GI-Workshop Entwurf und Dokumentation im rechnergestützten Facility Management, ed. A. Iwainsky, Wartburg, Eisenach, Deutschland,11.-13. Oktober 1995

Steuerung der Informationspräsentation in Informationssystemen mit dem automatisierten Bildschirmlayout

P. Lüders, R. Ernst
Institut für Datenverarbeitungsanlagen
TU Braunschweig, 38106 Braunschweig

Kurzfassung

Die Leistungsfähigkeit eines Facility Management Systems beruht neben der eigentlichen Datenhaltung und -verarbeitung zu einem großen Teil auf der zweckmäßigen Darstellung der vom Benutzer gewünschten Information. Es wird aufgezeigt, daß die grafische Benutzerschnittstelle eines Facility Management Systems durch Einführung des automatisierten Bildschirmlayouts wesentlich verbessert werden kann. Zu diesem Zweck wird die Leistungsfähigkeit des Hypertextsystems HyperADL, in dem das automatisierte Bildschirmlayout bereits integriert ist, analysiert und es werden mögliche Erweiterungen wie Änderung der grafischen Darstellung eines Fensterinhalts bei Änderung der Größe des Fensters oder eine Animation des Layoutverhaltens diskutiert. Vorrangiges Ziel ist dabei, die Handhabbarkeit des Layoutsystems zu verbessern und damit dessen Akzeptanz beim Benutzer zu erhöhen.[1]

1. Einleitung

Facility Management Systeme unterstützen den Benutzer u.a. bei der Administration und Planung von Objekten, die zu einem bestimmten Kontext gehören. Objekte können dabei beispielsweise Rechner in einem lokalen Netz oder Gebäude in einem gegebenen Bezirk sein. Damit gehören diese Systeme zu den Informationssystemen, die in [Char94] mit „ein System, das die von seinen Benutzern gewünschte Information aus einer Datenbasis selektiert, ggf. mehrere Selektionen kombiniert und in zweckmäßiger Darstellung (z.B. Grafik, Text) anzeigt oder dokumentiert" definiert werden. Mit dieser Definition lassen sich zwei wesentliche Teile eines Informationssystems voneinander trennen: einerseits die Datenhaltung und Bestimmung gewünschter Information und andererseits die zweckmäßige Darstellung der Information. Der folgende Beitrag befaßt sich ausschließlich mit dem letzteren Punkt, der grafischen Informationspräsentation.

Mit der zunehmenden Verbreitung der grafischen, fensterorientierten Benutzerschnittstelle lassen sich zwei Trends beobachten. Zum einen werden immer mehr Informationen grafisch dargestellt. Dies ist insbesondere dann von Vorteil, wenn der Zusammenhang einzelner Informationseinheiten verdeutlicht werden soll (Sprichwort: „Ein Bild sagt mehr als tausend Worte"). Beispiele hierfür sind Rechner in einem lokalen Netzwerk (Netzwerklayout) oder wechselseitig abhängige Arbeitsschritte in einem Entwicklungsprojekt (Netzplantechnik). Zum zweiten werden aber auch immer mehr Fenster der fensterorientierten Benutzerschnittstelle verwendet, in denen die gewünschte Information dargestellt wird. Beispielsweise wird in einem Fenster ein Lageplan der Rechner aufgezeigt, in einem anderen Fenster eine genaue

1. Dieses Projekt wird gefördert im Rahmen der industriellen Gemeinschaftsforschung (Nr. 5.3 BesNB) aus Mitteln des BmWi.

Beschreibung eines zuvor spezifizierten Rechners und in einem weiteren Fenster eine grafische Darstellung der momentanen Auslastung des Netzwerkes.

Der vermehrte Einsatz beider Techniken birgt aber auch erhebliche Nachteile. So müssen beispielsweise bei einem neu hinzukommenden Rechner die gesamten Grafikdarstellungen manuell überarbeitet werden. Andererseits muß der Benutzer bei der Verwendung mehrerer Fenster selbst für ein jederzeit gutes und übersichtliches Bildschirmlayout sorgen, da sich die Fenster andernfalls überdecken und somit gewünschte Information zwar dargestellt, aber für den Benutzer nicht sichtbar ist. Zudem werden im momentanen Stand der Technik Zusammenhänge von Fenstern und Objekten bzw. von Objekten in verschiedenen Fenstern nicht visuell aufgezeigt (mit Ausnahme von unterschiedlichen Farbgebungen). Diese Nachteile sind jedoch für eine effiziente und produktive Arbeit mit einem Informationssystem nicht akzeptabel. Eine Möglichkeit zur Verbesserung der Benutzerschnittstelle liegt in der Verwendung von Layoutalgorithmen, mit deren Hilfe sowohl die grafischen Informationsdarstellungen innerhalb der Fenster als auch die Fensteranordnungen, d.h. Plazierung und Größeneinstellung, selbst automatisch generiert werden. Damit würde der Benutzer von der zeitaufwendigen und mühsamen manuellen Tätigkeit des manuellen Layoutentwurfs befreit und eine kontinuierlichere und produktivere Arbeit wäre möglich.

Während für den Bereich der Darstellungen innerhalb der Fenster auf eine große Basis von für verschiedene Problemstellungen optimierten Layoutalgorithmen zurückgegriffen werden kann (Forschungsbereich des Computer Aided Schematics, CAS) /Ple*94/, /Bat*93/, ist der Einsatz von Layoutalgorithmen zur Steuerung des Bildschirmlayouts ein vergleichsweise junger Forschungsbereich. Hier liegen jedoch unter anderem bereits Erfahrungen über die interaktive Benutzung von Informationssystemen in Form des Hypertextsystems *HyperADL* (Hypertext with Automatic Display Layout) vor, in dem das automatisierte Bildschirmlayout bereits integriert ist /Lüde95a/, /Lüde95b/. Ein Hypertext besteht im Gegensatz zum traditionellen, linearen Text aus einzelnen *Karten*, die durch *Verweise* miteinander verbunden sind /Wood90/. Für den Begriff der Karten wird im folgenden auch synonym dazu von elementaren Informationseinheiten oder auch von Knoten gesprochen. Wie in einem Buch, in dem ein Verweis auf eine andere Textstelle zeigt, an der weiterführende Information zu finden ist, so führt auch der Verweis eines Hypertextes zu einer anderen Karte. Doch während in einem traditionellen Text ein Verweis eher selten ist, da er das zumeist gewünschte, sequentielle Lesen unterbricht, ist ein Verweis in einem Hypertext üblicherweise die einzige Möglichkeit, zu einer anderen Informationseinheit zu gelangen. Das implementierte System ist dabei als ein kartenorientiertes Hypertextsystem angelegt. Der Benutzer kann dann durch Selektion von Informationseinheiten eine Menge von Karten spezifizieren, für die dann durch geeignete Layoutalgorithmen ein Bildschirmlayout generiert wird. Damit sind alle momentan ausgewählten Karten auf dem Bildschirm ohne gegenseitige Überdeckung sichtbar. Ein Bildschirmabzug beim Lesen eines C++-Programmes zeigt das Bild 1.

Allerdings ist das System *HyperADL* ein reines Experimentiersystem und daher für den durchschnittlichen Benutzer aufgrund der komplexen Bedienung nicht geeignet. So wurden beim Entwurf des Systems durch die Konzentration auf die Plazierungsproblematik einige Problemstellungen wie das Routing der visuellen Verbindungen zwischen Objekten und Fenstern bzw. Objekten in unterschiedlichen Fenstern oder die visuelle Darstellung von Ankern (d.h. Quell- bzw. Zielpunkte der Verweise) nicht angemessen berücksichtigt. Zudem ist das Layoutverhalten vom Benutzer nicht zu jeder Zeit vorhersehbar, da für die Plazierung wichtige, interne Attribute der Karten nicht visualisiert werden. Dies stellt eine starke Beeinträchti-

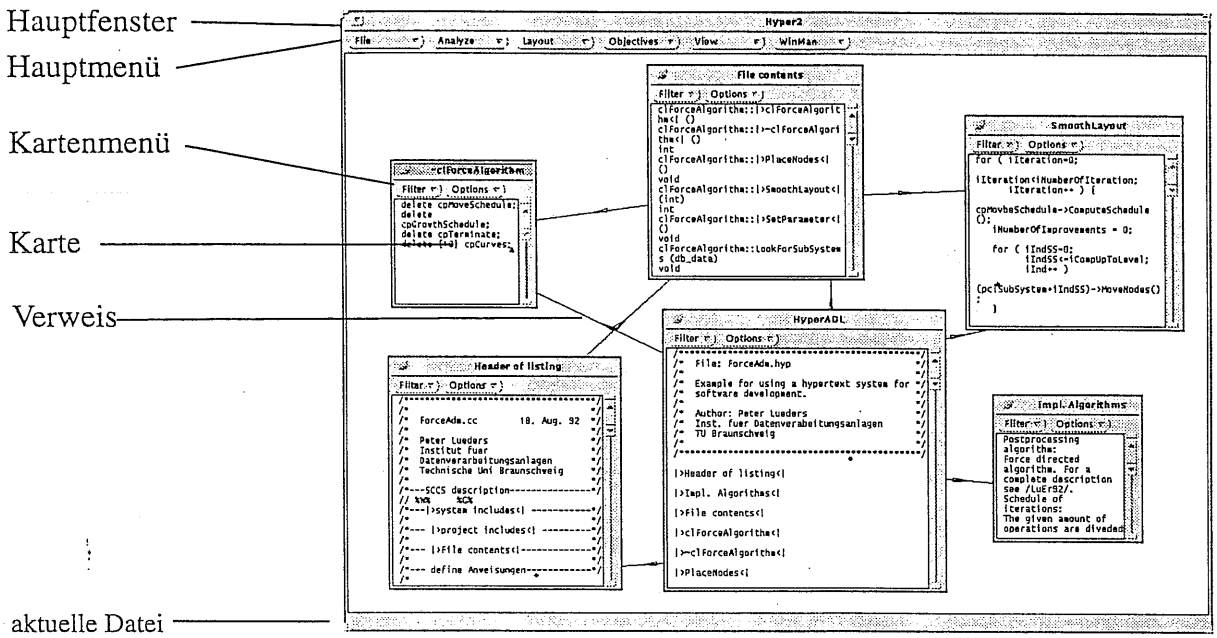

Bild 1: Bildschirmabzug beim Lesen eines C++ Programmtextes mit dem Hypertextsystem *HyperADL*.

gung für eine kontinuierliche Arbeit dar, da vom Benutzer nicht erwartete Systemreaktionen - hier Layoutberechnungen - zu Unterbrechungen im Arbeitsablauf führen.

Doch trotz dieser Einschränkungen scheint es sinnvoll zu sein, dieses System als Grundlage auch für den Aufbau der Benutzerschnittstelle eines Facility Management Systems zu verwenden, da hier eine ähnliche Aufgabenstellung vorliegt.

Die nun folgenden Ausführungen konzentrieren sich auf die Diskussion einiger offenstehender Problemstellungen, die im praktischen Einsatz des Experimentiersystems *HyperADL* auftreten:

- die Änderung der grafischen Darstellung des Fensterinhalts bei Änderung der Größe oder Form eines Fensters,
- Definition von Umgebungen zur einfachen Einstellung des Systemverhaltens und
- eine grafische Animation, die es dem Benutzer ermöglicht, die Layoutberechnung nachzuvollziehen.

2. Grafische Repräsentationen der Informationselemente

2.1 Größe und Form der Karten und der Anker

Das Aussehen der Karten folgt dem von Sun Microsystems propagierten Standard OpenLook. Das Aussehen einer Karte ist im Bild 2 aufgezeigt. Es ist ein Cmd-Frame, der durch einen Push-Pin (oben links) gekennzeichnet ist. Mit diesem Push-Pin ist es möglich, das Fenster bzw. die Karte vom Bildschirm zu löschen. Unterhalb der Titelzeile ist eine Menüzeile, in der Einträge vorhanden sind, mit denen der Benutzer Attribute der Karte setzen / ändern kann

oder auch Befehle wie beispielsweise Informationsfilter zur Selektion von momentan interessierenden Informationseinheiten aufrufen kann.

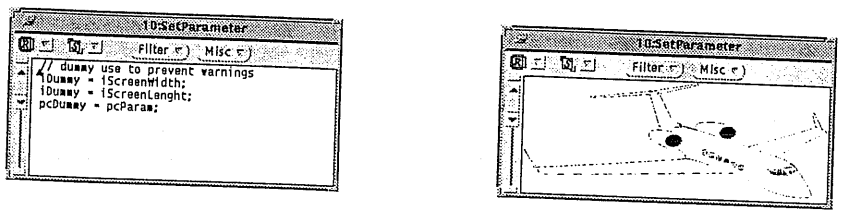

Bild 2: Repräsentation einer Karte.

Die Größe einer Karte - wobei davon ausgegangen wird, daß der Benutzer noch keine explizite Wunschvorgabe gesetzt hat - leitet sich aus dem Inhalt der Karte ab. Dazu wird entweder aus der Grafik oder es wird bei einer Textkarte aus dem darzustellenden Text und dem Textfont deren Höhe und Breite ausgelesen und als gewünschte Größe der Karte gesetzt. Vor dem Setzen werden die bestimmten Werte aber noch mit Maximal- bzw. Minimalwerten verglichen, die der Benutzer entweder für alle Karten oder nur für bestimmte Kartentypen vorgeben kann.

Ein wesentliches Problem beim Aufbau eines Hypertextes ist die grafische Darstellung der Quelle bzw. des Ziels eines Verweises, da ein Hauptteil der Navigation durch einen Hypertext durch das Folgen der Verweise geschieht. Und dieses wiederum erfolgt zumeist durch Mausaktionen wie beispielsweise ein Doppelklick über der *aktiven Fläche* des Ankers. Somit muß im weiteren zwischen Quell- und Zielankern unterschieden werden. Während bei Quellankern die aktive Fläche wichtig für die Bedienung ist, ist die grafische Darstellung eines Zielankers nur für die Informationsdarstellung relevant, der Ort selbst ist passiv und spielt sonst keine weitere Rolle (natürlich kann ein Zielanker aber auch gleichzeitig Quellanker eines anderen Verweises sein und damit wieder eine aktive Fläche beinhalten).

Eine wichtige Anforderung an die grafische Darstellung der Anker ist die Verdeutlichung des jeweils vorliegenden Verweistyps, beispielsweise ein 1:1-Verweis, ein 1:n-Verweis oder die Unterscheidung, zu welchem Typ von Information der Verweis führt. In einem engen Zusammenhang mit der unterschiedlichen Darstellung steht die Typisierung der Verweise. Diese ermöglicht eine eindeutige Zuordnung von grafischen Repräsentationen zu bestimmten Typen von Verweisen. Daraus folgt, daß jedem vorhandenen Verweistyp eine bestimmte Repräsentation für die Darstellung des Quell- und des Zielankers als Attribut zugeordnet ist, die zudem durch Änderung einer Konfigurationsdatei verändert werden kann. Das Bild 3 soll einige Möglichkeiten aufzeigen, wie Anker dargestellt werden könnten. In der Darstellung ist noch zu beachten, daß Verweisstellen auch am Rande der Karte markiert sind. Dies stellt den Anknüpfungspunkt für die visuelle Darstellung der Objektbeziehungen dar (auf diese Problematik des Routings wird im Rahmen dieses Beitrags aber nicht weiter eingegangen).

2.2 Verhalten der Karten bei Größenänderung

Bei der Neuberechnung eines Layouts kommt es üblicherweise zu Größen- bzw. Formveränderungen der dargestellten Karten. Für die praktische Handhabung des Systems ist es nun notwendig, Richtlinien vorzugeben, wie der Inhalt - Text oder Grafik - bei veränderter Kartengeometrie der Karte dargestellt wird.

Allgemein läßt sich die folgende, allgemeine Regel formulieren:

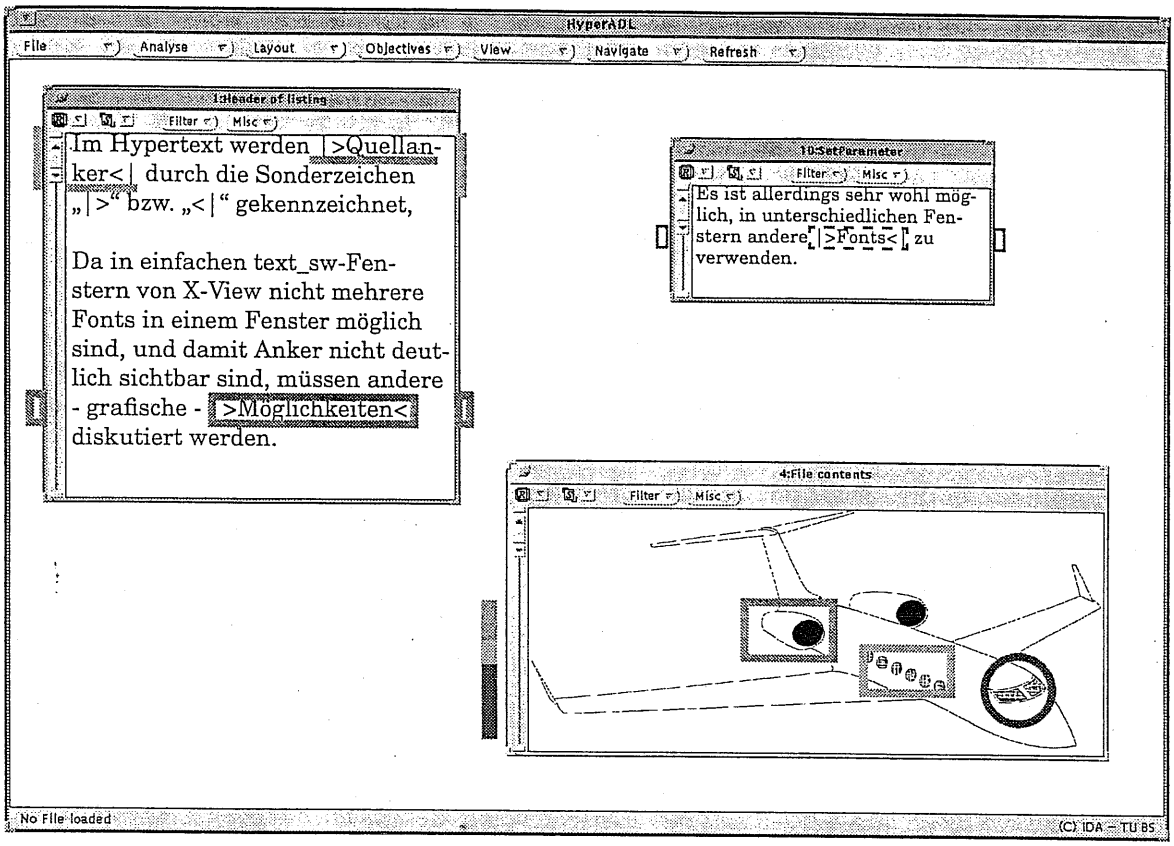

Bild 3: Beispiele für grafische Darstellungen von Quellankern.

Regel 1: Der bildliche Eindruck einer Karte soll sich bei einer Größen- / Formänderung so wenig wie möglich ändern.

Dieses Verhalten unterstützt dann die Orientierung des Benutzers im Layout, da sich durch die Layoutberechnung zwar die Position und die Geometrie einer Karte geändert hat, der Inhalt aber leicht wiederzuerkennen ist. Es ist aber offensichtlich, daß es zu dieser Regel zahlreiche Ausnahmen gibt, von denen einige im folgenden diskutiert und in das System integriert werden sollen.

Für alle Karten gleich sind jedoch die Grenzen zwischen einer Iconisierung des Fensters und der Re-Iconisierung bzw. die minimalen Abmessungen einer Karte. Dabei ist ein Icon ein grafisches Objekt fester Größe (60x60 Bildschirmpunkte), das stellvertretend für die Karte auf dem Bildschirm abgebildet wird. Daneben ist es auch möglich, über ein Attribut jeder Karte explizit ein individuelles Icon zuzuordnen, daß die Wiedererkennung wesentlich vereinfacht.

Regel 2: Eine Karte wird bei Unterschreitung der Breite von B_{icon} Bildpunkten als Icon dargestellt. Zudem muß die Höhe einer Karte stets größer als die Höhe H_{min} sein.

Dieser Regel liegt der Gedanke zugrunde, daß, eine Mindesthöhe vorausgesetzt, eine Karte ab einer gewissen Breite durch die Anordnung der Zugriffselemente (Titel, Menüs, Buttons) immer noch sowohl als möglicher Ausgangpunkt einer Benutzeraktion angesehen werden kann als auch als Informationsträger (Titel, Geometrie) wirkt. Erst bei Unterschreitung dieser

Breite kann auf die Menüelemente nicht mehr zugegriffen werden und damit entfällt die Funktion der Karte als Navigationselement.

2.2.1 Textkarten

Wie bereits erwähnt, enthalten Textkarten alphanumerischen Text, wobei der Font für alle Zeichen in dieser Textkarte gleich ist. Damit entfallen besondere Formatierungsmerkmale wie Fettdruck oder kursiv als Orientierungsmerkmale. Der bildliche Eindruck eines Textes basiert also allein auf den *Absatzgrößen* in der vertikalen und den *Zeilenumbrüchen* in der horizontalen Richtung. Zu unterscheiden sind dabei im folgenden zwei Typen von Textkarten, zum einen Karten mit einer festen Schriftgröße, im folgenden mit dem Attribut *statisch* belegt, und zum anderen Karten, die die Schriftgröße an die momentane Kartengröße anpassen, im folgenden mit dem Attribut *dynamisch* bezeichnet. Das Verhalten bei einer Größen- bzw. Formänderung der Karte folgt damit zu:

- Statisch: die Schriftgröße des Textes ist fest vorgegeben und ändert sich nicht. Das Verhalten des Umbruchmodus und der Verschiebung des Textes (siehe Bild 5) kann hingegen interaktiv durch ein Attribut gesetzt werden.

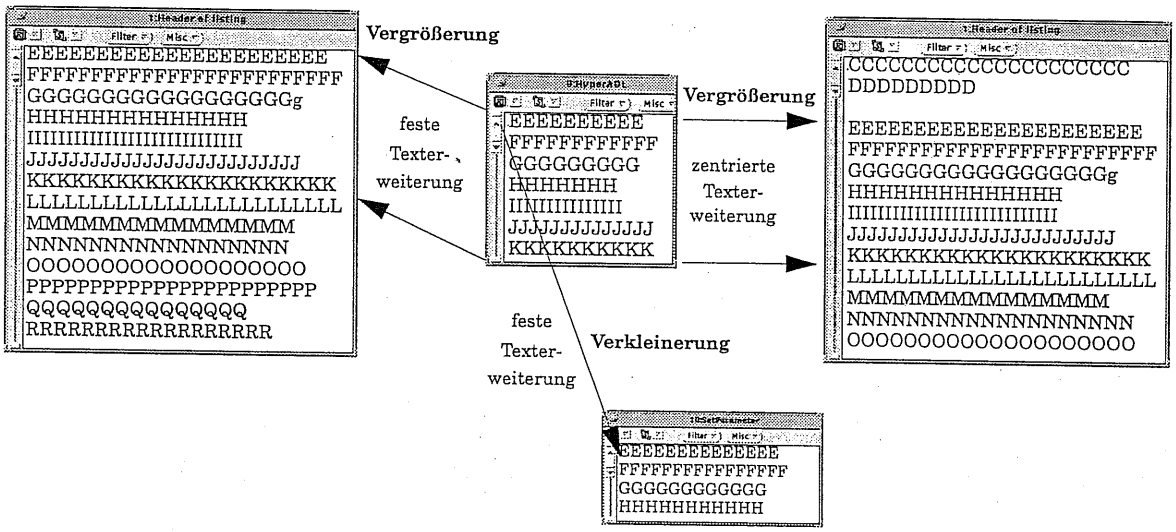

Bild 4: Textansichten bei Größenänderung einer Karte mit fester Schriftgröße.

- Adaptiv: Ist die Schriftgröße flexibel, so kann sie als Funktion der Höhe einer Textkarte betrachtet werden. Prinzipiell heißt das, daß die Schrift sich vergrößert, wenn die Höhe zunimmt, bzw. sich verkleinert, wenn die Höhe abnimmt. Zudem kann eine minimale bzw. maximale Schriftgröße vorgegeben werden, die nicht unter- bzw. überschritten wird. Auch kann eine bestimmte Höhe gesetzt werden, bei der ein vorbestimmter Font verwendet wird. Eine zweidimensionale Abhängigkeit (zusätzlich abhängig von der Breite) wird hier nicht betrachtet, da es nicht sinnvoll erscheint, die Schriftgröße einer breiter werdende Karte zu vergrößern.

In beiden Fällen gibt es für die Stellung des Textes zwei Möglichkeiten. Einerseits kann das erste dargestellte Zeichen bei der Formveränderung gleich gehalten werden, d.h. es wird am linken bzw. unteren Rand der Text abgeschnitten. Andererseits kann es aber auch bei einer Vergrößerung sinnvoll sein, gleichmäßig um den bisher dargestellten Text neue Textzeilen einzu-

setzen. Welche Möglichkeit jeweils angewendet werden soll, muß durch ein Attribut vorgegeben werden.

Voraussichtlich, wird der adaptive Typ mit einer sowohl nach oben als nach unten begrenzten Schriftgröße der am ehesten verwendete Kartentyp sein. Denn zumeist wird man eine angenehme Schriftgröße gewählt haben und es vorziehen, bei einer Vergrößerung mehr Text angezeigt zu bekommen als den vorliegenden Text in einem sehr großen Font (insbesondere bei einer Veränderung von einer sehr kleinen auf eine sehr große Größe).

Gleichwohl die Veränderung der Textgröße in einem direkten Verhältnis zur Kartenhöhe steht (gleiche Menge von Text sichtbar), ist die Verknüpfung der Standardgröße mit einer bestimmten Kartengröße notwendig. So soll ein kurzer Text (Beispiel: Warnung) bei einer kleinen Fenstergröße b_1, h_1 mit der Standardgröße 12 pt dargestellt werden, ein sehr großer Text (Beispiel: Veröffentlichung) soll hingegen erst bei einer sehr viel größeren Fenstergröße h_2, b_2 mit der Größe 12 pt dargestellt werden.

Insgesamt läßt sich die Änderung der Schriftgröße damit als Funktion der Höhe einer Karte angeben. Geht man von einem linearen Verhältnis zwischen Schriftgröße und Kartenhöhe aus, so folgt eine Geradengleichung für die Größe von *font* als Funktion von der Höhe *h* mit

$$font = a \cdot h + c, \qquad \text{(GL 1)}$$

wobei die Steigung *a* durch den Zusammenhang Font - Kartengröße gegeben ist und der Summand *c* die Lage der Geraden in Abhängigkeit des vom Benutzer vorgegebenen Default-Wertepaares (h_1,F_1) beschreibt. Dabei werden jedoch Werte für *font*, die kleiner als der minimale bzw. größer als der maximale Font sind, auf den gegebenen Minimal- bzw. Maximalwert gesetzt. Dieser Zusammenhang ist nochmals im Bild 6 grafisch dargestellt.

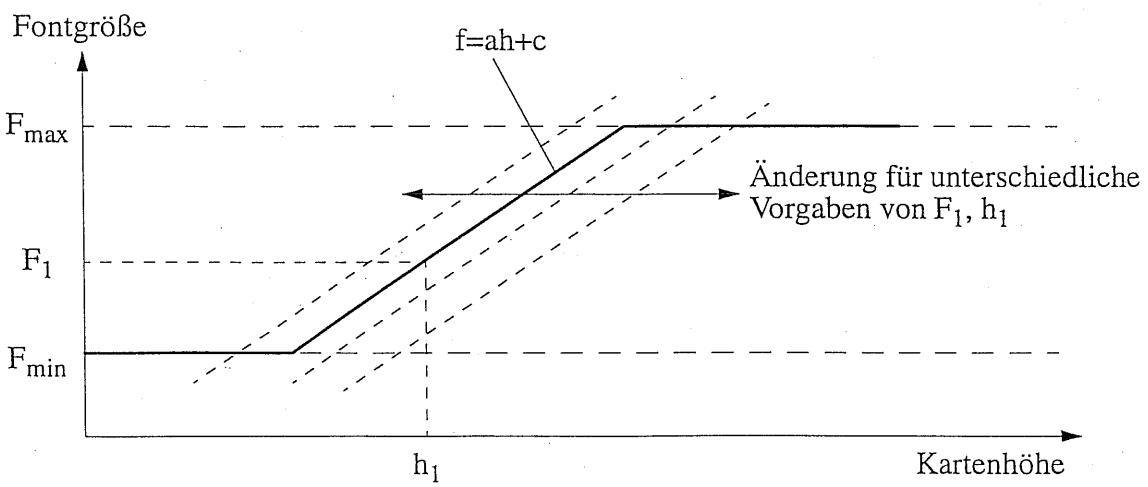

Bild 5: Abhängigkeit der Textgröße von der Höhe einer Textkarte bei Berücksichtigung vorgebbarer Parameter für maximale bzw. minimale Schriftgröße.

2.2.2 Grafikkarten

Die Funktionalität der Grafikkarten soll nur rudimentär gehalten werden und im wesentlichen die vielfältigen Möglichkeiten aufzeigen. So ist der Grundtypus der bislang vorgesehenen

Grafikkarten rein pixelorientiert. D.h. aus einer Datei wird die Bildbeschreibung eingelesen und entsprechende Bildpunkte im Kartenbereich gesetzt. Ohne unangemessen hohen Aufwand sind Skalierungen und Fensteroperationen möglich. Entsprechend den sich daraus ergebenden zwei Verhaltensweisen werden zwei Typen von Karten unterschieden:

- Statisch: das Bild wird 1:1 in den Hintergrund kopiert (oder mit einem festen Skalierungsfaktor) und die Karte fungiert nur als Sichtfenster auf dem Bild. Damit ändert sich das Bild nicht, es werden aber die Bereiche außerhalb der Karte geclippt. Als Fixpunkt dient wieder der Bildpunkt oben links. Zudem werden noch Scrollbars eingeblendet, um ein Verschieben des gerade sichtbaren Bereiches zu erlauben. Als Besonderheit muß noch hervorgehoben werden, daß sich hier das Fenster zwar nahezu beliebig verkleinern kann, eine Vergrößerung über die 1:1 Bildgröße jedoch nicht sinnvoll ist und unterbunden wird. Ebenfalls ist bei diesem Typ auch eine - im Rahmen der Bildgröße - beliebige Formänderung möglich da Verzerrungen des Bildes nicht auftreten können.
- Dynamisch: entsprechend der Geometrieänderung des Fensters wird auch das dargestellte Bild skaliert. Analog zum Textfenster wird dabei die Skalierung entsprechend der neuen Breite (Höhe) durchgeführt. Die passive Seite erhält daher noch einen Scrollbar, mit dem der Bildausschnitt verschoben werden kann.

3. Verstehen des Verhaltens des Layoutsystems

Soll der Benutzer mit der Benutzerschnittstelle kontinuierlich arbeiten, muß die Reaktion des Systems auf Eingaben seiten des Benutzers erwartungskonform sein. Entspricht beispielsweise das Layout nicht den Erwartungen des Benutzers, so wird er es wiederum manuell korrigieren und damit ist kein kontinuierlicher Arbeitsablauf möglich; das automatisierte Bildschirmlayout stellt keinen wesentlichen Fortschritt dar.

Es folgt, daß ein Benutzer das Verhalten des Systems verstehen muß um dann durch Änderung von einzelnen Parametern das Verhalten so einzustellen, daß es seinen Erwartungen genügt. Zum Verstehen des Verhaltens ist zum einen eine Visualisierung der Attribute einer Karte, wie beispielsweise die gewünschte Größe, die gewünschte Form etc. notwendig, und zum anderen ist eine Animation des Optimierungsverlaufes notwendig, durch die der Benutzer die Optimierung, hier insbesondere das Zusammenspiel der einzelnen Komponenten der Kostenfunktion versteht.

3.1 Darstellung der inneren Attribute der Karten

Dieser Punkt adressiert das Verständnis des Benutzers für das Verhalten des Systems (siehe Abschnitt 3.1). Gegeben sei das folgende Szenario: in einem Layout hat eine Karte - die einen hohen Wert für die gewünschte Größe hat - eine verhältnismäßig kleine Größe, da sie durch andere Karten in ihrer Vergrößerung behindert wird. Der Benutzer arbeitet nun eine Zeitlang nicht mit Fenstern, sondern studiert intensiv den Karteninhalt. Dadurch haben sich seine Interessen verschoben und der Benutzer ist eigentlich nicht mehr an dem Inhalt der Karte interessiert. Bei Aufruf eines neuen Layouts, bei dem die vormals große Karte vom Bildschirm gelöscht wird, vergrößert sich aber nicht die momentan interessierende Karte, sondern die, die im letzten Layout ihre Größe nicht erreichen konnte. Dadurch ist der Benutzer überrascht, da er die Zustände der Karten nicht mehr vor Augen gehabt hat.

Aus diesem Grund ist es notwendig, die Zustände einer Karte durch grafische Darstellung sichtbar zu machen und damit dem Benutzer auf das zu erwartende Verhalten der Layoutberechnung vorzubereiten. Eine Möglichkeit ist die Verwendung von Gauges (ähnlich Slidern)

die den momenten Wert der gewünschten Größe und Ähnliches bei Bedarf anzeigen. Ein möglicher Ort wäre der untere Rand einer Karte (Bild 7). Eine ähnliche Möglichkeit besteht in der Verwendung farbiger Elemente, wobei die Intensität der Farbe den Wert widerspiegelt.

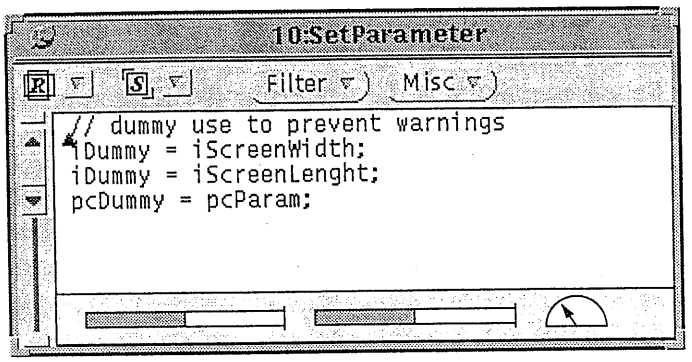

Bild 6: Möglichkeit zur grafischen Darstellung der inneren Zustände einer Karte.

3.2 Darstellung der Optimierung

Um ein Layoutergebnis nachvollziehbar zu machen, muß die Kostenfunktion in die einzelnen Komponenten aufgesplittet werden und diese sowie die Gesamtbewertung grafisch dargestellt werden. Eine Möglichkeit besteht darin, in einem gesonderten Fenster (vielleicht Tcl/Tk - Programmierung) die Kostenfunktion und die Summe über der Zeit darzustellen (Bild 8).

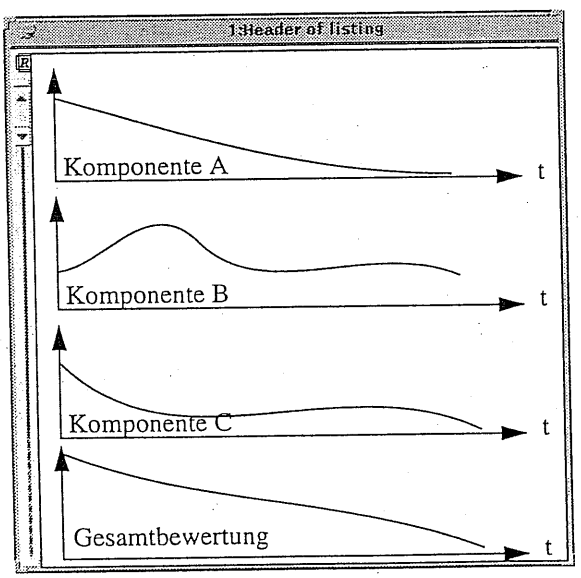

Bild 7: Verstehen des Optimierungsverlaufes durch Darstellung der Kostenverläufe der einzelnen Komponenten.

Damit kann der Benutzer die Optimierung in Hinblick auf die unterschiedlichen Komponenten der Kostenfunktion und deren Gewichtung in Relation setzen und das Ergebnis verstehen.

Als nächstes ist es nun notwendig, daß der Benutzer ein Gefühl dafür erlangen kann, wie eine Komponente ein Layout bewertet.

Die Darstellung des Einflusses eines einzelnen Objektes auf die Bewertung des Layouts durch die Kostenfunktion erfolgt am einfachsten dadurch, daß der Benutzer ein generiertes Layout modifiziert und die Änderungen der Kostenfunktion angezeigt bekommt. Dadurch ist es ihm möglich, das Layout bestimmter Szenarien zu verstehen und somit auch das Verhalten der Optimierung nachzuvollziehen. Hierzu kann ein dem vorherigen Abschnitt ähnliches Verfahren Verwendung finden, nur, daß nun die Änderung der Kosten nicht durch den Layoutalgorithmus, sondern durch Modifikationen seitens des Benutzers herrühren.

Die Darstellung der vom Benutzer oder von der Applikation vorgegebenen Randbedingungen sind ebenfalls wesentlich für das Verständnis eines Layoutverhaltens. Aus diesem Grund müssen sie ebenfalls grafisch visualisiert werden. Erst durch die vollständige Anzeige aller die Layoutberechnung beeinflussenden Elemente kann der Benutzer das Verhalten voll verstehen.

Mögliche Ansätze für die Darstellung der Randbedingungen sind Pfeile für Relationsbedingungen (z.B. A links von B, A am Rand, ...) und Flächen für gegebene Plazierungsbereiche. Die Darstellung sollte dann für einzelne Knoten ein- und ausschaltbar sein, um daß Gesamtbild übersichtlich gestalten zu können.

Dabei stellt sich natürlich die Frage, ob bei manuellen Verschiebungen die Randbedingungen berücksichtigt werden oder nicht. Darf der Benutzer ein Fenster entgegen gesetzten Randbedingungen verschieben; einerseits muß der Benutzer letztendlich die Kontrolle behalten, andererseits entstehen dadurch ungültige Layouts. Ein Kompromiß ist, daß die Aufhebung der Randbedingungen bestätigt werden muß.

3.3 Handhabung der Referenzlayouts

Ein Referenzlayout dient als Vergleichslayout für die aktuelle Layoutberechnung. Bei passend eingestellten Gewichtungsparametern der Kostenfunktion wird so daß neue Layout ähnlich dem Referenzlayout. Dies ist ein maßgeblicher Punkt für die Gewährleistung der topologischen Konsistenz in einer Layoutsequenz /Lüde95b/.

Im praktischen Einsatz sind eine Anzahl von Referenzlayouts notwendig, um dem Benutzer die Möglichkeit zu geben, für verschiedene Umgebungen unterschiedliche Referenzlayouts vorgeben zu können. Somit ist auch eine Schnittstelle zur Verwaltung der Referenzlayouts notwendig. Dabei kann jedes Layout als Referenzlayout unter einem Namen in einer Liste abgelegt werden und anschließend kann jedes Layout der Liste als Referenzlayout gesetzt werden. Zur Verwaltung der Liste ist weiterhin eine grafische Darstellung jedes Referenzlayouts sowohl auf dem Gesamtbildschirm als auch in einem externen Fenster ähnlich den Übersichten der Virtual Window Manager (z.B. fvmw, olvwm) notwendig. Mit diesen Möglichkeiten hat dann der Benutzer die Möglichkeit, für seine aktuelle Problemstellung das passende Referenzlayout vorgeben zu können.

Weiterhin kann der Benutzer sich in einer grafischen Darstellung verdeutlichen, wie Referenzlayout und aktuelles Layout zusammenpassen. Für diesen Zweck werden zwei Funktionen zur Verfügung gestellt. Einerseits werden in einer grafischen Darstellung des Referenzlayouts die aktuell auf dem Bildschirm dargestellten Karten eingetragen; damit kann sich der Benutzer die Kohärenz zwischen diesen Bildschirmlayouts aufzeigen lassen. Andererseits kann der Benutzer durch einen Vergleich der beiden Layouts sich veranschaulichen, wie ähnlich die Layouts sind.

Eine weitergehende Möglichkeit wäre noch eine formale Beschreibung, wann welches Referenzlayout eingesetzt werden soll. Beispielsweise könnte man setzen, daß beim Verfolgen eines Links, d.h. es erscheint ein zusätzliches Fenster, stets daß vorherige Layout als Referenzdatei gesetzt wird. Ebenso könnte man setzen, daß bei Rücknahmen von Informationsfiltern (d.h. die Liste der letzten Informationsfilter wird rückwärts durchlaufen) stets die davor verwendeten Referenzlayouts gesetzt werden. Diese Möglichkeit böte natürlich eine wesentlich höhere Flexibilität für zukünftige Erweiterungen.

4. Umgebungen

Die Handhabung des Systems wird durch die große Zahl von Parametern (im momentanen Stand 50, mit zunehmender Komplexität wachsend), deren Bedeutung und Wirkungsweise dem durchschnittlichen Benutzer nicht bekannt sein dürfte, erschwert. Dieses Problem soll durch die Einführung von Umgebungen adressiert werden. Dabei definiert eine Umgebung einen vollständigen Satz der Parameter, die das Verhalten des Systems vorgeben. Hat der Benutzer wahlfreien Zugriff auf mehrere Umgebungen, mit denen er die aktuellen Parametereinstellungen einfach überschreiben kann, so kann er beispielsweise eine Umgebung für das erste Suchen einer Informationsmenge und eine andere Umgebung zur Darstellung von Flußgraphen verwenden. Formal gesehen wird damit eine Abstraktion eingeführt, es wird nämlich eine Menge von Parametern durch ein bestimmtes Verhalten charakterisiert.

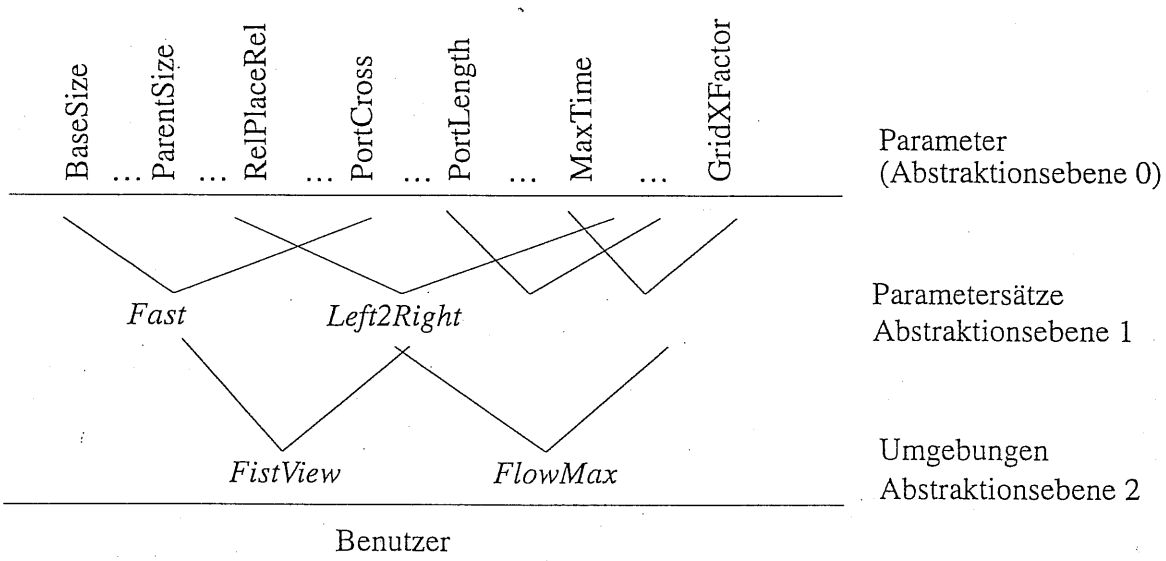

Bild 8: Veranschaulichung der Abstraktionsebenen.

4.1 Umgebung als Gesamtheit aller Parameter

Die einfachste Möglichkeit, eine Umgebung zu definieren, besteht wie oben bereits angedeutet nun darin, die Gesamtheit aller Parameter unter einem Namen abzuspeichern. Der Benutzer könnte sich so eine *Bibliothek von Umgebungen* mit genau definierten Eigenschaften aufbauen, aus der er bei bestimmten Gegebenheiten eine für die momentane Situation passende Umgebung auswählt und damit alle Parameter mit denen der gewählten Umgebung über-

schreibt. Dieser Ansatz würde einer einfachen Abstraktion entsprechen, d.h. einer Menge von Parameterwerten würde genau ein Verhalten (eine Umgebung) zugeordnet.

Als Nachteil für diesen Ansatz ist zu nennen, daß der Benutzer bei der Generierung einer Umgebung jedwede Auswirkung von Parametervorgaben kennen muß. Zudem müßte für jede noch so geringe Abweichung von Parametern eine eigene Umgebung aufgebaut werden, was zu einer hohen Zahl von Umgebungen führen würde, über deren Funktion und Verhalten der Benutzer während der Arbeit den Überblick behalten muß. Bei einem komplexen System wie der Layoutberechnung können beide Punkte jedoch für einen durchschnittlichen Benutzer nicht vorausgesetzt werden.

4.2 Zusammensetzung einer Umgebung aus Elementarumgebungen

Der eben beschriebene Ansatz kann nun weiter verfeinert werden, indem in das Abstraktionsmodell weitere Abstraktionsebenen eingeführt werden. Für bestimmte Teilmengen von Parametern können Umgebungen definiert und diese dann zu einer vollständigen Umgebung verknüpft werden. So könnte beispielsweise eine Umgebung *schnell* für die Layoutalgorithmen einerseits mit der Umgebung *Flußgraphen* der Kostenfunktion und in einem anderen Zusammenhang mit der Umgebung *verteilt* der Kostenfunktion verknüpft werden. Wenn nun bereits standardmäßig eine Anzahl von vordefinierten Umgebungen vorgegeben wird, kann der Benutzer durch die Abstraktion leicht auf diese zurückgreifen und zu Umgebungen miteinander verknüpfen und ist somit von einem konkreten Wissen der einzelnen Parameter befreit. So wäre es dem Benutzer leicht möglich, eine Hierarchie von Umgebungen zu verwalten.

Damit sind aber Definitionen von Operationen notwendig, die mit Umgebungen durchgeführt werden können. In einem ersten Ansatz sind dies allerdings nur eine Oder-Verknüpfung:

- Oder-Verknüpfung $(U_1 \vee U_2)$: hierbei werden die Parameter gesetzt, die in mindestens einer der Umgebungen enthalten sind. Ist ein Parameter nur in einer Umgebung enthalten, so wird er mit diesem Wert gesetzt. Ist ein Parameter in beiden Umgebungen enthalten, so wird er entweder mit einem Mittelwert besetzt (→ Schwierigkeiten bei textuellen Parametern oder True/False) oder enthält den Parameterwert der erstgenannten Umgebung (→ Reihenfolgeproblem).

Eine weitere Fragestellung ist die der bedingten Umgebung, beispielsweise soll ein Constraint zugeordnet werden, wenn der Knoten den Namen *Name1* trägt. Oder es soll eine bestimmte Umgebung dann verwendet werden, wenn eine bestimmte Voraussetzung erfüllt ist wie die erste Berechnung eines Layouts. Die dabei entstehende Fragestellung ist, ob bzw. inwieweit sich dieser Ansatz von Ansätzen der künstlichen Intelligenz unterscheidet, in denen aus einer Regelbasis heraus Regeln für das aktuell darzustellende Bildschirmlayout abgeleitet werden.

Mit der Einführung einer globalen und einer lokalen Umgebung, wobei die globale Umgebung Parameterwerte beispielsweise für die Layoutalgorithmen enthält, die lokale Umgebung dann für die momentane Datei wichtige Parameter (eventuell mit einer Bedingung verknüpft) ist auch eine Und-Verknüpfung möglich:

- Und-Verknüpfung $(U_1 \& U_2)$: hierbei werden die Parameter von U_1 durch die Werte von U_2 überschrieben (→ Reihenfolgeproblem).

4.3 Beispiele und Problemstellungen bei dem Einsatz von Umgebungen

Zur besseren Veranschaulichung sollen im weiteren einige Beispiele für Umgebungen und das Zusammensetzen von Umgebungen aufgezeigt werden. Einfach zu kleineren Umgebungen zusammenfaßbare Umgebungen sind:

- Parameter der Layoutalgorithmen (*first, fast, slow, best, ...*)
- Parameter der Layoutcharakteristika (*flow-chart, automate, screen_space, ...*)
- Parameter zur Einstellung des Kartenverhaltens (...)
- Welches Referenzlayout verwendet wird (*last, explizit, global, ...*)
- ...

Natürlich sind noch weitere logisch zusammengehörige Teilmengen identifizierbar, hier soll nur ein erster Eindruck vermittelt werden. Zudem sind noch Teilmengen denkbar, die nur einige Parameter unterschiedlicher Teilmengen enthalten. Wie (und ob überhaupt) diese zu handhaben sind, sei aber noch als weiteres Problem dahingestellt.

Für den Benutzer ist es während der Arbeit für das Verständnis des Systemverhaltens aber sehr wichtig, zu wissen, mit welcher Umgebung er gerade arbeitet (und welche Auswirkung sie auf das Verhalten hat). Wiederum die einfachste Lösung ist das alleinige Anzeigen des Namens der aktuell gewählten Umgebung. Im nächsten Schritt wäre die zusätzliche Darstellung der Verknüpfung der Umgebungen denkbar; als ausführlichste Darstellung eventuell eine weitergehende Beschreibung der einzelnen Umgebungen und deren Einfluß auf die endgültige Umgebung. Vielleicht ist es aber auch möglich, jede Umgebung durch ein grafisches Icon darzustellen und somit für den Benutzer bei passender Darstellung die Assoziation zwischen Umgebung und Verhalten einfacher zu gestalten.

5. Zusammenfassung

In diesem Beitrag wurde die Leistungsfähigkeit der Benutzerschnittstelle des Hypertextsystems *HyperADL* analysiert und Möglichkeiten zu Verbesserungen vorgeschlagen und diskutiert. Dazu wurde ein Ansatz vorgestellt, in dem jede Karte des Hypertextes Wissen über deren bestmögliche *Präsentation* bei unterschiedlicher Größe der Darstellungsfläche besitzt. Weiterhin wurden einige Möglichkeiten zur grafischen Animation des Layoutverhaltens aufgezeigt. In den Diskussionen der einzelnen Punkte wurden auch die zu erwartenden Problemstellungen skizziert. Obwohl die beschriebenen Ansätze offensichtliche Verbesserungen enthalten, steht sowohl eine qualitative wie auch eine quantitative Evaluation der Benutzerschnittstelle noch aus. Erst hiermit werden fundierte Aussagen über die Verbesserung der Leistungsfähigkeit der grafischen Benutzerschnittstelle von Informationssystemen durch das automatisierte Bildschirmlayout möglich.

6. Literatur

/Char94/ H.J. Charwar: Lexikon der Mensch-Maschine-Kommunikation, Oldenbourg Verlag, 2. Auflage, 1994.

/Bat*93/ DiBattista,G., Eades,P., Tamassia,R. und Tollis,I.G.: Algorithms for Drawing Graphs: an Annotated Bibliography, anonymous ftp: wilma.cs.brown.edu, 1993.

/Lüde95a/ P. Lüders, S. Stille, R. Ernst: An Algorithmic Approach to Automatic Display Layout. Erscheint in: „Software - Practice and Experience", John Wiley & Sons, 1995

/Lüde95b/ P. Lüders: Ein Beitrag zur Verbesserung der grafischen Benutzerschnittstelle durch das automatisierte Bildschirmlayout, erscheint im VDI-Verlag, Reihe Fortschritt-Berichte, 1995.

/Ple*04/ M. Plessow, J. Sieck, B. Stube und F. Thiede: Ein adaptierbares Layoutsystem für die Darstellung netzartiger grafischer Schemata, Konferenz für Computer aided Design, (1994).

/Wood90/ N. Woodhead: Hypertext & Hypermedia, Theory and Applications, Addison Wesly Publishing Company, 1990.