Erfahrungen beim Entwurf der Benutzerschnittstelle des mbp-tool-system unter Verwendung von Simulation und Prototyping

Klaus-Dieter Kreplin, Arno Schmidt, Klaus Werner Wirtz
mbp Mathematischer Beratungs- und Programmierungsdienst GmbH, Dortmund

Zusammenfassung: Probleme bei und Erfahrungen aus der frühen Konzeption der Benutzerschnittstelle des mbp-tool-system, einer laufenden Entwicklung des mbp, werden am Beispiel des JSP-Arbeitsplatzes (Unterstützungssystem für die Jackson-Strukturierte-Programmierung) diskutiert. Insbesondere wird auf die ergonomische Problematik eines grafikorientierten Werkzeugs, das auf alphanumerischer Hardware implementiert wird, eingegangen.

1. Jackson-Strukturierte-Programmierung (JSP)

Das mbp-tool-system, das derzeit beim mbp entwickelt wird, ist ein integriertes Software-Entwicklungs- und -Wartungssystem. Ein Schwerpunkt des mbp-tool-systems liegt in der Unterstützung sowohl aufeinander aufbauender, als auch zueinander alternativer Methoden des Software Engineering.

Eine dieser Methoden ist die Jackson-Strukturierte-Programmierung (JSP). Um die ergonomischen Probleme bei der Konzeption eines Werkzeugs, das die JSP unterstützt, deutlich zu machen, werden zunächst die Arbeitsschritte bei der Methodenanwendung sowie die Eingangs- und Ausgangsdokumente jedes Schritts kurz umrissen.

Im **Datenschritt** wird für jede Datei, die das zu entwerfende Programm verarbeitet oder produziert, ein Datenstrukturdiagramm entwickelt. Datenstrukturdiagramme sind Bäume, die von oben nach unten und von links nach rechts gelesen werden; sie beinhalten die problembezogene Sicht auf die Daten. Ein Beispiel findet sich in Abb. 1.

Im **Programmschritt** wird, unter Zuhilfenahme von Entsprechungen (inhaltlichen Abhängigkeiten) zwischen allen Datenstrukturdiagrammen, ein Programmstrukturdiagramm abgeleitet, das bereits die gesamte Kontrollstruktur des Programms beinhaltet. (Es ist formal wie ein Datenstrukturdiagramm aufgebaut.)

Im **Anweisungsschritt** werden die verarbeitenden Anweisungen in einer Liste zusammengestellt und dem Programmstrukturdiagramm zugeordnet. Die Anweisungen

ergeben sich aus der Aufgabenstellung. Sie können direkt in einer Programmiersprache oder weitgehend programmiersprachenunabhängig formuliert werden. Die Zuordnung der Anweisungen zum Programmstrukturdiagramm erfolgt grafisch durch das Anhängen der Anweisungsnummern an die Komponenten der Programmstruktur.

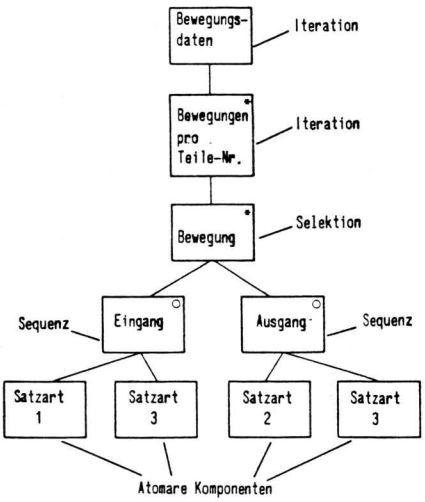

Abb. 1 Beispiel für ein Datenstrukturdiagramm

Im **Bedingungsschritt** werden konkrete Bedingungen für die im Programmstrukturdiagramm enthaltenen Kontrollkonstrukte "Wiederholung" und "Auswahl" formuliert und in einer Liste festgehalten.

Im **Textschritt** wird das Programmstrukturdiagramm in Strukturtext linearisiert. Der Strukturtext ist eine nach formalen Regeln aus dem Programmstrukturdiagramm mit zugeordneten Anweisungs- und Bedingungsnummern sowie den Listen der Anweisungen und Bedingungen abgeleitete textliche Form. Zur Übersetzung dieses Strukturtextes in Programmiersprachen gibt es bereits Werkzeuge, für COBOL beispielsweise den JSP-COBOL-Vorübersetzer.

In Abb. 2 sind die JSP-Methodenschritte, deren Ausgangspunkte und Ergebnisse sowie einige Rückkopplungen im Verlauf der Programmentwicklung im Zusammenhang dargestellt. Die Leistungen eines JSP-Werkzeugs sind ebenfalls angedeutet.

2. Benutzerorientierte Aspekte der JSP-Unterstützung durch Software-Werkzeuge

Die JSP-Methode ist gekennzeichnet durch ein schrittweises Vorgehen, verbunden mit einer starken Abhängigkeit zwischen den Arbeitsergebnissen der einzelnen Schritte. Zwei dieser Schritte, die Ableitung der Programmstruktur aus den Datenstrukturdiagrammen und den Entsprechungen und die Ableitung des Strukturtexts, sind streng formalisiert und daher automatisierbar.

Ein weiteres Merkmal ist die Verwendung grafischer Darstellungen: Datenstrukturen und Programmstrukturen werden als Baumdiagramme notiert; erst im letzten Schritt der Methodenanwendung, dem Textschritt, entsteht eine textliche Form des Programms.

Die Methodenanwendung erfolgt im konkreten Fall in der Regel iterativ, d.h. einige oder alle Schritte werden mehrmals durchlaufen, bis ein lauffähiges Programm entsteht, das den Vorgaben genügt. Bei diesen Rückkopplungen im Programmentwicklungsprozeß entsteht das Problem der Konsistenz zwischen überarbeiteten und unveränderten Teilergebnissen, das bei einem manuellen Vorgehen nicht so stark ins Gewicht fällt, bei einem teilautomatisierten Prozeß aber äußerst wichtig ist.

Die Entwicklung eines großen, nicht trivialen Programms nach JSP ist meist nicht in einem Zug zu erledigen. Kreative und denkintensive Phasen, wie das Erstellen der problemadäquaten Datenstrukturdiagramme oder das Finden und Zuordnen der Anweisungen, wechseln ab mit eher formalen und nach festen Regeln ablaufenden Tätigkeiten, wie etwa dem Ableiten der Programmstruktur.

Aus diesen Merkmalen der JSP-Methode ergibt sich bereits eine erste Menge von Anforderungen an die Software zur Unterstützung der JSP:

- Ein JSP-Werkzeug muß die klare und wohldefinierte Abfolge von Methodenschritten unterstützen.
- Die automatisierbaren Schritte sind zu automatisieren. Dies entlastet den Benutzer von Routinetätigkeiten und eliminiert Fehlerquellen.
- Das JSP-Werkzeug muß dem Benutzer das Arbeiten mit den gewohnten Baumdiagrammen ermöglichen, ihm also Formen der grafischen Interaktion bieten.
- Die Änderung und Erweiterung bereits durchlaufener Schritte muß problemlos

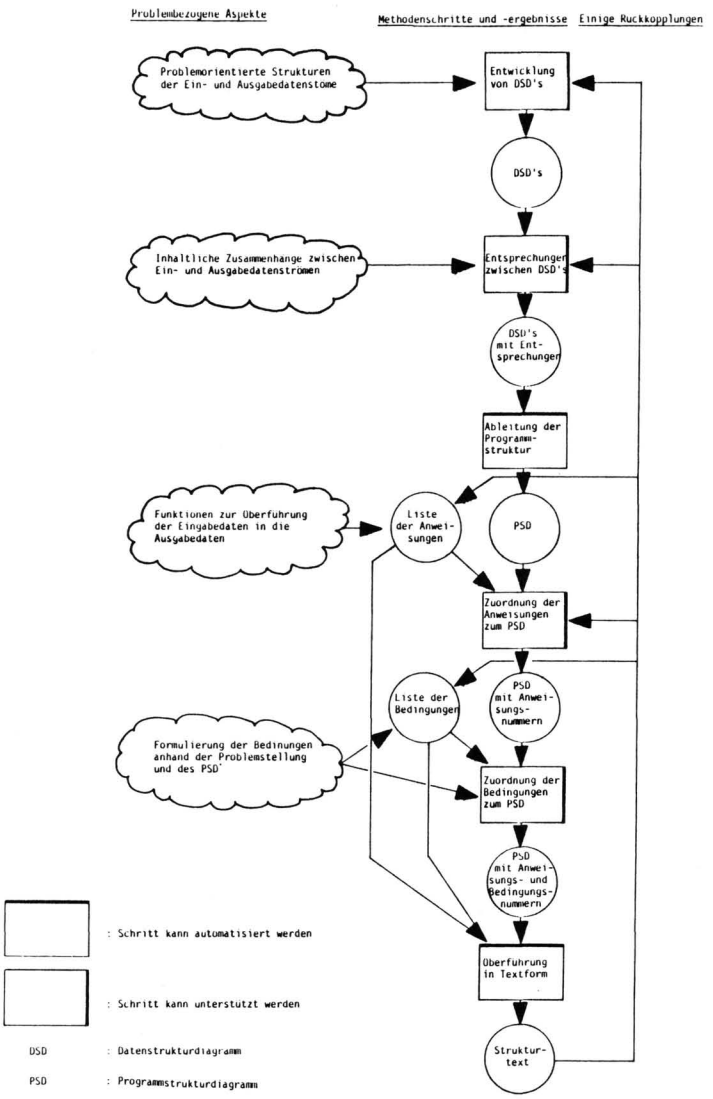

Abb. 2 Überblick über Jackson-Strukturierte Programmierung (JSP)

möglich sein. Hierbei ist sicherzustellen, daß Überarbeitungen - nach notwendigerweise inkonsistenten Zwischenzuständen - wieder zu einer geschlossenen, konsistenten Abhängigkeitskette zwischen den Teilergebnissen führen. Ziel der Werkzeugunterstützung sollte hier die Minimierung des vom Benutzer zu erbringenden Aufwands sein.

- Das Werkzeug soll dem notwendigen Wechsel zwischen Bildschirm- und Schreibtischarbeit Rechnung tragen, da kreative Arbeit am Bildschirm nicht immer möglich ist.

Aus den beschriebenen Leistungen eines JSP-Werkzeugs ergeben sich unmittelbar eine Reihe von Vorteilen:

- Automatisierung einiger formaler Teilschritte,
- Automatische formale Vollständigkeits- und Konsistenzprüfungen in anderen Schritten,
- Durch das Werkzeug garantierte formale Richtigkeit von Ergebnissen,
- Automatische Dokumentation der Programmentwicklung als Nebenprodukt.

Im Mittelpunkt der folgenden Ausführungen steht die Problematik, wie dem Benutzer die Leistungen des Werkzeugs zur Verfügung gestellt werden, also die Frage nach der Gestaltung der Benutzerschnittstelle.

Die Konzeption der Benutzerschnittstelle des bei uns in Entwicklung befindlichen JSP-Werkzeugs muß die folgenden extern gesetzten Vorgaben berücksichtigen:

- Es ist ein Dialog-Werkzeug zu entwickeln.
- An Hardware-Ausstattung können zunächst nur die gängigen, alphanumerischen Bildschirme mit 24 Zeilen und 80 Spalten vorausgesetzt werden. Für Papierausgaben stehen nur Schnelldrucker zur Verfügung.
- Das Werkzeug muß verschiedene Benutzergruppen jeweils angemessen unterstützen. Zu denken ist an
  - den erfahrenen Methoden- und Werkzeugkenner,
  - den Methodenanwender, der sich mit dem Werkzeug vertraut machen will,
  - und den gelegentlichen Benutzer ("casual user"), der das Werkzeug nur oberflächlich kennt und sich auch nicht weiter einarbeiten will, als es für die Lösung seines gerade anstehenden Problems notwendig ist.

Die Überlegungen zur adäquaten Schnittstelle für die erwähnten Benutzergruppen, die einheitlich für das mbp-tool-system angestellt wurden, führten zur Festlegung mehrerer sogenannter Dialogmodi:

- Eine auf Kürze und Prägnanz abgestellte kommando-orientierte Schnittstelle.
- Eine grundsätzlich ebenfalls auf Kommandos basierende Oberfläche, in der jedoch jederzeit zur Vervollständigung von Anweisungen und zur Information über verfügbare Kommandos in eine formularorientierte Eingabeart gewechselt werden kann.
- Ein - im Gegensatz zu den vorigen - stark vom Werkzeug geführter Dialogmodus, in dem das Werkzeug den Benutzer "an die Hand nimmt" und ihn durch die Methoden- und Werkzeugschritte führt.

Als folgenreichste und einschneidendste Entwurfsvorgabe für ein primär grafik-orientiertes Werkzeug ist zweifellos die Festlegung auf alphanumerische Bildschirme zu werten:

- Allgemein läßt sich weniger darstellen als auf grafischen Bildschirmen.
- Die oben erwähnten Dialogmodi, die eine teilweise sehr platzintensive Kommunikation mit dem Benutzer führen, schränken die ohnehin knappe Darstellungsfläche noch weiter ein.
- Maßstäbliche Verkleinerung ist auf alphanumerischen Bildschirmen nicht möglich. Es sind also Verfahren der "logischen" Verkleinerung zu konzipieren.
- Die reichlich vorhandene Software für Grafik-Geräte kann nicht genutzt werden.
- Es liegen nicht allzu viele Erfahrungen mit Grafik-Verfahren auf alphanumerischen Bildschirmen vor. Dieses Defizit vor allem führte zu der frühzeitigen Erprobung von verschiedenen Darstellungsformen in Simulationsmodellen und Prototypen.
- Generell gilt, daß der Benutzer ab einer bestimmten Größe seiner Baumdiagramme auf dem Bildschirm nicht mehr all das gleichzeitig im Blick behalten kann, was er bei manueller Vorgehensweise auf seinem Schreibtisch um sich herum ausbreiten kann. Zwangsläufig wird also die erwünschte gleichzeitige Betrachtung mehrerer Objekte umgewandelt in eine zeitlich sequentielle.

Weniger stark ins Gewicht fällt dagegen die Festlegung auf Schnelldrucker statt Plotter für Papierausgaben, da dort die Größe der erzeugten Zeichnung nicht so kritisch ist. Druckausgaben sind im Rahmen eines Dialogtools notwendige Hilfsmittel, die den Wechsel von Bildschirm- zu Schreibtischarbeit unterstützen.

Im folgenden Abschnitt wird daher weiter auf die Darstellung der JSP-Objekte auf alphanumerischen Bildschirmen eingegangen.

3. Ergonomische Überlegungen beim Entwurf des grafikorientierten JSP-Arbeitsplatzes

Durch die Vorgabe für den Entwurf des JSP-Editors, daß die volle Arbeitsfähigkeit bei Vorliegen von handelsüblichen alphanumerischen Bildschirmen und Zeilendruckern gegeben sein soll, ohne daß auf Grafik verzichtet wird, ergibt sich eine Reihe software-ergonomischer Probleme, die sich bei Verwendung von Grafik-Software und -Hardware nicht in dieser Art stellen. Ausgangspunkt ist dabei die grobe Rasterung der Darstellungsfläche des Bildschirms und die Beschränktheit des verwendeten Zeichensatzes, die sich beide nachteilig auf die Darstellung von Grafik auswirken. Durch die Notwendigkeit, bei der Darstellung von Verbindungslinien und Kastenumrandungen jeweils Elemente des Standardzeichensatzes verwenden zu müssen, ergibt sich eine relativ einförmige Darstellung. Zudem ist wegen des damit einhergehenden hohen Verbrauchs der zur Verfügung stehenden Darstellungsfläche eine räumliche Gliederung des Bildes durch unterschiedliche Abstände der einzelnen Bildelemente nur in sehr geringem Umfang möglich.

Um diesen sich ergonomisch negativ auswirkenden Eigenschaften entgegenzusteuern, werden zwei Techniken verwendet. Zum einen werden die durch die Hardware gegebenen Möglichkeiten, wie hellere Darstellung, anderer Schrifttyp, Cursorsteuerung etc. ausgenutzt, um visuelle Hervorhebungen zu erreichen. Der andere Aspekt betrifft mehr den Bildaufbau. In der Regel wird nur an einem Kasten in der Grafik tatsächlich gearbeitet, während die Umgebung nur Orientierungshilfe ist. Dieser "aktuelle" Kasten ist stets im Zentrum des Bildschirms lokalisiert und ändert beim Rollen des Fensters seine Position nicht, so daß das Auge einen räumlichen Fixpunkt auf dem Bildschirm hat.

Die Beschränkung des Darstellbaren durch die Bildschirmfläche führt bei größeren Grafiken zu der Forderung nach ausschnittweiser Darstellung und stufenweiser Verkleinerung. Bei letzterer führt die begrenzte Auflösungsfähigkeit und die Lesbarkeit zu Informationsverlusten (Texte werden gekürzt oder verschwinden ganz); trotzdem muß der Benutzer die Möglichkeit haben, lokal die volle Information visuell gleichzeitig (eventuell teilüberlagernd) anzufordern. Bei der ausschnittweisen Darstellung reichen Standardvorgehensweisen, wie z.B. das Rollen eines Fensters über einem festen Bild, aus methodischen und ergonomischen Gründen nicht aus, da dabei oft Ausschnitte gezeigt werden, die nicht sinnvoll sind. Softwaretechnisch aufwendigere Verfahren sind hier zu konzipieren, die es z.B. ermöglichen, für jeden Ausschnitt (Fenster) dynamisch eine sinnvolle Aufbereitung der Grafik zu erzeugen.

Der Auftragealgorithmus zur Erzeugung des jeweiligen Bildschirmteils wird in Kapitel 4 beschrieben. Die bildschirmorientierte Aufbereitung führt dazu, daß man z.B. nicht die Ausdrucke mehrerer Bildschirme aneinanderkleben kann, um das Gesamtdiagramm zu erhalten, da die Positionen gleicher Kästen nicht übereinstimmen. Entsprechend muß für den Listenausdruck des Gesamtbaumes eine andere Auftrageform gewählt werden. Es erfordert daher einen erhöhten Aufwand, den Listengesamtausdruck und das am Bildschirm Angezeigte miteinander zu korrelieren. Ein Beispiel für diese Vorgehensweise bringt Kapitel 4. Gerade dies zeigt, daß in der Regel eine Gewichtung von miteinander in Konflikt stehenden ergonomischen Prinzipien erfolgen muß.

4. Konzeption der Werkzeugoberfläche

Der Bildschirm ist aus der Sicht des Benutzers grundsätzlich in zwei Teile geteilt. Um signifikante Grafik-Ausschnitte zeigen zu können, belegt der Anzeige-Bereich die oberen 21 Zeilen. Die 22. Zeile trennt den Anzeige-Bereich von dem Kommunikationsbereich; die 23. Zeile nimmt die Eingabe auf und die 24. Zeile bildet die Meldungszeile. Die vom JSP-Arbeitsplatz generierten Baumdiagramme bestehen aus Kästen und Verbindungen. Ein Kasten besteht aus Fläche und Begrenzung. Dabei dient sowohl die Fläche als auch die Begrenzung der Wiedergabe von Information, die mit dem Kasten verbunden ist. Kästen können folgende Informationenen enthalten:

* im Kasten
  - max. 30 Zeichen langer Text (z.B. Kastenname)
* obere Kastenbegrenzung
  - Eintragung über eventuell vorhandene Bedingung
* untere Kastenbegrenzung
  - Kastennummer
  - Entsprechung
* rechte Kastenbegrenzung
  - Eintragung über Typ des Vaterkastens (Sequenz, Selektion, Iteration)

Um einen Gesamteindruck von der Gestaltung des Bildschirms zu erhalten, betrachte man Abb. 3.

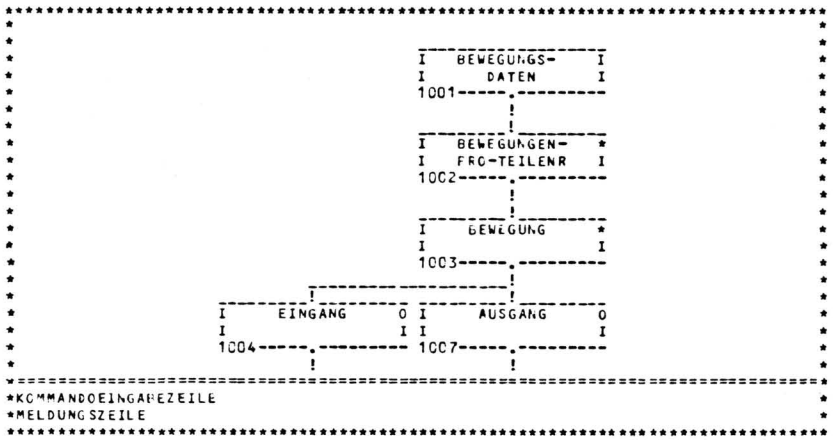

Abb. 3 Bildschirm-Nutzung bei ungeteiltem Anzeigebereich (max. 16 Kästen)

Die methodischen Entwicklungsschritte schlagen sich auch in der Gestaltung des Bildschirms nieder; z.B. wird für den Entwicklungsschritt "Entsprechungen eingeben" eine vertikale Teilung des Anzeigebereichs gewählt. Die Entwicklungsschritte "Bedingungen eingeben" und "Elementaranweisungen zuordnen" führen ebenfalls zu einer vertikalen Teilung des Anzeigebereichs, wobei links das Programmstrukturdiagramm gezeigt wird, während rechts Raum für die Bedingungen und Elementaranweisungen zur Verfügung steht (vgl. Abb. 4).

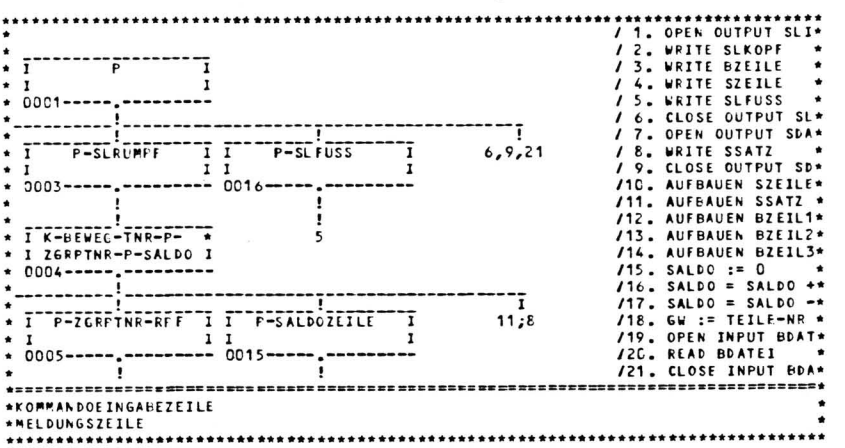

Abb. 4 Teilung des Anzeigebereichs bei Entwicklungsschritt "Elementaranweisungen der Programmstruktur zuordnen"

Da mit der Teilung des Anzeigebereichs der gezeigte Ausschnitt einer Grafik noch kleiner wird, haben wir zwei Verkleinerungsstufen für die Grafik vorgesehen, damit der Rollaufwand auf den jeweiligen Anzeigebereichen für den Benutzer nicht zu groß wird.

Bei der ersten Verkleinerungsstufe entfallen der Kastenrand und die zweite Zeile des Kastentexts; die Kastenbreite wird reduziert. Diese "logische" Verkleinerung bedingt einen Informationsverlust, indem die Kastentexte gekürzt werden. Trotzdem hat der Benutzer die Möglichkeit, lokal - für einen Kasten - die volle Information anzufordern. Diese wird dann in die Meldungszeile eingeblendet. In der zweiten Verkleinerungsstufe werden nur noch Kastennummern, Typsymbole und ein Hinweis auf vorhandene Entsprechungen dargestellt. Auch in dieser Stufe kann die nicht gezeigte Kasteninformation für jeweils einen Kasten angefordert werden. (Vgl. Abb. 5)

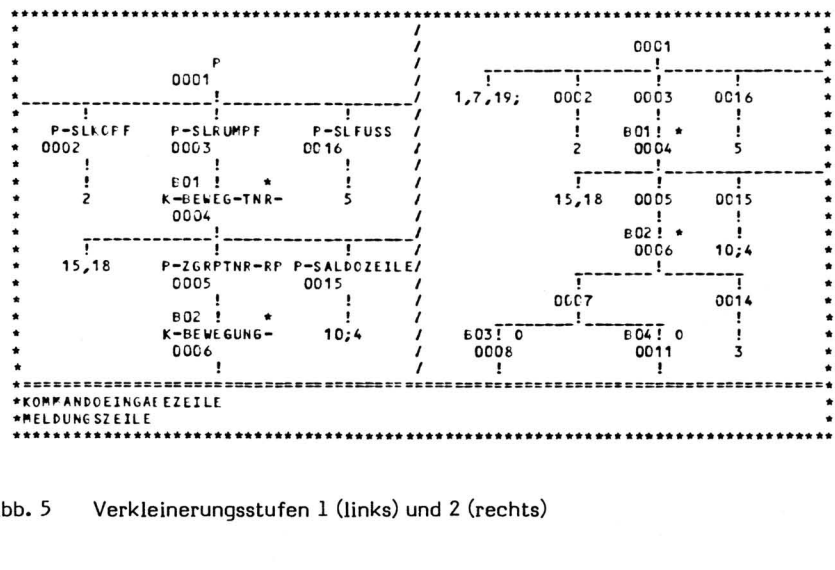

Abb. 5 Verkleinerungsstufen 1 (links) und 2 (rechts)

Zu erläutern bleibt, wie der Ausschnitt eines Gesamtdiagramms bestimmt wird, der auf einem gegebenen Bildschirmteil gezeigt wird. Die ergonomische Zielsetzung hierbei ist, dem Benutzer stets "sinnvolle" Bildausschnitte zu präsentieren. Der Begriff "sinnvoll" wird dahingehend präzisiert, daß der Bildausschnitt einen strukturell zusammenhängenden Teil des Gesamtbildes anzeigen soll. Für die JSP-Baumdiagramme bedeutet das:

- In jedem Werkzeugzustand ist ein "aktueller Kasten" definiert, entweder implizit durch vorausgegangene Operationen des Benutzers mit diesem Kasten oder explizit, indem der Benutzer diesen Kasten direkt ansteuert.

- Dieser aktuelle Kasten wird ins Zentrum des Anzeigebereichs plaziert.
- Um ihn herum werden sukzessiv und mit abnehmender Priorität seine Söhne, sein Vater, seine Brüder, seine Onkel, sowie weitere Vor- und Nachfahren aufgetragen. Das Vorhandensein von weiteren Kästen außerhalb des Bildausschnitts wird durch Verbindungslinien, die aus dem Bild laufen, angedeutet.

Ein Beispiel für das Ergebnis dieser Auftragevorschrift enthält Abb. 6. Dort ist das Gesamtdiagramm abgebildet, wie es z.B. auf Schnelldrucker ausgegeben würde, zusammen mit einem Bildschirmausschnitt. Die im Bildausschnitt enthaltenen Kästen sind im Gesamtbild umrahmt.

Diese Vorgehensweise ist ebenfalls ein Ergebnis unserer Versuche mit Prototypen. Dort zeigte sich, daß die Standardvorgehensweise, den Bildausschnitt als Fenster auf einem festen Diagramm zu bestimmen, nicht ausreicht, da strukturell unzusammenhängende (und damit aussagelose) und auch ganz leere Bildausschnitte nicht zu vermeiden sind.

Entsprechend diesem Vorgehen ist auch das Rollen über das Baumdiagramm nicht als Verschieben eines Fensters über einem festen Bild realisiert. Es kann vielmehr nur entlang der den Bildausschnitt verlassenden Kanten gerollt werden. Der vom Benutzer angegebene Rollfaktor definiert dann einen neuen aktuellen Kasten, auf den der oben beschriebene Auftragealgorithmus erneut angewendet wird.

Die vorgestellten Verfahren sind softwaretechnisch relativ aufwendig, dienen aber dem ergonomischen Ziel, dem Benutzer das Arbeiten mit dem Werkzeug möglichst einfach und angenehm zu machen, d.h. der Anpassung der Software an den Menschen.

5. Probleme bei und Erfahrungen aus der frühen Konzeption der Benutzerschnittstelle des JSP-Arbeitsplatzes

Die sich schon bei der Erstkonzeption andeutenden Schwierigkeiten bei der Realisierung des JSP-Arbeitsplatzes, insbesondere die methoden-spezifischen Aspekte der Ausgabe auf dem Terminal, ließen es uns angezeigt erscheinen, frühzeitig ein Simulationsmodell zu implementieren. Dabei spielten, neben der Darstellung der funktionsorientierten Aspekte des JSP-Arbeitsplatzes, die Aspekte der Werkzeugoberfläche, d.h. die Grafik-Ausgabe, eine wesentliche Rolle. Es traten nun Dinge wie

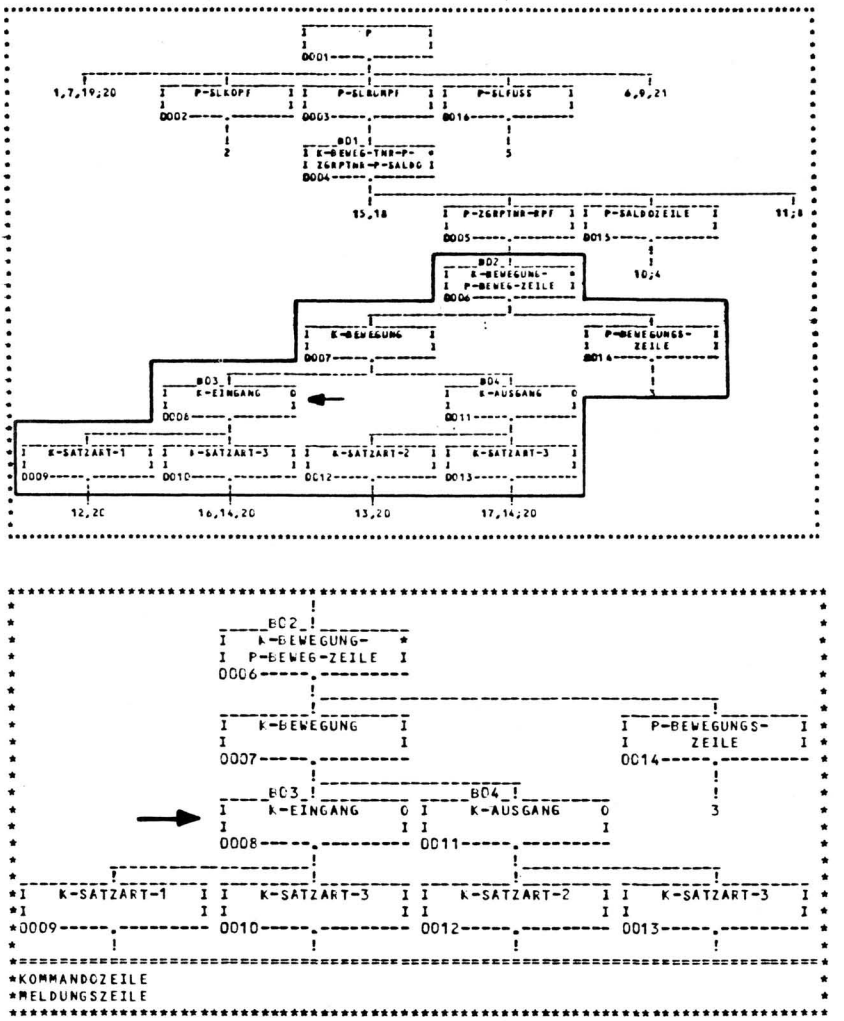

Abb. 6 Gesamtdiagramm und Bildschirmausschnitt (Aktueller Kasten: K-EINGANG)

arbeitsbezogene Abläufe, Hilfs- und Auskunftsfunktionen sowie Fehlerbehandlung in den Vordergrund, weil sie für die Werkzeugoberfläche von erheblicher Bedeutung sind. Jedoch zeigte sich, daß erste Entscheidungen über Bildschirmgestaltung und Dialogstruktur, wobei die methodenbezogene Vorgehensweise zu berücksichtigen war, für die folgenden Entwicklungsschritte bei der Realisierung des Werkzeugs

einen recht starken, normsetzenden Charakter gewonnen hatten.

Eine besondere Erfahrung bei der Konzeption des JSP-Arbeitsplatzes bestand darin, daß zunächst versucht wurde, die Werkzeugoberfläche für Grafik-Bildschirme zu entwerfen, um sie dann für alphanumerische Bildschirme zu vereinfachen. Hierfür wurde unter Verwendung vorhandener Grafik-Bausteine ein Simulationsmodell implementiert. Als Ergebnis konnten zwar hinsichtlich gewisser Aufbereitungsformen der Grafik Fortschritte erzielt werden, es zeigte sich aber sehr deutlich, daß eine derartige Vorgehensweise den spezifischen Eigenarten der alphanumerischen Terminals in keiner Weise gerecht werden konnte. Es wurde daher der Entwurf dahingehend geändert, daß für Grafik- und alphanumerische Bildschirme eine gemeinsame interne Schnittstelle entworfen wurde, daß aber die Werkzeugoberfläche für alphanumerische Bildschirme autonom unter Berücksichtigung der Restriktionen des Mediums neu konzipiert wurde.

Beim Neuentwurf wurde aufgrund der bis dahin positiven Erfahrungen ebenfalls auf die Technik der Implementierung von Prototypen zurückgegriffen. Allerdings stand in diesem Fall kein Werkzeug zur Verfügung, das die Prototyp-Entwicklung unterstützt, und verwendbare Bausteine waren nur in geringem Umfang vorhanden, so daß wir uns auf vorhandene Basissoftware stützen mußten. Daher konnte nicht durch immerwährende Modifikation des Simulationsmodells dieser oben erwähnten normsetzenden Kraft gegengesteuert werden. So kam es dazu, daß die Oberfläche des Simulationsmodells häufig für die zu realisierende Werkzeugoberfläche gehalten wurde.

Trotz dieser negativen Aspekte halten wir den Schritt, ein Simulationsmodell zu implementieren, für sinnvoll und nutzbringend. Nicht nur Schwachstellen des Entwurfs im Hinblick auf die Benutzerschnittstelle wurden deutlich, sondern auch Probleme, die den funktionalen Entwurf betrafen. Am Beispiel des Rollens wird deutlich,daß beides nicht völlig voneinander getrennt gesehen werden kann. Das Simulationsmodell sah nur das Verschieben des Bildschirms über die erstellte Graphik vor, wobei sich dann herausstellte, daß informationsträchtige Teile aus dem Bild "hinausgerollt" wurden. Eine Änderung, die sich hierauf bezog, war die Entscheidung, nur "logisch", d.h. entlang der den Bildschirm verlassenden Kanten zu rollen. Dies bedeutet, daß die Grafik immer wieder neu aufzutragen ist. Dabei spielt - im Gegensatz zu Grafik-Bildschirmen - weniger die Berechnung der Lage der einzelnen Darstellungselemente eine Rolle, sondern eher die Auswahl der im darzustellenden Ausschnitt enthaltenen Teile aus der Gesamtgrafik.

Nicht nur das Rollen, sondern auch das Verkleinern war eine Folgerung aus den Erfahrungen des Simulationsmodells. Dabei wird insbesondere die Komposition des gezeigten Grafik-Ausschnitts nach methodisch und ergonomisch sinnvollen Kriterien gestaltet.

Bereits im Simulationsmodell war ansatzweise die Teilung des Bildschirms vorgesehen, jedoch eröffnete die Verkleinerung weitere Möglichkeiten der Teilung des Anzeigebereichs.

Zusammenfassend läßt sich sagen, daß die Verwendung von Prototypen erhebliche Vorteile bietet, daß aber ohne ein entsprechendes Werkzeug die erreichbaren Vorteile kaum auszunutzen sind.

6. Literatur

N. Christensen: Elemente und Kontrollstrukturen für die Spezifikation von Dialogen. In: Notizen zum Interaktiven Programmieren, Heft 8/1982

N. Christensen, K.-D. Kreplin: Spezifikation von Dialogen versus Simulation, Vergleich zweier Sprachen. Fachgespräch "Simulation als Werkzeug für das Information System Engineering" der GI-Fachgruppe "Methoden und Modelle für die Entwicklung von Informationssystemen", 1982

W. Dette: Erfahrungen mit Programm-Prototypen. In: Softwaretechnik-Trends, Mitteilungen der GI-Fachgruppe "Software-Engineering", Heft 2-1/1982

W. George: Jackson trennt Entwurf und Implementierung. In: Computerwoche 16-22/1981

J. Griese: Software-Ergonomie. In: Angewandte Informatik, Heft 4/1982, S. 230-235

M. Jackson: Grundlagen des Programmentwurfs, 2. Aufl., Darmstadt 1982

K.-D. Kreplin: Simulation der Benutzerschnittstelle im mbp-tool-system. IKD 1982

O. V.: Developing Systems by prototypes. In: EDP-Analyzer, Bd. 19/1981, Heft 9

M. J. Spier, S, Gutz, A. I. Wasserman: The Ergonomics of Software Engineering - Description of the problem space. Symposium on Software Engineering Environments, Bad Lahnstein 1980

D. Vahle: Ein logisches Ein-/Ausgabesystem für formularorientierte Dialoge. In: Notizen zum Interaktiven Programmieren, Heft 8/1982

Anschrift der Autoren:
mbp Mathematischer Beratungs- und Programmierungsdienst GmbH
Semerteichstr. 47, 4600 Dortmund 1