der besten Möglichkeit muß dann im Weiteren erfolgen, evtl. unter Berücksichtigung von voreingestellten Parametern (sloppy)

Dieses Verfahren, wo jeweils nur eine Zeile betrachtet wird, heißt lokal optimierendes Umbruchverfahren. Man kann aber nicht nur für jeden Absatz einen lokal optimierenden Umbruch durchführen, sondern alle lokal akzeptablen Umbrüche berechnen, und aus ihnen dann einen unter globalen Gesichtspunkten optimalen aussuchen. Hierzu werden für die Ereignisse Dehnen, Stauchen und Trennung verschieden hohe Strafpunkte verteilt. Die lokalen Strafpunkte werden nun innerhalb eines Umbruchs (Absatz) aufsummiert und der Umbruch mit den wenigsten Strafpunkten ist der global beste.

2.2.3. Silbentrennung

Die beiden wichtigsten deutschen Trennregeln sind:

1. Mehrsilbige einfache und abgeleitete Wörter trennt man nach Sprechsilben, die sich beim langsamen Sprechen von selbst ergeben.
2. Zusammengesetzte Wörter und Wörter mit einer Vorsilbe werden dagegen nach ihren sprachlichen Bestandteilen, also nach Sprachsilben getrennt.

Diese Regeln können durch folgende Verfahren realisiert werden:

- Wörterbuchbasiertes Verfahren: Ein Wörterbuch enthält die Trennstellen aller in ihm vorhandenen Wörter.
- Musterbasiertes Verfahren: Es beruht auf verschiedene Buchstabenmuster, die erlaubte Trennstellen enthalten. Paßt ein Teilwort auf ein Muster, so wird an der erlaubten Trennstelle getrennt.
- optionale Trennung: manuelles Auszeichnen einer Trennstelle.

Für den Fall, daß ein vom Trennverfahren falsch getrenntes Wort sehr häufig vorkommt, ist es nützlich wenn man das Wort mit der korrekten Trennung in ein Ausnahmewörterbuch aufnehmen kann. Die obengenannten Verfahren kann man auch kombinieren, um einen höheren Wirkungsgrad zu erzielen.

2.2.4. Seitenumbruch

1. Schusterjungen und Hurenkinder: Weder die erste noch die letzte Zeile eines Absatzes sollte durch einen Seitenumbruch von dem Rest des Absatzes getrennt werden.
2. Keine Trennzeichen am Ende einer Seite
3. Überschriften: Überschriften sollten mindestens von 3-4 Zeilen gefolgt werden, bevor ein Seitenumbruch vorgenommen wird.

4. Figuren: Numerierte Figuren müssen in der Reihenfolge ihrer Numerierung plaziert werden.
   - Numerierte Figuren müssen nach dem ersten Verweis auf die Figur erscheinen.
   - Numerierte Figuren erscheinen oben oder unten auf der Seite, mit einem gewissen Minimal- und Maximalabstand zum Text.
   - Numerierte Figuren müssen von ihrem ersten Verweis aus sichtbar sein, müssen also in Büchern auf derselben Doppelseite erscheinen
   - In Ausnahmefällen darf eine Figur auch vor ihren ersten Verweis geschoben werden, wenn es anders nicht möglich ist, sie von diesem ersten Verweis aus sichtbar zu machen.
   - Abhängig von der Figurenbreite können sequenziell numerierte Figuren auch nebeneinander erscheinen.
5. Fußnoten: Fußnoten sind immer auf der selben Seite, auf der auch ihr Verweis steht. Sie können aber auch am Ende eines Kapitels zusammengefaßt werden, wenn es sich um Hinweise auf Literatur handelt, und diese kein Kürzel im Literaturverzeichnis haben (bzw. kein Literaturverzeichnis besteht). Weitere Ausnahmen sind hier nicht möglich.

Es sollte die Möglichkeit bestehen einen Seitenumbruch manuell auszuzeichnen und auch zu erzwingen. Um vorstehende Regeln einhalten zu können, sollte der vertikale Abstand zwischen Überschriften und Fließtext dehn-/stauchbar sein.

2.2.5. Verschiedene Formatierer

Wir haben verschiedene Teilbereiche, die der Formatierer bearbeiten muß. Im Kleinen z.B. die Formatierung einer Ligatur innerhalb eines Wortes, und im Großen z.B. der Umbruch von Seiten und dabei das richtige Plazieren von Abbildungen. Diese verschiedenen Teilbereiche sind nun aber für sich schon so komplex und spezifisch, daß wir für die einzelnen Bereiche eigene Formatierer benötigen.

Mögliche Formatierer wären:

- Wortformatierer: Er setzt Buchstaben zu Wörtern zusammen unter Beachtung von typographischen Regeln wie z.B. Kerning und Ligaturen.
- Absatzformatierer: Er faßt Wörter zu Zeilen und Zeilen zu Absätzen zusammen. Dabei müssen Wörter getrennt und die typographischen Regeln (siehe Abschnitt 2.2.1.) berücksichtigt werden (z.B. Blocksatz, Einzüge, geschützte Leerzeichen).
- Seitenformatierer: Er bricht die Satzfahne in Seiten um. Hierbei müssen bestimmte Regeln beachtet werden (siehe Abschnitt 2.2.4.). Z.B. Plazierung von Abbildungen und Fußnoten, Schusterjungen und Hurenkinder.

Hierbei ergibt sich aber schon das Problem, daß der Absatzformatierer Wörter, die schon vom Wortformatierer formatiert wurden, zum Zeilenumbruch evtl. wieder aufbrechen und trennen muß. Dieses Problem ist kein Einzelfall. Es gibt mehrere Situationen, in denen ein schon formatierter Bereich von einem anderen Formatierer wieder aufgebrochen werden muß. Hierbei kann es vorkommen, daß der vorherige Formatierer diesen Bereich erneut

formatieren muß. Das bedeutet, daß die Formatierer nicht rein sequenziell arbeiten dürfen. Sie müssen zeitlich verschachtelt fungieren, was einen hohen Bedarf an 'Kommunikation' zwischen den Formatierern mit sich bringt. Wie weit man Probleme in kleinere Formatierer unterteilt muß man natürlich im Einzelfall abwägen.
Sicher ist, daß diese Technik der interagierenden Formatierer einen hohen 'Kommunikationsaufwand' zwischen den Formatierern mit sich bringt.

3. Logische Struktur

Michael Jäschke, Gunnar Kavemann, Achim Mahnke, Jens Warnken

3.1. Motivation

Ein entscheidender Faktor, der die Möglichkeiten eines Systems zur Bearbeitung und Formatierung von Texten beeinflußt, ist die Art, in der das System Texte intern darstellt. Dabei ist wohl allen Textverarbeitungssystemen gemein, daß sie den reinen Text in irgendeiner Form von Zeichenketten speichern. Entscheidend ist jedoch, ob sie auch die Struktur eines Textes, also zum Beispiel dessen Einteilung in Kapitel, Unterkapitel, Absätze u.s.w. erfassen und für die Bereitstellung von Funktionen zur Manipulation dieser Struktur benutzen. So kennt Word zum Beispiel die Gliederung eines Textes, wenn die Absätze entsprechend markiert werden, und erlaubt in einer eigenen Sicht das schnelle Verschieben von Kapiteln. Weitere hilfreiche Funktionen, die auf den Struktur-Informationen aufbauen, sind die automatische Numerierung der Gliederungsebenen und die Generierung eines Inhaltsverzeichnisses.

Eine zweite Frage, die in diesem Zusammenhang eine große Rolle spielt, ist die der Speicherung von Formatierungs-Attributen. Viele Textsysteme vermerken die Formatierung eines Textteils (z.B. Fettdruck oder Unterstreichung) direkt im Text und diese können dann nur per Hand geändert werden, wenn Bedarf dazu besteht. Ein extremes Beispiel hierfür sind DTP-Programme, die nicht nur die Formatierung, sondern auch die Positionierung von Textteilen und Objekten (Bildern, Formeln etc.) fest im Text „verdrahten“.

Bei diesem Ansatz besteht das große Problem, einen einmal „per Hand“ gesetzten Text für andere Zwecke - die eine andere Formatierung erfordern - wiederzuverwenden, da dann mühselige Handarbeit geleistet werden muß. Eine Lösung dieses Problems besteht darin, dem Textsystem genaue Informationen über den Aufbau eines Dokumentes zu geben, indem man die einzelnen logischen Elemente wie Kapitel, Absatz, Formel, Bild, Hervorhebung u.s.w. kenntlich macht und Regeln zur Formatierung der logischen Elemente angibt. Wird eine neue Formatierung gewünscht, muß der Text nicht geändert werden, da er keine Informationen über sein konkretes Aussehen (seine physikalische Repräsentation) enthält. Stattdessen werden die Regeln zur Formatierung geändert und man erhält praktisch sofort die neue physikalische Repräsentation.

Diese Philosophie wird unserem System zugrunde liegen.

3.2. Bisherige Ansätze

In den heute verbreiteten Textsystemen existieren verschiedene Ansätze, die logische Struktur eines Textes zu erfassen. Zwei wichtige Formen, die hierbei unterschieden werden können, sind die sequentielle und die hierarchische Struktur.

Viele gängige Textsysteme, wie zum Beispiel Word in der DOS/Windows-Welt und Framemaker in der Unix-Welt, verwenden eine Darstellung des Textes als Sequenz von Objekten wie Absätze und Bilder (bei Framemaker auch Formeln und Tabellen). Absätze setzen sich in der Regel aus einer Folge von Zeichen zusammen; dies kann man jedoch nicht als Merkmal einer hierarchischen Struktur werten.

Durch Zuweisung eines Formates können Absätze als logische Elemente gekennzeichnet werden, was jedoch hauptsächlich auf die Formatierung des Textes wirkt. Bekommt ein Absatz z.B. den Typ „Listenpunkt“, dann erscheint der Text eingerückt, mit einem Punkt oder Strich davor. Wird er als „Zitat“ gekennzeichnet, dann ist er z.B. gegenüber beiden Rändern eingerückt und in Anführungszeichen eingeschlossen. Als einziges logisches Merkmal wird bei diesen Systemen die Information über die Reihenfolge der Absätze verwendet, indem z.B. Kapitel automatisch durchnumeriert werden.

Probleme treten hier auf, sobald logische Elemente eines Textes geschachtelt sind, wenn z.B. ein Zitat in einem Listenpunkt auftritt. Für diesen Fall müßte man in den Systemen mit sequentieller logischer Struktur einen neuen Absatztyp definieren, der erforderliche Formatierungsregeln, wie zusätzliche Einrückungen oder vielleicht eine kleinere Schrift enthält.

Der hierarchische Ansatz erlaubt einen einfacheren Umgang mit diesem Problem. Dem System ist bekannt, daß das Zitat zum Listenpunkt gehört und benutzt diese Information, um den Text entsprechend zu formatieren. Dieser Ansatz ist z.B. in LaTeX realisiert. Voraussetzung für diese Herangehensweise ist jedoch ein profundes „Wissen“ des Textsystems in Bezug auf die Formatierungsregeln, da komplexere Schachtelungen von Text-Elementen zu extremen Formatierungen führen können (wenn in dem Zitat z.B. noch eine Liste vorkommt). Auf der Habenseite steht jedoch eine Vereinfachung für den Benutzer, der sich in diesem Fall kein neues logisches Element definieren und keine Formatierungsparameter angeben muß.

Im folgenden werden Anforderungen an die Beschaffenheit der logischen Struktur formuliert und ein Beispiel für ein komplexeres Modell vorgestellt.

3.3. Anforderungen

3.3.1. Übergeordnetes Modell

Die logische Struktur ist nur eine von mehreren Repräsentationen, die ein Dokument auf dem Weg zu seiner endgültigen Form durchläuft. Ausgehend von der logischen Struktur werden durch die Formatierung physikalische Repräsentationen erzeugt, die in einer geeigneten Form das konkrete Aussehen des Dokumentes beschreiben.

Die Konzeption dieser physikalischen Repräsentationen muß überdacht und in ein globales Modell eingeordnet werden, das alle Repräsentationen sowie deren Beziehungen untereinander darstellt. Dieses zu definierende Modell bildet eine Grundlage für den Aufbau des Textsystems.

Als Beispiel für ein solches Modell soll hier das von Richard Furuta in [Fu89] verwendete kurz vorgestellt werden.

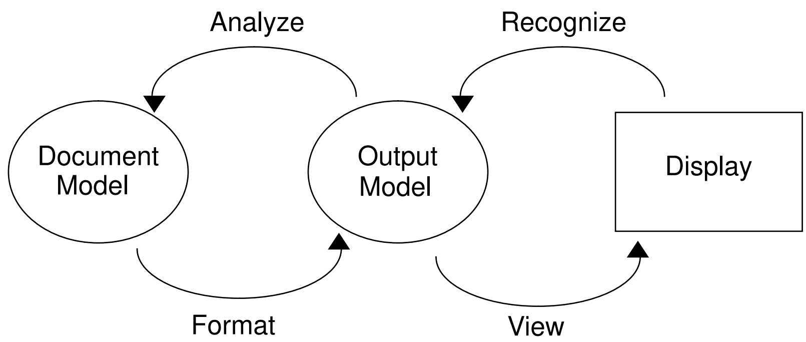

Abbildung 5. Repräsentations-Modell

Das Modell beinhaltet drei Repräsentationen des Dokumentes (siehe Abbildung 5). Das „Document Model“ stellt eine Abstraktion auf der Ebene einer logischen Beschreibung dar, bezeichnet also die Form des Dokumentes, die in diesem Abschnitt behandelt wird. Im „Output Model“ wird das Erscheinungsbild des Dokumentes beschrieben. Zeilen- und Seitenumbrüche, die Positionierung des Textes sowie Attribute wie Schriftgröße und -art sind hier festgelegt. Diese physikalische Beschreibung des Dokumentes kann ebenfalls abstrakt vorgenommen werden, z.B. in Postscript-Form. Als „Display“-Repräsentation wird das fertige Dokument bezeichnet, so wie es auf das jeweilige Medium (Papier, Bildschirm etc.) gebracht wird.

Das bearbeitete Dokument wird in seinem Entstehungsprozess zwischen diesen Repräsentationen transformiert. Die wichtigste Transformation ist der Formatierungsvorgang, der aus dem „Document Model“ das „Output Model“ generiert. Dem Anzeigen des Dokumentes muß eine Umsetzung des Output Model auf ein geeignetes Medium vorausgehen. Dies kann z.B. die Aufbereitung der physikalischen Repräsentation für einen Drucker oder einen Bildschirm durch entsprechende Treiber sein.

Die inverse Transformation zur Formatierung - in diesem Modell Analyse genannt - ist für unser System ebenfalls von großer Bedeutung, da wir ein interaktives Editieren von Dokumenten aus unterschiedlichen Sichten realisieren wollen. Es stellt sich dabei das Problem, aus den Sichten, die eine physikalische Repräsentation anzeigen, Informationen über Änderungen am Dokument in die logische Beschreibung zurückzutransportieren. Diesen Vorgang kann man annähernd mit dem in Abbildung 5 „Analyze“ genannten gleichsetzen. Abschnitt 5.3. beschäftigt sich mit Möglichkeiten zur Realisierung des Rücktransportes. Das Erkennen („Recognize“) von Text aufgrund einer physikalischen Vorlage (Stichwort „Optical Character Recognition“) ist für unser Textsystem nicht vorgesehen.

3.3.2. Generelle Anforderungen an die logische Struktur

Eine naheliegende und grundsätzliche Forderung ist die nach der Trennung von logischer und physikalischer Beschreibung. Die logische Struktur eines Dokumentes soll völlig unabhängig von jeglichen Definitionen seiner äußeren Form gebildet werden. In irgendeiner Form muß natürlich eine Verbindung von logischen Elementen und Formatierungsangaben existieren, diese sollen jedoch nicht für die Beschreibung der logischen Struktur herangezogen werden.

Das Dokument muß sinnvoll in atomare Objekte unterteilt werden können. Diese logischen Elemente wie z.B. „Satz“ oder „Absatz“ dienen als kleinste adressierbare Objekte. Bei der Festlegung solcher Objekte ist insbesondere auf die Granularität und die innere Struktur zu achten.

Die logischen Elemente müssen in eine hierarchische Struktur gebracht werden, d.h. zusammengehörige Objekte werden zu übergeordneten Objekten zusammengefaßt; das größte Objekt ist das Dokument als Ganzes. Durch die hierarchische Struktur erhält das System Informationen über den Aufbau des Dokumentes, was für die Bereitstellung von mächtigen Editier- und Formatierungsfunktionen - wie z.B. der automatischen Numerierung von Objekten - vonnöten ist.

Neben der Zusammenfassung von Objekten zu übergeordneten Einheiten muß es eine Möglichkeit geben, Objekte miteinander zu verbinden, die nicht im Sinne der hierarchischen Struktur zusammengehören. Ein gutes Beispiel für solche Verbindungen sind z.B. Referenzen auf andere Textstellen (logische Elemente), ohne die Verbindungen auf diesen Zweck und diese Form beschränken zu wollen.

Um verschiedene Dokumentenklassen beschreiben zu können, muß es dem Benutzer möglich sein, neue logische Elemente - und auf deren Basis auch eine neue hierarchische Struktur - zu definieren. Zu klären ist, ob völlig neue atomare Objekte kreiert werden, oder ob es genügt, die hierarchische Struktur der bestehenden verändern zu können. Dies hängt wiederum von der Form der gewählten atomaren Objekte ab. Unabhängig von der Ausgestaltung muß es jedoch möglich sein, jede beliebige Dokumentenklasse (zumindest für wissenschaftliche Texte) zu definieren.

Über die Verbindung von Objekten untereinander hinaus ist die Verbindung mit externen Objekten, die nicht direkt im Textsystem gespeichert sind, von Bedeutung. Als externe Objekte kommen z.B. Felder aus Datenbanken (auch Literaturdatenbanken) oder Grafiken in Betracht. Die Einbindung solcher Objekte ermöglicht unter anderem das Zurückgreifen auf stets aktuelle externe Daten; außerdem können auf diese Weise spezielle Funktionen zur Bearbeitung von bestimmten Objekten ausgelagert werden, um von spezialisierten Werkzeugen ausgeführt zu werden (z.B. Formelsatz).

3.3.3. Anforderungen an die Gestaltung der Beschreibung

Neben den Anforderungen an die Funktionalität der Beschreibung einer logischen Struktur gilt es, sich Gedanken über deren Gestaltung zu machen.

Die Beschreibung muß so beschaffen sein, daß sie eine effiziente Implementierung der vorgesehenen "Erweiterten Funktionalität" zuläßt. Als Beispiel sei hier die Unterstützung der gemeinsamen Erstellung von Dokumenten angeführt. Wollen mehrere Benutzer gleichzeitig ein gemeinsames Dokument bearbeiten, muß es in der logischen Struktur geeignete Elemente geben, auf dessen Basis die Synchronisation (z.B. gegenseitiger Ausschluß) stattfinden kann; speziell die Granularität der logischen Elemente ist hier von Bedeutung.

Da die qualitativ hochwertige Formatierung eines Dokumentes sehr aufwendig ist, liegt der Wunsch nach einer effizienten Methode hierfür sehr nahe. Die Beschreibung der logischen Struktur sollte diese ermöglichen. In die Gestaltung der logischen Struktur müssen auch Überlegungen zu den inversen Transformationen einfließen, da der Rücktransport von Informationen in diese Struktur hinein nicht einfach sein wird.

Die letzte Anforderung betrifft die Repräsentation der logischen Struktur selbst. Sie sollte so klar und verständlich darzustellen sein, daß auch ungeübte Benutzer in der Lage sind, sich eigene generische Strukturen zu erzeugen, um verschiedene Dokumentenklassen beschreiben zu können.

3.4. Ein Beispiel: 'tnt' von Richard Furuta

Nachdem die Anforderungen an eine Beschreibung der logischen Struktur eines Dokumentes gezeigt wurden, wollen wir diese an einem Beispiel näher erläutern. Dazu nehmen wir das Modell 'tnt' ('tnt' ist also keine endgültige Beschreibungssprache der logischen Struktur, sondern ein Modell für eine solche).

'tnt' steht für 'strict tree, not strict tree' und ist von Richard Furuta entwickelt worden. Die ersten Publikationen, die 'tnt' erwähnen, erschienen im Jahre 1986. 'tnt' beschreibt die logische Struktur eines Dokumentes durch einen Baum.

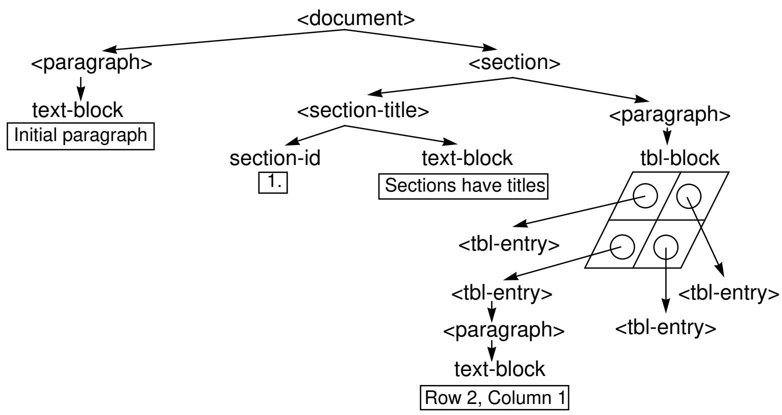

Abbildung 6. Ein ‘tnt’-Baum

Furuta wählte diese Darstellung, da sie zwei Beziehungen zwischen den Knoten des Baumes angibt:

- die Baumstruktur zeigt sehr schön, wie die Objekte der logischen Struktur sich aus anderen Objekten zusammensetzen. (Abbildung 6: <document> ist ein zusammengesetztes Objekt aus <paragraph> und <section>, oder man sagt: <paragraph> und <section> sind in <document> enthalten).
- linke Kindsknoten erscheinen in der physikalischen Repräsentation ‘vor’ rechten Kindsknoten [Abbildung 6: <paragraph> wird im Ausdruck vor <section> erscheinen (d.h. z.B. weiter oben auf einer Seite)].

Weiterhin beschreiben die Äste des Baumes die logische Struktur des Dokumentes und die Blätter des Baumes den Inhalt.

Blätter des ‘tnt’-Baumes können irgendeine freie Struktur beinhalten. Es gibt nur die Vorgabe, daß diese freien Strukturen irgendwann in atomare Objekte oder Verbindungsknoten terminieren müssen. Atomare Objekte sind die kleinsten Elemente in einem Dokument. Atomare Objekte können z.B. einzelne Zeichen oder auch ganze Absätze sein. Verbindungsknoten sind Knoten, die den Baum, der sie beinhaltet, terminieren lassen und dabei auf den Startknoten eines anderen Baumes verweisen.

In einer Testimplementation von ‘tnt’ mit Namen ‘pedtnt’ sind Blätter des Baumes unter anderem Textblöcke, die aus dem atomaren Objekt ‘String’ bestehen. Weiterhin sind Formelblöcke für mathematische Formeln und Tabellenblöcke implementiert. Die Tabellenblöcke sind als Matrix von Verbindungsknoten organisiert. Jeder Verbindungsknoten verweist auf einen ‘tnt’-Baum, der den Tabelleneintrag darstellt (evtl. wieder eine komplizierte Struktur). Siehe dazu auch Abbildung 6, in der der letzte <paragraph> aus einem Tabellenblock besteht.

Dadurch, daß die Blätter des ‘tnt’-Baumes eine freie Struktur beinhalten können, lassen sich beliebige neue Blöcke definieren, die z.B. ein Einbinden von Grafik in ein Dokument ermöglichen, oder die eine Verbindung zu Elementen aus anderen Programmen herstellen. Dabei ist nur die oben genannte Vorgabe an die Termination der freien Struktur zu beachten.

Strukturierte Dokumente können durch einen oder mehrere ‘tnt’-Bäume dargestellt werden. Wird mehr als ein ‘tnt’-Baum benutzt, repräsentiert jeder einzelne Baum einen Strom von Objekten in dem Dokument. Dabei werden diejenigen Objekte in einem Strom zusammengefaßt, die die Reihenfolge ihres Auftretens beim Umwandeln der logischen Beschreibung des Dokumentes in die physikalische Repräsentation immer beibehalten. Kapitel in einem Buch sind z.B. Objekte, die in einem Strom zusammengefaßt werden können. Objekte eines Dokumentes, die bei der Umwandlung der logischen Beschreibung des Dokumentes in die physikalische Repräsentation ihre Position verändern können (Fußnotentext oder eingebundene, mitfließende Grafiken sind Beispiele für diese Objekte), werden in separate Objektströme geschrieben.

Einzelne Objektströme darstellende ‘tnt’-Bäume können über „Links“ miteinander verbunden werden. „Links“ sind den Verbindungsknoten ähnlich, denn auch „Links“ haben ihren Ursprung in den Knoten eines ‘tnt’-Baumes und verweisen auf den Startknoten eines anderen ‘tnt’-Baumes. Allerdings können den „Links“ sichtbare Referenzzeichen zugeordnet werden, die in der physikalischen Repräsentation des Dokumentes erscheinen. Fußnote und Fußnotentext wäre eine Anwendung dieser „Links“.

Bleibt nur noch zu sagen, daß dieses Modell sicherlich eine gute Grundlage bietet, auf der wir unsere Beschreibung der logischen Struktur entwickeln können.

4. Funktionalität

4.1. Standardfunktionalitäten

Jens-Martin Brinkmann, Frank Hettling

4.1.1. Was sind Standardfunktionalitäten?

Standardfunktionen ermöglichen erst das Arbeiten mit einem Text. Ohne die elementaren Funktionen Einfügen, Löschen, Speichern und Drucken wäre es nicht einmal möglich, einen Text zu erzeugen und diesen dauerhaft auf einem Medium festzuhalten. Aber Standardfunktionen bieten dem Benutzer noch mehr. Sie ermöglichen ihm ein komfortables und vor allem schnelles Arbeiten mit dem Text. Im Laufe der Zeit sind die Ansprüche an Textverarbeitungsprogramme immer mehr gestiegen. Durch die immer weiter sinkenden Preise auf dem Computermarkt ist es heute fast jeder Person möglich, sich einen Computer zuzulegen. Die Folge ist, daß es auch immer mehr Programme gibt, und daß die Zahl von Textverarbeitungsprogrammen stetig steigt. Eine grundlegende Standardfunktionalität kann deshalb für ein Programm nicht genau festgelegt werden, da sich die Maßstäbe durch die rasche Entwicklung immer wieder ändern. Benutzer stellen auch immer mehr allgemeinere und spezielle Anforderungen an Textverarbeitungsprogramme (siehe auch Mehrbenutzerbetrieb). Alle Standardfunktionen arbeiten grundsätzlich auf der logischen Struktur des Textes.

4.1.2. Logische Struktur & Standardfunktionalität

Die simpelste logische Struktur ist die Anordnung von Zeichen hintereinander zu einem Text. Höhere logische Strukturelemente sind z.B.: Worte, Paragraphen, Kapitel, Überschriften, Absätze, ...

Die Relation zwischen diesen einzelnen Elementen ergibt dann die logische Struktur. Ein Beispiel: Nach einer Kapitelüberschrift folgt ein Absatz mit einem oder mehreren Wörtern. Es gibt nun Standardfunktionen, die die logische Struktur ändern oder nur auf ihr arbeiten. Funktionen, die auf ihr arbeiten, können z.B. Funktionen der Formatierung sein (Beispiel Satzumbruch oder Darstellen eines Inhaltsverzeichnisses). Funktionen, die die logische Struktur ändern, wären z.B. das Löschen einer Überschrift oder das Einfügen eines neuen Absatzes.

4.1.3. Übergreifende Standardfunktionen

Man kann Standardfunktionen auf mehrere Bereiche aufteilen. Da wären z.B. Typographie, Formatierung, erweiterte Funktionalität, Ausgabe und Export. Standardfunktionen fallen nicht nur in einen Bereich hinein. So hat das Löschen einer Überschrift nicht nur Konsequenzen für die logische Struktur sondern auch für die Formatierung und z.B. für den Mehrbenutzerbetrieb. Durch das Löschen einer Überschrift, müssen nachfolgende Überschriften vielleicht neu numeriert werden.

Welche Änderungen müßten in der logischen Struktur vorgenommen werden?

- Zusammenfügen von 2 Absätzen.
- Neunumerierung der restlichen Überschriften.
- Änderung des Inhaltsverzeichnisses.

Welche Änderungen ergeben sich für die Typographie und Formatierung?

- Es entstehen neue Zeilenumbrüche.
- Es entstehen neue Seitenumbrüche.
- Es muß neu getrennt werden.
- Objekte müssen neu plaziert werden.
- Absatzstil muß geändert werden.

Welche Konflikte können beim Mehrbenutzerbetrieb entstehen?

- Überschrift ist priviligiert und darf nicht gelöscht werden.
- Absatz kann in den Bereich eines anderen Benutzers fallen.

...

4.1.4. Standardfunktionen und ihre Bereiche

Zu dem Bereich Typographie und Formatierung zählen hauptsächlich:

- Funktionen des Layout (Schriftart, Schriftstile, Seitenformat, ...)
- Funktionen für Worttrennung, Umbruch, Korrektur (Wörterbuch, ..)

Zu dem Bereich Ausgabe und Export zählen hauptsächlich:

- Die Dateioperationen (Speichern und Laden von Objekten)
- Die Erzeugung von Austauschformaten (ASCII, Postscript, TeX,...)

Zu dem Bereich erweiterte Standardfunktionalitäten zählen hauptsächlich:

- Mehrbenutzerbetrieb (siehe erweiterte Standardfunktionalitäten)

Zu dem Bereich logische Struktur zählen hauptsächlich:

- Einfügen, Ändern und Löschen von Strukturelementen (Kapitel erzeugen, ...)

4.1.5. Funktionalität auf der Benutzungsoberfläche

Abschließend noch ein kleiner Punkt zur Softwareergonomie. Auf der Benutzungsoberfläche muß das richtige Maß an Funktionalität angesiedelt werden. Deshalb ist es angebracht, einige Funktionen zu größeren Funktionen zusammenzufassen oder sie in ihnen zu verbergen. Für das Zusammenfassen der Funktionen können wir die oben genannten Bereiche benutzen.

4.2. Erweiterte Funktionalität

Guido Frick, Kai Hofmann, Trinh Le

Bei den erweiterte Funktionen handelt es sich um Funktionen, die bisher noch nicht, oder nur von sehr wenigen Programmen zur Verfügung gestellt werden. Wir erhoffen uns von diesen Features eine größere Flexibilität und vor allem eine angenehmere Arbeitsumgebung für den Anwender. Hierzu gehören vor allem folgende Punkte:

- Multiview (mehrere Sichten auf ein Dokument)
- Querverweise (für eine flexible Gestaltung von Dokumenten)
- Mehrbenutzerbetrieb (für größere Projekte, die an dem selben Dokument arbeiten)
- Sehr flexible UNDO/REDO-Funktionen
- Versionsverwaltung unter Berücksichtigung des Mehrbenutzerbetriebes

4.2.1. Installation/Deinstallation

Ein komplexes Softwareprodukt besteht in der Regel aus einer Vielzahl von Dateien, die in bestimmten Ordnern bzw. Verzeichnissen vorhanden sein müssen, damit das gesamte Programm einwandfrei ablaufen kann. Abhängig von einer konkreten Rechnerumgebung müssen in verschiedenen Dateien Anpassungen während der Installation vorgenommen werden. Um ein sicheres und fehlerfreies Installieren zu gewährleisten, muß dieser Vorgang programmgesteuert durchgeführt werden.

Möchte man mit einem Programm nicht mehr arbeiten, löscht man alle Dateien und Ordner bei denen man annimmt, daß sie zu dem jeweiligen Programm gehören. Hier kann es nicht nur dazu kommen, daß einige Dateien übersehen werden, sondern im schlimmsten Fall werden Dateien gelöscht, die nicht zu diesem Programm gehören. Es ist daher sinnvoll ein Hilfsprogramm diese Deinstallation durchführen zu lassen, da hier die gesamten Informationen vorliegen, welche Dateien wo vorhanden sind und an welchen Stellen Änderungen durchgeführt wurden (z.B. in Initialisierungsdateien).

4.2.2. Multiview

Auf ein Dokument sollten mehrere Fenster erzeugt werden. Dabei soll das System verschiedene Benutzungssichten zur Verfügung stellen:

- WYSIWYG-Sicht,
- Steuerzeichen-Sicht,
- ASCII-Sicht,
- Postscript-Sicht (zur Feinjustierung).

Die WYSIWYG-Sicht entspricht dem Aussehen des späteren Dokumentes auf dem Papier. Der Benutzer kann schon bei der Bearbeitung seines Dokumentes die Druckausgabe kontrollieren und sie nach seinen Wünschen verändern.
Unter Steuerzeichen-Sicht verstehen wir eine Sicht, die einer $\TeX$-Sicht ähnlich ist. Diese Steuerzeichen sind besonders wichtig beim geometrischen Zeichnen. Beispielsweise sind Tangenten von einem Kreis mit gegebenen Koordinaten exakt und einfacher zu erzeugen, wenn diese direkt eingegeben werden können. Auch bei der Dokumentenerstellung können wir die Steuerzeichen verwenden, anstatt einer Menüsteuerung mit der Maus.
Außerdem finden wir es wichtig, eine Baumstruktur der Gliederung zu erhalten, die manipuliert werden kann. Der direkte Zugriff auf Kapitel bzw. Unterkapitel ist durch Anklicken möglich.

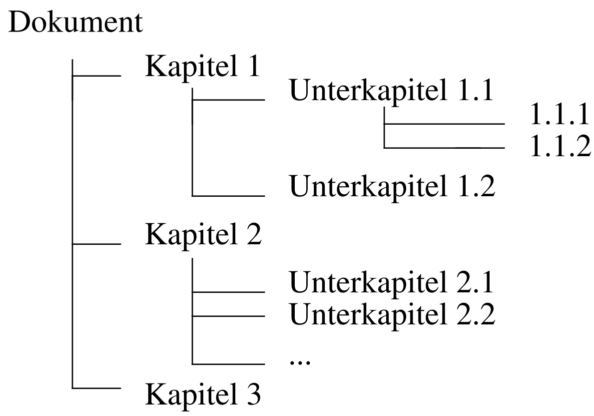

Abbildung 7. Baumstruktur

4.2.3. Textteile einfalten

Der Text kann eingefaltet werden, um z.B. Teilbereiche auszublenden. Hierdurch wird ein besserer Überblick über das Dokument geschaffen, da bekannte Teile einfach eingefaltet werden können. Dies kann z.B. bei Programmtexten sehr sinnvoll sein. Beim Einfalten werden ausgewählte logische Objekte der logischen Struktur einfach ausgeblendet. Die Falte wird durch eine entsprechende Markierung dargestellt.

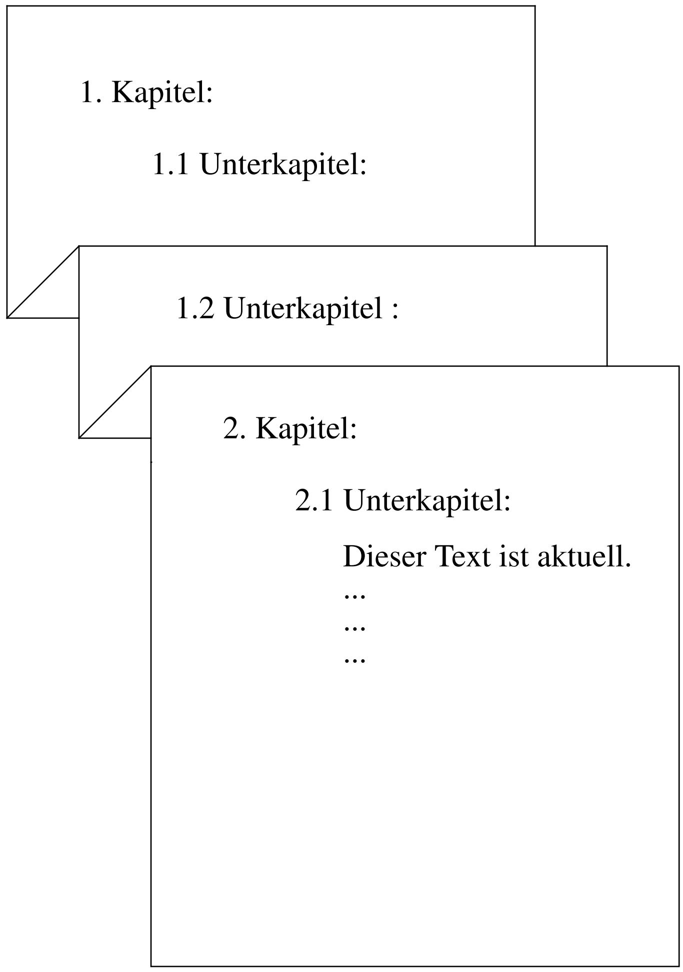

Abbildung 8. Texteinfalten

4.2.4. Vorlesen von Texten

Diese Aufgabe ist unserer Meinung nach nicht so wichtig. Trotzdem ist es interessant, ein geschriebenes Dokument bzw. Teile hieraus vorlesen zu lassen. Dabei sollte eine Unterscheidung zwischen englischen und deutschen Texten vorgenommen werden.

4.2.5. Querverweise

Zur Realisierung von verschiedenen Funktionen wollen wir die sehr flexible Methode von Querverweisen benutzen. Hierbei unterteilen wir die Querverweise in externe und interne. Diese Unterteilung geschieht aufgrund der externen Kommunikation, welche durch die externen Verweise nötig wird.

Die Querverweise sollen in der internen Struktur eingebunden sein. Für den Benutzer sollen sie explizit durch die Verwendung eines Schalters sichtbar gemacht werden können. Dies könnte durch die Verwendung einer farbigen Hinterlegung des betroffenen Teilbereichs geschehen. Querverweise müssen hierbei durch den Benutzer sowohl gesetzt als auch gelöscht werden können. Auch erscheint es sinnvoll, innerhalb eines gewissen Rahmens die freie Definition von Querverweisen zu ermöglichen.

4.2.5.1. Interne Querverweise

Bei dieser Art von Querverweisen muß gewährleistet sein, daß es möglich ist, Marken zu setzen, an deren Position eine entsprechende Art von Verweisen (z.B. alle Kapitelüberschriften für ein Inhaltsverzeichnis) aufgelistet werden kann. Diese Auflistung muß sowohl unsortiert, als auch sortiert möglich sein. Weiterhin sollen Notizverweise beim Anklicken angezeigt und editiert werden können.

Durch interne Querverweise sind folgende Funktionen als Minimum zu realisieren:

- Inhaltsverzeichnis
- Index
- Glossar
- Literaturverzeichnis
- Formelverzeichnis
- Fuß- und Endnoten
- Kopf- und Fußzeilen [für “running title”]
- Notizen (in schriftlicher Form, und optional in sprachlicher Form)
- Sowie freidefinierbare Querverweise, für z.B. eine Chemikalienliste.

4.2.5.2. Externe Querverweise

Externe Querverweise dienen als Kontakt zur Außenwelt. Hierbei sind vorallem folgende Funktionen ins Auge zu fassen:

- Der Import von Datenfeldern aus einer Datenbank.
- Die sich aus obigem Punkt ergebende Funktion der Serienbriefe.
- Der Import von Objekten, wie Grafiken, Tabellen und evtl. Ton.
- Die unter „Interne Querverweise“ dargestellten möglichkeiten für externe, programmeigene Dokumente

Eine freie Definition von externen Querverweisen ließe sich evtl. mit Hilfe einer Makrosprache realisieren.

4.2.5.3. Formulare erstellen

Wir stellen uns vor, daß die Erstellung verschiedener Formulare mit Hilfe unseres Systems schnell möglich sein sollte. Ein Beispiel dafür ist die Erstellung von Scheinformularen. Die Struktur des Scheins ist gespeichert, nur noch die persönlichen Daten des Studenten sind einzutragen. Die Struktur des Scheins wird als logische Struktur definiert, mit deren Hilfe dann das Formular dargestellt und ausgefüllt wird.

Hierbei kommen auch wieder die Querverweise zum Zuge, da z.B. die Daten, welche in den Schein eingetragen werden aus einen Datenbank geholt werden können. Somit können auch Serienbriefe usw. realisiert werden.

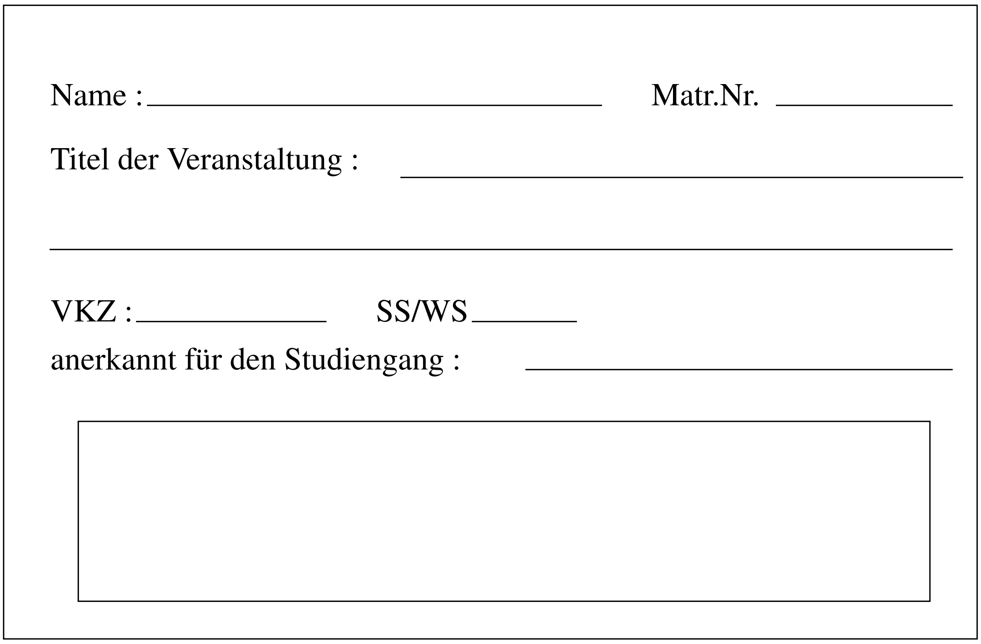
Name : ____________ Matr.Nr. ____________
Titel der Veranstaltung : ____________
____
VKZ : ____________ SS/WS ____________
anerkannt für den Studiengang : ____________

Abbildung 9. Formular

4.2.6. Versionsverwaltung

Ein Dokument ist in der Regel das Ergebnis eines langen Erstellungsprozesses, erst recht, wenn es von mehreren Personen bearbeitet worden ist. Der eine schreibt heute einen neuen Absatz, ein weiterer korrigiert morgen Textstellen usw. Ein Dokument durchläuft somit in seinem Erstellungsprozeß sehr viele Zustände.

Möchte man die "Entstehungsgeschichte" eines solchen Dokumentes nachvollziehen, so haben wir eine Möglichkeit zu schaffen, solche Zustände zu kennzeichnen und zu speichern, um bei Bedarf hierauf wieder zurückzugreifen. Neben dem reinen Dokumentenzustand soll erkennbar sein, welche Person diese oder jene Textstelle eingegeben oder verändert hat und wann. Es ist einsichtig, daß es nicht sinnvoll ist, jeden Zustand festzuhalten. Habe ich lediglich einen neuen Buchstaben eingegeben, so hat sich mein Dokument noch nicht derart geändert, daß ich dies in der "Entstehungsgeschichte" des Dokumentes nachvollziehen möchte. Wie finden wir nun aber geeignete Größen von Änderungen, um hierin einen neuen zu protokollierenden Dokumentenzustand zu erkennen? Im folgenden wollen wir einen solchen Zustand als "Version" des Dokumentes bezeichnen.

Nach jeder Sitzung, wenn alle gleichzeitig an einem Dokument arbeitenden Benutzer ihre Arbeit beendet haben, entsteht eine neue Version. Möchte man einen aktuellen Stand als Version festhalten, so wäre es nicht sinnvoll dies nur dadurch zu erreichen, daß alle Benutzer sofort die Bearbeitung zu beenden haben. Für einen solchen Fall muß es die Möglichkeit geben, explizit eine Funktion "Versionsfestlegung" aufzurufen. An alle noch an diesem Dokument arbeitenden Benutzer wird eine Nachricht gesendet, daß sie einen evtl. gesperrten Absatz (siehe Mehrbenutzer-Betrieb) freizugeben haben, etwa der Form:

"Versionsfestlegung - Bitte gesperrten Absatz freigeben!".

Wir haben oben schon angesprochen, daß neben der textuellen Erscheinung ebenfalls Autoren und Bearbeitungszeiten innerhalb der Versionsverwaltung ersichtlich sein sollen. Dies reicht jedoch noch nicht aus. In den Fällen, in denen z.B. Textteile gelöscht werden, geht dieser Textteil für die Version "verloren". Die Person, die die Löschung vorgenommen hat, wird ebenfalls nicht festgehalten. Wo kein Text, da kein Autor.

Es müssen also alle Änderungen und Löschungen festgehalten werden, um eine informative Versionsverwaltung zu gewährleisten. Autorenvermerke sollten sich aus Gründen der Übersicht jedoch nur auf Wörter, nicht auf Buchstaben innerhalb eines Wortes beziehen. Es ist zu überlegen, ob Formatierungen oder nur Zeicheneingaben bzw. -änderungen mit einem Vermerk versehen werden.

Bei dem folgenden Beispiel werden die Autorenvermerke für geschriebenen Text in “<...>” und gelöschten Text in “[...]” dargestellt:

Fritz tippt ein:

<Fritz> .... Halla Welt ...

Karl korrigiert das falsche Wort:

<Fritz> .... [Karl] Halla <Karl> Hallo <Fritz> Welt ...

Es soll jederzeit möglich sein, neben der Ansicht der Entwicklungsgeschichte eines Dokumentes, zu einer Vorversion des Dokumentes zurückzukehren und diese als aktuelle Version festzulegen. Dies führt jedoch dazu, daß alle höheren Versionsstände nicht mehr für dieses Dokument gültig sein können.

Da bei der Versionsverwaltung festgehalten wird wann welche Personen was geändert bzw. eingegeben haben, müssen hier Aspekte des Datenschutzes diskutiert werden:

Ziel der Speicherung ist es festzuhalten, wann an einer Textstelle gearbeitet wurde und wer der Autor ist. Mann kann ermitteln, ob ein Textteil überarbeitet wurde, wie aktuell z.B. ein Absatz ist (wann er erstellt wurde) und ob Änderungen in Texten anderer Autoren vorgenommen wurden. Eine personenbezogene Auswertung, wann eine bestimmte Person wie lange an einem Dokument gearbeitet hat, wird vom Programm nicht unterstützt. Dieses wäre jedoch manuell möglich, indem alle Autorenvermerke einer Person herausgeschrieben und ausgewertet werden. Es ist ersichtlich, daß dieses gerade bei umfangreichen Dokumenten mit einem enormen Aufwand verbunden ist. Die Möglichkeit einer programmgesteuerten Auswertung der Autorendaten im gespeicherten Dokument durch ein Fremdprogramm kann ggf. durch eine Codierung dieser Daten erschwert werden.

4.2.7. Mehrbenutzerbetrieb

Um das Bearbeiten von großen Dokumenten von Personengruppen zu fördern, soll ein Mehrbenutzerbetrieb möglich sein. Das dabei auftretende Problem, daß zwar mehrere Personen gleichzeitig das Dokument editieren, nicht aber gleichzeitig an ein und demselben Textabschnitt arbeiten, soll wie folgt gelöst werden:

Ein Teilbereich, der von einem Benutzer bearbeitet wird, wird für die anderen Benutzer, welche gleichzeitig an demselben Dokument arbeiten, gesperrt sein.

Diese Sperrung soll für alle anderen Benutzer durch eine Markierung des gesperrten Bereichs sichtbar gemacht werden. Dies könnte z.B. durch Hinterlegen des Abschnittes mit einem Grauton geschehen.

Als Teilbereiche, die gesperrt werden, sollten folgende über ein Menü einstellbar sein:

- Absatz
- Unterkapitel
- Kapitel.
- evtl. andere logische Elemente

Das Sperren des entsprechenden Bereiches geschieht bei der ersten Eingabe eines Zeichens innerhalb eines Bereiches automatisch. Freigegeben wird dieser Bereich dann entweder durch ein explizites Kommando, oder aber spätestens nach dem Verlassen des Bereiches durch eine Cursorbewegung.

Weiterhin sehen wir es als erforderlich an, daß es den Benutzern möglich ist, untereinander Kontakt aufzunehmen. Dies soll mittels des UNIX-Kommandos TALK geschehen.

Hierbei ist es wünschenswert, daß alle Benutzer, die gerade mit dem System arbeiten, in einer Auswahlbox angezeigt werden, so daß die Kontaktaufnahme übersichtlicher wird. Hierfür ist eine eindeutige Identifikation der einzelnen Benutzer erforderlich.

Eine weiter möglichkeit der Kommunikation unter verschiedenen Benutzern, die hierbei nicht gleichzeitig „Online“ sein müssen ist folgende Idee:

Jeder Nutzer hat die möglichkeit, eine art Folie über den Text zu legen, um auf dieser Notizen zu vermerken. Die anderen Nutzer können dann auf dieser „Notiz-Folie“ sehen, was der jeweilige Autor gemeint hat.

4.2.8. Undo/Redo

Hemmend für einen flüssigen Arbeitsablauf bei dem Umgang mit einem Computer ist die Angst, durch Fehleingaben geleistete Arbeit zunichte zu machen. Möchte man prüfen, ob der Text nicht doch mit dieser oder jener Formatierung besser wirkt, so dauert es ziemlich lange, bis der Text gesichert, umformatiert und der alte Zustand wieder eingeladen ist, da die alte Formatierung doch besser aussah.

Abhilfe kann hier eine Undo/Redo-Funktionalität schaffen, die es ermöglicht, zu unterschiedlichen Vorzuständen zurückzukehren (Undo) oder eine solche Rückkehr wieder zurückzunehmen (Redo). Dieses soll sich nicht nur auf die zuletzt getätigte Änderung beziehen, sondern möglichst viele Zustandsänderungen erfassen. Hierzu betrachten wir die Eingaben, die eine solche Zustandsänderung eines Dokumentes verursachen:

- Einfügen von Zeichen und Objekten (z.B. Grafiken, Querverweise)
- Löschen von Zeichen und Objekten
- Ändern der Position von Zeichen und Objekten
- Änderungen auf der logischen Struktur

Wir müssen nun geeignete Größen von Eingaben finden, die logisch zusammengehören und einen “Aktionsabschnitt” beschreiben. Einen solchen “Aktionsabschnitt” nennen wir im folgenden “Undo-Block”. Dieser muß die Eigenschaft besitzen, daß man von ihm eine inverse Operation bilden kann, um den ursprünglichen Zustand vor der Änderung wieder herzustellen.

Es scheint sinnvoll, die Eingabe eines einzelnen Zeichens nicht als eigenständigen Undo-Block zu betrachten. Alle Eingaben, die zwischen zwei Cursor-Bewegungen getätigt werden, inklusive Löschungen innerhalb dieses Aktionsabschnittes, werden als ein Undo-Block aufgefaßt. In allen anderen o.g. Fällen kann jede Operation als eigenständiger Undo-Block gelten. Man kann sich die Folge der Undo-Blöcke als einen Stack vorstellen, auf den jeder neue Block gelegt wird. Mit dem Befehl UNDO wird der Zustand vor den Eingaben des letzten Undo-Blockes wiederhergestellt. Dieser Block wird nun von dem Undo-Stack genommen und auf den Redo-Stack gelegt. Den UNDO-Befehl kann man nun solange wiederholen, bis kein Undo-Block mehr auf dem Undo-Stack vorhanden ist. In diesem Fall wäre der Urzustand des Dokumentes vor der Bearbeitung erreicht.

Wurde nach einem UNDO kein neuer Undo-Block erzeugt, so kann mit dem Befehl REDO das jeweils vorherige UNDO rückgängig gemacht werden. Der entsprechende Undo-Block wird vom Redo-Stack genommen und erneut auf den Undo-Stack abgelegt. Jedes Erzeugen eines Undo-Blockes führt zum Löschen des Redo-Stacks. Warum? Betrachten wir folgendes Beispiel:

Eine Eingabe e_1 überführt ein Dokument von Zustand z_a nach z_b

Auf den Undo-Stack wird dieser Undo-Block e_1 gelegt.

$$z_a \xrightarrow{e_1} z_b$$

Undo-Stack: e_1

Redo-Stack: leer

Ein UNDO führt nun die von e_1 inverse Operation e_{-1} aus und überführt den Zustand z_b zurück zu z_a:

$$z_b \xrightarrow{e_{-1}} z_a$$

Undo-Stack: leer

Redo-Stack: e_1

Ein folgendes REDO würde erneut e_1 durchführen und zu z_b zurückkehren. Erfolgt stattdessen eine weitere Eingabe e_2, so ergäbe sich ein dritter Zustand z_c, von z_a durch e_2. Zu beachten ist jetzt, daß der Redo-Stack an dieser Stelle gelöscht werden müßte. Um zu zeigen, daß dieses notwendig ist, belassen wir ihn mit e_1:

$$z_a \xrightarrow{e_2} z_c$$

Undo-Stack: e_2

Redo-Stack: e_1

Charakteristisch für UNDO bzw. REDO ist es, daß nach Anwendung dieser Operationen immer zu einem bereits vorher dagewesenen Zustand zurückgekehrt wird. Würden wir nun ein REDO durchführen, so ergibt sich durch Anwendung von e_1 ein vierter Zustand z_d:

$$z_c \xrightarrow{e_1} z_d$$

Undo-Stack: e_1 e_2

Redo-Stack: leer

Aus diesem Grund muß bei jedem Erzeugen eines Undo-Blockes der Redo-Stack gelöscht werden. Es ist also nur unmittelbar nach einem UNDO bzw. REDO möglich, den Befehl REDO durchzuführen.

5. Ausgabe und Datenaustausch

Bernd Rattey, Christian Schaefer, Kai Siegele

Aus der Aufgabenstellung, geeignete Ausgabe- und Austauschformate für unser Projekt zu sondieren, ergibt sich ein direkter Realitätsbezug zu heutigen Standards und Implementierungen.

Da die Erfahrungen unserer Gruppe zumeist aus dem Umgang mit der DOS- und Windowswelt stammen, sind auch unsere Ergebnisse sehr stark von dieser Domäne geprägt. Ein weiterer Grund für diese starke Ausrichtung in Richtung PC-Welt liegt auch darin begründet, daß es vielfältige Quellen (Bücher, Zeitschrift usw.) zu unserem Thema gibt.
Bei der Untersuchung dieses Themenkomplexes für unsere Zielplattform, dem SUN-System, hatten wir das Problem, daß aufgrund des geringen Bekanntheitsgrades von Standardprodukten und deren geringe Verfügbarkeit am Fachbereich, sich keine eindeutigen Aussagen hierzu machen lassen. Vielmehr ist das SUN-System durch den Einsatz von Public Domain (PD)- Programmen geprägt, welche wiederum mit keinen einheitlichen Standards operieren.

Es existiert ein gewisser Widerspruch in unserem Kapitel. Die Austauschformate sind stark auf die DOS-Welt zugeschnitten, während das erste Ausgabemodell stark auf die SUNs ausgerichtet ist. Man hätte die Ausgabe auch auf anderen Rechnern zusätzlich beschreiben können, dies hätte aber den Rahmen gesprengt.

Weiterhin sind im Teil Ausgabemodell auch Aspekte, die über das Thema Anforderung hinausgehen. Wir haben uns die Gedanken jedoch gemacht und halten es für sinnvoll, sie hier mitaufzuführen.

Für das weitere Verständnis, werden wir nun für die von der Adobe Company erschaffene druckerunabhängige Beschreibungssprache Encapsulated PostScript den Begriff des EPS einführen.

5.1. Datenaustausch mit Fremdprogrammen

5.1.1. Gründe für die Integration von Fremdprogrammen

Der zukünftige Benutzer einer Textverarbeitung benutzt gewöhnlich auch andere Programme. Für ihn ist es wichtig, daß Daten, die in Fremdprogrammen erfaßt wurden, auch in der Textverarbeitung zur Verfügung stehen. Sinnvoll ist der Datenaustausch mit folgenden Programmarten:

- Datenbanken
- Tabellenkalkulationen
- Textverarbeitungen
- Grafikprogramme
- Formelsatzprogramme

Für die Kommunikation mit diesen Programmarten sprechen folgende Gründe:

5.1.1.1. Datenbanken

Eine Standardanwendung im Büro ist die Erstellung eines Serienbriefes. Ein Brief mit dem gleichen Inhalt wird dabei mit verschiedenen Adressaten ausgedruckt, die aus einer Datei entnommen werden. Gewöhnlich muß zur Erzeugung eines Serienbriefes zunächst eine Textvorlage, die mit entsprechenden Befehlen versehen ist sowie eine Steuerdatei, in der die einzufügenden Datensätze in einem besonderen Format abgespeichert sind, erstellt werden. Aus diesen Bestandteilen wird von der Textverarbeitung ein Serienbrief generiert. Zur Erzeugung eines solchen Serienbriefes sollte die FORAUS-Textverarbeitung entsprechende Befehle zur Verfügung stellen.

Jedes auf dem Markt erhältliche Datenbankprogramm stellt mittlerweile Befehle zur Erzeugung von Listen zur Verfügung. Dadurch ist es möglich, daß Daten in druckaufbereiteter Form ausgegeben werden können. Schwierigkeiten treten jedoch auf, wenn solche Listen in ein Dokument integriert werden sollen, um ein einheitliches Erscheinungsbild und eine fortlaufende Seitennumerierung zu gewährleisten. Wenn der Anwender in der Textverarbeitung die Möglichkeit hat, in seinem Dokument Daten eines Datenbankprogramms einzufügen, werden solche Schwierigkeiten vermieden.

5.1.1.2. Tabellenkalkulationen

Auch Tabellenkalkulationsprogramme stellen Befehle zur Druckaufbereitung der durchgeführten Berechnungen zur Verfügung. Einzelne Felder können bereits in einer anderen Schriftart gesetzt werden, außerdem besteht die Möglichkeit, Grafiken zu integrieren. Schwierigkeiten hat der Anwender auch hier, wenn die Tabelle zusammen mit einem begleitenden Text ausgegeben werden soll. Auch hier muß die Textverarbeitung entsprechende Befehle zur Verfügung stellen.

5.1.1.3. Textverarbeitungen

Wenn der Anwender bereits mit einer Textverarbeitung gearbeitet hat und nun eine Neue einführen will, ist es für ihn besonders wichtig, vorhandene Dokumente problemlos übernehmen zu können. Für viele potentielle neue Anwender ist es abschreckend, wenn der Text eventuell neu formatiert oder gar neu eingegeben werden muß. Entsprechende Routinen zur Übernahme von Dokumenten anderer Textverarbeitungen sollten daher in der FORAUS-Textverarbeitung vorhanden sein.

Wenn der Datenaustausch zwischen zwei Textverarbeitungen gewährleistet ist, können in einem Betrieb von mehreren Benutzern unterschiedliche Textverarbeitungen benutzt werden. Viele Benutzer haben weder Zeit noch Muße, den Umgang mit einer neuen Textverarbeitung zu erlernen. Außerdem ist die Bedienung einer Textverarbeitung häufig eine Geschmacksfrage. Während einige Anwender lieber die Befehle mit Tastenkombinationen aufrufen, bevorzugen andere die Aktivierung von Aktionen über ein Menü.

5.1.1.4. Grafik und Formelsatzprogramme

Auch die in einem Grafikprogramm erstellten Bilder sollen häufig in ein Dokument integriert werden. Das Gleiche gilt für mathematische Formeln. Die Erzeugung von Formeln soll zwar auch in der FORAUS-Textverarbeitung möglich sein. Wenn jedoch Berechnungen durchgeführt werden müssen, ist dazu ein spezielles Programm notwendig.

Stehen in einer Textverarbeitung keine Befehle zur Integration von Formeln und Grafiken zur Verfügung, muß im Text Freiraum eingefügt werden, in welchem die Grafiken und Formeln nach dem Ausdruck des Dokumentes eingeklebt werden.

5.1.2. Formate zum Datenaustausch

Datenaustausch soll mit Programmen möglich sein, die wegen ihrer häufigen Verbreitung gewissermaßen einen Standard darstellen. In der Regel gibt es für Datenbanken, Tabellenkalkulationen und Textverarbeitungen leider keine einheitlichen Formate. Jedes Programm speichert seine Daten in einem anderen Format ab. Entsprechend schwierig gestaltet sich der Datenaustausch zwischen verschiedenen Programmen. Mittlerweile können die meisten Textverarbeitungen zwar Dateien anderer Programme einlesen bzw. erzeugen: gleiches gilt für Tabellenkalkulationen und Datenbankprogramme. Dazu muß jedoch stets eine interne Funktion zur Konvertierung aufgerufen werden.

Aus diesem Grunde werden wir zunächst die elementaren Formate vorstellen, die unser System auf jeden Fall beherrschen muß. Im Anschluß gehen wir auf die Formate der unterschiedlichen Programme unter verschiedenen Betriebsystemen ein.

Als einziges Standardformat steht auf allen Rechnern das ASCII-Format zur Verfügung. Sämtliche Programme können mittlerweile ASCII-Dateien lesen und erzeugen. Leider ist die Darstellungsmöglichkeit von Sonderzeichen beschränkt, außerdem gehen sämtliche Formatierungen verloren.

Zur logischen Auszeichnung von Dokumenten steht SGML zur Verfügung. SGML (Standard Generalized Markup Language) ist eine von der ISO normierte Sprache, die besonders im Verlagswesen zum Austausch von Dokumenten weit verbreitet ist. Der Text wird bei der Auszeichnung vom Autor in Textelemente wie Überschriften, Zitate und Aufzählungen eingeteilt. Mit Hilfe von Textverarbeitungen, die mit SGML-Strukturen arbeiten, können den Textelementen anschließend Layout-Funktionen, wie z.B. Schriftart, zugeordnet werden. Nur wenige Textverarbeitungen wie Daphne und Grif sind jedoch in der Lage, das SGML-Format einzulesen und zu verarbeiten. Die meisten zur Zeit auf dem Markt erhältlichen Textverarbeitungen arbeiten mit grafisch orientierten Auszeichnungsverfahren. Dabei wird eingegeben, welche Textstücke in welcher Weise gesetzt werden sollen. Das SGML-Format wird leider nicht unterstützt.

Zur Erzeugung von Druckausgaben in hoher Qualität hat sich PostScript durchgesetzt. Die Sprache dient zur Beschreibung von Text und Grafiken. Das PostScript-Format basiert auf einer ASCII-Darstellung und ist daher vom Betriebssystem unabhängig. Es eignet sich auch zum Im- und Export von Grafiken.

Im Bereich der Grafik haben sich neben PostScript zwei Standards etabliert, die von vielen Programmen auf unterschiedlichen Betriebssystemen im- und exportiert werden, hierbeihandelt es sich um das sogen. GIF-Format (Graphic Interchange Format) und das PCX-Format, wobei ersteres am weitesten verbreitet ist.

Für Tabellenkalkulationen, Datenbanken und Formelsatzprogramme haben wir keine übergreifenden Standards ausmachen können, der Bereich ist stark rechnerabhängig.

Wir müssen uns daher überlegen, welche rechnerabhängigen Formate unser System verarbeiten sollte. Für jedes der Formate müssen Routinen zur Umwandlung implementiert werden. Denkbar ist hier beispielsweise das WORD-Format, das auf vielen Rechnern bereits vorhanden ist. Eine genaue Bestimmung der Formate muß dann in der Spezifikation definiert werden.

Eine lose Zusammenstellung einiger Programme befindet sich in der im Anhang befindlichen Tabelle.

Die Konvertierungsroutinen sind hier weniger wichtig, da eine genaue Formatspezifikation in der Literatur zu finden ist.

Das Format der meisten Exportdateien wird ausführlich beschrieben in [Bor90]

5.1.3. Möglichkeiten Fremdprogramme zu integrieren

In der FORAUS-Textverarbeitung müssen Routinen vorhanden sein, um Fremdprogramme integrieren zu können. Folgende Möglichkeiten haben wir uns dabei überlegt:

5.1.3.1. Datenaustausch über eine Clipboard-Funktion

Eine Clipboard-Funktion ist eine Art Hilfsprogramm, in das aus einem Anwendungsprogramm heraus Daten abgespeichert bzw. ausgelesen werden können. Die Zwischenablage funktioniert nach dem folgenden Prinzip: Der Anwender bearbeitet z.B. in einer Textverarbeitung ein Dokument. Wenn er nun einen Textausschnitt ausschneidet oder ihn zum Kopieren markiert, wird dieser ohne weiteres Zutun des Benutzers in die Zwischenablage geschrieben. Nun kann die Textverarbeitung verlassen, und ein anderes Programm aufgerufen werden. Wählt der Benutzer dort die Option *Einfügen*, wird, ohne daß weitere Aktionen des Benutzers notwendig sind, der Text aus der Zwischenablage eingefügt. Die Idee der Clipboard-Funktion wurde erstmalig auf dem Apple MacIntosh realisiert und wird vom Betriebssystem optimal unterstützt.

Im Clipboard sind in MS-Windows fünf verschiedene Formate intern gespeichert. Jedes in das Clipboard eingefügten Objekte wird in eines der fünf Formate umgewandelt. Damit stellt das Clipboard nur eine Konvertierung der Dateien dar (siehe unten).

Eine ähnliche Funktion wird auch unter XWindows/XView angeboten. Dort wird lediglich ein Zeiger auf das zu bearbeitende Objekt übergeben. Entsprechende Konvertierungsroutinen (siehe unten) müssen die Daten in das eigene Format umwandeln.

Der Datenaustausch über eine Clipboard-Funktion ist leicht zu handhaben und funktioniert im Allgemeinen ohne Probleme.

5.1.3.2. Referenzen auf andere Programme

Statt die Daten, die man mit einem Fremdprogramm erzeugt hat, in das Dokument einzufügen, wird lediglich ein Befehl in den Text geschrieben, der auf eine Datei verweist. Dazu werden in der Regel Ikonen oder leere Flächen der entsprechenden Größe verwendet. Wenn das Dokument schließlich ausgedruckt wird, fügt die Textverarbeitung den Inhalt der Datei an der gekennzeichneten Stelle ein. Dazu wird das referenzierte Objekt in das eigene Format umgewandelt und in die logische Struktur des Dokumentes eingefügt. Ein Vorteil dieses Verfahrens ist es, daß die Verarbeitung des referenzierten Objektes unabhängig von der Textverarbeitung geschehen kann. Ein weiterer Vorteil ist die geringere Größe des Textdokumentes.

5.1.3.3. Konvertierung der Dateien anderer Programme in das eigene Format

Die letzte Möglichkeit besteht in der Bereitstellung von Konvertierungsroutinen. Bevor Daten anderer Programme ins Dokument integriert werden, wird ein Übersetzungsprogramm aufgerufen, welches die Dateien eines Fremdprogrammes in ein eigenes Format konvertiert. Umgekehrt sollte auch die Übersetzung des eigenen Formates in Fremdformate möglich sein. Jedes Programm legt seine Daten in einem spezifischen Format ab. Dieses Format sollte beim Datenaustausch von der FORAUS-Textverarbeitung erkannt werden.

Über die Art und Anzahl der letztendlich verwendeten Konvertierungsroutinen muß in der späteren Spezifikation festgelegt werden. Als Entscheidungshilfe kann die im Anhang aufgeführte Tabelle 1 angesehen werden.

Der Datenaustausch über Konvertierungsroutinen hat den Vorteil, daß die Formate in der entsprechenden Literatur sehr gut dokumentiert sind.

Es entsteht der Nachteil, daß die Übersetzung bei längeren Dateien eventuell recht zeitaufwendig ist. Außerdem können nicht einzelne Teile einer Datei eingelesen werden.

5.2. Die Ausgabe

5.2.1. Das Idealbild

Wir werden zunächst kurz das Idealbild skizzieren, da die gesamte Anforderung das Idealbild einer Textverarbeitung charakterisieren soll. Die Ausgabe unserer Textverarbeitung hängt von vier Komponenten ab. Zum Einen haben wir den Monitor als Ausgabemedium. Um den Monitor vom Rechner anzusteuern, benötigt man Grafikkarten, die die Steuerung realisieren. Die beiden Komponenten beeinflussen die Ausgabe eines Dokumentes auf dem Bildschirm.

Eine weitere Art von Ausgabe ist die auf Geräten, die ein physikalisches Ergebnis des Dokuments produzieren, gemeinhin fallen hierunter alle Arten von Druckern.

Der letzte Aspekt der Ausgabe ist der Export, der im vorhergehenden Abschnitt detailliert erläutert wurde.

Um nun das Idealbild der Ausgabe zu skizzieren, beschreiben wir die drei erstgenannten Klassen.

Unsere Textverarbeitung sollte idealerweise jede Art von Monitoren unterstützen, um auf die individuelle Rechnerkonfiguration eines jeden Benutzers einzugehen. Es sollten also folgende Klassen von Monitoren unterstützt werden:

- Farbmonitore
- Schwarzweißmonitore
- LCD-Bildschirme
- Plasma-Bildschirme

Bei den Grafikkarten gibt es für bestimmte Rechner inzwischen eine unübersehbare Anzahl von verschiedenen und inkompatiblen Grafikkarten. Ein für die Ausgabe wichtiger

Punkt ist, daß alle Grafikkarten mit ihren Optimierungen unterstützt werden, um ein bestmögliches Bild auf dem Monitor zu erreichen. So müssen z.B. spezielle Grafikkarten mit eigenen Prozessoren auf der Karte anders angesteuert werden als “normale” Karten. Mit solchen Karten muß der Vorgang des Bildschirmblätterns (Scrollen) nicht mehr von der CPU des Rechners berechnet werden, sondern vom Prozessor der Grafikkarte.

Das Thema Grafikkarten ist sehr rechnerabhängig. Während es auf den SUN-Rechnern fast nur die CG-3-Grafik gibt, existieren in der MS-DOS-Welt Dutzende von völlig unterschiedlichen Karten und Standards.

Daraus folgt für unsere Textverarbeitung, daß sie möglichst viele der Standards unterstützen sollte und entsprechende Treiber zur Verfügung stellen müßte.

Die letzte Klasse der Ausgabemedien, die Drucker, läßt sich in diverse Unterklassen unterteilen. Auch hier muß man davon ausgehen, daß die BenutzerIn die vorhandenen Geräte in maximaler Qualität nutzen möchte. Folgende Klassen von Druckern sollten unterstützt werden:

- Belichtungsmaschinen
- Postscriptdrucker
- Laserdrucker
- Tintenstrahldrucker
- Nadeldrucker
- Farbdrucker
- Thermodrucker

Die Betonung bei der Unterstützung der Drucker liegt darin, daß das Gerät in wirklich maximaler Qualität angesteuert wird. Es ist unbefriedigend, wenn man einen hochwertigen Drucker, der in der Lage ist in einer Qualität von 600 DPI zu drucken, mit nur 180 DPI unterstützt. Das Ergebnis ist dann minderwertig.

Beim Betrachten der Aufzählungen stellt man fest, daß die Ausgabe einer idealen Textverarbeitung äußerst variantenreich ist. Eine Unterstützung aller Monitor-, Grafikkarten- und Druckerklassen ist viel zu aufwendig. Zunächst muß für jedes Gerät ein Treiber entwickelt werden und dann müssen die Treiber auch immer auf den aktuellsten Stand gebracht werden, da die Entwicklung der oben aufgeführten Geräte ständig voranschreitet und die Produkte eine relativ geringe Produktlebenszeit besitzen.

Aus den Gründen und denen in der Einleitung zu diesem Abschnitt gemachten Prämissen folgt, daß wir in der Anforderung nicht versuchen, alle diese Besonderheiten zu berücksichtigen. Unser Ansatz wird nicht alle Geräte unterstützen und das Treiberkonzept nicht verwenden. Trotzdem wird aber eine Vielzahl von Klassen damit abgedeckt, die für die Belange einer Universität ausreichen.

5.2.2. Die Bildschirmausgabe

Da es sich bei der hier zu beschreibenden Textverarbeitung um ein Mehrsichtensystem handelt, gehen wir zunächst auf die unterschiedlichen Sichten ein. Wir gehen davon aus, daß folgende Sichten existieren:

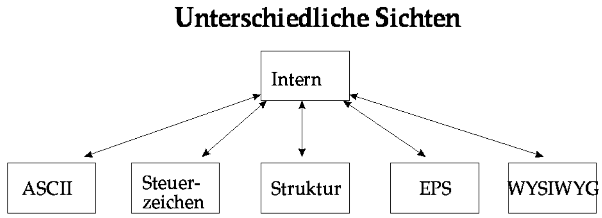

Abbildung 10. Unterschiedliche Sichten

Die Sicht EPS ist eigentlich keine eigenständige Sicht sondern nur ein Format, zur besseren Übersicht wird sie hier aber als Sicht aufgeführt.

Für die Darstellung der Sichten läßt sich eine Zweiteilung der Sichten vornehmen: Die Sichten ASCII, Steuerzeichen und Logische Struktur bestehen aus ASCII-Zeichen. Sie können daher in normalen Textfenstern dargestellt werden. Dieses läßt sich relativ einfach realisieren.

Aus dem Rahmen fällt die Sicht WYSIWYG. Die Darstellung dieser Sicht ist nicht simpel und die nun folgenden Ausführungen behandeln ein Modell zur Ausgabe des WYSIWYG-Sicht.

Die Ausgabe der WYSIWYG-Sicht läßt sich nicht mit ASCII-Zeichen bewerkstelligen, da sich das Dokument aus verschiedenen Zeichensätzen aufbauen kann oder die einzelnen Zeichen attributiert sein können (Fett, Kursiv, ...). Weiterhin kann das Dokument auch Grafiken und Sonderzeichen wie z.B. griechische Buchstaben für mathematische Formeln enthalten. Daher muß die WYSIWYG-Sicht als Grafik dargestellt werden.

5.2.2.1. Idee des Ausgabemodells

Das nun folgende Modell zeigt einen Weg auf, wie man das Dokument in der WYSIWYG-Sicht als Grafik darstellen kann.

Der Aufbau der Grafik soll unter Ausnutzung der Routinen des Betriebssystem geschehen. Jedes Betriebssystem und/oder die darauf aufsetzende Benutzungsoberfläche[1] liefert schon eine Anzahl von Routinen, die die Darstellung von z.B. Texten in Grafikfenstern unterstützen. Das Problem liegt nun darin, daß die Routinen von System zu System völlig unterschiedlich sein können.

1. wir fassen die beiden Begriffe im folgenden zu “*System*” zusammen

Daher sieht unser Modell vor, daß der Aufbau des WYSIWYG-Fensters in einem zentralen Modul vorgenommen werden soll. Das zentrale Modul (Black Box) besitzt genau definierte Schnittstellen, die für jedes System identisch sind. Der innere Aufbau der Black Box ist dagegen von System zu System unterschiedlich, da die Black Box die Routinen des Systems ausnutzen soll.

Folgende Grafik verdeutlicht das Modell:

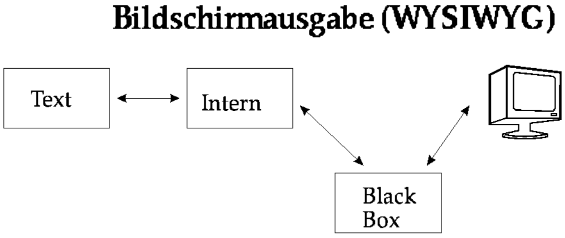

Abbildung 11. Bildschirmausgabe

Der Text des Dokuments, der vom Formatierer in die gewünschte Form gebracht worden ist, wird über die Black Box auf dem Bildschirm dargestellt.

Gegeben sei folgende Anweisung des Formatierers:

> Setze den Text “Hallo Welt” 10cm vom oberen und 3cm vom linken Rand in der Schrift Times Roman 12 Punkt mit dem Attribut Fett und umrahme den Text mit einem dünnen Rahmen.

Die Anweisung ist völlig unabhängig vom System und sieht daher für alle System exakt so aus. In der Black Box existieren nun Schnittstellen, in denen der Black Box die Informationen (Fett, Times Roman, “Hallo Welt”, ...) übergeben werden.

Die Black Box führt die Anweisung nun mithilfe der Routinen des entsprechenden Systems aus und setzt den Text auf dem Bildschirm.

Aus dem Gesagten folgt, daß die Black Box für jedes System neu geschrieben werden muß, während alle anderen Teile der Textverarbeitung wie z.B. der Formatierer für alle Systeme identisch sind.

5.2.2.2. Die Black Box

Im vorhergehenden Abschnitt wurde gesagt, daß die Black Box die Routinen des Betriebssystem und der Benutzungsoberfläche (des “Systems”) ausnutzen soll, um die WYSIWYG-Sicht aufzubauen. Eine andere Möglichkeit wäre, in der Black Box eine eigene Benutzungsoberfläche zu integrieren, was natürlich zu einer sehr guten Anpassung der Black Box an die Bedürfnisse der Textverarbeitung führen kann. Mit einem solchen Ansatz tauchten aber genau die Probleme auf, die wir bei den Bemerkungen zum Idealbild der Ausgabe gemacht haben.

Aus dem Grund soll unsere Black Box auf ein System aufsetzen. Da wir uns an den realen Gegebenheiten orientieren wollen und die Realität an der Universität SUN heißt, schlagen wir als Benutzungsoberfläche XWindows vor. Für XWindows existiert eine Erweiterung, XView, die die Programmierung von XWindows erleichtert und viele Routinen bietet, die für die Black Box nützlich sein können.

Die Black Box kann dann z.B. die schon vorhandenen Zeichensätze für den Aufbau von Fenstern, usw. nutzen.

Der wichtigste Punkt der Black Box ist die Definition der Schnittstellen. Ein Fehler wäre beispielsweise folgende fiktive Idee:

Wir wissen, daß XView 25 Zeichensätze hat. Daher geben wir jedem Zeichensatz eine Nummer und definieren die Schnittstelle nun so, daß der Formatierer nur noch die Nummer an die Black Box übergibt.

Das führt natürlich dazu, daß man auch auf anderen Systemen genau 25 Zeichensätze unterstützen muß und kann. Weiterhin muß man dann auf einem anderen System genau die Zeichensätze haben, die es unter XView gibt, denn sonst ist die Darstellung eines Dokumentes auf zwei unterschiedlichen Systemen nicht identisch.

Die Black Box sollte daher folgenden Kriterien genügen[1]:

- Die Anzahl der Schnittstellen sollte minimal sein. Dadurch wird gewährleistet, daß eine Änderung der Schnittstelle (z.B. Hinzunahme eines weiteren Parameters) in den betroffenen Modulen einfacher vorzunehmen ist.
- Die Schnittstellen sollten schmal sein, d.h. sie sollten nicht zu komplex sein. Sie sollen so knapp wie möglich sein, die Black Box soll möglichst unabhängig sein. Beispielsweise soll nicht vorgegeben werden, ob im obigen Beispiel erst der Rahmen gezeichnet und dann der Text danach ausgegeben wird, oder ob die Reihenfolge verdreht wird.
- Das Geheimnisprinzip soll in der Black Box gewahrt bleiben.

Die genaue Definition der Schnittstellen der Black Box muß dann in der Spezifikation festgelegt werden.

5.2.2.3. Bewertung des Modells

Das Modell, daß auf den ersten Blick simpel und einleuchtend erscheint, hat neben einigen Vorteilen auch Nachteile, daher wollen wir eine Bewertung vornehmen.

5.2.2.3.1. Vorteile

Da die Grafik, die die WYSIWYG-Sicht darstellt, in der Black Box komplett mit "eigenen" Routinen entwickelt werden soll, ist es relativ leicht möglich, im WYSIWYG-Fen-

1. Diese Punkte haben nichts mit dem Idealbild einer Textverarbeitung zu tun, sie sollen nur helfen, zum Verständnis der Black Box beizutragen...

ster Änderungen vorzunehmen, also zu editieren. Mit den eigenen Routinen kann man die Position jedes Wortes, Buchstabens oder anderen Objektes speichern. Wird nun per Maus ein Bereich angeklickt, kann der sich dort befindliche Text erkannt werden und eine Änderung auch in den anderen Sichten stattfinden.

Weiterhin verzichten wir im Unterschied zum zweiten Modell auf Fremdprogramme, wir bleiben also unabhängiger.[1]

Durch die Nutzung der Routinen des Betriebssystems und der Benutzungsoberfläche wird zum Einen der Umfang der Textverarbeitung auch für unsere Bedürfnisse akzeptabel, zum Anderen lehnen wir uns auch an Standards an, die dem späteren Benutzer vertraut sind, damit vereinfacht sich die Bedienung des Systems.

Die letzten beide positiven Aspekte beziehen sich auf das System, das an der Universität existiert. Speziell für die Implementierung der Black Box unter XWindows/XView gilt, daß es sich dabei um ein frei kopierbares System handelt, dessen Sourcen vorliegen. Daher tauchen keine Probleme mit Lizenzen auf. Weiterhin ist das System recht gut dokumentiert, was die spätere Implementierung vereinfacht.

XWindows ist inzwischen für mehrere Systeme verfügbar und stellt einen Quasistandard dar. Aus dem Grunde ist es möglich, die Black Box eventuell auch in anderen Systemen unverändert zu übernehmen oder nur geringfügig zu modifizieren. Die Benutzungsoberfläche XWindows ist beispielsweise auch auf Intel-Rechnern (MS-DOS-Welt) mit dem freien Betriebssystem LINUX verfügbar.

5.2.2.3.2. Nachteile

Ein Nachteil liegt darin, daß nur die Ausgabemedien unterstützt werden, die auch vom Betriebssystem und der Benutzungsoberfläche unterstützt werden.

Der Hauptnachteil ist jedoch das Zusammenspiel mit dem Drucker. Da die Bildschirmsicht unabhängig von der Ausgabe des Druckers berechnet wird, können sich Differenzen zwischen den beiden Sichten ergeben, dies wird nach der Vorstellung des Modells zur Druckausgabe näher erläutert.

5.2.3. Ausdruck

Für den Prozeß des Druckens taucht das Problem auf, daß wir mit minimalem Aufwand auf möglichst vielen Druckern ein hochwertiges Ergebnis erhalten wollen.

Daher soll der Text, dargestellt durch die Sicht “Intern” nach Encapsulated PostScript (kurz EPS) gewandelt werden. EPS ist eine sogenannte Druckerbeschreibungssprache, ein PS-fähiger Drucker liest eine EPS-Datei und führt die Befehle aus. Um einen Kreis mit dem Radius 200 an der Position (100,100) zu zeichnen, enthält die Datei beispielsweise den Befehl

1. Vgl. Abschnitt, alternatives Modell

moveto(100,100)
circle(200)

Wenn die Sicht “Intern” nun in eine EPS-Datei (im folgenden auch EPS-Sicht genannt) gewandelt worden ist, kann das Dokument auf einem PS-fähigen Drucker in maximaler Qualität gedruckt werden.

Die Grafik verdeutlicht das Prinzip:

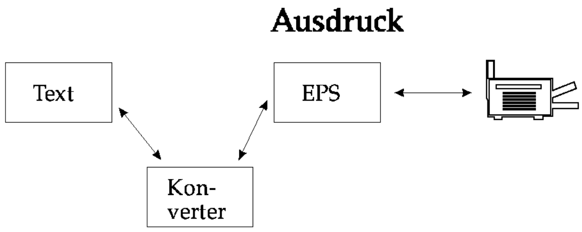

Abbildung 12. Ausdruck

Damit wird natürlich nur eine Klasse von Druckern unterstützt. Um auch auf anderen Druckern zu einem Ausdruck zu kommen, muß die EPS-Datei mithilfe eines Wandlungsprogrammes in eine entsprechende Bitmapgrafik konvertiert werden. Hierfür existiert das frei kopierbare Programm GhostScript, das eine EPS-Datei auf den verschiedensten Drukker ausgeben kann. So lassen sich auch andere Arten von Druckern wie beispielsweise Laser- oder Nadeldrucker nutzen.

5.2.3.1. Vorteile

Ein Vorteil dieses Modells ist die hohe Qualität, die sich so erreichen läßt. Für die Sprache PostScript existiert eine Vielzahl von Zeichensätzen, die verwendet werden können. Auch Grafiken lassen sich so problemlos in das Dokument einbinden.

PostScript hat sich in den letzten Jahren zu einem weit verbreiteten Standard entwickelt, der systemübergreifend existiert. Damit ist das Modell eine systemunabhängige Lösung, die Portierbarkeit der Textverarbeitung ist für die Ausgabe sehr einfach.

Während PS-fähige Drucker in den letzten Jahren noch sehr teuer waren, werden sie nun immer billiger. Es ist abzusehen, daß ihr Verbreitungsgrad zunehmen wird.

Ein Effekt der Lösung ist, daß die zusätzliche Sicht EPS existiert. Damit ist das Exportieren des EPS-Formates automatisch möglich, es werden dafür keine weiteren Übersetzer benötigt.

5.2.3.2. Nachteile

Die Lösung ist für PS-fähige Drucker sehr effizient. Alle Drucker, die nicht in die Kategorie fallen, benötigen jedoch ein Fremdprogramm zur Umwandlung, hier ist pro Ausdruck ein entsprechender Aufruf nötig, der unter Umständen einige Zeit in Anspruch nimmt.

Das Hauptproblem der vorgestellten Lösung ist jedoch der Umstand, daß die Sichten Bildschirm und EPS unterschiedlich berechnet werden. Es kann im WYSIWYG-Fenster zu einer Darstellung des Dokumentes kommen, die nicht der des Druckbildes entspricht. Wie groß die Abweichungen sein können, läßt sich hier nicht bestimmen.

Es werden zwei Maschinen zur Berechnung der Darstellung des Dokumentes verwendet, die ihre Daten evtl. aus unterschiedlichen Quellen beziehen. So stammt der Zeichensatz zur Darstellung auf dem Bildschirm aus XWindows, der für den Drucker ist ein Postscriptzeichensatz. Jede der beiden Maschinen skaliert den Zeichensatz evtl. unterschiedlich, sodaß z.B. ein zusätzlich im Dokument befindlicher Kreis im Vergleich zum Text auf dem Bildschirm kleiner wirkt als auf dem bedruckten Papier.

Eine wichtige Festlegung für die Spezifikation wird daher sein, die Maschinen mit denselben Algorithmen und Quellen zu versorgen. So muß der Bildschirmaufbau z.B. mit Postscriptzeichensätzen erfolgen, die auch für die Drucker existieren.

5.3. Alternatives EPS - Modell.

Für das nun folgende alternative Ausgabemodell, welches wir im Anschluß vorstellen möchten, werden wir vorab erst einmal eine Paar Annahmen treffen, damit sich ein solches Modell unter Berücksichtigung des derzeitigen Wissenstands auch realisieren läßt.

Annahmen zum Modell:

- Es wird angenommen, daß es uns möglich ist, eine Rücktransformation von Daten aus der WYSIWYG-Sicht mit Hilfe ihrer EPS-Repräsentation in die interne Datenstruktur unseres FORAUS-Textes durchzuführen.
- Weiterhin wird angenommen, wenn eine solche Rücktransformation möglich ist, daß der Implementierungsaufwand in der uns zur Verfügung stehenden Zeit vertretbar ist.

Annahmen zu technischen Voraussetzungen:

- Es wird vorausgesetzt, daß EPS fähige Ausgabemedien zur Verfügung stehen z.B. für den Bildschirm das Programm GhostView. Als Minimalanforderung gilt hier, daß der EPS Bildschirminterpreter mit der gleichen Auflösung arbeitet soll, wie das angeschlossene Druckermedium (Laserdrucker).
- Auch wird vorausgesetzt, daß die Programme mit denen ein Datenaustausch betrieben werden soll, eine EPS - Schnittstelle besitzen, über welche man den Datenaustausch betreiben kann.

5.3.1. Von FORAUS über EPS nach WYSIWYG

Nachdem wir die Voraussetzungen für unser Modell hinreichen spezifiziert haben, wollen wir damit beginnen, den ersten Schritt zu erklären. Dieser besteht in der Transformation von FORAUS-Textdaten über ihre EPS -Repräsentation hin in ihre WYSIWYG-Darstellung.

Wir beginnen unsere Erklärung mit der Annahme, daß es einen kontinuierlichen und ausgezeichneten Textstrom aus der internen Struktur in den FORAUS-Formatierer gibt (vgl. Abbildung 10). Dieser hat die Aufgabe, den Textstrom in einzelne Seiten zu unterteilen, sie physikalisch zu beschreiben und den Text entsprechend seiner Auszeichnungen zu setzen. Zu den physikalischen Eigenschaften einer solchen FORAUS-Textseite gehören z.B.:

- interne Seitennummer
- Seitengröße
- Standardfont
- Standardfontgröße.

Jede so erzeugte Seite wird in einem FORAUS-Seitenfile abgelegt und bei der FORAUS/EPS- Seitenverwaltungseinheit angemeldet. Diese soll nach unserer Meinung folgende Funktionalitäten besitzen:

- jede FORAUS-Seite soll in ihre aktuelle EPS-Notation mittels FORAUS->EPS Übersetzer überführt werden
- jede Änderung in der internen Struktur, sei es textuell oder in den Auszeichnungen, zieht eine entsprechende Neuformatierung und Neuübersetzung der betroffenen Seiten nach sich
- die direkte Visualisierung von Änderungen wird erzwungen, wenn sie den angezeigten WYSIWYG-Seitenbereich betrifft
- eine allgemeine Steuerung der WYSIWYG-Sicht, d.h. die Steuerung des Anzeigeprozesses der EPS-Textseiten mittels eines entsprechenden EPS-Interpreters (GhostView) auf dem Bildschirm

Wird nun durch die FORAUS/EPS-Seitenverwaltungseinheit ein Übersetzungsvorgang einer FORAUS-Textseite nach EPS angestoßen, so wird dieser mittels der physikalischen Seitenbeschreibung, den Auszeichnungen und dem Text eine WYSIWYG-Darstellung in eindeutiger EPS-Notation erzeugt. Ist dieser Übersetzungsvorgang abgeschlossen, so wird der Seitenverwaltungseinheit mitgeteilt, daß die FORAUS-Seite in einer eindeutigen EPS-Notation vorliegt.

Ab dieser Stelle bietet unser Modell nun drei Möglichkeiten zur Ausgabe auf einem entsprechenden Medium. Die erste besteht in der Darstellung auf dem Bildschirm, die zweite in der direkten Ausgabe auf einem postscriptfähigen Drucker und die dritte in der EPS-Notation selbst, als Austauschformat für andere Programme.

5.3.2. Von WYSIWYG über EPS und FORAUS in die interne Struktur

Bevor wir nun aber die Vor- und Nachteile eines solchen Modells vorstellen, wollen wir erst noch den von uns erdachten Rückweg in die interne Struktur vorstellen (vgl. Abbildung 14). Wir beschränken uns hierbei auf die Betrachtung einer Seite, da der Vorgang für alle weiteren Seiten gleich ist. In der Rücktransformation nimmt eine von uns erdachte “Beziehungseinheit” eine zentrale Rolle ein. Auf der einen Seite stehen Kommandos und Änderungen, die auf sie einwirken, auf der anderen Seite steht das FORAUS-Seitenfile und die entsprechende EPS-Notation sowie die interne Struktur.

Ihre Aufgabe besteht nun darin, die Kommandos, Änderungen und Seitenfiles miteinander in Beziehung zu setzen, um somit auf die interne Struktur einwirken zu können. Um eine solche Beziehung herstellen zu können, muß die “Beziehungseinheit” die nachfolgenden Relationen ermöglichen.

- Sie muß einen Bezug zwischen der aktuellen Bildschirmposition und der dazugehörigen Position in der EPS-Notation herstellen.
- Dann muß sie eine Relation zwischen der EPS-Position und der im FORAUS-Seitenfile aufbauen.
- Sind die beiden vorangegangenen Beziehungen aufgebaut, so muß ein Bezug zwischen der internen Struktur und dem FORAUS-Seitenfile hergestellt werden.
- Erst jetzt, da alle Relationen aufgebaut sind, kann nun mit der Auswertung der Kommandos und den Änderungen begonnen werden. Hierfür ist von besonderer Bedeutung, daß zwischen einer Auszeichnung und ihrer EPS-Umsetzung eine eindeutige Beziehung besteht. Ist das Kommando oder die Änderung eindeutig den betroffenen Objekten z.B. einer Auszeichnung zugeordnet worden, so kann nun die Beziehungseinheit mit der Festschreibung der Änderung in der internen Struktur beginnen. Anschließend wird der Prozeß zur Neuformatierung und Neuübersetzung initiiert.

Nachdem wir nun das Modell in seinen beiden Richtungen beschrieben haben, wollen wir nun die Vor- und Nachteile eines solchen Modells, sowie die mögliche Produktunterstützung aufzeigen. Zu den Vorteilen eines solchen Modells zählen,

- daß das verwendete Ausgabeformat (EPS) ein betriebssystemunabhängiges und weitestgehend genormtes Format ist.
- Dokumente müssen für die Ausgabe und für den Austausch nur einmal aufbereitet werden, somit entfällt die spezifische Aufbereitung für das jeweilige Ausgabemedium.
- Von dem Projekt FORAUS wird jeweils nur ein EPS-Interpreter für die jeweilige Klasse von Ausgabemedien implementiert, wobei man hier auf eine Produktunterstützung für Klassen der Bildschirme und der Nadeldrucker zurückgreifen kann.
- Es wird für jede Klasse von Dokumenten immer eine WYSIWYG-Sicht erstellt. Art und Umfang der Auszeichnungen spielen dabei keine Rolle.
- Für jede Sicht auf das FORAUS-Dokument braucht nur ein und derselbe Übersetzungsprozeß für die Hin- und Rückrichtung erzeugt werden.

Zu seinen Nachteilen zählen,

- ob die von uns am Anfang gemachten Annahmen zutreffend sind.
- Weiterhin können Konflikte mit den Funktionalitäten der Gruppe Advanced-Feature in dem Bereich des Mehrbenutzerbetrieb auftreten. Die Konflikte in diesem Bereich könnten bei den Zugriffsrechten und dem Filehandling liegen.
- Es ist auch nicht geklärt, ob die gegebene Produktunterstützung sich auf die von uns gewünschten Funktionalität erweitern läßt und wenn, ob der Aufwand gerechtfertigt ist. In diesem Zusammenhang ist auch nicht die lizenzrechtliche Seite abgeklärt.
- Ein weiteres Gebiet, welches mit Problemen bei diesem Modell behaftet ist, ist das der Integration von Fremdprogrammen. Hier kommt nur die Einbindung per EPS-File in Frage, da die des Clipboards und der der Referenz zuvor eine Speicherung im EPS-Format bedingen, weil interne Bitformate durch das Modell nicht unterstützt werden.

Da wir nun die Vor- und Nachteile betrachtet haben, kommen wir nun zur einer kurzen Produktbeschreibung von GhostView und GhostScript, welche wir als Produktunterstützung für unser Modell gewählt haben.

GhostView und GhostScript sind EPS-Interpreter, die die Ausgabe von EPS-Dateien in sehr hoher Qualität auf dem Bildschirm und auf dem Nadeldrucker realisieren. Beide Produkte sind in der Programmiersprache C geschrieben und liegen in ihren Quelltexten im Netz vor. Da sie in C implementiert worden sind, können sie auf die unterschiedlichsten Systeme portiert werden. Dies ist ein weiterer Grund für ihre Verwendung als Ausgabehilfsmittel, wenn die Portierbarkeit des System mit eine Rolle im Entscheidungsfluß spielen soll.

Wägt man nun aber die Vor- und Nachteile eines solchen Modells mit seinen Annahmen ab, so muß man zu dem Schluß kommen, daß eine Realisierung unter den gegebenen Umständen in der von uns angeführten Form nicht möglich sein wird. Wohl aber, daß das EPS-Format eine große Rolle bei den Austauschformaten spielen wird.

Von FORAUS über EPS nach WYSIWYG !

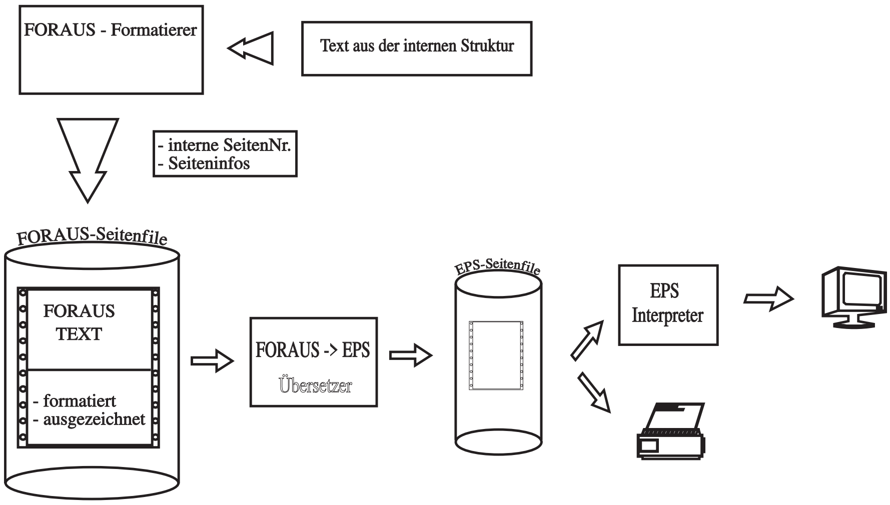

Abbildung 13. Foraus -> WYSIWYG

Von WYSIWYG über EPS und FORAUS in die interne Struktur !

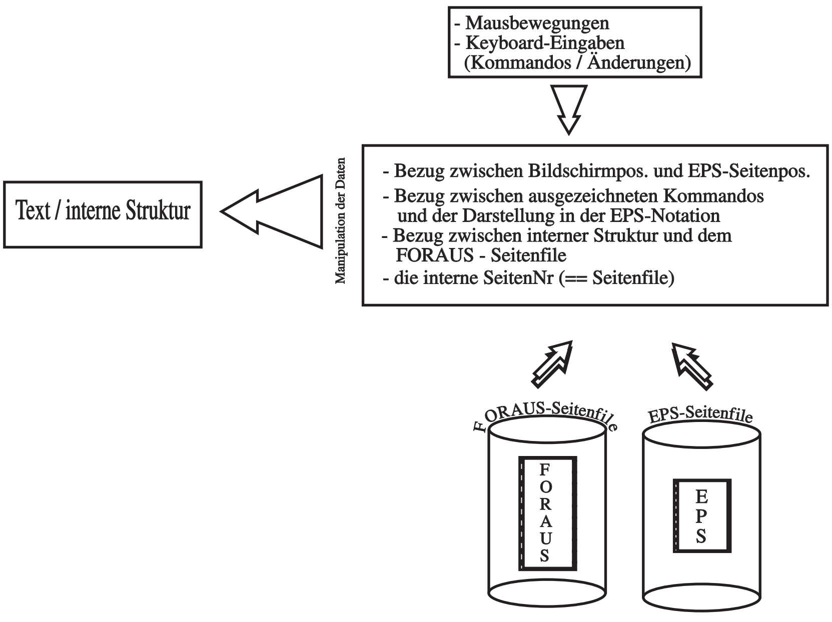

Abbildung 14. WYSIWYG -> Foraus

6. Softwareergonomie

Lu Lei, Van Son Le, Rainer Schmidtke, Volker Schmidtke

Die Software-Ergonomie ist in den letzten Jahren zu eines der zentralen Themen in der Softwaretechnik geworden. Ihre Bedeutung steigt mit dem funktionellen Angebot von Software-Systemen. Dies wird leicht ersichtlich, wenn man heutige Software, insbesondere Textverarbeitungen unter ergonomischen Aspekten betrachtet.

Einfache Editoren (z.B. vi , ed und andere) mit sehr eingeschränkter Funktionalität zählen schon längst zur Vergangenheit. Obwohl sie kaum mehr Gestaltungsmöglichkeiten zuließen als elektronische Schreibmaschinen, waren schon diese Textverarbeitungsprogramme kompliziert zu bedienen. Der Benutzer mußte viele komplexe Tastenkombinationen erlernen, um mit dem Programm etwas anfangen zu können.

Heute ist die Anzahl der Funktionen innerhalb eines Textverarbeitungsprogramms so stark angestiegen, daß es allein mit Tastenkombinationen wohl kaum bedienbar wäre.

Es stellt sich also die Frage, auf welche Art und Weise der Informatiker die SW- Oberfläche gestaltet, um dem Benutzer den Umgang mit seinem Computer möglichst leicht zu gestalten. Dabei spielt natürlich die Zielgruppe eine wichtige Rolle. Gerade bei Textverarbeitungen kann man jedoch davon ausgehen, daß ein großes Spektrum von Anwendergruppen mit dem System arbeiten werden. Folglich muß ein Programm sowohl für den Computerneuling (Anfänger) als auch für den fortgeschrittenen Anwenderkreis konzipiert werden.

Dies bedeutet sowohl eine gute Unterstützung bei den ersten Versuchen als auch eine stärkere Individualisierbarkeit bei der routinemäßigen Arbeit. Schließlich soll der Computereinsatz für den Benutzer eine Effizienzsteigerung (trotz zunehmender Komplexität der Software) bei seiner Arbeit bringen.

Wie diese Ziele der Software-Ergonomie am Beispiel einer Textverarbeitung umgesetzt werden können, soll im folgendem Text behandelt werden. Dabei ist natürlich nur eine Skizzierend der Motivationen von Ergonomierichtlinien und Standards möglich, da die vollständige Thematik sonst den Rahmen dieser Arbeit bei weitem sprengen würde. Trotzdem sollen zumindest einige objektive Methoden zur Überprüfung der Ergonomie von Software aufgezeigt werden. Sie sollen am Ende dieser Semesterarbeit in Anforderungen an ein möglichst ergonomisches Textverarbeitungsprogramm angewendet werden.

6.1. Grundlagen der Wahrnehmung und Informationsverarbeitung

Was ist überhaupt Software-Ergonomie?

Der Begriff der ‘Software-Ergonomie’ ist abgewandelt aus dem allgemeinen Begriff der Ergonomie, wie er aus der Psychologie und Kognitionswissenschaft kommt. (Als Kognitionswissenschaft bezeichnet man die Lehre von der Entwicklung all der Funktionen beim Menschen, die zum Wahrnehmen eines Gegenstandes oder zum Wissen über ihn beitragen.)

In der Psychologie / Kognitionswissenschaft definiert man den Begriff der Ergonomie als die...

Lehre von der (Anpassung) des Arbeitsystems an die menschlichen Eigenschaften, Fähigkeiten, Kenntnisse (und Bedürfnisse), wobei

- die physischen Eigenschaften weitgehend fest liegen (z.B. Greifräume)
- die kognitiven Eigenschaften weitgehend plastisch sind (z.B. Qualifizierung).

Die Definition der Software-Ergonomie mußte jedoch begrifflich angepaßt werden:

Der Mensch paßt sich durch Qualifizierung an das interaktive System an und der Mensch paßt das (flexible) System an seine Arbeitsbedingungen an.

Allein diese Definition stellt schon ergonomische Grundanforderungen an das System. So muß ein ergonomisches Software-System interaktiv und zugleich flexibel anpaßbar sein. Auch wird deutlich, daß ein Computer Arbeitsabläufe mitbestimmt und somit in direktem Zusammenhang zu den Arbeitsbedingungen steht. Hinter diesen eher trivialen Forderungen verbergen sich allerdings sehr unterschiedliche Auffassungen.

Die Frage ist also:

a) Welche Art von Interaktion ist gemeint?

- Computergesteuerte/-orientierte Interaktion
- Benutzergesteuerte/-orientierte Interaktion

b) Welche Art der Anpaßbarkeit ist gemeint? (vergl. [Opp89])

- angepaßt:
  angepaßt an die Anforderungen des vorgesehenen Einsatzes als Ausgangswert für die Benutzung
- anpaßbar:
  anpaßbar durch einen bestimmten Benutzer an seine Anforderungen
- anpassungsfähig:
  anpassungsfähig durch das System selbst aufgrund dessen Analyse der Benutzeranforderung

6.2. Die M - M und M - R Kommunikationsmodelle

Das Problem bei einer ergonomischen Gestaltung von Oberflächen ist, daß der Computer als Werkzeug zur Informationsverarbeitung eingesetzt wird. Informationsverarbeitung ist jedoch notwendigerweise mit Kommunikation verbunden.

Welche Form der Kommunikation mit dem Benutzer ist nun die richtige?

Um hier eine Entscheidung treffen zu können, ist es zunächst wichtig, die Kommunikation zwischen Mensch und Rechner mit der Kommunikation, wie sie zwischen zwei Menschen stattfindet, zu vergleichen.

In der Kognitionsforschung wird dieser Ansatz konkreter formuliert:

Notwendig ist eine vergleichende Betrachtung

- menschlicher Denkprozesse (Gehirn) und
- maschineller Datenverarbeitungsprozesse (Computer)

verallgemeinernd als

- “informationsverarbeitende Systeme” (IVS)

mit verwandten Eigenschaften und Verhaltensweisen.

Hierzu existieren bereits Kommunikationsmodelle, wie sie zum Beispiel in [Opp89] beschrieben werden.

6.2.1. Das Mensch - Mensch - Kommunikationsmodell

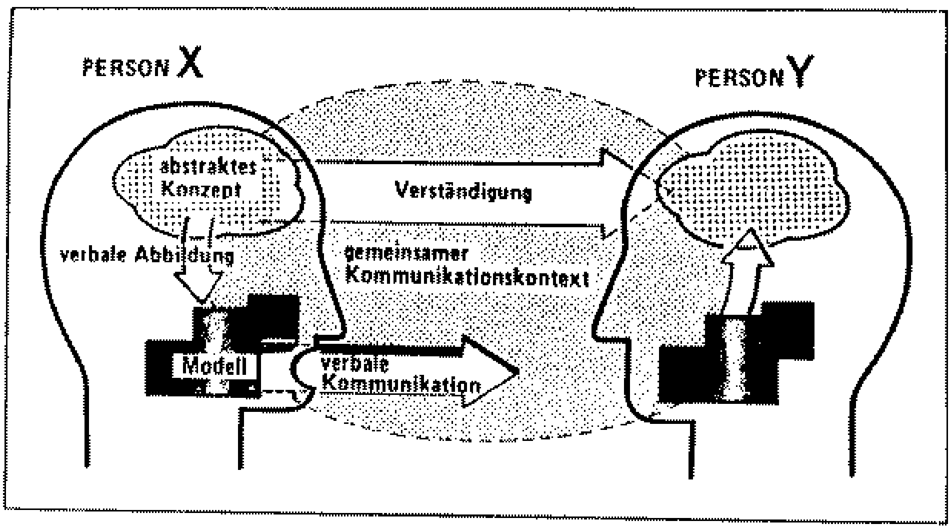

Abbildung 15. Das Mensch-Mensch Kommunikationsproblem

Merkmale:

- Der Mensch entwickelt eine Idee (abstraktes Konzept) und kann das verbal abgebildete Modell einem anderen Menschen mitteilen, wenn sie einen gemeinsamen Kontext (Sprache, Wissen, Zielsetzung...) haben.
- Durch den Kommunikationsprozeß wird der gemeinsame Kommunikationskontext beider Personen vergrößert. Beide Personen 'lernen' vom - und 'belehren' den anderen.
- Das übertragende Modell kann auf vielfältige Art interpretiert und hinterfragt werden. Je nach Erfahrung des Empfängers wird das Modell unterschiedlich ausgelegt.
- Rückfragen (durch direktes Feedback) und Themensprünge (durch flexible Handlungsweise) sind jederzeit möglich.
- Es fließen mehrere Meinungen zusammen, werden analysiert und zu einem (oder mehreren) Resultat (-en) zusammengestellt.
- Die Gesprächspartner passen sich an dem Kontext des anderen im Dialog selbst an. Je nach Situation (z.B. auch Zeit und Umgebung (Ort)) findet das Gespräch auf unterschiedlichste Weise statt.
- Der Gesprächspartner kann sich oft in das Denkschema des anderen hineinversetzen und erkennt relevante und irrelevante Informationen / optimale und suboptimale Vorgehensweisen.
- neben der verbalen Kommunikation werden Informationen auch mittels Gestik, Mimik etc. übertragen.

6.2.2. Das Mensch - Rechner - Kommunikationsmodell

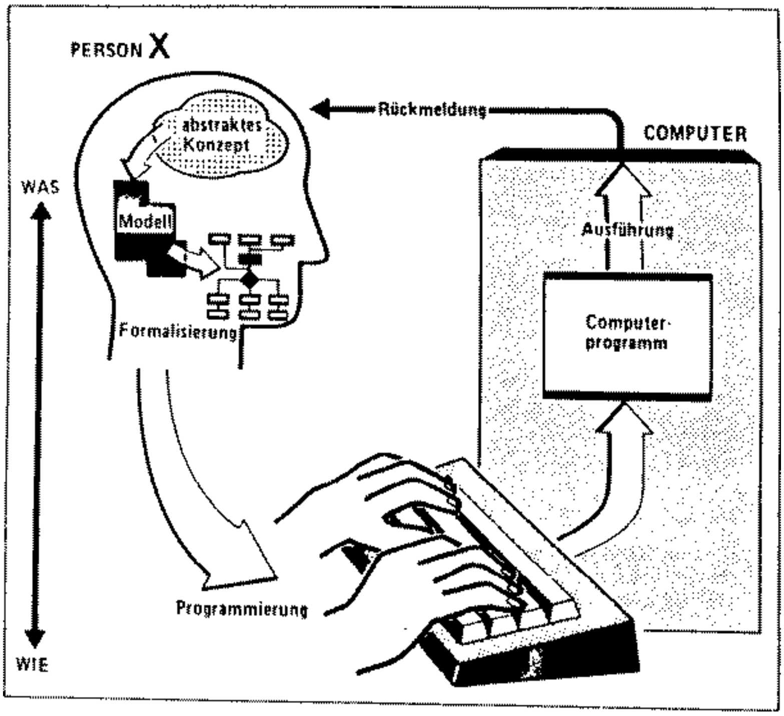

Abbildung 16. Das Mensch-Rechner Kommunikationsproblem

Merkmale:

- Der Mensch formalisiert das Modell, das durch das abstrakte Konzept erzeugt wurde, und teilt es dem Rechner als Programm mit. Es finden also mehrere Abbildungen durch Übersetzungsprozesse statt!
- Wie gut die Programmierung ist, hängt von der Qualität der Formalisierung und der Qualität der Programmierung (Programmierungsmöglichkeiten des Systems) ab.
- Erst nach Ausführung des Computerprogramms erhält der Benutzer eine Rückmeldung, die seine Formalisierung (und eventuell das Modell) beeinflußt bzw. revidiert.
- Das Vertrautmachen mit dem System ist schwer, da hierzu noch keine Erfahrung bzw. Wissen existiert.
- Der Benutzer ist gezwungen, sich ein Verhaltensmodell der Maschine zu erstellen, aus dem er auf komplizierte Weise Merkmale extrahiert und diese wiederum in sein (aktualisiertes Zustands-) Modell einsetzt.
- Der Computer verarbeitet nur ein eingeschränktes Bild des realen Modells, da niemals der gesamte Kontext erfaßt wird.
- Der Wissenserwerb ist meist einseitig.

Ein gutes, sowie ein schlechtes Beispiel einer Dialogschnittstelle zeigen die folgenden Abbildungen. Hier wird am Beispiel eine Patientenverwaltung die unterschiedliche Modellbildung aufgezeigt, die sich bei Verwendung von ergonomischen bzw. nicht-ergonomischen Schnittstellen ergeben.

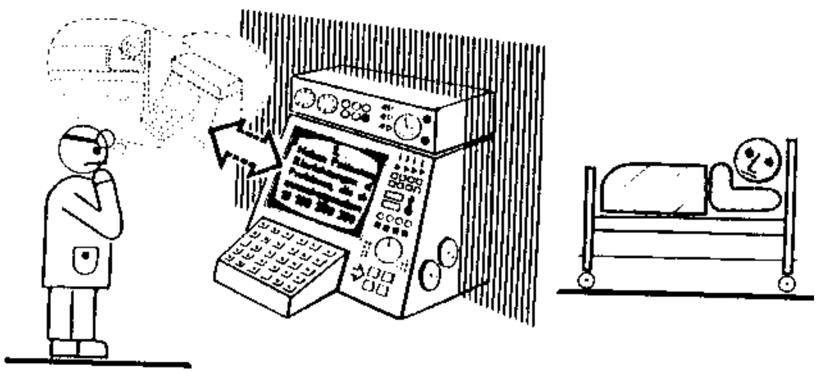

Abbildung 17. Ungünstige Mensch - Maschine Interaktion

Ein günstig gestaltetes System setzt die erfaßten Zustandsgrößen des Patienten bereits so um, daß der Arzt als Benutzer diese Information weitgehend nahtlos in sein “inneres Patientenmodell” einsetzen kann. Er ist dadurch in der Lage, notwendige Konsequenzen rascher und sicherer ableiten zu können.

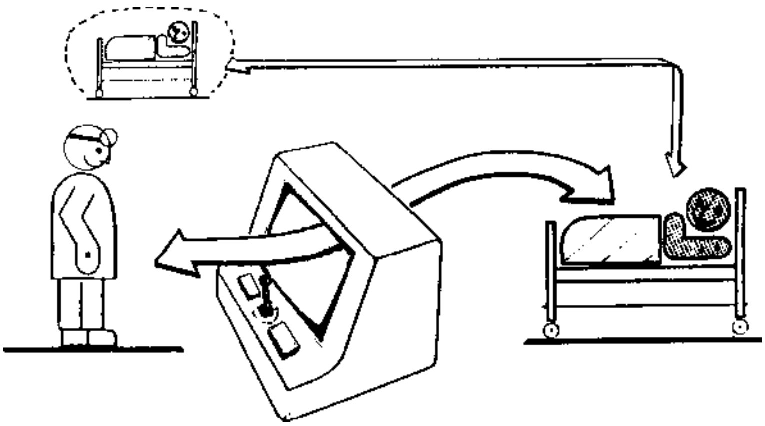

Abbildung 18. Ergonomische Mensch - Maschine Interaktion

6.3. Das IFIP als abstraktes Schnittstellenmodell

Bei jeder Oberflächengestaltung stellt sich spätestens beim Programmieren der Benutzungsoberflächen die Frage, wie man Schnittstellenmodelle entwirft, die den Informationsfluß möglichst flexibel, elegant und einfach kontrollieren und beschreiben.

Als besonders geeignetes Modell hierzu hat sich das IFIP (International Federation of Information Processing) -Benutzungsschnittstellen-Modell herausgestellt, welches sich durch eine klare Struktur auszeichnet.

Das IFIP-Modell wurde gegenüber der Benutzer - Dialog - Programmfunktionen - Kette, die mehr einem computergesteuerten Ansatz wiederspiegelt, stark verfeinert.

Wesentlicher Unterschied ist die Berücksichtigung der Arbeitswelt mit ihren Beziehungen zu den repräsentierenden Modellen im Benutzer und Computer, sowie die Unterteilung des Dialog-Begriffs in Ein-/Ausgabe, Dialog und Werkzeug. Letztere Unterteilung ist besonders wichtig, wenn man ein modulares Programm und einen benutzergesteuerten Dialog realsieren möchte.

Das Schema des Datenflusses wird anhand der Grafik unten deutlich. Im folgenden möchte ich deshalb mehr auf die einzelnen Phasen des Modells und ihre typischen Merkmale eingehen. Dabei ist der Benutzer (USER) und der Rechner weniger von Bedeutung, da sie zwar die aktiven Organe im Modell sind, aber für die Software- Entwicklung keine größere Rolle spielen.

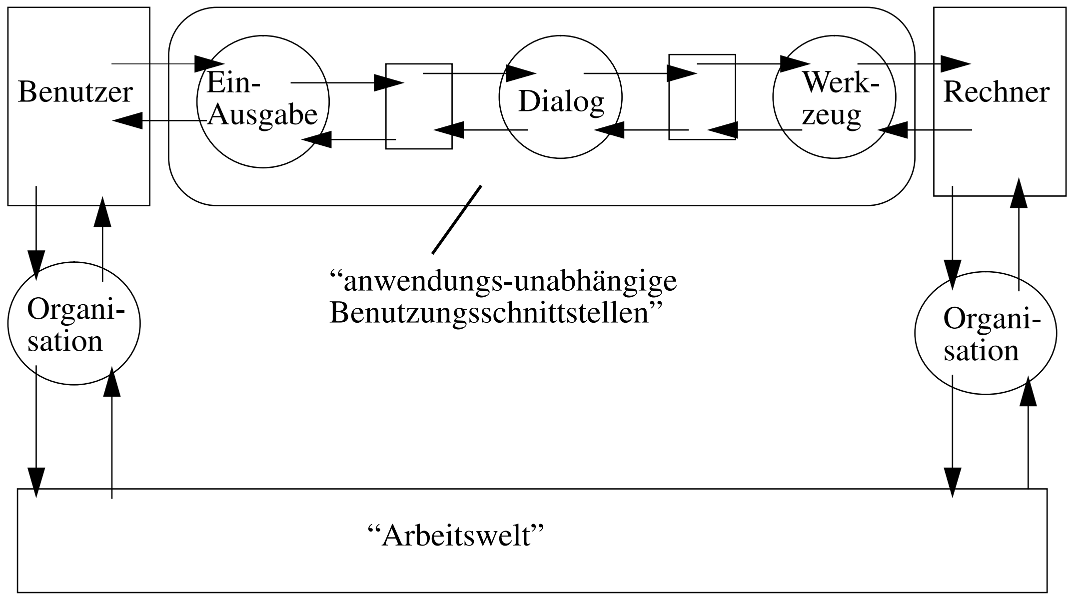

Abbildung 19. Der Aufbau des IFIP-Benutzungsschnittstellen-Modells

1. Ein-/Ausgabe - Schnittstelle (engl. PRESENTATION):

Wie der Name schon sagt, wird hier die grafische Form der Ein- und Ausgabe festgelegt.

Bestandteile sind:

- Elemente (Buchstaben, geometrische Grundelemente...)
- Gruppen (Menüs, Ein-/Ausgabe-Zonen, Masken...)
- Bilder, d.h. “dynamische Gruppen” (Fenster, Dialogboxen, Meldungen...)
- Metaphern, d.h. Bilder 2. Ordnung, Analogien (Desktop-Metapher, Karteikasten-Metapher)

Die Wahrnehmung von Objekten (Gestaltwahrnehmung) ist deshalb gerade hierbei von größter Wichtigkeit, da in dieser Schicht das Aussehen der einzelnen Objekte bestimmt wird!

2. Dialog-Schnittstelle (engl. CONTROL):

- enthält Interaktionsarten (sprachliche-,textorientierte-, menüorientierte-, funktionsta-stenorientierte Auswahl) und Interaktionsstile (Objekt-Funktion- oder Funktion-Objekt - Bindung, sowie Wahl zwischen computer- und benutzergesteuerte Dialogführung)
- legt Steuerungsmechanismen fest: WYSIWYG, Festlegung der Form der direkten Manipulation (d.h. permanente Sichtbarkeit der interessanten Objekte)...

Neben der visuellen Wahrnehmung (Gestaltwahrnehmung) sind hier auch die psychologischen Handlungstheorien bei der Gestaltung zu berücksichtigen, da der Benutzer am Bildschirm interaktiv wird.

3. Werkzeug - Schnittstelle (engl. INTERFACE):

- definiert Regeln über den Zugriff der BenutzerInnen auf die Software-Werkzeuge (z.B. Info geben, Zusammenfassen von Prozeduren, Makros, Bestimmung des Geltungsbereiches der Kommandos über Parameter, Struktur- und Konsistenzprüfung der Funktionen (z.B. Fehler ausgeben, wenn illegale Funktionen oder die Funktionen auf falsche Objekte angewendet werden))
- sollte eine ausgewogene Größe des Vokabulars (Objekte) bestimmen, da sonst der Benutzer zu viele Objekte erlernen muß.

Da es hierbei größtenteils um den Ablauf der Benutzung des Computers geht, stehen hier besonders die psychologische Handlungstheorien im Vordergrund.

4. Organisations - Schnittstelle (engl. PUBLIKATION (-FUNCTIONS))

In dieser Schnittstelle sind Regeln über den Zusammenhang der Arbeitsaufgaben verschiedener BenutzerInnen definiert. Da sie in direkter Verbindung mit der “Arbeitswelt” steht, stellt sie die eigentlich Manipulationsmöglichkeiten des Modells zusammen.Sie bestimmt Strukturen und Datenflüsse im „Arbeitswelt“-Modell. Aber auch andere Daten für die Organisation werden hier behandelt.

6.4. Gestaltwahrnehmung

Da der Computer neben wenigen akustischen Signalen, dem Benutzer im Wesentlichen eine optischen Rückmeldung seiner Aktionen zurück liefert, ist es für den Informatiker interessant, wie es dem Benutzer möglichst leicht gemacht werden kann, die oft vielfältigen Informationen auf dem Bildschirm zu erfassen.

Psychologen und Kognitionswissenschaftler führen dieses Problem zurück auf die Mustererkennung und -verarbeitung.

Wir können zwei Arten von Formen des Wissens / der Wissensspeicherung unterscheiden:

a) Deklaratives Wissen

- Propositionen (beschreibende, sprach-artige Repräsentation)
  (Sammlung von (seq.) Fakten und Aussagen))
- Bilder (abbildende, wahrnehmungs - orientierte Repräsentation)

b) Prozedurales Wissen

(in Form von Funktionen, Handlungsabläufen, Schemata)

Der Kognitionswissenschaftler David Marr konnte aufgrund von Experimenten 1982 einige grundsätzliche Vorgänge des Wahrnehmens von Objekten durch den Menschen erklären.

Er stellte seine Erkenntnisse in seinem David Marr - Skizzen - Modell zusammen. Es war Resultat von Versuchen mit Assoziationsbilder (Klecksbilder), die eine Reihe von Probanten zu deuten hatten. (aus:[Gar89], S. 311 ff)

Nach dem David Marr Skizzen-Modell unterteilt sich die Wahrnehmung in 3 Schritten:

1. Fläche wird in geometrischen Figuren zerlegt (in Komponenten und Subkomponenten) durch Ausnutzung der strukturell informationstragenden Teile (Knick-, Wende-, Krümmungspunkte), Weglassung redundanter Teile.

   Dies führt zu einer symbolischen Repräsentation (als Einheit wahrgenommene Objekte). Sie wird auch bezeichnet als primäre Skizze.

   Es erfolgt eine Schemata-Anwendung. Es wird eine Folge von bestimmten Arten von Informationen aufgestellt, die auf spezifische Art und Weise organisiert sind (komplexe angeeignete Strukturen). Sie werden durch Erfahrungen etc. gebildet.
2. Bilden und gruppieren der Merkmale (funktionelle und gestaltspezifische Merkmale), Analyse der primären Skizze durch symbolische Prozesse.
3. Möglichst eindeutige Identifizierung eines Gegenstandes einschließlich seiner Komponenten mittels (top-down-) Wissen und Erfahrung.

Bei der Erkennung / Zerlegung gelten folgende Gestaltgesetze:

- Gesetz der Nähe

  In einer Menge gleichartiger Elemente schließen sich in unserer Wahrnehmung die räumlich nahe beieinanderliegenden zu einer Gruppe zusammen.

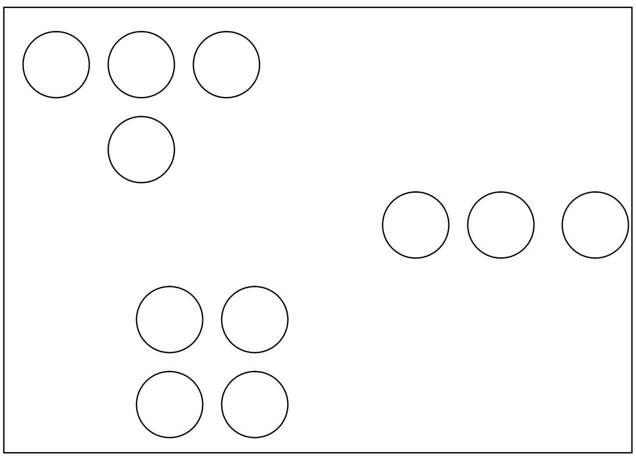

Abbildung 20. Das Gesetz der Nähe

- Gesetz der Gleichheit/Ähnlichkeit

  Bei Darbietung verschiedener Elemente werden gleiche, gleichartige oder ähnliche Elemente zu einer Gruppe zusammengefaßt. Die Gleichartigkeit kann sowohl durch Form als auch durch die Farbe der Bildelemente bewirkt werden.

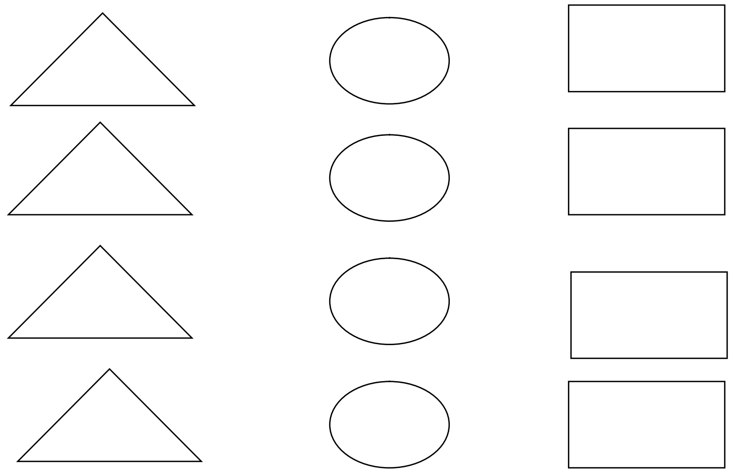

Abbildung 21. Das Gesetz der Gleichheit

- Gesetz der guten Fortsetzung

  Es werden die Elemente besser wahrgenommen, deren gerade oder gekrümmten Linien am wenigsten verändert oder unterbrochen werden.

- Gesetz der Geschlossenheit

  Linien, die Flächen umschließen, werden gegenüber nicht-geschlossenen Linienzügen als Einheit wahrgenommen.

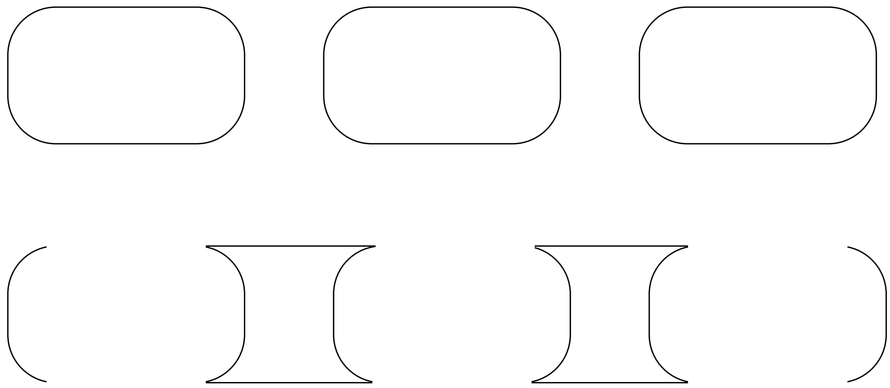

Abbildung 22. Das Gesetz der Geschlossenheit

- Figur - Hintergrund - Beziehung.

  Figur und Hintergrund stehen in einer Beziehung zueinander. Die Entscheidung, welche Elemente Figuren und welche Hintergründe sind, erfolgt durch ihre geometrischen Form (z.B. geometr. Elemente auf heterogenen Hintergrund).

- Gesetz der Prägnanz

  Unvollständige Gestalten werden „idealisiert“ wahrgenommen, z.B. wird eine nicht ganz symmetrische Figur bei ungenauer Betrachtung als symmetrisch wahrgenommen, oder eine unvollständige Gestalt wird vervollständigt. Gestalten werden also immer nach bestimmten Kriterien ergänzt bzw. optimiert.

  (Methode der Expressionisten: Unsichtbares wird vor dem geistigen Auge sichtbar!)

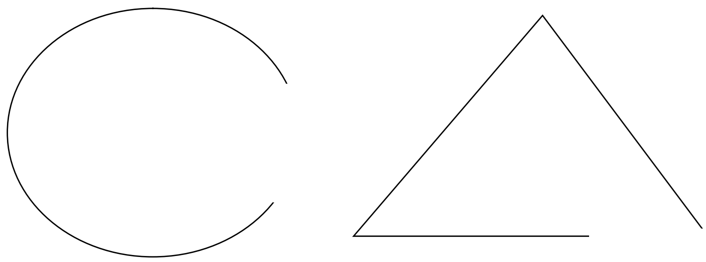

Abbildung 23. Das Gesetz der Prägnanz

- (Gesetz der Symmetrie)

  Ein geschlossener Bereich (z.B. Grafik-Rahmen) wird umso eher als Figur gesehen, je symmetrischer er ist.

... und deren Kombinationen.

Je mehr Gestalt-Gesetze erfüllt sind (zusätzlich: farbliche Trennung), um so prägnanter sind die Figuren.

Jedoch: Man kann Objekte, die lediglich durch Farbe gebildet worden sind, nur schwer erkennen. Auch das gleichzeitige Benutzen von gesättigten Farben sollte vermieden werden.

Bei der Verwendung von Farben ist weiterhin die Tatsache zu berücksichtigen, daß ca. 8% der Menschen farbenblind sind (meist rot-grün-Blindheit).

Abgeleitete ergonomische Ziele:

- Ermöglichung einer Schemata - Bildung der Reize (Muster) zwecks besseren Wiedererkennens von gelernten Merkmalen (Assoziationen) (eindeutige Piktogramme, Informationsgliederung, einheitl. Strukturen, Konventionen über Dialog-Aussehen (Farbe / Ort / Zeit), Zusammenhänge verdeutlichen).
- keine Reizüberflutung (z.B. durch viele bewegte Objekte (da Bewegungsempfindlichkeit der Netzhaut!))
- visuelle Trennung von nicht-zusammenhängenden Informationen
- Funktions-Objekt-Kopplung, da Merkmale der Objekte auch deren Funktionen sein können (z.B. Designer-Stühle /-Tische: Erkennen des Objekts durch das Erkennen seiner Funktion (Stühle haben das Potential, auf ihnen zu sitzen!))
- Minimierung der Blickbewegungen
- Zustandorientierung durch Maskenstruktur
- Blinken und andere blickfixierende Effekte nur bei besonders wichtigen Meldungen benutzen. Niemals mehrere Effekt-Meldungen zur gleichen Zeit.

6.5. Die psychologische Handlungstheorie

Die Wahrnehmung von Objekten ist nur ein kleiner Teil dessen, was der Benutzer am Bildschirm zu leisten hat. Wie schon im Mensch-Rechner Kommunikationsproblem angesprochen, ist für ein planendes Vorgehen eine Idee als Modell, die Möglichkeit der Manipulation des Computers und eine Rückmeldung des Systems von besonderer Wichtigkeit. Durch die Rückmeldung des Programms kann der Benutzer Rückschlüsse auf den aktuellen Zustand seiner Programmierung schließen. Erst dann kann er eventuell sein Vorgehenskonzept der Situation anpassen. Die Erforschung der Wege des Planen und Handelns beim Menschen ist daher von großem Interesse.

Ansatz:

Der Mensch als tätiges (arbeitendes) Wesen. Veränderung der Welt nach selbstgesetzten Zielen und mit selbst entwickelten Werkzeugen.

Das menschliche Vorgehen kann man durch vier Phasen beschreiben:

1. Handlungsziele bilden
2. Weg zum Ziel suchen
3. Handlung ausführen
4. Ergebnis überprüfen

Bedingungen für diese Vorgehensweise sind:

- eine zyklische Einheit / hierarchische Gliederung
- die Kontrolle der Handlung (= Handlungsregulation) durch Freiheitsgrade, Tätigkeitspielräume, Entscheidungsspielräume des Benutzers.

Voraussetzungen für eine Kontrolle durch den Benutzer ist:

- eine Transparenz (Durchschaubarkeit, Erlernbarkeit)
- die Vorhersagbarkeit des Systems
- eine Kompetenz (Fähigkeit zur Entwicklung effizienter Handlungsweisen) bei Computer und Benutzer

Eine Vergrößerung der Kontrolle kann durch das Konzept der "vollständigen Handlung" und durch direktes Feedback (grafisch / akkustisch) erreicht werden.

Abgeleitete ergonomische Ziele für unser Textverarbeitungssystem:

- Intuitives System: virtuelle Werkzeuge (Metaphern)
- Interaktives System: Direkte Manipulation, Mehrfenster-Technik
- Objekt - Funktion und Funktion - Objekt - Konzept
  (Objekte: Dokument, Kapitel, Absatz, Text...),
- wenige, prägnante (Grund-) Funktionen: Cut, Copy, Paste... ("generische Kommandos"), damit der Benutzer nicht zu viele Kommandos erlernen muß.
- möglichst flexible, kontrollierte Handhabe
- benutzerorientierter / -gesteuerter Dialog
- Abstrahierungsmöglichkeiten (Zoom, Hierarchie(n) - Darstellung...)

6.6. Richtlinien für eine ergonomische Dialoggestaltung

Unter "ergonomischer Dialoggestaltung" versteht man die Anforderungen an den Dialog, den Mensch und Dialogsystem führen, damit der Mensch seine Arbeitsaufgabe unter ergonomisch günstigen Bedingungen planen und ausführen kann."

Günstig sind solche Bedingungen, die psychische Eigenschaften des arbeitenden Menschen berücksichtigen, z.B. die Aufmerksamkeitsspanne, die begrenzte Kapazität des Kurzzeit- Gedächnisses, den Übungsgrad, das innere Modell des Benutzers von Arbeitsmitteln und Ausführungsbedingungen, seine Neugier, seine Bedürfnisse, seine sichere Orientierung [DIN88, Seite 6].

Hierbei sind die Teile eines Datenverarbeitungssystems, die vom Benutzer am Bildschirmarbeitsplatz zur Abwicklung eines Dialog zur Verfügung stehen, unter dem Begriff "Dialogsystem" zusammengefaßt.

Bei der Gestaltung der Benutzungsoberfläche möchten wir uns an der DIN 66234 Teil 8 orientieren.Die Gestaltungsgrundsätze der Norm folgen dem Leitgedanken, daß die Software an die unterschiedlichen Eigenschaften der Benutzer und an die gegebenen Arbeitsaufgaben anpaßbar zu gestalten ist. Da auf der Ebene einer allgemeine Beschreibung verblieben wird, läßt die DIN bei der Gestaltung zukünftiger Dialogsysteme einen großen Gestaltungsspielraum. Vielmehr werden Mindestanforderungen festgelegt, die exemplarisch durch Anwendungsvorschläge ergänzt werden. Die Reihenfolge der Nennung der Einzelkriterien stellt keine Hierarchie dar, sie sollen vielmehr immer in kombinierter Form Anwendung finden, wobei sie je nach Anwendungsfall unterschiedlich zu gewichten sind.

Es sind die Punkte

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlerrobustheit

sowie

- Belastungsoptimierung
- Lernförderlichkeit/Erlernbarkeit (ISO 9241 Teil 10)
- Kooperationsförderlichkeit
- Individualisierbarkeit (Adaptierbarkeit) (ISO 9241 Teil 10)
- Arbeitnehmerdatenschutz

zu berücksichtigen.

6.6.1. DIN 66 234 - Teil 8

6.6.1.1. Aufgabenangemessenheit

Ein Dialog ist aufgabenangemessen, wenn er die Erledigung der Arbeitsaufgabe des Benutzers unterstützt, ohne ihn durch Eigenschaften des Dialogsystems unnötig zu belasten [DIN88].

Die Software kann den auftretenden Sachproblemen nicht angemessen gerecht werden, wenn die Systementwicklung praxisfern erfolgte. Oft wird vom Software-Entwickler entschieden, welche Methode zur Textgestaltung zu benutzen ist und welche Teile der Arbeit vom Computer übernommen werden bzw. welche beim Benutzer liegen. Die Eigenschaft des planvoll handelnden Menschen darf durch undurchschaubares Systemverhalten nicht behindert werden, sowie langfristig nicht zum Abbau der Fähigkeit zum Planen führen.

6.6.1.2. Selbstbeschreibungsfähigkeit

Ein Dialog ist selbstbeschreibungsfähig, wenn dem Benutzer auf Verlangen Einsatzzweck sowie Leistungsumfang des Dialogsystems erläutert werden können und wenn jeder einzelne Dialogschritt unmittelbar verständlich ist oder der Benutzer auf verlangen dem jeweiligen Dialogschritt entsprechende Erläuterungen erhalten kann [DIN88].

Dieses ist insbesondere bei großen Programmen - wie wir eines vorhaben - wichtig, da hier die Anzahl der bereitgestellten Funktionen vom normalen Benutzer nicht mehr überschaubar ist. Zur Durchführung eines Planes bedarf der Benutzer jederzeit Erklärungshilfen, damit er Systemzusammenhänge versteht und diese in seine Planarbeit integrieren kann.

6.6.1.3. Steuerbarkeit

Ein Dialog ist steuerbar, wenn der Benutzer die Geschwindigkeit des Ablaufs sowie die Auswahl und Reihenfolge von Arbeitsmitteln oder Art und Umfang von Ein- und Ausgaben beeinflussen kann [DIN88].

So kann eine Benutzerführung für den ungeübten und gelegentlichen Benutzer sinnvoll, für den geübten allerdings kontraproduktiv sein. Steuerbar ist der Dialog dann nicht, wenn das System Eingaben in einen bestimmten Takt verlangt, wie z.B. durch frühzeitiges Erlöschen der Bildanzeigen. Bei Textverarbeitungssysteme ist die Wahl der Eingabegeräte, sowie die Erstellung von Makros unter dieses Kriterium zu subsumieren, denn der Aufruf von mehreren Dialogschritten und Textblöcken durch einem einzigen Kommando erhöht die Steuerbarkeit des Sytems beträchtlich.

Die Steuerbarkeit eines Systems bringt jedoch auch Probleme mit der Selbstbeschreibungsfähigkeit mit sich. Der erfahrene Benutzer wird sich das System nach seinen Bedürfnissen anpassen, wodurch die Benutzerschnittstelle uneinheitlich wird.

6.6.1.4. Erwartungskonformität

Ein Dialog ist erwartungskonform, wenn er den Erwartungen des Benutzers entspricht, die sie aus Erfahrungen mit bisherigen Arbeitsabläufen oder aus der Benutzerschulung mitbringen sowie den Erfahrungen, die sie sich während der Benutzung des Dialogsystems und im Umgang mit dem Benutzerhandbuch bilden [DIN88].

Datenverarbeitungsspezifische Bedienungselemente - wie z.B. Scrollen - widersprechen den Erwartungen des Laien, der gerade bei einer Oberfläche, die sich der Schreibtisch-Metapher bedient, blockweises Blättern erwartet. Der computer-kundige Benutzer jedoch kennt und erwartet diese Art der Bedienung.

6.6.1.5. Fehlerrobustheit

Ein Dialog ist fehlerrobust, wenn trotz erkennbarer fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis mit minimalem oder ohne Korrekturaufwand erreicht wird. Dazu müssen dem Benutzer die Fehler zum Zwecke der Behebung verständlich gemacht werden [DIN88].

Fehler sind Bestandteil von Lernvorgängen und sollen den unerfahrenen Benutzer ermutigen das System und seine Möglichkeiten zu erforschen. Hierbei ist es unerläßlich, daß das System so reagiert, daß verhängnisvolle Aktionen annulierbar und nötigenfalls eine Fehlerkorrektur möglich ist. Undefinierte Systemzustände sind für ein sicheres Gefühl beim Arbeiten also absolut zu vermeiden.

6.6.2. ISO 9241 - Teil 10 und andere

6.6.2.1. Belastungsoptimierung

Das Dialogsystem soll bei Arbeitenden weder eine Unter- noch eine Überforderung erzeugen. Wie oben schon gefordert, sind unterschiedliche Benutzer zu unterscheiden, sowie sind sämtliche Systemkomponenten adaptierbar zu gestalten.

6.6.2.2. Lernförderlichkeit/Erlernbarkeit (ISO 9241 Teil 10)

Die Erlernbarkeit eines Systems hängt von vielen, zum Teil schon genannten Kriterien ab. Besonders die Komplexität (ist möglichst zu verbergen) der Textverarbeitungs-Software, vor allem die Quantität von möglichen Befehlen, stellt eine Schwierigkeit dar. Es ist deshalb darauf zu achten, daß verständliche, in sich abgeschlossenene Module gebildet werden, die in Phasen nacheinander erlernt werden können. Ein Laien- und Experten-Modus ist hierzu unerläßlich, hat jedoch den Nachteil, daß ein Moduswechsel stattfinden muß und somit zur Belastung des Benutzers führt (Lösung: viele, sanfte Modiübergänge).

Hauptaufgabe des Benutzers ist es nicht zu wissen, wie eine Aktion ausgelöst werden kann. Vielmehr ist eine intuitive Bedienung anzustreben. Die Steuerbarkeit, Erwartungskonformität und Fehlerrobustheit von Programmen sind ebenfalls wichtige lernfördernde Kriterien und wurden oben schon behandelt.

6.6.2.3. Kooperationsförderlichkeit

Unter Kooperation versteht man die computergestützte Gruppenarbeit. Das Dialogsystem soll den Benutzern ermöglichen, die in der Gruppe abgesprochenen Arbeiten gemeinsam zu lösen. Beispiele für solche Systeme mit unterschiedlichen Zielsetzungen sind:

- Message handling systems (Email)
- Co - authoring systems
- Meeting support systems
- Group decision support systems
- Coordination systems

6.6.2.4. Individualisierbarkeit (Adaptierbarkeit)

Die Individualisierbarkeit geht über das Kriterium der Steuerbarkeit hinaus, da hier die individuellen Bedürfnisse des Benutzers im Vordergrund stehen. Zu nennen ist beispielsweise eine manipulierbare Maus (Zahl der Knöpfe, Geschwindigkeit) oder die Tastenbelegung (Wo war noch das ß?). Darüber hinaus können Dialoge und Kommandos (Makros) individuell zusammenstellbar und somit effizienter im Sinne von belastungsmindernd gestaltet sein.

6.6.2.5. Arbeitnehmerdatenschutz

Der Arbeitnehmerdatenschutz soll gewährleisten, daß nur Daten gespeichert, aber auch protokolliert werden, die zur Aufgabenbewältigung notwendig sind, und daß die Berechtigung des Zugriffs auf Datenbestände gewährleistet wird. Dazu sind verschiedene Mechanismen einsetzbar, deren genaueren Betrachtung jedoch nicht im Rahmen dieser Arbeit stattfinden kann.

6.7. Grundlegende Prinzipien der graphischen Benutzungsoberfläche OPEN LOOK

Die graphische Benutzungsoberfläche OPEN LOOK ist eine Spezifikation für das “Aussehen und Gefühl” einer Window-Umgebung von multifunktionalen Computersystemen. Nach dieser Spezifikation werden OPEN LOOK Applikationen entwickelt, um den Benutzern konsistente Window-Umgebungen zu schaffen. Es gibt einige grundliegende Prinzipien von der Gestaltung der OPEN LOOK Applikationen, die in [Sun90] beschrieben sind.

6.7.1. Einfachheit

OPEN LOOK definiert Objekttypen, die auf dem Bildschirm gezeigt werden, und Konventionen, wie die Objekte benutzt werden. Die Objekte und die darauf operienden Kontrollen sind Elemente in OPEN LOOK Applikationen.

Die Elemente, die durch die Applikation präsentiert werden, sollten gut gestaltet sein, so daß sie von den Benutzern einfach bedient werden können. OPEN LOOK bietet dazu standardisierte Elemente an, z.B. base window mit Kontrollfläche, Menü, Arbeitsfläche etc.

Abbildung 24. Ein typisches OPEN LOOK Fenster

Die Objekte und Kontrollen sollten deutlich benannt und gekennzeichnet sein und den entsprechenden Realen-Sachverhalte wiederspiegeln. Hier ist ein Überblick von OPEN LOOK Kontrollen:

Abbildung 25. Beispiele für OPEN LOOK Kontrollen

6.7.2. Konsistenz

- Maus-Benutzung

Maus-Buttons sollten in Applikationen konsistent benutzt werden, z.B. die Voreinstellung zum Auswählen von Objekten ist der linke Knopf, vorgesehen ist der rechte Knopf für das Menü. Bei der Drei-Button-Maus ist der mittlere Knopf für das Addieren/Reduzieren von ausgewählten Objekten.

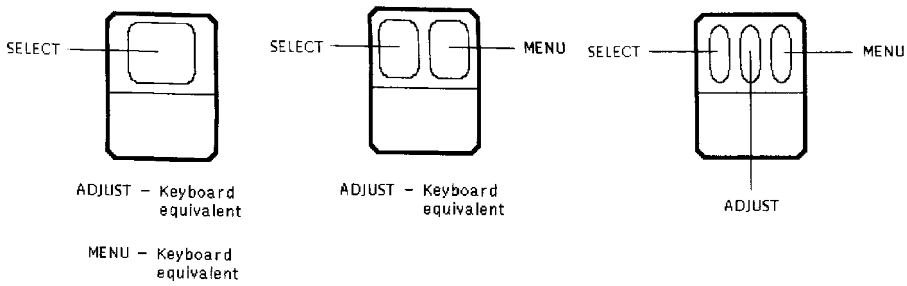

Abbildung 26. Zuordnung der Maustasten

- Tastatur-Benutzung

In OPEN LOOK können die Benutzer Daten auf einem vorübergehenden Speicherplatz, dem „Clipboard" speichern. Dadurch werden Daten zwischen oder innerhalb von Applikationen bewegt oder kopiert. Auf der Tastatur gibt es dazu sogenannte Core-Funktions-Tasten, wie z.B. cut, copy, paste u.s.w., die den Umgang mit dem Clipboard beschleunigen.

Es gibt weitere Core-Funktions-Tasten wie help, properties, undo, stop, die einem den Zugriff auf diese Grundfunktionen beschleunigen sollen.

- standardisierte Konventionen

Die standardisierten Funktionen sollten in allen Applikationen konsistent bleiben. Zum Beispiel enthalten die Datei-Operationen die meistbenutzten Operationen wie Datei öffnen, löschen, speichern u.a. Zur Organisation der Funktionen sind verschiedene Benutzer zu berücksichtigen.

- Benutzung der Objekte

Es besteht zwei Verfahren in OPEN LOOK, wie man Objekte manipuliert: „select-then-operate" und die direkte Manipulation. Mit dem ersten Verfahren wählt man zuerst Objekte, die man benutzen möchte, und dann die entsprechende Dialoge über sie. Mit direkter Manipulation werden Operationen direkt auf den graphischen Repräsentationen von Objekten, Attributen und Relationen ausgeführt.

6.7.3. Effizienz

Zur Erhöhung der Leistungsfähigkeit minimiert OPEN LOOK durch Pop-Up Menüs und Tasten-Äquivalenz u.a. die Tastenbetätigungen und Mausbewegungen.

- Pop-Up Menü / Pop-Up Window

Jede Arbeitsfläche hat ein Pop-up Menü. Das Menü enthält die Kontrollen, die in dieser Arbeitsfläche oft benutzt werden. Damit reduziert man viele Maus-Bewegungen. Hier ist ein Beispiel von Pop-Up Menü in Texteditoren (s.u.). Pop-Up Menüs können auf dem

Bildschirm fixiert werden und so immer zugänglich sein, wenn seine Push-Pin zu ist. Diese “Push-Pin”-Metapher ist eine besondere Eigenschaft von OPEN LOOK, die den Zugang zu häufig benutzten Menüs erleichtert.

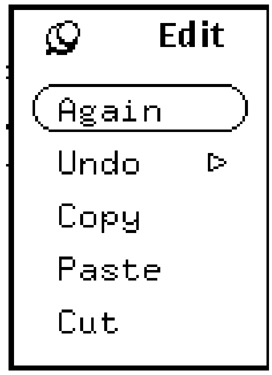

Abbildung 27. Beispiel eines Pop-Up Menüs

- Tastatur-Äquivalenz und Accelerators

In OPEN LOOK sind einige Tastenkombinationen für spezielle Funktionen definiert, z.B. CTRL-U steht für UNDO. Standard-Accelerators(cut/copy/paste) sind für Core-Funktionen definiert.

- Scrollbars

Eine weitere Eigenschaft von OPEN LOOK sind die Scrollbars, die Mausbewegungen dadurch reduzieren, daß die Knöpfe zur Bewegung nicht am Ende des Srollbars sind, sondern direkt am Aufzug zur Positionierung dienen.

6.8. Die allgemeinen Prinzipien der Apple-Benutzungsoberfläche

Die Prinzipien der Benutzungsoberfläche von Apple-System sind ähnlich wie bei OPEN LOOK, der Atari-Oberfläche und Windows für den PC. Ich glaube, daß ihr alle die Bedienung dieser Funktionen kennt, deshalb möchte ich nicht die Bedienung noch einmal wiederholen. Damit wir Zeit und Speicherplatz sparen können, werde ich nur einige grundlegende Funktionen der Benutzungsoberfläche beschreiben, wie sie in [App87], S. 2-11 beschrieben werden.

Diese sind:

- Metapher
- direkte Manipulation
- See-and-point
- Kontrolle
- Freeback und Dialog u.s.w...

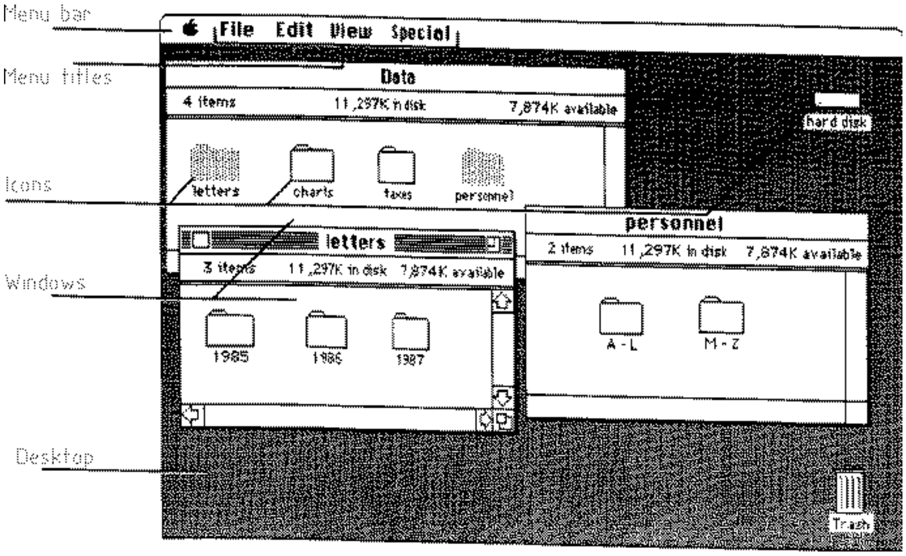

Abbildung 28. Die Oberfläche des Apples

Wie man in der obigen Abbildung sieht, ist bei Apple eine grafische Interaktion dominierend. Ohne Piktogramme, Symbole und Icons ist eine Benutzungsoberfläche bei Apple nicht denkbar. Die Desktop-Metapher versinnbildlicht eine Handlungsweise, die dem Umgehen mit Objekten auf einem realen Schreibtisch ähneln soll. Unterstützt wird die Handlungsweise durch die direkte Manipulation einzelner Objekte am Bildschirm. Alle Objekte, die auf dem Bildschirm angeordnet sind (also sichtbar für den Benutzer sind), können auch manipuliert werden (Prinzip “see - and - point”). Eine unmittelbare Rückmeldung (Feedback) ist dabei sehr wichtig.

6.8.1. Ästhetische Vollständigkeit

Neben den allgemeinen Richtlinien, die oben schon genannt wurden, gibt es eine Reihe von Gestaltungsprinzipien. Sie beschreiben das Aussehen der einzelnen grafischen Objekte und sollten möglichst dem Apple-Standard entsprechen.

Grundsätztlich gilt dabei:

- Durch die einfache, direkte Bedienung und Untermenüs wird die Arbeit der BenutzerInnen sehr viel erleichtert.
- Die BenutzerInnen haben Kontrolle über die Arbeitsfläche am Monitor.
- Die Kommunikation mit der Applikation geschieht mittels grafischer Interaktion.

Abbildung 29. Hauptmenü vor und nach Öffnen eines Untermenüs

Ein wesentlicher Punkt für den Graphik-Designer ist, daß beides, Graphik und Text, gut kombiniert werden können, d.h. die Graphiken werden leicht in den Text durch die einfache Objekteinbindung in das Fenster importiert. Man manipuliert also immer irgendwelche grafischen Objekte mit anderen Objekten. Diese Objekte können beliebiger Natur sein, also auch Texte.

6.8.2. Farben

Auch die Farbgebung unterliegt bei Apple einem Standard. Durch die Belegung der Farben mit Bedeutungen, gewinnen grafische Objekte an Ausdruckskraft. Sie können frühzeitiger und eindeutiger vom Benutzer erkannt werden.

- Färbung

Die drei Grundfarben sind rot, blau, grün oder gelb. Aus diesen Farben kann man durch Kombination der Farbwerte beliebig viele Farben erzeugen. Beim Apple sind 256 oder mehr Farben darstellbar. Davon sind jedoch nur 128 Helligkeitsstufen (hues) durch das menschliche Augen eindeutig unterscheidbar. Studien darüber haben gezeigt, daß das Gehirn nur vier bis sieben Farben gleichzeitig effizient erfassen kann. Generell ist jedoch der Einsatz von Farben gut geeignet für die Darstellung von Graphikobjekten und das optische Aussehen des Bildschirms.

- Helligkeit

Bei Farbmonitoren soll der Hintergrund grau sein, damit die Farben gut erkannt werden. Es wird damit eine Abgrenzung zu den Objekten auf dem Desktop sichtbar.

6.8.2.1. Farbenstandard

Farben können dazu eingesetzt werden, Objekte zu verbinden oder auch zu separieren, wenn es sinnvoll ist. Mögliche Anwendungsgebiete sind:

- die Unterscheidung von Bildschirmflächen,
- das Trennen von funktionellen Gruppierungen,
- das Aufzeigen von Beziehungen zwischen Objekten
- und die leichte Idendifizierung “crucial features”.

Die Bedeutung der Farben werden wie folgt festgelegt:

- Rot: Stop, Error, Fehler. (z.B. Der Laufwerk läuft immer noch. Man soll Diskette nicht herausnehmen.)
- Gelb: Warnung, Achtung oder Verzögerung.
- Grün: Power on, bereit.

6.8.2.2. Die allgemeinen Prinzipien des Farbendesigns

- Monitor: Manche Leute haben keinen Farbmonitor sondern nur einen Monochrom-Monitor. Dies muß bei der Farbgebung berücksichtigt werden.
- Drucken: Beim Ausdrucken von Objekten oder Graphiken werden die Farben durch einen Farbdrucker nicht so exakt ausgedruckt. Man sollte deshalb schon aus diesem Grund zu viele Farbnuancen vermeiden.

Abbildung 30. Eine Dialogbox kommt auch ohne viel Farben aus

6.8.2.3. Beschränkung bei der Farbenbenutzung

Farben sind gut für Zeichnungen und Graphiken. Sie erhöhen die Ausdruckskraft einer Grafik. Trotzdem gibt es auch Beschränkungen der Benutzung der Farben durch folgende Gründe:

- Zu viele Farben am Bildschirm machen die Augen schnell müde.
- Die Benutzungsobersfläche sollte nicht alle Farben benutzen. Meistens werden nur wenige Farben für ganz bestimmte Zwecke und Objekte (z.B. Ordner) gebraucht.
- Das Desktop ist am Besten in schwarz und weiß zu halten, da hierdurch ein besserer Kontrast zu den anderen Objekten am Bildschirm entsteht. Die Objekte können so leichter unterschieden werden.

6.8.3. Bearbeitung von Textdokumenten

Für das Ausschneiden und Einfügen von Textpassagen in einem Textdokument oder in Dialogmasken werden Standardfunktionen bereitgestellt. Die eingefügten Zeichen sollen direkt nach der Eingabe auf dem Bildschirm angezeigt werden. Bei Applikationen, die mehrere Textzeilen zulassen, sind automatsche Zeilenumbrüche vorgeschrieben.

Als besonders intelligent wird das Ausschneiden und Einfügen von Textblöcke angegeben, denn hier werden insbesondere die Leerzeichen zwischen den Worten soweit automatisch angepaßt, das immer korrekte Wortabstände entstehen.

Die Auswahl der Textblöcke ist soweit geregelt, daß durch einem einzelnen Mausklick der Textcursor positioniert, mit einem Doppelklick das Wort unter dem Textcursor und durch einem dreifachen Klick mit der Maus der Satz oder der Absatz selektiert wird. Auf diesem Block sind sodann Funktionen wie cut, copy und paste anwendbar. Zur Erweiterung von Textblöcke sind zudem noch die Shift-Tasten mit dem Mausklick vorbelegt (Shift-click). Weiterhin ist es möglich, mit einem Gummiband eine Bereichsselektion vorzunehmen, welches zum Beispiel auch für mehrer Textspalten anwendbar ist.

6.9. Abschließende Betrachtungen

1. Überprüfung der Entwicklungsziele und die daraus resultierende Implementierung durch den Anwender:

Grundlage für die Entwicklung eines Systems sollte eine Untersuchung von Handlungs- und Entscheidungsabläufen durch die Zielanwendergruppe sein.

Dies könnte z.B. durch einen weiteren Fragebogen oder durch Benutzungsoberflächen - Testphasen beim Erstellen des Prototypen und während der Implementierung geschehen. Die Testphasen werden üblicherweise schon in der Entwurfsphase festgelegt.

2. Weiterentwicklung:

Erstrebenswert ist ein offenes System der Modularisierung (mit hinzuladbarem Funktionsumfang). Dies sollte Arbeit in der Entwurfsphase sein.

3. Umsetzung auf andere Rechnersysteme:

Eine Portabilität auf anderen Systemen ist fraglich, da insbesondere bei der Fenster-Technik große Unstimmigkeiten herrschen. Auch die rechnerspezifischen Richtlinien sind unterschiedlich. Ein Kompromiss aus einer effizienz-orientierten Benutzungsoberfläche (OpenLook) und einer metapher-orientierten Benutzungsoberfläche (Apple) ist schwierig.

Wir haben bei der Implementierung der IFIP-Schichten zu entscheiden, ob die Bedienung auf jedem System

- gleich sein soll (zwar erwartungskonform unter den Rechnersystemen, jedoch nicht erwartungskonform aus der Sicht des Benutzers) oder
- unterschiedlich sein soll (nicht erwartungskonform unter den Rechnersystemen, jedoch erwartungskonform aus der Sicht des Benutzers).

Die Benutzungsoberflächen sind in den Ansteuerungen der Grafikelemente sehr unterschiedlich. Deshalb ist eine strenge Modularisierung notwendig.

Alle Programmteile (also auch die von uns genutzten) müssen adaptierbar an das Zielsystem sein. Dies führt automatisch zu der Überlegung für einen Einsatz von GUIM's (engl. Abk.: Graphical User Interface Manager) wie z.B. HyperLook, zur Unterstützung des Designs und der Implementierung.

Die Vorteile eines solchen GUIM-Systems sind:

- PRESENTATION-CONTROL - Schicht (Ein-/Ausgabe-Schnittstelle) ist relativ systemunabhängig (da modular!)
- Sie sind effizient und flexibel bei der Implementierung und bei Änderungen an den Benutzungsoberflächen. Abstraktion der Entwicklung stellt technische Belange bei der Implementierung in den Hintergrund.
- Bei der Erstellung der Benutzungsoberfläche findet eine direkte Rückopplung durch direkte Manipulation an der Oberfläche statt.
- Die ersten Prototypen sind relativ rasch verfügbar.
- Eine Anpassung an verschiedene Sprachen ist leicht möglich.
- Die Benutzungsoberfläche ist weitgehend wiederverwendbar und in sich konsistenter.
- Es kann eine Prototypen - Erstellung in Zusammenarbeit mit den künftigen Benutzern des Systems geschehen. Wünsche und Vorschläge können während des Designerprozesses besser berücksichtigt werden.

Nachteile:

- Programmierer muß sich an das (neue) Konzept anpassen.
- Eventuell müssen die Programmierschnittstellen an den GUIM angepaßt werden.
- Eine zu starke Verschachtelung zwischen PRESENTATION und APPLICATION-FUNCTION - Schicht könnte einen Bypass zu unserer Applikation notwendig machen. Eine uneinheitliche Struktur des Programmes wäre die Folge.
- Man ist relativ stark auf die angebotenen Funktionen des GUIM's angewiesen.

4. Weitere Aufgaben in der Entwurfsphase können sein:

- Entwurf einer entsprechenden Modularisierung.
- Erarbeitung von Grunderscheinungsbildern der Dialogformen zwecks Vereinheitlichung der Benutzungsoberfläche: Piktogramme, Masken, Dialogverhalten, Hilfesysteme...

7. Umfrage zur Nutzung

Jan Hiller, Andreas Hinken

7.1. Zweck der Umfrage

Um eine bedarfsgerechte Systemspezifikation aufstellen zu können, ist es in jedem Falle sinnvoll, die Benutzer solcher Systeme zu befragen, um ggf. die eigenen Vorstellungen modifizieren zu können.

Für dieses Projekt haben wir eine Befragung durchgeführt. Die von uns konzipierte Textverarbeitung soll hauptsächlich für die Verarbeitung wissenschaftlicher Texte auf Workstation-Basis gestaltet werden. Die Auswertung unserer Umfrage hat allerdings ergeben, daß zu 70% PC Benutzer, die größtenteils Briefe und allgemeine Texte bearbeiten, befragt worden sind. Deswegen halten wir die Ergebnisse nur für bedingt verwertbar.

Aus Gründen der Vollständigkeit publizieren wir unsere Umfrageauswertung trotzdem und hoffen, daß die Ergebnisse zumindestens als grobe Orientierungshilfe dienen können.

7.2. Auswertung

7.2.1. BenutzerInnenprofil

Insgesamt wurden 41 BenutzerInnen befragt, von denen ein Großteil PC-BenutzerInnen sind (69%). Andere Systeme wurden vereinzelt berücksichtigt (1-3 Befragte/System).

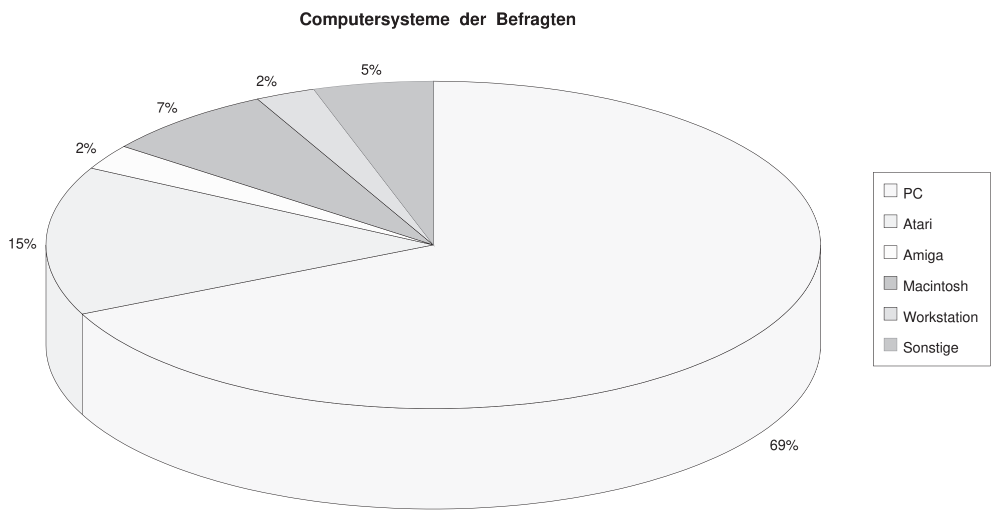

Abbildung 31. Verwendete Systeme

Die Nutzung der Computersysteme erfolgt beruflich und privat fast gleichermaßen.

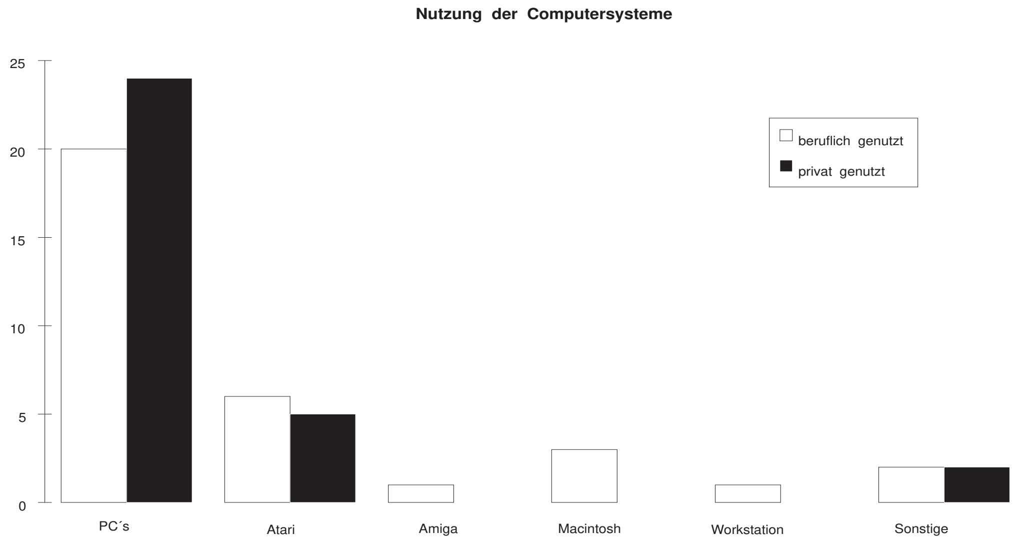

Abbildung 32. Nutzung der Systeme

Wie bereits eingangs erwähnt, bilden Briefe und allgemeine Texte eine Hauptanwendung der verwendeten Systeme. Unter dem Auswertungspunkt “Sonstige” sind Textarten zusammengefaßt, die jeweils nur einmal genannt worden sind. Darunter folgende: Formulare, Adreß-, Literatur-, Artikellisten, Rechnungen, Diplomarbeiten, Bewerbungen, Etiketten.

Wir vermuten, daß die hohe Zahl der Nennungen von Briefen und allgemeinen Texten darauf zurückzuführen sind, daß diese als Beispiel auf dem Fragebogen genannt worden sind.

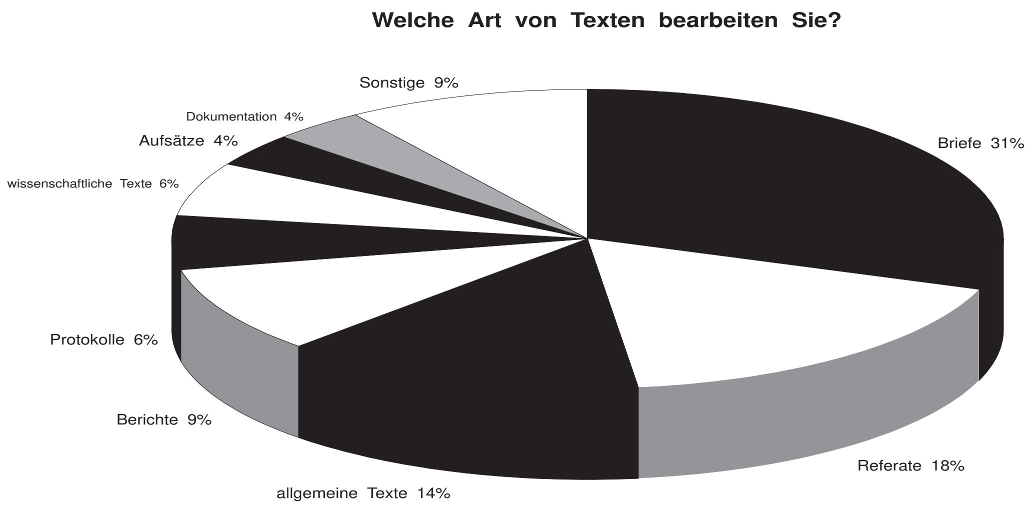

Abbildung 33. Art der verarbeiteten Texte

7.2.2. Wissensstand über die Software und Nutzung von Funktionen

Die befragten BenutzerInnen glauben 68% der Funktionalität ihrer Software zu kennen. Von den vorhandenen Funktionen werden nach den Angaben 48% bei der täglichen Arbeit genutzt.

Zu den hauptsächlich verwendeten Möglichkeiten der Textverarbeitungen zählt die Einbindung von Grafiken und Tabellen, gefolgt von Text-/Layoutfunktionen. Zu diesen zählen wir z.B. Textattribute und Formatierung. Da diese Funktionen bei fast jedem Text zur Anwendung kommen, scheint die Häufigkeit der Verwendung zu gering. Auch an dieser Stelle führen die aufgeführten Beispiele (Tabellen, Grafik) zur Verfälschung der Umfrageergebnisse.

7.2.3. Resonanz auf die vorgeschlagene Funktionalität

Deutlich zu erkennen ist der Wunsch nach einem WYSIWYG, welches dem Anspruch dieses Wortes gerecht wird. Die Möglichkeit von Objektimporten ist ebenfalls als gut bewertet worden, vermutlich aus der häufigen Verwendung von Tabellen und Grafiken im Text resultierend. Deutlich am schlechtesten schnitt das Vorlesen eines Dokuments ab, während alle anderen Vorschläge als mittelmäßig befürwortet worden sind.

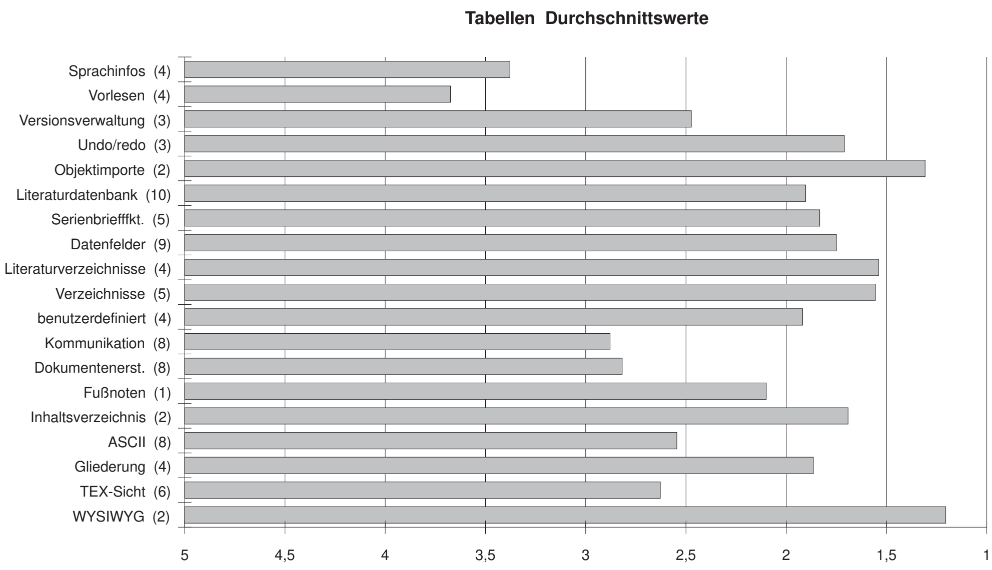

Abbildung 34. Verwendete Funktionen

7.2.4. Textuelle Auswertung

Wir haben die zusätzlichen Anregungen der Befragten zusammengefaßt. Aufgrund der inhaltlichen Vielfalt der Wünsche, beschränken wir uns auf die folgende Tabelle:

Anregungen und Wünsche	Anzahl
gutes Benutzerhandbuch	1
gute Online-Hilfe	4
möglichst einfache Bedienung/nicht überladen	15
Funktionskonfiguration	2
Formatierungsleiste mit Funktionen	1
besseres WYSIWYG	5
Unterschiedliche Zoomgrößen	4
Steuerzeichensicht	2
verschiedene Sichten/Ausschnitte in unterschiedlichen Fenstern	1
mehrere Texte gleichzeitig bearbeiten	1
Formularmodus	1
Unterstützung beim Layouten	9
freie Platzierung von Grafik und Formeln	5
typografische Eigenschaften für schöne Texte	6
einfacher Wechsel zwischen Formulartypen	1
gute Fußnotenverwaltung	1
Stichwortverzeichnis selbst erstellen	1
Text auf Vollständigkeit prüfen	1
Rechtschreibung/Grammatik/Thesaurus/Lexikon	6
guter Trennalgorithmus	2
Makros	2
Hot-Keys	4
Formeleditor	2
Grafikeditor	3
Fonteditor	2
gute Im-/Exportmöglichkeiten	5
Fremdprogramme einbinden	5
Dateioperationen	4
Statistik	1
gute GESCHWINDIGKEIT	5
klein und kompakt	2
Spracheingabe	3

Abbildung 35. Gewünschte Funktionalität

7.2.5. Fazit

Erst bei der Auswertung fielen uns die Mängel des von uns gestalteten Fragebogens auf. Zur Verbesserung der Befragung hätten einige Fragen die Streuung der Antworten stärker eingrenzen müssen. Statt der Aufzählung von einigen Beispielen hätte die Verwendung von Fragen im "Multiple-Choice"-Stil genauere Ergebnisse geliefert und gleichzeitig zur Vermeidung der Häufung der von uns genannten Beipiele geführt.

Weiterhin hat die mangelnde Disziplin der Befrager im Hinblick auf die korrekte Beantwortung der Fragen dazu beigetragen, daß unbrauchbare Ergebnisse entstanden sind. Auch die fehlende Koordinierung bei der Auswahl der Befragten bezüglich der verwendeten Systeme war sicherlich daran beteiligt.

In einer zweiten, neubearbeiteten Umfrage werden wir versuchen diese Fehler zu vermeiden, um ein aussagekräftiges Ergebnis für das Projekt vorlegen zu können.

Anhang

Anhang A: Liste der Standardfunktionalität

Block-Funktionen:

- Block Anfang
- Block Ende
- Block speichern
- Block laden
- Block drucken
- Block als ASCII speichern
- Block als ASCII laden
- Block ausschneiden
- Block einfügen
- Block kopieren
- Block löschen
- Block auf das Klemmbrett
- Block vom Klemmbrett holen
- Block sortieren
- Block linksbündig
- Block rechtsbündig
- Block zentrieren
- Block in Blocksatz
- Block doppelbündig
- Block dreifachbündig
- Block als Titel
- Block als Kapitel
- Block als Paragraph
- Block als Überschrift
- Block als Unterüberschrift
- Block als . . .
- Block als normaler Text
- Block als Kopfzeile
- Block als Fußzeile
- Block in andere Schriftgröße absolut

- Block in anderer Schriftgröße relativ
- Block kursiv
- Block einrücken
- Block unterstreichen
- Block hochstellen
- Block runterstellen
- Block in Kleinschrift
- Block in Großschrift
- Block invertieren
- Block mit Schatten
- Block nur Aussenlinien
- Block fett
- Block dünn
- Block breit
- Block in Normalschrift
- Block in anderer Schriftart
- Block in anderen Farbe
- Block in normaler Farbe
- Block spiegeln
- Block drehen
- Block als Fußnote
- Fußnote als Block
- Block nicht trennen
- Block in einer Sprache trennen
- Block Korrektur mit Wörterbuch
- Block als Zitat
- Block als Autor
- Block übersetzen
- Block als Mail
- Mail als Block
- Block in Form pressen

Bilder-Funktionen:

- Bilder laden
- Bilder speichern
- Bild vergrößern
- Bild verkleinern
- Bild drehen
- Bild spiegeln
- Bild Invertieren
- Bild an Drucker anpassen
- Bild an Bildschirm anpassen
- Farben des Bildes verändern
- Bild als Gleitobjekt in den Text
- Bild als Festobjekt in den Text
- Text fließt um das Bild
- Bild auf das Klemmbrett
- Bild vom Klemmbrett holen
- Bild verschieben
- Bild mit Tastenkombination belegen
- Bild löschen

Tabellen:

- Tabelle laden
- Tabelle speichern
- Tabelle auf das Klemmbrett
- Tabelle vom Klemmbrett holen
- Tabelle als Gleitobjekt in den Text
- Tabelle als Festobjekt in den Text
- Text fließt um die Tabelle
- Tabelle verschieben
- Tabelle in anderer Farbe
- Tabelle mit Tastenkombination belegen
- Tabelle löschen

Formeln:

- Formel laden
- Formel speichern
- Formel auf das Klemmbrett
- Formel vom Klemmbrett holen
- Formel als Gleitobjekt in den Text
- Formel als Festobjekt in den Text
- Text fließt um die Formel
- Formel vergrößern
- Formel verkleinern
- Formel mit Tastenkombination belegen
- Formel in anderer Farbe
- Formel löschen

Datei:

- Text anlegen
- Text anlegen als Brief, Buch, Artikel, Broschüre, Zeitung, Flugblatt, . . .
- Text laden
- Texte laden und anhängen (Texte verknüpfen)
- Text speichern mit oder ohne Backup
- Text speichern
- Text alle X Minuten Speichern
- Text als ASCII laden
- Text als ASCII speichern
- Text als Graphik speichern

Drucken:

- Druckertreiber laden
- Treiber einstellen
  - Drucken auf Einzelblatt
  - Drucken auf Endlospapier
  - Einzelblatteinzug automatisch
  - Einzelblatteinzug manuel
  - Drucken von bis
  - Anzahl der Exemplare
  - Rand einstellen

- Seitenformat einstellen
- Seitenverkehrt drucken
- Invers drucken
- Sortiert drucken
- Unsortiert drucken
- Helligkeit einstellen
- Auflösung einstellen
- Vergrößern
- Verkleinern
- Gedreht drucken
- Druckvorgang abbrechen
- Druckerport auswählen
- Einstellung laden
- Einstellung speichern

• Text drucken
• Text als ASCII drucken
• Text als Graphik drucken
• Text über Postscript drucken
• Alle Druckfunktionen auch als Serienbriefe und als Farbdruck
• Seite in mehreren Stücken drucken (Seite zu groß)

Wörterbuch:

• Wörterbücher laden
• Wörterbücher speichern
• Wörterbuch für den Text auswählen
• Wörter einfügen
• Wörter löschen
• Wörter korrigieren
• Wörter suchen
• Wörterbuch einschalten
• Wörterbuch ausschalten
• Wörterbuch löschen

Hilfsfunktionen:

• Funktionstasten belegen
• Belegung laden
• Belegung speichern

- Menüs mit Tastenkombination belegen
- Belegung speichern
- Belegung laden
- Funktionen zu Makros zusammenfassen
- Makros laden
- Makros speichern
- Makros ansehen
- Makros mit Maus erzeugen
- Makros mit Tastenkombinationen belegen
- Makros können mit Maus aufgerufen werden
- Makro löschen
- Makros ändern
- Undo/Redo
- Lineal zeigen
- Tabulatoren setzen/löschen/verschieben fürs Dokument oder Block (links rechts Punkt Mitte)
- Seitenübersicht
- Ganzseitenansicht
- Zoom auf Seitenteile
- Bilder zeigen
- Tabellen zeigen
- Formeln zeigen
- Suchen/Ersetzen (einmalig alles abfragen vorwärts rückwärts im Block)
- Zeichensätze laden
- Zeichensätze löschen
- Datum einfügen als Text
- Datum einfügen als Symbol
- Zeit einfügen als Text
- Zeit einfügen als Symbol
- Inhaltsverzeichnis einfügen
- Fußnoten auf die Seite unter den Absatz / unter das Kapitel / an das Ende des Dokumentes
- Autor und Quellen an das Ende des Zitates / unten auf die Seite / an das Ende des Absatzes / ans Ende des Kapitels / ans Ende des Dokumentes

- Autorenverzeichnis einfügen
- Quellenverzeichnis einfügen
- Variablen Definieren
- Variablen mit Daten belegen
- Einfügen oder Überschreiben
- Linie in den Text zeichnen
- Text umranden

Seiten:

- Seite einfügen
- Seite löschen
- Seite laden
- Seite speichern
- Seite auf das Klemmbrett
- Seite vom Klemmbrett holen
- Seite verschieben
- Seite einfügen

Dokument:

- Größe festlegen (Din A4 hoch quer . . . Frei)
- Ränder festlegen
- Schriftart festlegen
- Schriftgröße festlegen
- Spalten festlegen
- Einseitig doppelseitig
- Abstand Kopfzeile
- Abstand Fußnote
- Abstand Fußzeile
- Größe Fußnoten releativ-absolut
- Seitennummer Start auf Seite X
- Seitennummer Start mit Nummer X

Layout:

- Schriftart ändern: Absolut-Relativ, Fürs Dokument, Zeile, Absatz, Block,
- Schriftgröße: Absolut-Relativ, Fürs Dokument, Zeile, Absatz, Block
- Buchstabenabstand: Vergrößern, Verkleinern, Absolut-Relativ zur Schriftart
- Maximaler Minimaler Buchstaben- und Wortabstand: Vergrößern,
- Verkleinern, Absolut-Relativ (Schriftart in cm oder Punkt),
- Zeile, Absatz, Dokument, Block
- Schrift: Fett, Dünn, Breit, Klein, Kursiv, Schatten, Groß, Klein, Hochstellen,
- Tiefstellen, Drehen, Invers, Gespiegelt, Normal

Anhang B: Datenaustauschformate

Die zu importierenden Programme werden in der folgenden Tabelle kurz aufgeführt:

Programmart	**Betriebssystem**	**Programmname**	**Exportformat**
Tabellenkalkulation	MS-DOS	Louts 1-2-3	.WK1
Tabellenkalkulation	MS-DOS	Quattro Pro	.WKQ
Tabellenkalkulation	Windows	Excel	.XLS
Tabellenkalkulation	MS-DOS	Lotus 1-2-3	.WKS
Datenbankprogramm	MS-DOS	dBase	.DBF
Datenbankprogramm	MS-DOS	Paradox	.DB
Datenbankprogramm	MS-DOS	Oracle	.ORA
Datenbankprogramm	Unix	Informix	.IDX/.DAT
Textverarbeitung	MS-DOS	WordPerfect	.DOC
Textverarbeitung	Mac	MacWord	.DOC
Textverarbeitung	Windows	WinWord	.DOC
Textverarbeitung	Unix	Framemaker	.DOC
Grafikprogramm	MS-DOS	DrawPerfect	.PCX

Tabelle 1. Zu importierende Programme

Anhang C: Softwareergonomie-Kriterienkatalog

- Kriterien zur Informationsgestaltung
- Kriterien zur Selbstverwaltung des Systems
- Kriterien zu den Steuerungsmöglichkeiten des Benutzers
- Kriterien zu Systemmeldungen und Fehlerbehandlung

Anhang C.1: Informationsgestaltung am Bildschirm

1. Betonung und Hervorhebung
   - kein Übergebrauch von Markierungsarten (<2 Intensitätsstufen, <6 Farben, nur Cursorblinken)
2. Darstellung auf dem Bildschirm
   - Anzeige einer vertikalen und horizontalen Cursorposition im Lineal
   - Einblenden von Steuerzeichen (Art und Umfang einstellbar)
   - einstellbare Darstellungsgröße (Zoom) für jedes Fenster
   - Lupe
   - selektierte Objekte kennzeichnen
   - ähnliche Informationen haben ähnliche Darstellung
   - Icons/Abbildungen statt umfangreichen Text
   - analoge Anzeigen statt Zahlen
   - Formen und Farben aus der Erfahrungswelt des Benutzers
   - Erfahrungen aus anderen Anwendungen berücksichtigen
3. Ebenenkonzept (Menge der dargestellten Information steuerbar)
   - optionales Ausblenden von redundanten Text mit best. Auszeichnung oder auf best. Ebene (Ein, Rahmen, Aus)
   - optionales Ausblenden von Grafikobjekten (beliebig differenzierbar), Farbebenen und Notizen (für jeden Benutzer eine Ebene)
   - Ebenen schützbar (Funktionen darauf)
4. Darstellung von Formatzeilen
   - Optionales Ausblenden von Formatzeilen (Einzugsmarken etc.)
   - Verwendung einer Formatzeile im Kopfteil
   - Verwendung von Formatzeilen im Text
5. Gestaltung des Textcursors
   - Textcursor ist mit der Maus zu bewegen
   - Bewegung zeichenweise, wortweise (Satz, Absatz, Seiten, Kapitel)
   - Bewegung zum Textanfang/-ende
   - beim Systemstart erfolgt die Positionierung an der zuletzt gewählten Textstelle
   - nach einem Funktionsaufruf oder einem Dialogaufruf steht der Cursor auf einer sinnvollen Ausgangsposition
   - Zeichen unter dem Cursor ist sichtbar (Transparenz auch in Farbe)

- horiz./vertik. Textcursor im Linieal
- Variabler Textcursor (horizontal/vertikal Anpassung an Buchstaben)
- autom. Textcursorpositionierung (frei, Textgebunden, Zeilenende)

6. Gestaltung des Grafikcursors
   - Position nur durch den Benutzer änderbar (keine autom. Sprünge)
   - Positionsanzeige (digital und/oder im Linieal)
   - Änderung der Darstellung bei Funktionswechsel (Milbe,Fadenkreuz ..)
   - Unterstützung der Objektpositionierung durch Hilfslinien, Hilfsraster, magnetische Linien und Raster
   - zuschaltbare Positionierungshilfe (proportionales Aufziehen, horizontales und/oder vertikales Sperren, Objektbereich sperren)
7. Positionierungshilfe
   - Objekte einpassen/ausrichten (links, rechts, oben, unten, horiz., vert., zentr.)
   - Objekte innerhalb von einstellbarer/vorgegebener Schranken verschiebbar
8. Rollen (Scrolling)
   - Beeinflussung von Rollgeschwindigkeit, Sprungweite und Sprungadresse durch den Benutzer
   - Benutzerangepaßte Scrollgeschwindigkeit
   - automatisches Scrollen bei Berühren der Fenstergrenze
9. Sichten
   - einheitliche Dialoge und Darstellungen bei unterschiedliche Sichten
10. Kooperation
    - transparente Gruppenarbeit
    - Grad der zulässigen Dokumentenänderung bestimmbar (Funktionsselektion, für alle oder einzelne Personen)

Anhang C.2: Selbstverwaltung des Systems

1. Datenträgerverwaltung
   - automatische Rückspeicherung von Texten
   - persistente Arbeitsumgebung
   - automatisches Laden von zugehörigen Programmen bei Aufruf des Dokumentes
2. Schutz vor unbeabsichtigtem Löschen
   - Bestätigung vor kritischen Funktionen (mehrstufig)
   - Default-Insert-Modus zum Schutz vor versehentlichen Überschreiben von Text
3. Antwort-/ Verarbeitungszeiten
   - benutzergerechte Antwortzeiten (max. 2 Sekunden)
   - erwartungskonforme Verarbeitungszeiten
   - Statusanzeige bei längeren Prozessen (Drucken, Laden..)

- Benutzer nicht nötigen, keine zeitlich begrenzten Dialoge (z.B. Popup darf nicht verschwinden, wenn Maustaste losgelassen wird)

4. Zeilenumbruch / Reformatierung
   - automatischer Zeilenumbruch
   - automatischer Seitenumbruch
5. Konsistenzprüfung
   - sind alle Referenzen und Objekte verknüpft? (Text zum Bild, Bild zum Text)
   - Ist die Folge der Textsegmente korrekt? (Inhaltsverzeichnis, Text, Literaturverzeichnis ...)
6. lokaler Datenschutz
   - Passwortvergabe
   - Benutzerrechte

Anhang C.3: Steuerungsmöglichkeiten des Benutzers

1. Wahl des Eingabemediums
   - Wahlfreiheit des Eingabemediums (Wechselnotwendigkeit minimieren)
   - Einsatz von Zeigehilfen (z.B. Lichtgriffel) für Bewegungsbefehle
   - Einsatz von Tastatur und Zeigehilfen für Steuerkommandos
   - Unterstützung mehrerer Tastaturen, Tastaturbelegung erwartungsgemäß
   - gleiche Tastenbelegung in verschiedene Dialoge
   - handschriftliche Eingaben (z.B. Korrekturen, Notizen)
2. Wahl der Kommunikationsform
   - Einstellung benutzerabhängiger Kommunikationsformen zu Beginn des Dialogs (Laien -/ Expertenmodus)
   - Wahlfreiheit der Kommunikationsform / Bereitstellung unterschiedlicher Kommunikationsformen zur Zielerreichung
   - mögliche Zusammenstellung einzeloptimaler Dialogschnittstellen, Verwendung schon bekannter Module
   - ähnliche Arbeitsaufgaben haben ähnliche Dialoge
   - ähnliche Objekte haben ähnliches Verhalten
   - gleiche Aktionen haben gleiches Ergebnis
   - Dialoge und Menüs statt Kommandos
   - Angebot von generischen Kommandos (konsistente Funktionen, weniger Betriebsmittel)
   - Angebot von Menüs, Formulareingaben
   - Angebot beschrifteter Funktionstasten für häufig gebrauchte oder universelle Funktionen (Cut, Copy, Paste, Undo, Redo)
   - abgeschlossene Kommunikation (Wechsel vermeiden)
   - Ablage von Blöcke, Seiten, Kapitel auf dem Desktop
   - Funktionen mit (unerwünschten) Nebeneffekten vermeiden

- Dialoge sollten möglichst wenig der Arbeitsfläche überdecken (z.B. transparente Dialoge)
- nicht zu kleine Objekte (Mikrobewegung motorisch schwer)
- geringer Abstand zusammenhängender Kommandos (geringe Mausbewegung)

3. Gestaltung von Dialogmasken
   - sinnvolle Maskengliederung, keine Überladung (Selektion mit Scrollmenüs)
   - dokumentabhängige Speicherung von Formular-Eingabewerten als Standardvorgaben für die Zukunft (Default-Werte)
   - Anzeige der ursprünglichen und der geänderten Daten
   - enthält eine abgeschlossene Teilaufgabe (keine Verstreuung, Dialog im Fenster)
   - Positionierung immer gleich oder wählbar: zentriert, Mausposition
   - Maske auch nach Bestätigung der Eingabe sichtbar (als Default; auch abschaltbar)
   - je nach erforderliche Genauigkeit digitale und analoge Objekte
   - Tastaturkürzel einheitlich, jederzeit sichtbar (z.B. Funktion)
   - Ordnung in der Maske (kurze Auswahlzeiten/Lernphase, alphabetisch, nach Häufigkeit der Anwendung, nach Wichtigkeit, in funktionale Gruppen geordnet)
   - Gruppenbildung, 3 - 7 Elemente
   - Abbruch/Fortsetzung eines Dialogs jederzeit möglich
   - setzen, suchen, ersetzen und markieren im gleichen Dialog

4. Gestaltung von Menüs
   - aufgaben-/ benutzerorientierte Menügestaltung
   - Auflisten von Tastenkommandos (Kürzel) in den Menüpunkten
   - optionale Sichtbarkeit der Menüs (push-pin Menüs)
   - kurze, konsistente, eindeutige Benennung der Felder ohne redundante Informationen
   - hierachische Menüs (Submenüs) mit möglichst geringer Tiefe
   - Ordnung im Menü (s.o.)

5. Gestaltung von Kommandoeingaben
   - leicht verständliche Syntax
   - gestufte Kommandokomplexität
   - einheitliche, transparente Bezeichner (möglichst keine Abkürzungen)

6. Angebot von direkter Manipulation
   - semantische Direktheit (Verfügbarkeit der Funktionen gemäß der Benutzerintension)
   - operationale Direktheit (Handlungseinheit ohne Zwischenschritte, möglichst einstufige Benutzeraktionen)
   - informationelle Anzeigen selektierbar/editierbar (Seiten-, Zeilenanzahl)
   - formale Direktheit (Verständlichkeit, WYSIWYG)
   - permanente Sichtbarkeit der interessierenden Objekte
   - autom. Übernahme in Masken

7. Wahl der Objektdarstellung am Bildschirm
   - Wahl der Benutzungsoberfläche
   - Manipulation der Fenster (Größe, Position ....)
   - Auflösungsunabhängige Darstellung (verschiedene Rechner/Monitore)
8. Benutzerprogrammierung
   - Möglichkeiten der Funktionstastenprogrammierung
   - Abgleich mit bestehenden Funktionstastenprogrammierungen
   - Anzeige der Funktionstasten
   - Makroprogrammierung (Makrorekorder, persistent, global, lokal)
   - Programmierungshilfen
   - Menugeneratoren (Belegung von Popup-Menüs)
9. Parameter-Steuerung
   - Parameteranzahl steuerbar (mehrstufig nach Benutzerkenntnis)
   - passiv: sinnvolle (in Bezug auf den letzten Dialog) Vorbelegung
   - aktiv: Default-Werte speicherbar in Liste
   - persistente Parameter durch automatische Rückspeicherung (abschaltbar, bei Programmende, nach x Minuten, nach Dialogende, bei kritischen Funktionen, manuell)
   - Parameter für aktuelles Dokument (intern), Standard - Dokumententyp (global)
10. Abbruch / Rückgängigmachen von Dialogschritten
   - Abbruch einer Funktion möglich
   - Bereitstellung einer UNDO-Funktion (aktive, [passive] Steuerung)
   - Rückgängigmachung mehrerer Dialogschritte
   - Darstellung der möglichen Undo/Redo-Schritte
11. Unterstützung bei Handlungspläne
   - Erstellung individueller Pläne für eine Aufgabe
   - autom. / manuelles setzen von Checkpoints
   - visualisieren von Pläne (Was habe ich schon gemacht?)
   - mögliches Abbrechen und Wiederaufnehmen von Pläne
   - möglicher Einstieg in zurückliegenden Zuständen (Ändern, Löschen)
   - erkennen von Handlungen und Auslösung von Aktionen (mögliche Rückfrage) oder Dialogaufbau (z.B. Selektion “Linienobjekt” => Einblendung des Liniendialog für Attributwahl)
   - Konsistensprüfung von Plänen
12. Individualisierbarkeit / Adaptierbarkeit
   - Grad der möglichen (maximalen) Änderung nicht zu hoch (gemeinsamer Nenner muß bleiben, einige Änderungen nicht sinnvoll, z.B. Aussehen von Icons)
   - optional selbständige Anpassung (adaptive Systeme), notwendige Information des Benutzers
   - anpaßbare Tastaturbelegung, Wiederholgeschwindigkeit
   - anpaßbare Maus (1-3 Knöpfe, Klickgeschwindigkeit, logarithm. Bewegung ...)

Anhang C.4: Systemmeldungen und Fehlerbehandlung

1. Systemmeldungen
   - erwartungskonformes Feedback auf Benutzereingaben
   - akustisches Feedback (optional)
   - kein irrelevantes Erregen von Aufmerksamkeit
   - Meldungen stets an der gleichen Position (einstellbar)
   - Systemzustandsmeldungen bei voraussehbar länger dauerden Verarbeitungsprozesse (benutzer-/ aufgabenbezogen)
   - Angaben über Restverarbeitungdauer abrufbar
   - angepaßt an Kenntnisse/Sprache des Benutzers (Default: deutsch)
   - Darstellen, welches Fachwissen oder Kenntnisse aus z.B. Schulungen erforderlich sind
2. Fehlermeldungen / Fehlerbehandlung
   - Fehleingaben durch Vorgaben und Selektionen verhindern
   - Rückkehr zur gesicherten Systemzustand jederzeit ermöglichen
   - keine vollständige Neueingaben von fehlerhaften Kommandos
   - Benutzer- und situationsangepaßte Fehlermeldungen
   - positiver Tonfall, angemessene sprachliche und inhaltliche Darstellung
   - mehrstufige Fehlermeldungen (kurze/genaue Fehlerbeschreibung, weitere Referenzen)
   - optional automatischer Stufenwechsel bei korrekter Anwendung einer Aktion oder Häufung eines Fehlers
   - Fehlerursache beschreiben (Verletzung der Datenintegrität, programminterne Fehler, fehlerhafte Funktionsanwendung auf Objekte (Kombinierbarkeit), logische Folgefehler, Warum ist ein Kommando nicht verfügbar?...)
   - Korrekturhinweise (mehrere Alternativen)
   - automatische Korrektur (abschaltbar)
   - Unterstützung bei der Lokalisierung von Fehlern
3. Hilfesystem / Hilfmeldungen
   - passive Hilfe (Wahl über Funktionstaste oder Kommando)
   - aktive Hilfe (suboptimale Handlungen werden erkannt)
   - statische Hilfe (Online-Manual)
   - Funktionswahl mit kurzem Text einblenden
   - dynamische Hilfe (situationsbezogene, benutzerangepaßte Hilfe)
   - aktive dynamische Hilfe (Verhaltensanalyse, System gibt Vorschläge)
   - Vorschlag eleganterer und einfacherer Lösungsmöglichkeiten als vorgenommen (Makroprogrammierung, Funktionsreihenfolge...)
   - Hilfe zu Interaktionstechniken: Übersicht der Funktionstastenbelegung/Menüs/Masken und Felder (Abhängigkeit der Parameter untereinander); Erklärung von Handlungsabläufe; Welche Angaben müssen/können vorliegen
   - Hilfe zur Funktionalität:Was geht autom.? Welche Funktionen gibt es?

- Hilfe zur Adaptierbarkeit: Was kann angepaßt werden? Wie? Was läßt sich autom. anpassen?
- Hilfe zur Steuerbarkeit: Wo bin ich? Wie komme ich hierher? Was kann ich hier tun? Wo kann ich hin? Welches sind hier sinnvolle Aktionen?
- informative verständliche, konsistente Hilfsmeldungen
- Übereinstimmung mit geschriebener Dokumentation bezüglich Inhalt und Darstellung
- mehrstufige Hilfe wählbar (kurzer,normaler,ausfürlicher Text)
- Hilfsmeldung so einblenden, daß Ursache der Hilfsforderung sichtbar bleibt
- Ergänzung mit Beispiele
- eigene Hilfetexte zufügbar
- Rechtschreibhilfe, Thesaurus
- Glossar, Kurzreferenz

4. Lernförderlichkeit / Erlernbarkeit
   - Modellbildung unterstützen
   - Minimierung der zu lernenden Funktionen/Kommandos/Begriffe/Syntax (auf Bekanntes aufbauen)
   - Tutorfunktion (Übungslexionen, Folge steuerbar)
   - Qualifizierungsmöglichkeit durch erweiterte Aktionsmöglichkeiten

Anhang D: Fragebogen

Jan Hiller, Andreas Hinken

Universität Bremen
- FB Informatik -
Projekt: Formatierung von Dokumenten mit verschiedenen Sichten

Fragebogen zu Textverarbeitungssystemen

1. Grundinformationen

1.1 Auf welchem Rechnersystem arbeiten Sie? (PC, Mac, Workstation)

1.2. Welche Textverarbeitung benutzen Sie zur Zeit? (Winword, Word)

1.3. Benutzen Sie die Textverarbeitung beruflich oder privat?

1.4. Welchen Stellenwert hat die Textverarbeitung bei Ihrer Tätigkeit?

1.5. Wie gut sind Sie mit Ihrer Textverarbeitung vertraut?

1.6. Welche Art von Texten be- bzw. verarbeiten Sie am häufigsten? (Briefe, Texte)

2. Funktionalität

2.1. Wieviel Prozent der Funktionen Ihrer Software glauben Sie zu kennen?

2.2. Wieviel Prozent der Funktionen Ihrer Software benutzen Sie?

2.3. Welche Funktionen Ihrer Software benutzen Sie? (Tabellen, Grafik im Text)

2.4. Welche Funktionen empfinden Sie als störend bzw. überflüssig?

2.5. Sind Sie mit der angebotenen Funktionalität zufrieden?

2.6. Welche Funktionen lassen sich besonders gut, welche besonders schlecht bedienen?

2.7. Welche Eigenschaften wünschen Sie sich von Ihrer “Traum”-Textverarbeitung?

3. Unsere Vorstellungen

Frage	Note	Zusätze
3.1. Multi-View		
3.1.1. WYSIWYG		
3.1.2. TEX-artige Steuerzeichen		
3.1.3. grafische Sicht der Gliederung		
3.1.4. ASCI-Sicht		
3.1.5. Inhaltsverzeichnis		
3.1.6. Fußnoten		
3.2. Mehrbenutzerbetrieb		
3.2.1. Erstellung von Dokumenten		
3.2.2. Kommunikation zwischen Usern		
3.3. aktive Querverweise		
3.3.1. benutzerdefinierte Verweise		
3.3.2. Verzeichnisse		
3.3.3. Literaturverzeichnisse		
3.4. externe Querverweise (keine Kopien)		
3.4.1. Datenfelder (z.B. Excel)		
3.4.2. Serienbrieffunktion		
3.4.3. Literaturdatenbankanbindung		
3.4.4. Objektimporte (z.B. Grafik)		
3.5. Entwicklungsgeschichte		
3.5.1. Undo/Redo-Funktion		
3.5.2. Versionsverwaltung		
3.6. Sprachausgabe		
3.6.1. Vorlesen des Dokumentes		
3.6.2. Einbauen von Sprachinfos		

Bewertung 0=keine Meinung, 1=sehr gut ... 5=schlecht

4. Zusätzliche Vorschläge und Anregungen

Anhang E: Abbildungsverzeichnis

Anhang F: Literaturverzeichnis

[App87] Apple Computer Inc.: „Human Interface Guidelines: The Apple Desktop Interface”; Addison Wesley; 1987

[Bor90] Günter Born:“Referenzhandbuch Dateiformate“;Addison-Wesley Verlag;1990

[Bru89] Brüggemann-Klein, A.: „Einführung in die Dokumentenverarbeitung“; B.G. Teubner, Stuttgart 1989

[DIN88] Deustsches Institut für Normung: „DIN 66 234 Teil 8, Bildschirmarbeitsplätze, Grundsatz ergonomischer Dialoggestaltung”; Berlin, Köln; 1988

[FK89] Fromme, K.; Krekel, D.: „Vergleich der Benutzerschnittstellen unter VM/ CMS (Rel. 4) und unter NOS/VE (Rel. 1.2.3.) auf der Basis eines Kenngrößenkatalogs”; in: Angewandte Informatik 6/89; S. 235 - 240

[Fu89] Richard Furuta, ‘Concepts and Models for Structured Documents’, in „Structured Documents“, eds., André J.; Furuta, R.; Quint, V.; , S. 7-38, Cambridge University Press; 1989

[Gar89] Gardner, H.: „Dem Denken auf der Spur”; Stuttgart: Ernst Klett Verlag für Wissen und Bildung; 1989

[GJ89] Gierlach, D.; Jankowski, R.: „Operationalisierung der Grundsätze ergonomischer Dialoggestaltung nach DIN 66 234 Teil 8 - Beispiel Textverarbeitung” in: Angewandte Informatik 5/89; S. 189 - 196

[KTR84] Klocke, H.; Trispel, S.; Rau, G.: „Entwicklung einer Mensch-Rechner Schnittstelle für ein Anästhesie-Informationssystem unter Berücksichtigung ergonomischer Gesichtspunkte”; in: Angewandte Informatik 5/84; S. 197 ff

[Opp89] Oppermann, R.:„Gestaltung der Mensch-Maschine-Kommunikation”; in: Informationstechnik 3/89; S.181 - 189

[Rub91] Rubinstein, R., „Digital Typography“; Addison-Wesley; 1991

[Sun90] Sun Microsystems Inc.: „OPEN LOOK Graphical User Interface Application Style Guidelines”; Addison-Wesley; 1990

Anhang G: Index