ERGONOMISCH GUTE SOFTWARE DURCH DATENBANK ORIENTIERTEN ANWENDUNGSGENERATOR

H. Beck, Augsburg

Zusammenfassung: Eine wichtige Ursache fuer Unzufriedenheit und Angst vor Bildschirmarbeitsplaetzen ist die Undurchschaubarkeit der Systeme und insbesondere der Software fuer die Benutzer. Demgegenueber loest ein System, welches durchschaubar ist, welches es waehrend der Arbeit erlaubt vom zu bearbeitenden Feld zur zugehoerigen "Programmierung" zu springen, eine Reihe von ergonomischen Problemen. Dies gilt insbesondere fuer die Erlernbarkeit, Selbsterklaerungsfaehigkeit und Kontrollierbarkeit.

Ein solches System mit fuer Benutzer und Maschine gleichermassen verstaendlicher Programmierung wird hier vorgeschlagen. Die Programmierung erfolgt zunaechst felderweise mit einfachen Tests und Aktionen. Die Felder werden zu "intelligenten" Formularen, aehnlich den Spread-Sheet-Kalkulatoren zusammengefasst. Diese werden ihrerseits zu einem Baum, der die gesamte Anweisung umfasst, gebuendelt. Sowohl die Daten, wie das "Programm", werden in einer Datenbank gespeichert um Aenderungen leicht durchfuehren zu koennen und um fuer Benutzer und "Programmierer" getrennt Zugriffsrechte vergeben zu koennen. Dies erhoeht die Verlaesslichkeit des Systems.

1. Einleitung:

Fuer weite Kreise der Bevoelkerung ist der Computer oder die EDV ein Ausloeser von Unbehagen und Angst. Neben drohendem Arbeitsplatzverlust durch den Computer war hierfuer wohl die mangelnde Anpassung der EDV an die Physiologie und Psychologie des Menschen die Ursache. Waehrend im Bereich der physiologischen Anpassung in Deutschland beispielhafte Normierungsarbeit geleistet wurde, steckt die Forschung und Normung im Bereich der psychischen Anpassung noch in den Anfaengen (vgl. [1] S. 231 Erkenntnisstand...).

Ist die Ursache hierfuer die Tatsache, dass sich Technologen, Struktur- und Naturwissenschaftler nicht gern mit Psychologie beschaeftigen und Psychologen meist zuwenig von Software verstehen? Es fehlt auch an Messmethoden, um die Qualitaet und insbesondere die ergonomische Qualitaet der Software zu messen [2], [3], [4].

Ein bedeutender Teilaspekt der Unzufriedenheit und Angst vor der EDV liegt meines Erachtens darin, dass der Benutzer das System nicht "durchschauen" kann und ihm und insbesondere dem ihm unverstaendlichen Code "ausgeliefert" ist. Hier soll dagegen eine Art der Programmierung vorgeschlagen werden, die fuer den Benutzer verstaendlich, lesbar ist und zwar parallel zur Verarbeitung. Dies soll erreicht werden, durch Verwendung von "intelligenten" Arbeitsblaettern fuer die Programmierung im kleinen (Detailprogrammierung) und durch die Zusammenfassung dieser Arbeitsblaetter zu einem Baum zur Beschreibung der Gesamtanwendung. Neben den im naechsten Abschnitt zu besprechenden, ergonomischen Vorzuegen wie Selbsterklaerungsfaehigkeit, Kontrollierbarkeit etc. bietet dieses System folgende Vorteile:

- Benutzer koennen kleinere Anwendungen selber erstellen
- Benutzer koennen Anwendungen aendern.

Die Aenderungsfreundlichkeit wird durch die Verwendung einer Datenbank unterstuetzt. Somit erweist sich das Verfahren als nuetzlich fuer die Softwareimplementierung, Validierung, Wartung und Pflege. Zusaetzlich kann auf Teile der Dokumentation verzichtet werden, da der Baum der "intelligenten" Arbeitsblaetter selbstdokumentierend ist. Auf gruendliche Analyse und den Entwurf des Systems kann allerdings nicht verzichtet werden. Hier koennen ergonomische Aspekte mit einem System wie Benorsy [5] eingebracht werden.

Der Anwendungsbereich des hier beschriebenen Verfahrens duerfte insbesondere in interaktiven Anwendungen im kommerziellen Bereich liegen. Die stuermische Entwicklung auf dem Gebiet der Spread-Sheet-Kalkulatoren z.B., [6] - [12] zeigt den Bedarf in dieser Richtung.

2. Ergonomische Forderungen und Softwareerstellungsmethoden

Hier wollen wir den Forderungen, die an ergonomisch gute Software gestellt werden, Massnahmen und Methoden der Softwareerstellung gegenueberstellen. Ein Hauptgrund fuer unzureichende Ergonomie der Software duerfte in der spaerlichen Unterstuetzung liegen, die fast alle hoeheren Programmiersprachen dem Programmierer hierfuer bieten. Anhand der anschliessend aufgelisteten Methoden kann solche Unterstuetzung in das Programmiersystem eingebaut werden.

Forderung	Methode
Problemangemessenheit	individuelle, anpassbare Software
Anpassbarkeit	programmieren durch Benutzer
Erlernbarkeit	einheitliche Struktur, einfache Konzepte
Selbsterklaerungsfaehigkeit	sichtbare Aktionen, benutzerverstaendliche Programmierung, "Hilfestufen"
Kontrollierbarkeit	einfache Schritte, Zugriff zur Beschreibung des Ablaufs
Korrigierbarkeit	"Undo"-Funktion, Schreiben in Puffer
Verlaesslichkeit	sichere Datenorganisation
Fehlertoleranz	Ueberpruefen von Eingaben

Tab. 1 Ergonomische Forderungen contra Softwaretechnische Methoden

Zusaetzlich sind zu beachten:

- angemessene Reaktionszeiten
- nicht zuviel, nicht zuwenig, gutstrukturierte Information auf dem Schirm
- Vermeidung von Langeweile und Stress

Insbesondere aus der Forderung der Verlaesslichkeit und der Anpassbarkeit ergibt sich, dass eine Datenbank fuer die Speicherung sowohl der Daten wie auch des Programmes vorzuziehen ist.

3. Gesamtstruktur des Anwendungsentwicklungssystem

Eine Anwendung wird in dem vorgeschlagenen System gegliedert in 3 verschiedenen Detailierungsstufen:

- Einzelfelder
- 2 dimensionale Zusammenfassungen von Einzelfeldern zu Arbeitsblaettern
- Zusammenfassung der Arbeitsblaetter zum "Programmbaum"

Diese Art der Strukturierung geht davon aus, dass fuer die Programmierung im kleinen (Detailprogrammierung) direkt gezeigte Felder und zweidimensionale Zusammenfassungen hiervon, optimal sind, da dies den sehr guten menschlichen Faehigkeiten der Verarbeitung 2-dimensionaler Bilder entgegenkommt. Dies ist im Gegensatz zu der weitgehend linearisierten Darstellung in hoeheren Programmsprachen zu sehen. Dieser Vorteil geht aber bei sehr komplexen Anwendungen verloren, es entstehen dann unuebersichtliche Gebilde. Hier ist die Verdichtung, Vergroeberung und Abstraktion der Information erforderlich und die Zusammenfassung dieser Information zu einem hierarchischen System. Dies wird im 5. Abschnitt genauer beschrieben.

Die "Programmierung" der Anwendung erfolgt durch Festlegung der Gestalt der Arbeitsfelder, der Arbeitsblaetter und des Strukturbaumes und parallel hierzu der Festlegung der zugehoerigen Verarbeitungsschritte. Die Verarbeitungsinformation wird von der Maschine direkt abgearbeitet und kann jederzeit vom Benuetzer angesehen werden. Dabei ist der entscheidende Unterschied zu herkoemmlicher Programmierung, dass direkt vom Arbeitsfeld automatisch die entsprechende Verarbeitungsinformation gefunden werden kann und nicht in einem Quellprogramm (welches oft nicht vorhanden ist) an verschiedenen Stellen die notwendige Information zusammengesucht werden muss (siehe 8., Bild 2).

4. Die Detailprogrammierung

Fuer die Programmierung im Detail benutzen wir "intelligente" Arbeitsblaetter. Diese intelligenten Arbeitsblaetter bestehen aus einzelnen Arbeitsfeldern und zweidimensionalen Zusammenfassungen von Arbeitsfeldern. Einfache Formen solcher Arbeitsfelder sind als Masken bekannt und erfreuen sich grosser Beliebtheit, insbesondere als Benutzerschnittstelle zu Datenbanken.

Intelligent moechte ich hier ein Arbeitsblatt nennen, wenn nicht nur Lage und Groesse von Feldern festgelegt werden, sondern gekoppelt an die Felder auch Verarbeitungsinformationen vorliegen. Weiterentwicklungen auf der Basis von Einzelfeldern sind z.B. durch Datastar [7], auf der Basis von sich wiederholenden Einzelfeldern mit EASI [8] und auf der Basis einer zweidimensionalen Matrix durch Visicalc [6] realisiert werden. "Intelligente" Arbeitsblaetter wie Viscalc [6], Kopien und Weiterentwicklungen hiervon haben in den USA in den letzten Jahren eine enorme Verbreitung gefunden und sollen hier kurz unter ihrem neu gepraegten Namen beschrieben werden.

4.1 Spread-Sheet-Calculators

Mit "spread-sheet-calculators" oder "work-sheet-calculators" wird eine neue Klasse von Programmierwerkzeugen bezeichnet. Diese bestehen aus einem 2-dimensionalen, meist ungefaehr 64 x 256 Felder grossen Arbeitsblatt, welches ausschnittweise (oft mehrere Ausschnitte gleichzeitig) am Bildschirm sichtbar ist. Jedes dieser Felder kann dann mit Daten (Text oder Zahlen) oder mit Formeln beschrieben werden. Meist findet nach jedem neuen Eintrag eine sofortige Neuberechnung aller Formeln statt. Um dieses Rechenblatt herum gruppieren sich weitere Hilfsprogramme, wie Abspeichern oder Ausdrucken des Arbeitsblattes oder von Teilen, Uebergabeprogramme zu anderen Programmen etc.

Ein wesentlicher Faktor fuer die rasch steigende Beliebtheit dieser spread-sheet-calculators duerfte die Erfuellung ergonomischer Forderungen sein. So ist die Selbsterklaerungsfaehigkeit und Kontrollierbarkeit fuer jeden Benutzer gegeben.

Ebenso ist die Anpassbarkeit und Problemangemessenheit gegeben, da durch den Benutzer Anwendungen erstellt bzw. geaendert werden koennen. Allerdings ist dieses Verfahren fuer groessere Anwendungen ungeeignet. Erweiterungen durch Uebergang zu mehreren Arbeitsblaettern sind z.B. im Multiplan [11] realisiert worden. In Logicalc [12] werden auch einfache, logische Operationen erlaubt. In den im folgenden beschriebenen "intelligenten" Arbeitsblaettern werden noch etwas weitergehende "Programmiermoeglichkeiten" erlaubt und gleichzeitig ein Schutz des "programmierten" Arbeitsblattes eingefuehrt.

4.2 Das "intelligente" Arbeitsblatt

Den bisher betrachteten "spread-sheet calculators" fehlt meines Erachtens die Sicherheit, Verlaesslichkeit und Fehlertoleranz, um sie fuer kommerzielle Routineanwendungen verwenden zu koennen.

Hier fehlt meines Erachtens zum einen die Trennung zwischen Programmierung und Verarbeitung. Wir wollen deshalb statt des einfachen Arbeitsblattes ein doppeltes verwenden, welches aus Benutzerblatt und Programmblatt besteht. Zum Programmblatt werden fuer den Benutzer nur Leserechte vergeben, waehrend der Programmierer Lese- und Schreibrechte erhaelt. Diese Rechte lassen sich einfach verwalten, da beide Blaetter von einer Datenbank verwaltet werden. Zusaetzlich koennen durch die Kombination von Pruefungen und Aktionen (siehe 4.3) falsche Eingaben einfach abgefangen werden. Ein weiterer Mangel aelterer Spread-Sheet-Calculators besteht in fehlenden Integrationsmoeglichkeiten zu Gesamtanwendungen.

Dieser ist in neuen Produkten z.B. 1-2-3 [9] und Visi-On [10], teilweise ueberwunden. In unserem Vorschlag sind zu diesem Zweck die Koepfe und Fuesse der Arbeitsblaetter bereits mit der Information versehen, die als Anker fuer die Integration in einen Baum von "intelligenten" Arbeitsblaettern dienen.

4.2.1 Das Benutzerblatt

Die Groesse des Benutzerblattes ist waehlbar (10 x 80 bis 72 x 132 Zeilen x Spalten). Der Defaultwert betraegt 20 x 80. Das Blatt ist in Felder einteilbar, die als Einzelfelder oder als 2-dimensionale Tabellen von Feldern auftreten koennen. Die Einzelfelder erhalten die Namen A bis Z, die Felder in Tabellen die Namen von A1 bis Z132. Waehrend die Groessen der Felder frei waehlbar sind, gilt dies in den Tabellen fuer die Spaltenbreite. Weiterhin koennen Zeilen als wiederholbar definiert werden (der Rest verschiebt sich hierbei). Plaetze fuer die Felder werden durch @ @ fuer alphanumerische Felder und #...#, #...# fuer Zahlenfelder reserviert.

Das Benutzerblatt wird (falls es groesser als 20 x 80 ist) durch scrolling automatisch so verschoben, dass die jeweils aktuellen Felder sichtbar sind.

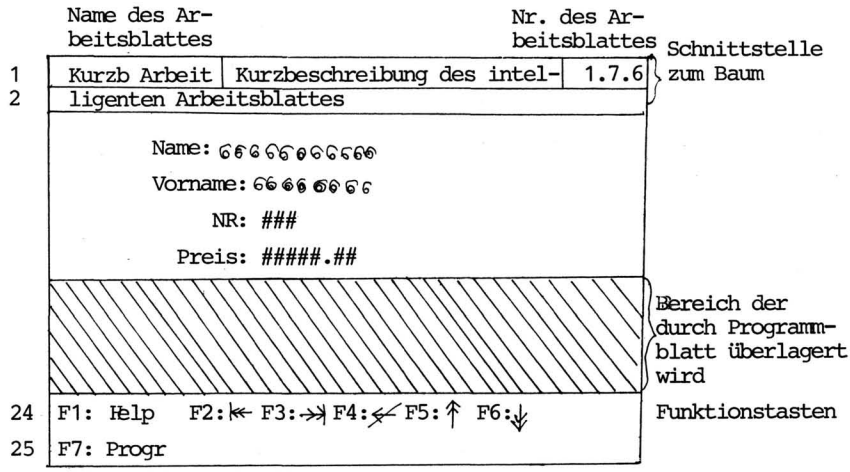

Bild 1 Beispiel eines intelligenten Arbeitsblattes

Die cursor keys sind ebenfalls aktiv um einfaches navigieren auf dem Arbeitsblatt zu erlauben, zum Beispiel zur Datenerfassung per Telefon. Diese Option kann abgeschaltet werden.

Auf Wunsch kann durch Druecken des "Programmkeys" der zugehoerige Programmblattausschnitt gezeigt werdem. Zusaetzliche Hilfeinformation wird durch die Helptaste angefordert.

4.2.2 Das Programmblatt

Da die Einzelfeldprogrammierung mehr Platz beansprucht, als die physische Felddarstellung auf dem Benutzerblatt, bestehe das Programmblatt aus der Liste der Einzelfeldbeschreibungen des Benutzerblattes. Um trotzdem die visuelle Beziehung zwischen Benutzerblatt und Programmblatt herzustellen wird nach dem Druecken der Helptaste gleichzeitig das aktuell zu bearbeitende Feld, wie auch in einen extra Felde auf dem Schirm, der zugehoerige Programmblattausschnitt angezeigt. Es lassen sich zusaetzlich hardcopies eines Benutzerblattes inclusiv Feldnamen und des zugehoerigen Programmblattes abrufen. Die Abarbeitung des Programmes erfolgt in alphabetischer Ordnung der Feldnamen, soweit nicht durch die im folgenden beschriebene Einzelfeldprogrammierung anderes festgelegt ist.

4.3 Einzelfeldprogrammierung

Die Einzelfeldprogrammierung bestehe aus einer Folge von Pruefungen und Aktionen. In halbformaler Schreibweise bedeutet das:

<Einzelfeldprogramm>: = {<test>:}<action>;
                        {<test>:}<action>;

Der Teil in den geschweiften Klammern kann hierbei auch fehlen. Die Interpretation sei hierbei: die Folge wird der Reihe nach abgearbeitet. Jede Aktion, die einem erfolgreichen Test unmittelbar folgt, wird ausgefuehrt.

Ein <test> ist eine Verknuepfung von einfachen Tests durch "und", "oder" und "nicht".

Ein einfacher Test lautet z.B.

A3 in {a, b, c, d}

oder

B7 < D6

oder Laenge (B7) = 5, d.h. es sind nur einfache Vergleiche zugelassen.

Eine Aktion kann

- einem Feld einen arithmetischen Ausdruck zuordnen (A: = C2 + 5,3),
- eine Nachricht dem Benutzer anzeigen (show "bitte Eingabe wiederholen"),
- ein Unterprogramm aufrufen ("nun wird die erste Spalte aufsteigend sortiert": call sort (A1, A20, steigend),
- ein Datenfeld holen (get Adresse.Ort, key in A5),
- ein Datenfeld beschreiben (put Adresse.Name, key in B),
- die vorausgehende Aktion wiederholen (redo)
- Cursorkeys abschalten,
- Feld verstecken (nur im Programmblatt sichtbares Feld),
- Sprung zu einem anderen Arbeitsblatt,
- Sprung zu einem anderen Arbeitsfeld,
- Antwort auf "Hilfe"

Spruenge, insbesondere Rueckwaertsspruenge sollten sehr sparsam verwendet werden.

Fuer die Programmierung des Einzelblattes werden die Befehle den Funktionstasten zugeordnet, so dass diese schnell erfolgen kann.

5. Globale Programmierung

Obwohl durch intensive Anwendung der Fenstertechnik auch relativ grosse Anwendungen durch ein grosses Arbeitsfeld programmiert werden koennen, ist ab einer gewissen Groesse die 2-dimensionale Darstellung ergonomisch nicht mehr guenstig (man denke hierbei auch an riesige Leitstaende und an die Cockpits von Flugzeugen). Grosse Anwendungen erfordern die Verdichtung und Abstraktion der Information.

5.1 Der Baum aus intelligenten Arbeitsblaettern

Die Verdichtung der Information wird hier durch den Namen (<15 Zeichen) und die Kurzbeschreibung der Arbeitsblaetter (<200 Zeichen) erreicht.

Statisch sollen die Arbeitsblaetter hierarchisch geordnet sein, wobei die Arbeitsblattnummer der Position des Blattes im Baum dargestellt. Ausschnitte dieser Hierarchie sollen auf Wunsch ange-

zeigt werden koennen. Um innerhalb einer Anwendung und ueber alle Anwendungen hinweg eine einheitliche Benutzerschnittstelle zu erreichen, werden einige Operationen fuer das navigieren in dem Anwendungsbaum standardmaessig vorgegeben. Dies sind insbesondere die Funktionen "vorwaerts zum naechsten Blatt", "rueckwaerts zu vorhergehendem Blatt", "hoch zur naechst hoeheren Stufe". Dadurch ist ein erheblicher Teil der globalen Programmierung schon durch die Vorgabe der Arbeitsblattnummern gegeben.

Zusaetzliches Navigieren wird durch die Aktion Sprung zum Arbeitblatt Nr. xxxx erreicht, welches es zum Beispiel gestattet, einfach Arbeitsblaetter fuer eine Menueauswahl zu definieren.

5.2 Anschlussmoeglichkeiten weiterer Bausteine

Waehrend in den vergangenen Jahren separate Produkte wie Spread-Sheet-Kalkulatoren, Reportwriter , Maskengeneratoren, Menueprozessoren und graphische Ausgabegeneratoren angeboten wurden, wird jetzt zunehmend Wert auf integrierte Produkte gelegt. Die Integration sollte dabei, sowohl den Datenaustausch zwischen den Produkten, wie auch die einheitliche Benutzerschnittstelle umfassen. Ersteres wird hier durch die gemeinsame Datenbank sichergestellt, letzteres kann hier erreicht werden, indem die wesentlichen Module der Bausteine aus intelligenten Arbeitsblaettern heraus aufgerufen werden. Die notwendigen Parameter muessen dann in intelligenten Formularen zuvor erfasst werden.
Eine weitere attraktive Moeglichkeit, komplexere Unterprogramme einzubinden, waere, diese als Struktogramme wie in Graphcal [13] zu definieren und einzubinden.

Allgemein gilt allerdings, dass einheitliche Benutzerschnittstellen nur ueber verstaerkte Normierungsanstrengungen zu erreichen sind.

6. Die Datenbank

Die Datenbank soll, wie bereits oben erwaehnt, sowohl zur Speicherung der Daten wie des "Programms" (d.h. der Benutzer- und Programmarbeitsblaetter, wie auch des Programmbaumes) dienen.

6.1 Datenspeicherung

Um der ergonomischen Forderung der Korrigierbarkeit entgegenzukommen, und um eine leistungsfaehige "Undo" Funktion zu haben, sollen erst nach der ersten Aktion auf einem neuen Arbeitsblatt die geaenderten Daten des alten Arbeitsblattes auf den permanenten Speicher zurueckgeschrieben werden. Das heisst waehrend der Bearbeitung eines Arbeitsblattes werden alle geschriebenen Daten in einem Puffer zwischengespeichert.

6.2 Programmspeicherung

Fuer die Programmspeicherung ist zu beachten, dass zumindest fuer kommerzielle Datenverarbeitung die Benutzung wesentlich haeufiger als die Programmierung ist und deshalb guenstige Zugriffszeiten erforderlich sind. Deshalb wird die folgende 2-stufige Datenorganisation vorgeschlagen. In der aeusseren Stufe werden die Arbeitsblaetter verwaltet. Primaerschluessel ist hierbei die Blatt Nr., Sekundaerschluessel sind die Kurzbezeichung und die Beschreibung des Arbeitsblattes. Zusaetzlich gespeichert sind die hierfuer vergebenen Benutzer- und Programmierrechte und die Indexinformation fuer die feldinterne Verarbeitung. Diese "Saetze" sollen physikalisch hintereinander auf der Platte stehen, so dass kurze Ladezeiten garantiert werden.

Innerhalb dieser grossen Saetze sind als kleine Datenbank die Relationen Text und Einzelfeldprogramm gespeichert. Auch hier ist variable Satzlaenge erforderlich, als Schluessel dienen Position (Zeile/Spalte) fuer den Text und der Feldname fuer die Einzelfeldprogramme.

Der Nachteil dieser 2-stufigen Organisation wird eine gewisse Umorganisationszeit nach groesseren Programmveraenderungen sein, die aber dann auch in Kauf genommen werden kann.

7. Zum Anwendungsinterpreter

Der Anwendungsinterpreter sollte sich vollstaendig im Arbeitsspeicher befinden. Er laedt die entsprechenden "intelligenten" Arbeitsblaetter und fuehrt dann diese der Reihe nach aus. Gleichzeitig muessen laufend die "Helptaste" und die anderen Funktionstasten abgefragt

werden, oder in einem multi-task-system eine von diesen Tasten stammende Unterbrechung akzeptiert werden. Der Durchsatz eines solchen Systems muesste fuer interaktive Anwendung bei Verwendung einess modernen 16-bit Mikroprozessor pro Benutzer zufriedenstellend sein. Dies zeigen Versuche mit dem aehnlich organisierten System EASI [14].

8. Schlussbemerkungen

Ein Hauptnachteil der konventionellen Programmierung von interaktiven Anwendungen ist meines Erachtens durch die relativ komplexe Zuordnung zwischen Benutzerschnittstelle und Programm gegeben. Demgegenueber ist mit "intelligenten" Formularen die Zuordnung auf einen Blick ueberschaubar (siehe Bild 2). Meist liegt das Quellprogramm nicht einmal vor.

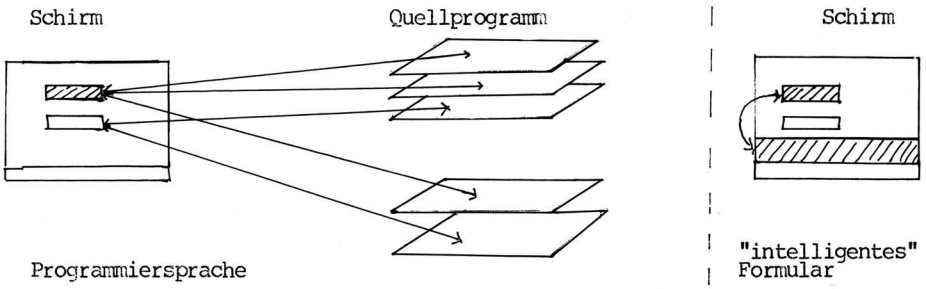

Bild 2: Zuordnung Programm/Benutzerschnittstelle

Mit der hier vorgeschlagenen Methode wird dieser Mangel ueberwunden.

Die weiteren wichtigen Punkte, naemlich
- gleiche Information fuer Mensch und Maschine
- 2-dimensionale "intelligente" Arbeitsblaetter
- baumartige Zusammenfassung der Arbeitsblaetter
- Benutzung einer Datenbank

sollen eine Anwendung so
- durchsichtig
- uebersichtlich
- korrigierbar
- sicher

machen, dass ein Benutzer erfolgreich und zufrieden damit arbeiten kann. Ein Teil der Benutzer wird bei der Benutzung die "Programme" so kennenlernen, dass sie lernen, eigene neue "Programme" zu erstellen.

9. Literatur

[1] Griese, J.; Software Ergonomie, Angewandte Informatik 4/82, 230 - 235

[2] Cordes, R. E.; SUIT (Software-User Interface Test), Human Factors Paper HFP 80 - 19, NCR Corp., 1980

[3] Hofmann, J.; Vorueberlegungen fuer ein Instrumentarium zur Gestaltung von Mensch-Rechner-Schnittstellen, Notizen zum Interactiven Programmieren, Heft 8, Maerz 1982, 15 - 22.

[4] Kofer, G. R.; Some Software Integration Technology Concepts for: saving money while doing empirical user research - Office Information Systems, INRIA 1982.

[5] Essig,H., Heibey,H.-W., Kuehn,M., Rolf A.; BENORSY, Formalisiertes Verfahren zur benutzerorientierten Systemrevision, Bericht Fachbereich Informatik, Universitaet, Hamburg 1980.

[6] VISICALC, Visicorp., San Jose, 1979

[7] DATASTAR, Micropro International, San Rafael, 1980

[8] EASI, NCR GmbH, 1981

[9] 1-2-3, Lotus Development Corporation, Cambridge 1982

[10] VISI-ON, Visicorp., San Jose, 1978

[11] Multiplan, Microsoft, Bellevue, 1982

[12] Logicalc, SPI, San Diego, 1981

[13] Willimann, L.-S.; Struktogramme als Programmiersprache, Angewandte Informatik 3/82, 184 - 190.

[14] Beck,H; Visuell Definieren statt Programmieren, Notizen zum Interaktiven Programmieren, Heft 8, Maerz 1982, 3-14.

Dr. H. Beck
NCR GmbH
Ulmerstr. 160

D-8900 Augsburg