4.0 ENTWICKLUNG LANDWIRTSCHAFTLICHER SOFTWARESYSTEME - ENTWURF UND PROGRAMMIERUNG VON NOTWENDIGEN SOFTWAREKOMPONENTEN

4.1 Softwaretechnologie - Methoden und Prinzipien der Programmentwicklung für landwirtschaftliche Anwendungen

Überlegungen zu Möglichkeiten und Methoden der Programmentwicklung kamen erst Ende der 60er Jahre auf.
Die sogenannte Softwarekrise hatte sie ausgelöst. Darunter ist eine Demoralisierung bei Programmentwicklern zu verstehen, die durch

- fehlerhafte Programme,
- schlecht konzipierte Programme,
- wachsende Kosten der Erstellung,
- fehlende Konstruktionsprinzipien,
- hohe Kosten der Wartung (= Fehlerbeseitigung)

initiert wurde (Stetter, 1984, S.11).

Seit dieser Zeit schenkte man einer Teildisziplin der Informatik, der "Softwaretechnologie", der Lehre von der planmäßigen Entwicklung und praktischen Gewinnung und Anwendung von Gesetzmäßigkeiten und Methoden bei der Erstellung von Software (Stetter, 1984, S. 14), zunehmende Aufmerksamkeit. Unter Software versteht man dabei nicht nur die Programme selbst, sondern auch z.B. ihre Dokumentation. Mit dem Begriff Technologie hoffte man, mit ingenieurmäßigem Vorgehen die Mängel in der Programmerstellung zu erfassen und zu überwinden.

4.1.1 Erwünschte Programmeigenschaften

Um Software mit Hilfe der Softwaretechnologie planmäßig entwickeln zu können, muß man sich über die gewünschten Qualitätskriterien des Endproduktes im Klaren sein. Stetter (1984, S.32) gibt in Weiterentwicklung zu Boehm (1978) die Kriterien für die Güte eines Programmes in Abbildung 16 auf Seite 31 an.

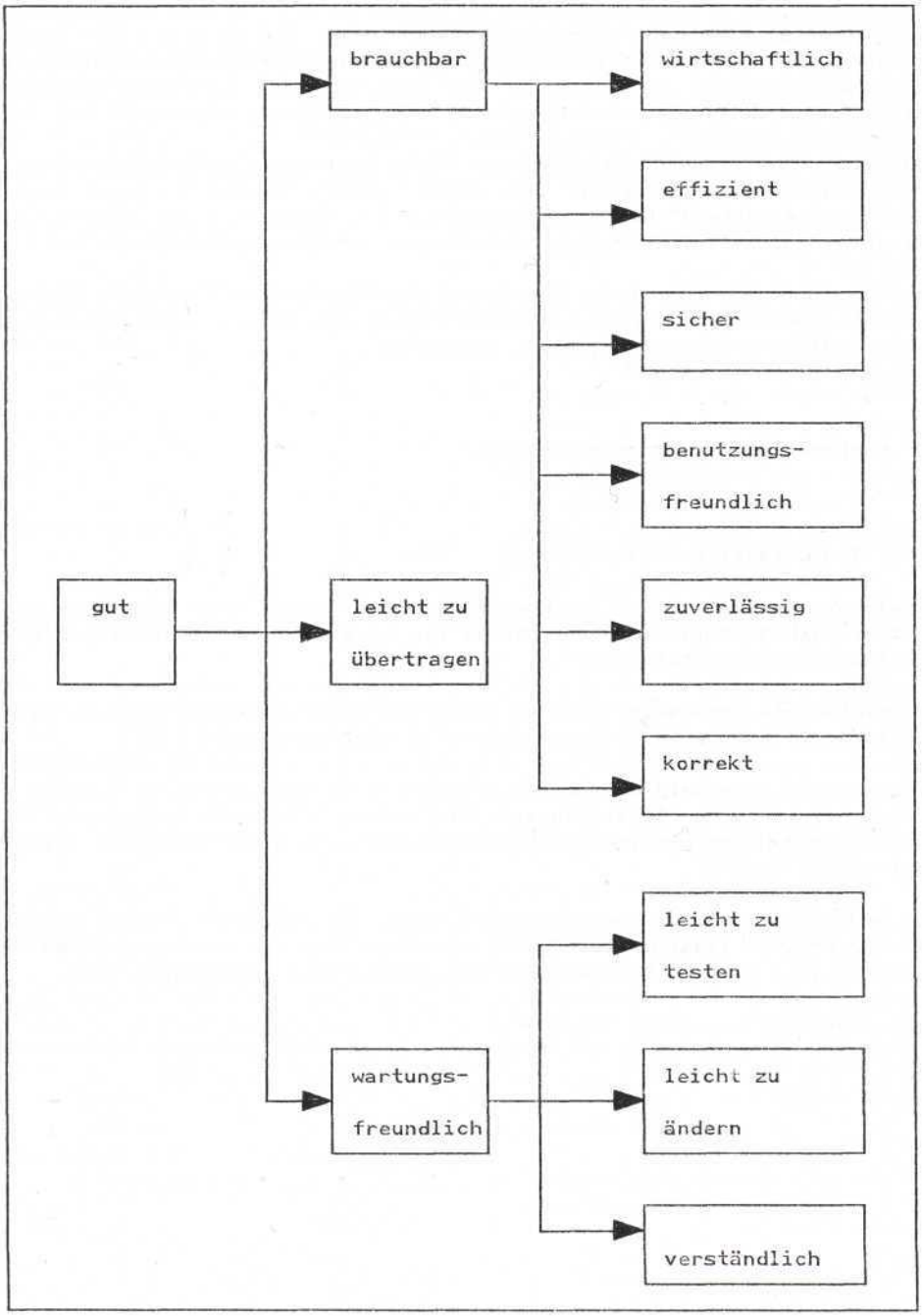

Abbildung 16: Kriterien zur Gütebeurteilung eines Programmes (in Anlehnung an Stetter, 1984, S.32)

Den aufgeführten Attributen wird sicherlich jeder, der sich mit der Programmentwicklung oder -anwendung beschäftigt, zustimmen können. Das Problem einer solchen Qualifizierung liegt aber in der Praxis in der fehlenden Möglichkeit, alle Attribute quantifizieren, d.h. testen und messen zu können.

Die Attribute "effizient", "sicher" und "benutzerfreundlich"[16] können letztlich nur subjektiv beurteilt werden. Die Eigenschaften "leicht zu übertragen" und die "Wirtschaftlichkeit" haben für Softwarehäuser sicherlich ein anderes Gewicht als für die Programmentwicklung innerhalb von Forschungsprojekten.

Große Bedeutung ist allerdings den Attributen "wartungsfreundlich", "zuverlässig" und "korrekt" beizumessen. Systematisch durchgeführte Programmtests können und sollen diese Kriterien objektiv überprüfen.

4.1.2 Methoden der Programmentwicklung

4.1.2.1 Traditionelles Phasenkonzept

Innerhalb des Bereiches der Softwaretechnologie haben sich mittlerweile die verschiedensten Konzepte und Vorschläge für eine geplante, effektive Softwareentwicklung herauskristallisiert.

Das traditionelle Phasenkonzept geht dabei von einer Stufenentwicklung aus, die in Abbildung 17 auf Seite 33 dargestellt ist (Balzert, 1982, S.15).

Die Phasen sind innerhalb des Konzeptes nicht fest standardisiert, sondern jeder Autor hat seine eigenen Bezeichnungen. Die grobe, sinngemäße Abfolge der Phasen stimmt jedoch bei den meisten Vorschlägen überein. Eine Übersicht gibt dazu Balzert (1982, S.469).

Das zu entwickelnde Programm durchläuft dabei die einzelnen Phasen streng sequentiell. Am Ende jeder Phase erfolgt eine Bewertung des Planergebnisses. Fällt sie negativ aus, muß das Programm die jeweilige Phase wiederholen.

---

[16] Siehe zur Begriffseingrenzung Kernler (1985, S.65-66).

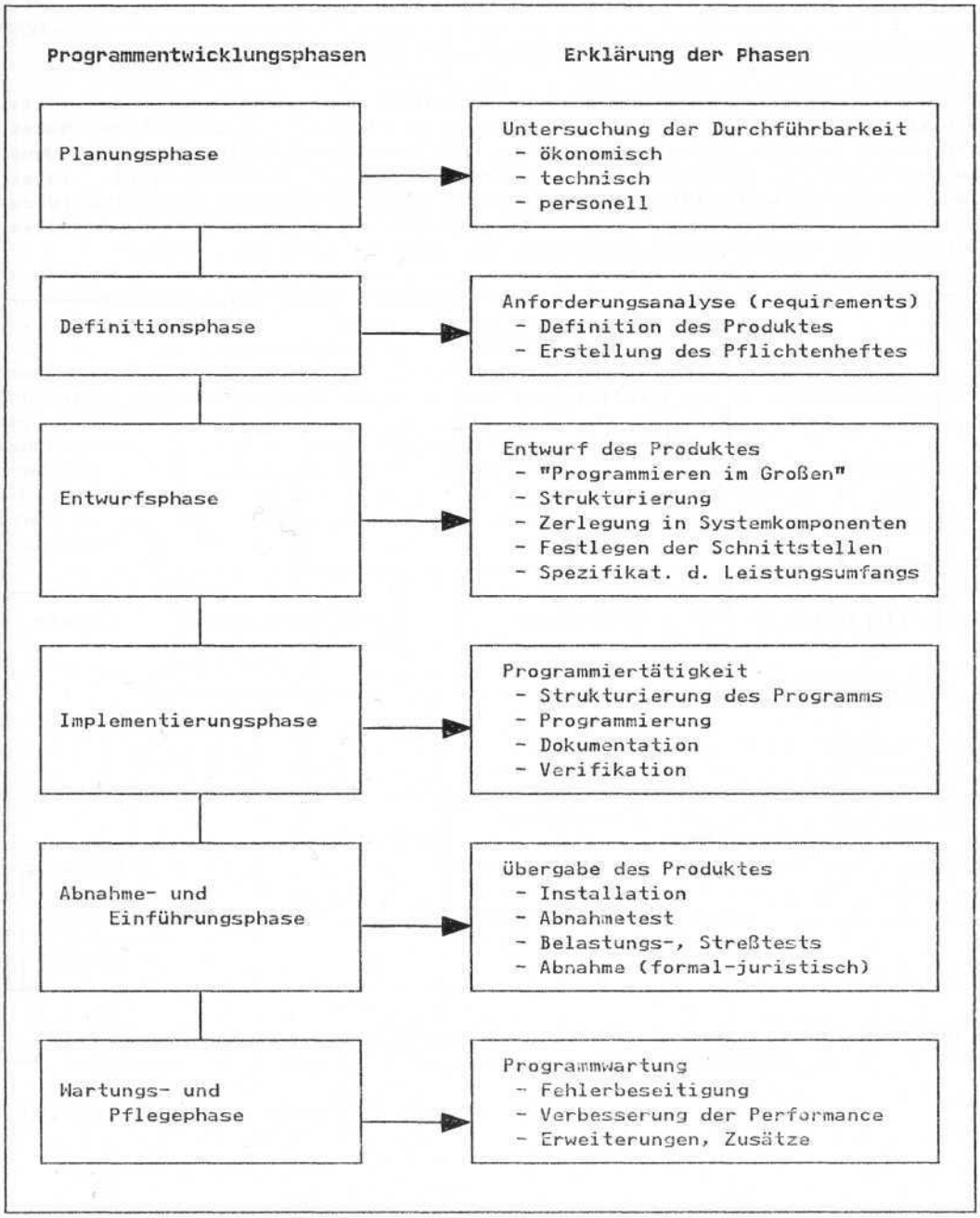

Abbildung 17: Programmentwicklung: traditionelles Phasenkonzept

4.1.2.2 Prototyping (Erstellung eines Prototypen)

Ein weiteres Verfahren zur Programmentwicklung ist das Prototyping.

Unter Prototyping ist der Prozeß der Erstellung eines Prototypen (übersetzt: Urbild, Muster, Rohling) zu verstehen (Kreplin, 1985, S.73). Zielsetzung dieses Verfahrens ist es, möglichst frühzeitig ein lauffähiges Programm zur Verfügung zu haben, das als Grundlage für die Diskussion mit dem Anwender dient. Dabei ist der erstellte Prototyp nicht mit einer billigen Attrappe - wie häufig zu Werbezwecken eingesetzt - vergleichbar, sondern ein systematisch entwickeltes Gebilde, das hauptsächliche Wesenszüge des späteren Programmes trägt.[17]

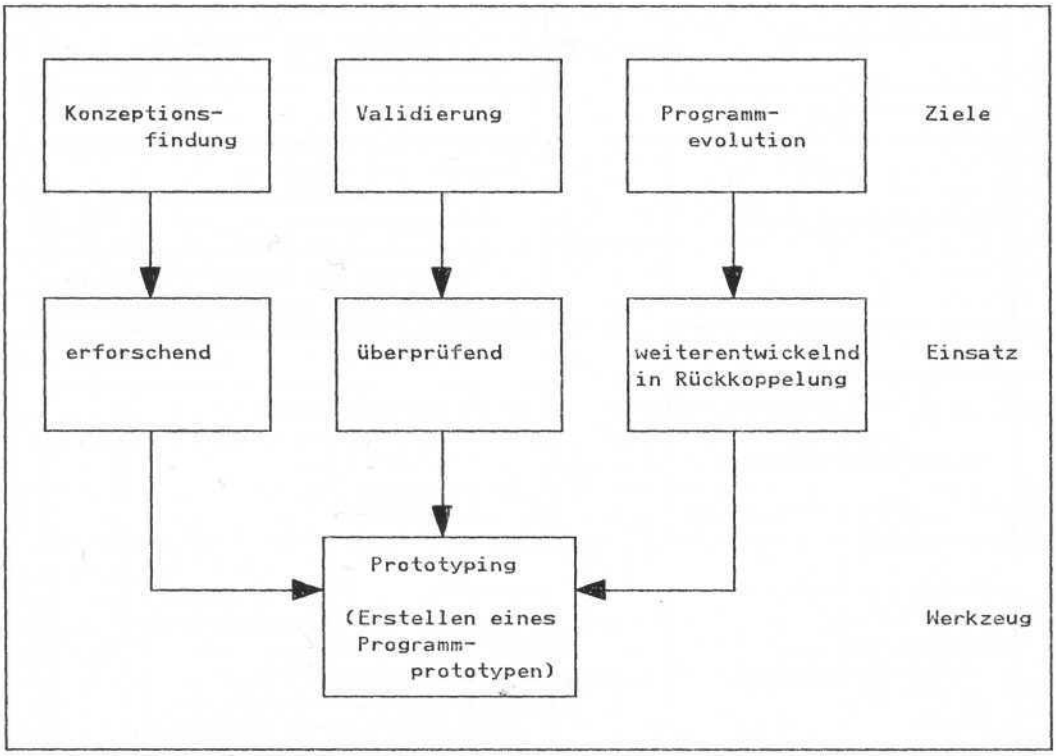

Abbildung 18: Einsatzmöglichkeiten des Prototypings (Erstellung eines Programmprototypen)

---

[17] Vgl. zur Klassifizierung unterschiedlicher Prototyping-Ansätze auch Hallmann (1985).

In Weiterentwicklung von Floyd (1984, S.1-18) läßt sich die Erstellung des Prototypen in dreierlei Weise einsetzen (Abbildung 18 auf Seite 34):

- Das erforschende Prototyping ist ein Hilfsmittel zur Konzeptionsfindung. Es ist dann besonders hilfreich, wenn von der Anwenderseite des Programmes unklare Anforderungen bestehen. Diese können dann mit Hilfe des Prototypen spezifiziert werden.

- Das Prototyping kann auch zur Programmüberprüfung (Validierung) durch den Anwender benutzt werden, bevor es seine endgültige Gestalt annimmt.

- Das weiterentwickelnde Prototyping ist eine Form der Softwareentwicklung, die auf Versionen beruht, wobei der Anwender ein arbeitsfähiges Programm erhält, das dann von Version zu Version weiterentwickelt wird.

Besondere Anforderungen stellen sich innerhalb der Wissenschaft. Sie möchte einerseits in der Programmentwicklung für die Umsetzung von Forschungsergebnissen in praxisnahe Programme sorgen. Andererseits will sie die Programme nicht unabhängig von einander, sondern miteinander vernetzt erstellen.
Innerhalb des Pflanzenbaus steht damit die Programmentwicklung zwischen den Anforderungen von Landwirten und Beratern als Programmanwendern und der Wissenschaft und Beratung als Lieferanten von wissenschaftlichen Ergebnissen und Ideen (Abbildung 19).

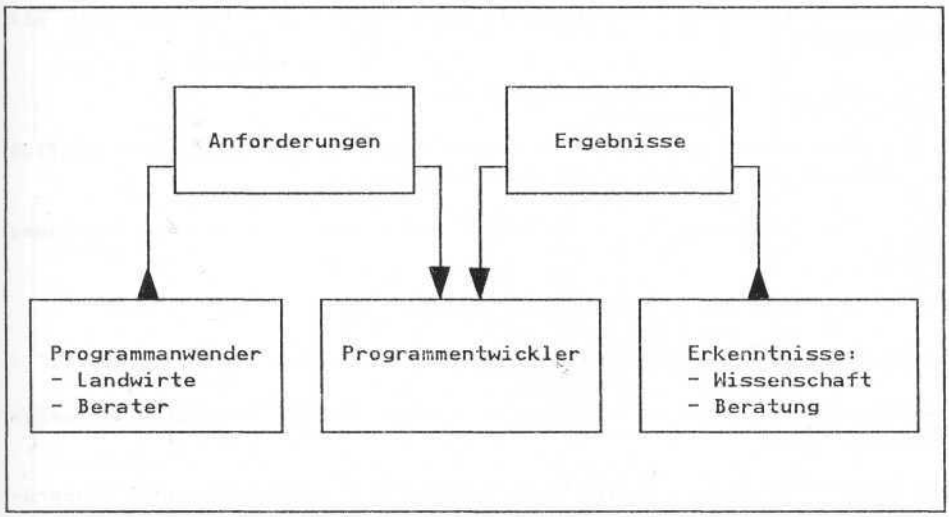

Abbildung 19: Beziehungen zwischen Programmentwicklern, Programmanwendern und der Wissenschaft

4.1.2.3 Gewählte Kombination zwischen Phasenkonzept und Prototyping zur Programmentwicklung

Während sich die Entwicklung der ersten Programme unserer Arbeitsgruppe mehr oder weniger nach dem Phasenkonzept richtete, ist heute zur Erstellung integrierter, miteinander vernetzter Programme eine Kombination aus Phasenkonzept und Prototyping besser geeignet (zur Vertiefung dieses Themas siehe Boehm, 1984, S.494-519).

Der Nachteil des Phasenkonzeptes bestand darin, daß die zukünftigen Benutzer häufig erst in der Implementierungsphase zu spät mit dem fertigen Programm in Berührung kamen. Häufig war es dann auf Grund anderer Benutzervorstellungen nötig, große Teile des Programmes aufwendig ändern zu müssen (zu dieser Problematik siehe auch Bergermeier, J.H., 1984 A, S.189).

Die heute verwendete Kombination von Phasenkonzept und Prototyping ist in Abbildung 20 auf Seite 37 dargestellt.

Da an der Erstellung integrierter Programme mehrere Entwickler gleichzeitig arbeiten, dient jede Phase der Koordination der Einzelprogrammentwicklung und ihrer Prototypen. Diese werden in diesem Konzept nicht nur in Form von lauffähigen Programmen (wie z.B. in der Implementierungsphase) erzeugt, sondern stellen auch Entwurfs- oder Definitionspläne in den Anfangsphasen dar. Die jeweiligen Programmentwickler diskutieren über die Prototypen sofort mit Beratern, Anwendern und Wissenschaftlern und koordinieren ihre weitere Entwicklung dann mit anderen Prototypen.

Durch dieses Verfahren ist sichergestellt, daß

- fachkundige Berater, Spezialisten und zukünftige Anwender sofortigen Einfluß auf die Entwicklung der Programme ausüben können,
- bisherige Kenntnisse, vorhandenes Wissen und Erfahrungen in die Programme einfließen,
- zukünftige Anwender das Programm eher akzeptieren,
- die Koordinierung aller Programme schon in einem frühen Stadium beginnt,
- die nächste Phase erst dann in Angriff genommen wird, wenn die vorherige abgeschlossen wurde.

Der Einsatz des Prototypings erfolgt dabei sowohl erforschend als auch überprüfend (siehe Abbildung 18 auf Seite 34).

Es handelt sich dabei nicht um ein in den USA vielfach anzutreffendes "rapid prototyping", bei dem es nach dem Verfahren "quick and dirty" darum geht, möglichst schnell einen "Wegwerf-Prototypen" zu erzeugen (Kreplin, 1985, S.77).

Ziel ist vielmehr einen zur Weiterentwicklung fähigen Prototypen zu erstellen, der dann in Richtung Endprodukt modifiziert werden kann. Damit ist ein im Gegensatz zum Wegwerf-Prototypen bisher geleisteter Implementierungsaufwand nicht verloren.

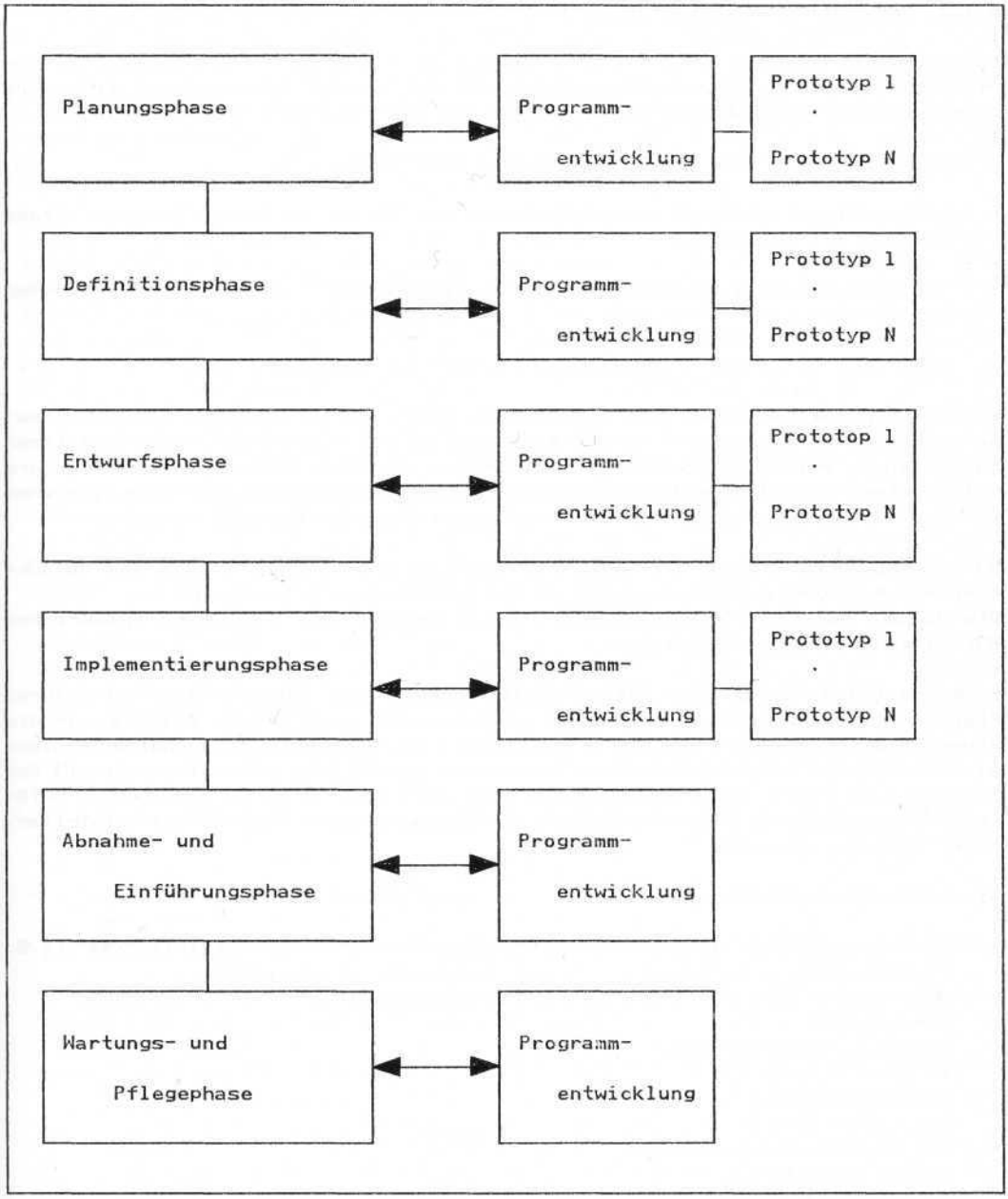

Abbildung 20: Kombination zwischen Phasenkonzept und Prototyping (Erstellen eines Programmprototypen)

4.1.3 Schwerpunkt: Programmentwurf

Sollen unterschiedliche Mitarbeiter in längeren Zeiträumen miteinander in Beziehung stehende Programme entwickeln, so ist dem Programmentwurf besondere Aufmerksamkeit zu schenken. Dieser muß Gewähr dafür bieten, daß

- jedes der Programme leicht lesbar und verständlich ist,
- Programmteile schnell ausgetauscht werden können, ohne das Gesamtprogramm umbauen zu müssen,
- jedes Programm zusammenhängend und in Teilausschnitten effizient zu testen ist,
- das Programm in den grundlegenden Strukturen einheitlich ist.

Die alte, bisweilen immer noch eingesetzte Methode, Programme mittels Programmablaufplänen zu entwerfen[18] (siehe Abbildung 21 auf Seite 40), führt fast immer zu Gebilden, die zwar zunächst ihre Funktion erfüllen können, die aber andere Mitarbeiter, gerade wenn die Programme komplexer sind, kaum noch nachvollziehen können. Sie spiegeln sozusagen den besondern Stil des Erstellers wider.

Um die obigen Forderungen zu erfüllen, ist als Entwicklungsmethode die "strukturierte Programmierung" mit top-down Entwurf besonders geeignet.
Die beiden Begriffe "strukturierte Programmierung" und "top-down" Entwurf bedürfen einer näheren Erklärung.

Dijkstra (1969, S.222-226, 1972, S.1-18) prägte den Begriff des "structured Programming" und verstand darunter verschiedene methodische Ansätze, um die Programmzuverlässigkeit und Durchsichtigkeit zu verbessern. Er hatte festgestellt, daß die Fehlerdichte eines Programmes direkt proportional zur Anzahl der Sprung-Anweisungen (GOTO-Anweisungen) ist und propagierte deswegen eine GOTO-lose Programmierung, die ausschließlich lineare oder D-Strukturen (Stetter, 1984, S.68 f.) enthält.

Diese Strukturen bestehen aus

- den Anweisungen, die den sequentiellen Ablauf nicht beeinflussen (z.B. arithmetische Anweisungen, Eingabe- und Ausgabeanweisungen),
- der Sequenz (der Aufeinanderfolge von Anweisungen),
- der Auswahl
  (Verzweigungsanweisung:
  IF Bedingung A
  THEN Anweisung 1
  ELSE Anweisung 2)
- der Wiederholungsanweisung
  (WHILE Bedingung A
  DO Anweisung 1)

---

[18] Vgl. zur Theorie von Ablaufdiagrammen Bauer (1984, S.393-401).

Tatsächlich konnten Böhm, Jacopini (1966) und Stetter (1984, S.72-73) nachweisen, daß jedes Programm aus linearen Anweisungen aufzubauen ist.

Die Verwendung von Sprungbefehlen ist für das strukturierte Denken in Sequenz, Auswahl und Wiederholung hinderlich, da der Programmentwickler mit ihrer Hilfe ständig "Notbremsen" ziehen kann.

Typisches Beispiel:
Nach der Programmierung eines Algorithmus merkt man am Ende, daß zur Problemlösung ein Rücksprung an den Anfang erforderlich ist. Die Verwendung des Sprungbefehls zwingt dann den Programmierer nicht dazu, den Algorithmus neu, d.h. unter Einbeziehung einer Wiederholung, zu strukturieren.
Kommt diese Situation bei größeren Programmen häufig vor, entwickelt sich etwas, was als "Spaghetti-Code" zu bezeichnen ist. (Eine didaktisch gut aufbereitete Zusammenfassung findet sich bei IBM (1983)).

Das Struktogramm (siehe Abbildung 21 auf Seite 40) ist das geeignete Hilfsmittel für den strukturierten Programmentwurf (zur Technik seiner Anwendung siehe Nassi, Shneiderman, 1973, S.12-26). Es verhindert durch seine Darstellungsweise einerseits Sprungbefehle, erscheint andererseits wesentlich klarer als etwa ein vergleichbares Ablaufschema.

Die top-down Entwicklung oder auch "Schrittweise Verfeinerung" (stepwise refinement) ist als deduktive Methode ein weiteres mit den Methoden der strukturierten Programmierung korrespondierendes Hilfsmittel.[19] Diese Methode löst ein vorhandenes Problem immer von "oben nach unten" oder vom Ganzen zum Detail, d.h. nach einer groben Formulierung des Problems wird dieses in mehreren Schritten soweit verfeinert, bis überschaubare Programmmodule entstanden sind.

Die Zerlegung sollte dabei nach folgendem Prinzip durchgeführt werden (Wirth, 1975):

- Die durch die Zerlegung entstandenen Teile müssen die Lösung des Ganzen ermöglichen.
- Die Zerlegung muß sinnvoll sein.
- Die gewählte Zerlegung sollte sich einer gewünschten Endzerlegung annähern.

Für eine mögliche oder gewünschte Endzerlegung nennt die Literatur (Stetter, 1984, S.136) Größenordnungen von 20 bis 50 Anweisungen je Modul.[20]

---

[19] Ausführliche Beschreibung siehe Wirth (1971, S.221-227), Wirth (1974), Kimm (1979, S.121-129), Dijkstra (1972, S.1-82).

[20] Vgl. zur genauen Bestimmung des Modulbegriffs Dennis (1977 A, S.128-131).

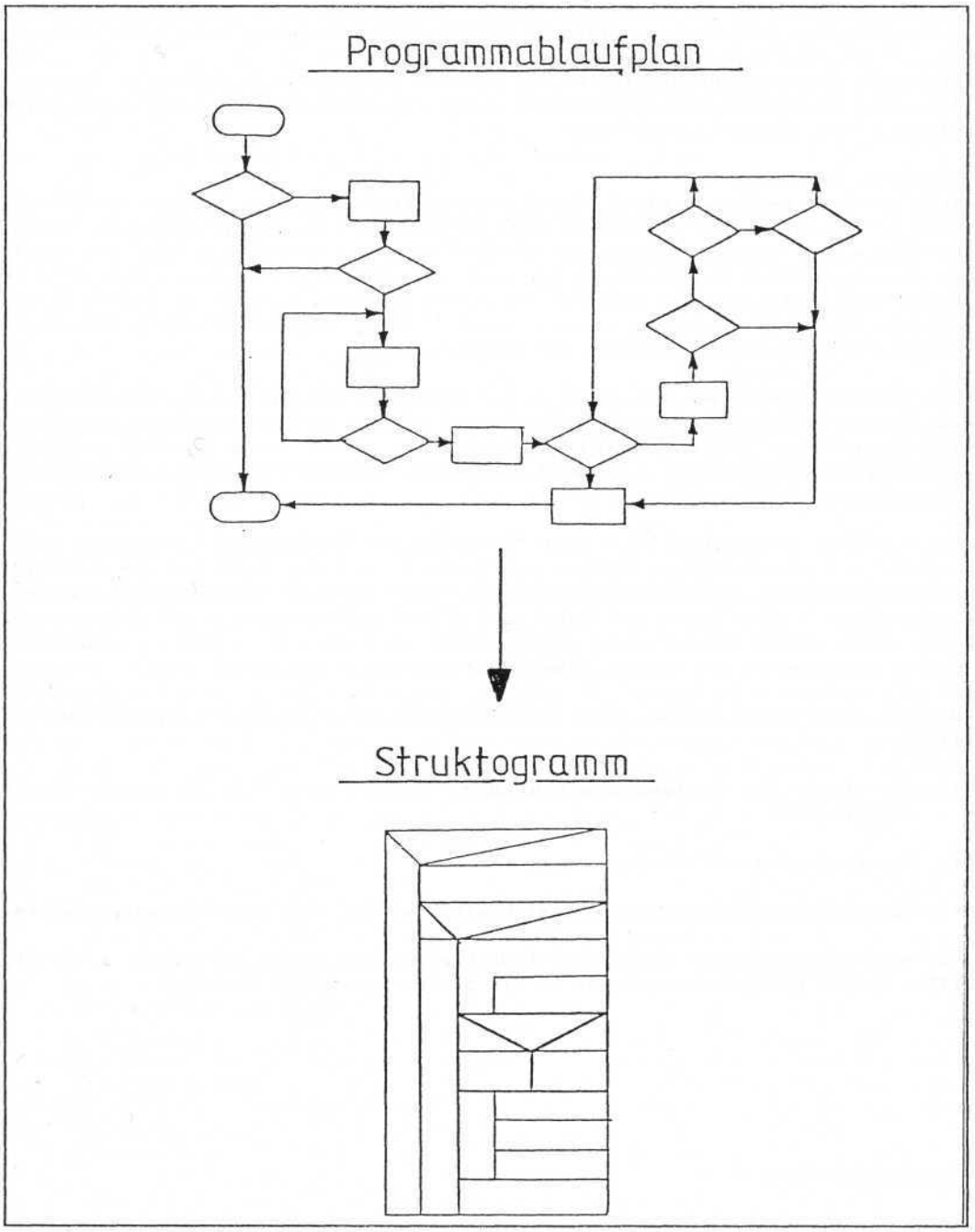

Abbildung 21: Vergleich zwischen einem Programmablaufplan und einem Struktogramm

Das Ergebnis der top-down Methode läßt sich in Form von Baumdiagrammen aufzeigen (siehe dazu Abbildung 22).

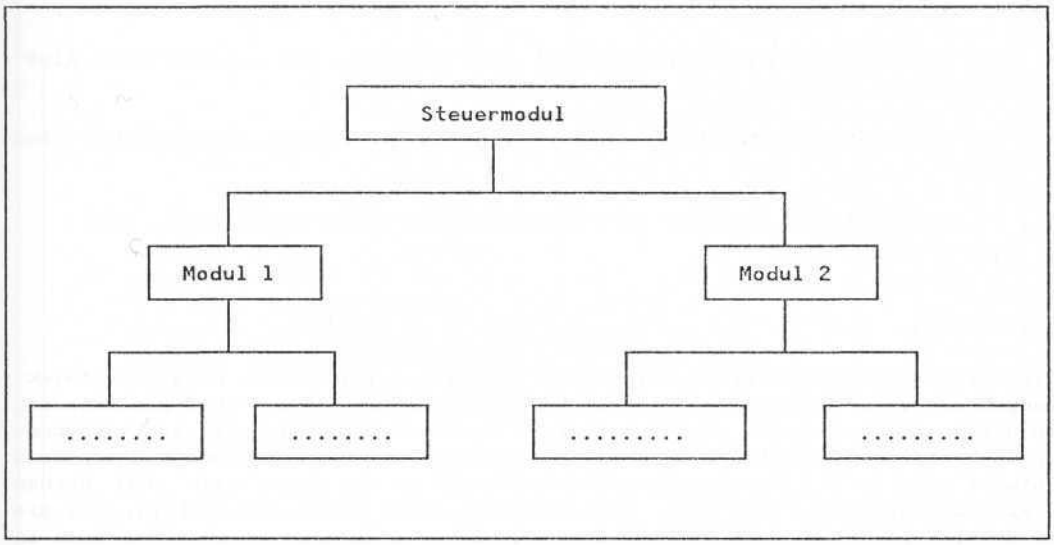

Abbildung 22: Baumdiagramm

Jedes Modul kann dabei wieder in Form eines Struktogrammes (siehe Abbildung 21 auf Seite 40) dargestellt werden. Ein Programmbeispiel in Cobol erläutert kurz dieses Entwicklungskonzept (Abbildung 23 auf Seite 43).

Der "erste Schritt" zeigt den Ablauf des Gesamtprogramms. Die Perform-Anweisungen rufen dabei jeweils Programmabschnitte auf, die spezielle Funktionen, wie die Initialisierung der Maske oder das Löschen des Bildschirms übernehmen. Im "zweiten Schritt" sind diese Abschnitte dargestellt. Auch sie können wiederum weitere Programmabschnitte aufrufen. Diese Methode verfeinert daher über mehrere Stufen die Programmfunktionen bis zu den Basisanweisungen der Programmiersprache.[21]

Bei bestimmten Programmodulen hat es sich als sinnvoll erwiesen, die Jackson-Methode[22] anzuwenden. Sie basiert auf den Prinzipien der "strukturierten Programmierung", gibt aber Entscheidungshilfen darüber, welche Sequenzen, Auswahlanweisungen oder Wiederholungen, bei ihm Sequenz, Selektion und Iteration genannt, für ein Modul notwendig sind und wie die Anweisungen zusammenzufügen sind.

---

21 Eine zusammenfassende Darstellung der verschiedenen Entwicklungsprinzipien für Programme enthält Stahlknecht (1985, S. 230-238).

22 Zur Methode selbst siehe Jackson (1976, 1977, S.577-581, 1983).

Die Grundidee dieser Vorgehensweise besteht darin, die Struktur des Moduls oder Programms aus der Datenstruktur abzuleiten. Zur Unterstützung konzipierte Jackson eine graphische Darstellungsweise, in der die Daten und Kontrollstrukturen als Baumstruktur zu beschreiben sind. Voraussetzung für die Anwendung dieser Methode ist jedoch, daß

- zum einen Datenstrukturen vorliegen (die Anwendung ist bei einfachen Algorithmusberechnungen nicht möglich),
- zum anderen Datenstrukturen in Form von Baumhierarchien darzustellen sind und
- im einfachen Fall Eingabe- und Ausgabestrukturen identisch sind.

Erfahrungen

Bisherige Erfahrungen eines halben Jahres zeigen, daß dieses Programmentwicklungskonzept für Programmiererteams, bestehend aus Doktoranden und Diplomanden, im universitären Bereich einzusetzen ist. Nach der Entwicklung eines Dialogprogramm-Prototypen in Cobol bzw. C mit strukturierter Programmierung und top-down Entwurf sind in der Programmierung Unerfahrene in der Regel nach zwei Wochen Einarbeitungszeit in der Lage, produktiv Programme so zu entwickeln, daß sie einerseits den fachlichen Anforderungen genügen, andererseits ohne Probleme in bestehende Anwendungen eingepaßt werden können.

Selbst bei Diplomanden hat sich mittlerweile herausgestellt, daß in der verfügbaren Bearbeitungszeit von sechs Monaten für die Diplomarbeit ein effektives Arbeiten an Teilproblemen sowohl bezüglich der wissenschaftlichen Vorbereitung als auch von der Umsetzung in Programme her möglich und auch sinnvoll ist.

Diese Erfahrung ist insofern erfreulich zu beurteilen, als die Programmierung an sich damit in unserer Arbeitsgruppe bald zum "normalen" akademischen Handwerkszeug gehören wird und der Schwerpunkt der Arbeit in der inhaltlichen Strukturierung und Aufbereitung bestehender Probleme liegen kann.

Erster Schritt :

```
PROCEDURE DIVISION


RAHMEN.
   PERFORM LOESCHEN BILDSCHIRM.              => Bildschirm löschen
   PERFORM INITIALISIEREN-MASKE.             => Dateizugriff für
   PERFORM AUSGEBEN-MASKE.
   PERFORM MASKE-BEARBEITUNG UNTIL ENDE.              etc.
   PERFORM LOESCHEN-BILDSCHIRM.

RAHMEN-ENDE.
   EXIT PROGRAM.
```

Zweiter Schritt :

```
LöSCHEN BILDSCHIRM.
   CALL XXXXXXX.


INITIALISIEREN-MASKE.
   PERFORM READ-DATEI.
   PERFORM UEBERTRAGEN-DATEN.


AUSGEBEN-MASKE.
   CALL DISPLAY-SECTION.
   CALL XXXXXX ......
      .
      .
      .
```

Abbildung 23: Schrittweise Verfeinerung (Stepwise refinement)

4.2 Softwareergonomie - Anwendergerechte Gestaltung der Benutzeroberfläche landwirtschaftlicher Programme

4.2.1 Begriffsbestimmung

Die Ergonomie bedient sich naturwissenschaftlicher Methoden zur Beschreibung von

- menschlichen Eigenschaften,
- Arbeitsabläufen, -plätzen, -organisationen und -umgebungen,

um durch Anpassung der Arbeit an menschliche Eigenschaften

- vor allem die Gefahr einer organischen Überlastung durch Arbeit so unwahrscheinlich wie möglich zu machen,
- aber auch dem Menschen höchsten Nutzen aus dem Einsatz seiner Fähigkeiten und Fertigkeiten sowie dem Gebrauch von Arbeitsmitteln und Gütern zu gewährleisten (zitiert nach Lauret in Bullinger, 1986).

In der Softwareergonomie, verstanden als aufgaben- und benutzergerechte Gestaltung von Software (Bischoff, 1985, S.143), stellt die Verbindung zwischen Mensch und Computer, auch als Mensch-Computer Schnittstelle oder Benutzeroberfläche bezeichnet, das primäre Erkenntnisobjekt dar.

Sie beschäftigt sich nach Bullinger (1986, S.2), ähnlich wie die Ergonomie, mit der

- Analyse

  menschlicher Eigenschaften,

  von Fähigkeiten, Fertigkeiten bei informatorischen Tätigkeiten,

  von Arbeitsaufgaben,

- der Gestaltung

  der Funktionszuordnung im Mensch-Rechner System,

  der Funktionalität von Mensch-Rechner Schnittstellen,

  der Informationsdarstellung und

  der Dialogführung,

- der Evaluation (Beurteilung)

  der Belastungs- und Beanspruchungssituation,

  der Leistung unter Berücksichtigung von Fehlerverhalten und

  der Akzeptanz beim Endbenutzer.

Die Softwareergonomie tritt als neues Wissenschaftsgebiet der Informatik erst in den letzten Jahren zunehmend in den Vordergrund. In den Anfängen der EDV, beim Einsatz sogenannter Batch-Programme (Programme, die nach Anstoß durch den Benutzer dann unabhängig von ihm nach einer gewissen Zeit Ergebnisse liefern), waren Gedanken zur Gestaltung der Mensch-Rechner Schnittstelle überflüssig.

Erst nach der zunehmenden Entwicklung von Dialogprogrammen mit beiderseitiger Kommunikation zwischen Rechner und Mensch (Benutzer) rückt die Gestaltung des Kommunikationsablaufes zunehmend in den Vordergrund, zumal sich dadurch ein neues Kommunikationsmedium zu den traditionellen Medien gesellt hat: der computergesteuerte Bildschirm.

Er erlaubt im Gegensatz zu anderen Medien, wie Zeitung, Radio oder Fernsehen, eine Zweiweg-Kommunikation zwischen dem Autor des Dialogs und seinem Benutzer. Über sie kann der Ablauf des computergestützten Dialogs vom Benutzer in wesentlichen Zügen bestimmt werden (falls er richtig gestaltet und programmiert wurde).

- "Bitte das nächste Bild",
- "Vorige Seite anzeigen",
- "Zurück zur Auswahlseite" oder
- "Weitere Schlagdaten anzeigen"

sind beispielsweise die Möglichkeiten, mit der der Benutzer sich dem Mensch-Rechner Dialog gegenüber bemerkbar machen kann.

Dabei sind die Mehrzahl der heute angebotenen Dialoge besonders für den unerfahrenen Benutzer mit Problemen verbunden. Nievergelt (1983 A, 1983 B, S.4) hat seine Hauptschwierigkeiten in Form der folgenden vier Fragen zusammengefaßt:

- Wo bin ich?

  Der Bildschirm sieht nicht wie erwartet aus.

- Was kann ich hier tun?

  Die Menge der aktiven Befehle ist unklar.

- Wie kam ich hierhin?

  Vielleicht wurde eine falsche Taste gedrückt.

- Wo kann ich hin und wie komme ich dorthin?

  Die Möglichkeiten des Systems sollen erforscht werden.

Für den Entwurf von interaktiven Systemen forderte er, daß sie die jederzeitige Beantwortung dieser Fragen durch den Benutzer sicherstellen sollen.

Innerhalb der Programmentwicklung für die Landwirtschaft hat man der Softwareergonomie bisher keine Bedeutung geschenkt.[23] Sie ist aber auch in diesem Bereich besonders wichtig, da die Effektivität zukünftiger Programme für den Benutzer von der an die landwirtschaftlichen Verhältnisse angepaßten Gestaltung des Dialogs abhängig ist.

Daher analysieren die folgenden Kapitel drei grundlegende Faktoren, die die Schnittstelle beeinflussen und zeigen Erfahrungen und Erkenntnisse für ihre Optimierung auf.

1. Welches Profil hat der landwirtschaftliche EDV-Benutzer?

2. Wie müssen Bildschirmmasken (Bildschirmseiten zur Ein- und Ausgabe von Daten und Informationen) gestaltet werden?

3. Welche für die Landwirtschaft geeigneten Dialogformen gibt es, wie und wann sind sie einzusetzen?

4.2.2 Benutzerprofil: der Landwirt und Berater als Programmanwender

Das Benutzerprofil kennzeichnet den Benutzer in seinen speziellen Fähigkeiten und Erfahrungen im Umgang mit Dialogprogrammen.

Prinzipiell ist es nach Schmitt (1983, S.124-125) sinnvoll, trainierte, erfahrene Benutzer von untrainierten, seltenen Benutzern zu unterscheiden.

Untrainierte Benutzer haben

- wenig oder keine Erfahrung mit Dialogsystemen,
- geringe Kontaktzeiten mit Dialogsystemen,
- beschäftigen sich nur einen sehr geringen Teil ihrer Zeit mit Computer-Dialogen.

Trainierte Benutzer

- haben erhebliche Erfahrungen mit Dialogsystemen,
- beherrschen zumindest eine nicht zu einfache Dialogsprache vollständig (Lotus, DBase etc.),
- sind in der Terminalbedienung geübt.

Im landwirtschaftlichen Bereich tätige EDV-Benutzer (praktische Landwirte, aber auch Mitarbeiter in den öffentlichen Verwaltungen, in Industrie und Wissenschaft) zählen in den meisten Fällen zu den untrainierten Benutzern. Dies liegt zum einen an der bisher noch relativ geringen Verbreitung des EDV-Einsatzes, zum anderen in der Tatsache begründet, daß die meisten heute in und für die Praxis eingesetzten Programme nicht kontinuierlich während des gesamten Jahres zu benutzen sind. Beispiele sind die EDV-Düngeplanung, Programme zur Prognose von Pflanzenkrankheiten und Programme zur Berechnung von optimalen Futtermischungen.

---

[23] Vgl. zu ersten allgemein gehaltenen Ansätzen Blokker (1986, S.67-68).

Dabei ist zu beachten, daß für die Berater und Landwirte die EDV nur ein Hilfsmittel von vielen in der Betriebsführung ist.

Für Dialogschnittstellen, die für untrainierte Benutzer geeignet sind, ergeben sich besondere Anforderungen:

- Anbiete-Technik

  Werden vom Benutzer Eingaben erwartet, so kann das System ihn dadurch unterstützen, daß eine Liste zulässiger Eingaben eingeblendet wird. Dies zeigt Abbildung 24 und Abbildung 25 auf Seite 48 an Hand zweier Beispiele. In der Abbildung 24 kann der Benutzer eine Fruchtart erfassen. Das Programm listet ihm zur Hilfe und Orientierung dazu bereits gespeicherte Fruchtarten auf.

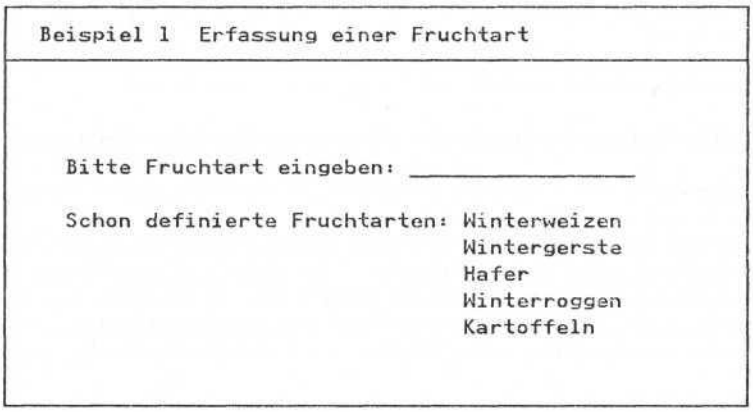

Abbildung 24: Anbiete-Technik am Beispiel der Erfassung von Fruchtarten

Diese Anbiete-Technik kann bei mehreren zu erfassenden Daten auf einer Bildschirmseite auch dynamisch über ein Fenster (Window) realisiert werden. Das Bildschirmfenster (in Abbildung 25 auf Seite 48 gezeigt) öffnet sich erst dann, wenn der Benutzer mit seinem Cursor in das Feld "Hauptfrucht" springt und schließt sich, wenn er dieses wieder verläßt.[24]

---

[24] Vgl. als Beispiel für den praktischen Einsatz dieser Technik in einem landwirtschaftlichen Programm auch Leeuwen-Pannekoek (1986, S.84).

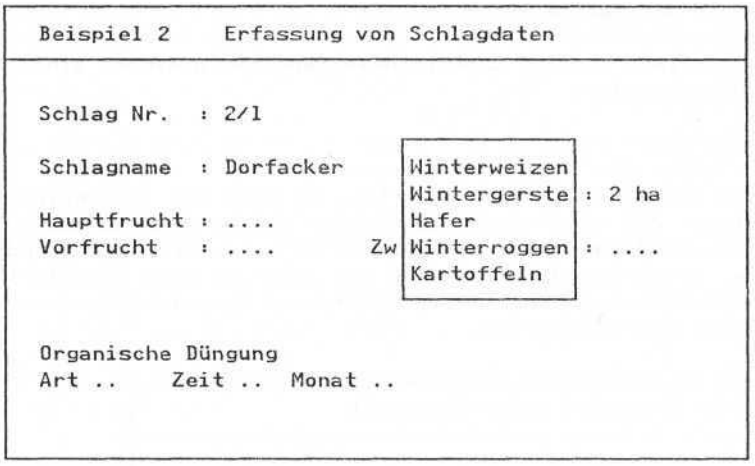

Abbildung 25: Anbiete-Technik durch den Einsatz von Fenstern (Windows)

- Zustandsanzeigen

  Verunsicherung tritt beim Unerfahrenen insbesondere dann auf, wenn nach seiner Reaktion, z.B. durch Betätigen der Datenfreigabe-Taste das Programm zwar wie gewünscht im Ablauf fortfährt, dies aber für ihn nicht sichtbar ist (z.B. längeres Durchsuchen von Datenbeständen). Hier ist es sinnvoll, vom Programm eine System- oder Zustandsmeldung, wie

Befehl akzeptiert, Programm arbeitet.

  auf dem Bildschirm einzublenden.

- Vermeidung von Spezialtasten

  Die Bedienung eines Programmes in den Hauptfunktionen sollte über Datenfreigabe und mit durch Cursortasten verschiebbaren Lichtbalken erfolgen. Dies ist für ungeübte Benutzer wesentlich einfacher und klarer als das Suchen und Finden von Spezialtasten wie "F1-F10, CRTL F1-F10, ALT F1-F10" auf unter Umständen mit dem Rechner sich ändernden Tastaturen.

- Programmhilfen bei Fehlern

  Fehler beim Benutzer gehören zur Normalität, fehlende Programmhilfen auch!

  Dabei zählen gerade sie, richtig eingesetzt, zu den entscheidenden Vorzügen für den Ungeübten. Als Hilfen sind Beispiele besser geeignet als abstrakte formale Beschreibungen. Abbildung 26 auf Seite 49 zeigt dies an einem Beispiel.

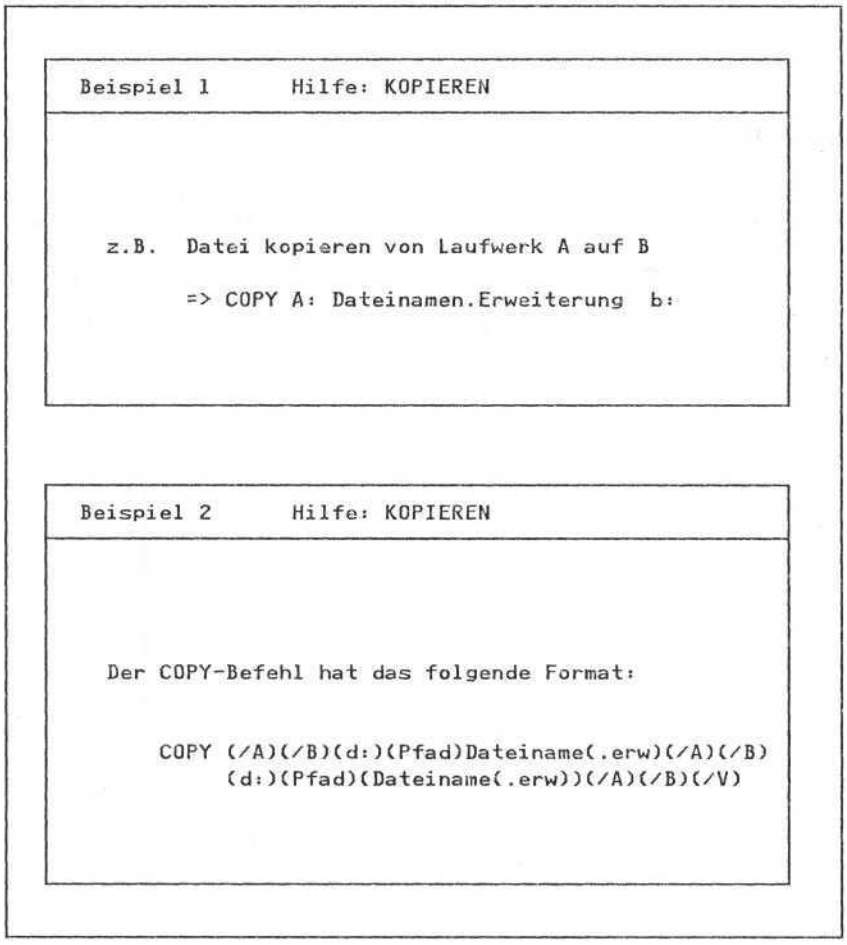

Abbildung 26: Programmhilfen

Während diese Anforderungen für Programme, die ungeübte Benutzer verwenden, im Allgemeinen gelten, beschreibt das Kapitel "Eignung verschiedener Dialogtechniken für den landwirtschaftlichen Einsatz" auf Seite 62 die Anforderungen speziell für den Pflanzenbau an Hand praktischer Beispiele aus bestehenden Dialogprogrammen.

4.2.3 Gestaltung von Bildschirmseiten

4.2.3.1 Grundsätze der Gestalttheorie

Die Gestaltung von Bildschirmseiten ist für die Optimierung der Benutzeroberfläche bedeutend.

Eine gute Maskengestaltung - bei dieser benötigt man im Mittel einen geringeren geistigen und manuellen Aufwand für die Lösung einer Aufgabe im Dialog als bei schlecht gestalteten Masken (Bertin, 1982) - ist kein Zufall, sondern kann entsprechend den Gesetzesmäßigkeiten des menschlichen Wahrnehmungsprozesses bewußt gestaltet werden (Lindsay, 1977).
Diese Wahrnehmungsprinzipien beruhen auf den Grundsätzen der Gestalttheorie (Ziegler, 1985, S.55).

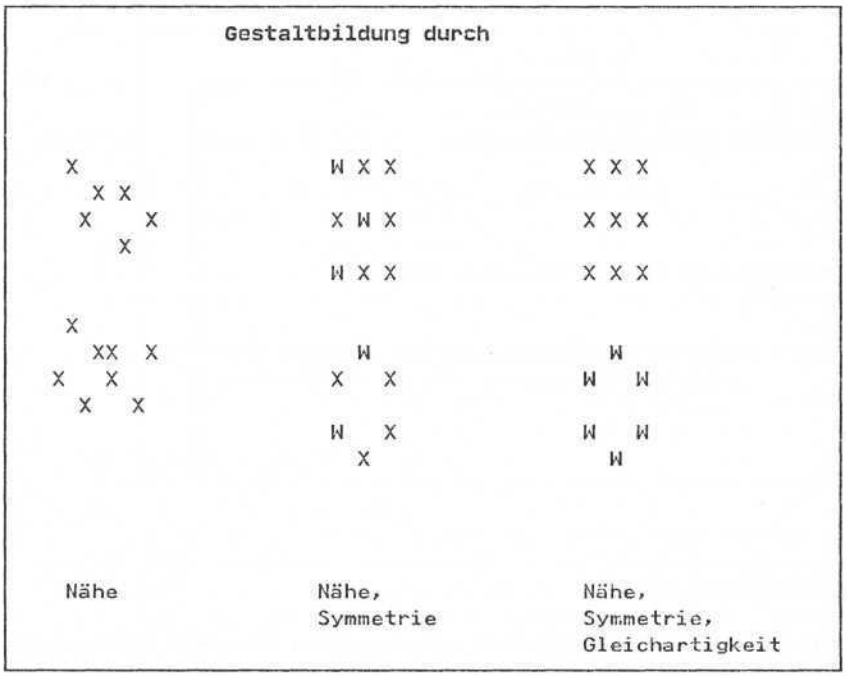

Abbildung 27: Gestaltbildung nach dem Prägnanzsatz

Aus der Gestalttheorie werden nachfolgend zwei wichtige Annahmen herangezogen (Wertheimer, 1923, S.301-350):

1. Prägnanzsatz

   Bei der Bildung von Gruppen im Sehfeld oder Ausbildung von Figuren setzt sich immer eine Strukturierung durch, die sich nach ausgezeichneten Ganzeigenschaften, wie Einfachheit und Symmetrie bestimmt.

Abbildung 27 verdeutlicht den Zusammenhang. Es bilden sich immer Gruppen von Buchstaben heraus, die nahe beieinander liegen. Dieser Effekt verstärkt sich noch durch symmetrische Anordnungen der Elemente bzw. durch die Gleichartigkeit der Elemente in den Gruppen.

Abbildung 28 und Abbildung 29 auf Seite 52 zeigen zur Verdeutlichung dieses Satzes zwei Beispiele in Form von Bildschirmmasken bestehender Programme. Das Bildschirm-Design des ersten Beispiels in Abbildung 28 ist unglücklich gewählt. Die Maske ist kopflastig und nicht klar strukturiert. Die Augen des Benutzers irren zwangsläufig über die Maske, um für sein Verständnis zunächst Ordnung und Übersicht zu gewinnen.

```
BALIS - TU München/LE Ackerbau
++++++++++++++++++++++++++++++++++++++
 Erntejahr : 19..    Schlagnummer: ../.
 Schlagbezeichnung  : ..............
 Fläche : ....  ha
 Hauptfrucht: .... Ertragsziel: ......
 Bodenuntersuchung : 19..  CAL .  DL .
 Bodenart : ..      Krumentiefe :  .. cm
 Ackerzahl: ..      Humusgehalt : .... %
 P2O5 : ..  K2O: .. MGO: .. mg/100 g
 pH-Wert: ...        Nmin: ... kg/ha

 Vorfrucht          : ..............

 Vorvorfrucht       : ..............

 Vorvorvorfrucht    : ..............

++++++ nächster Schlag ../. Auswahl =>.
 ....................................
 ....................................
```

Abbildung 28: Gestaltung einer Bildschirmmaske der Schlagkartei in Bildschirmtext

Die Bildschirmmaske in Abbildung 29 auf Seite 52 (Pohlmann, 1987 C) zeigt dagegen eine klare und übersichtliche Struktur. Sie kommt vor allen Dingen dadurch zustande, daß zwischen den Eingabefeldern symmetrische Beziehungen (spaltenweise Anordnungen) bestehen und die Maskenstruktur logische Abschnitte von Datengruppen unterstützt. Mit zwei bis drei Blicken kann der Benutzer den Inhalt der Maske erfassen.

```
| Dungplan | Schlagdaten | 14.06.87 |

Schlagnummer ../.
Schlagname  .....                    Größe ..... ha
Hauptfrucht  ....                 Ertragsziel ..... dt /ha


Ernterückstände/Zwischenfrucht
Art ....  Menge .... dt/ha          Art ....  Menge ..... dt /ha


Wirtschaftsdünger
Art ....  Menge .... dt/ha  Monat ..  Art ....  Menge .... dt/ ha  Monat ..
    ....        .... dt/ha        ..      ....        .... dt/ ha  Monat ..


Bodenuntersuchung (nach CAL)
Bodenart ..  N-Faktor ...  pH-Wert ...     CaO(kg/ha) ...  P2O5 ... K2O ...
oder Bedarfswerte (z.B. nach EUF) in kg/ha          N ...  P2O5 ... K2O ...

| ESC Auswahlmenü | <----¬ Daten erfassen |
```

Abbildung 29: Gestaltung einer Bildschirmmaske der Düngeplanung am Personal-Computer

2. Gesetz der guten Gestalt

Nach Neumann (1976) umfaßt dieses Gesetz mehrere "Faktoren". Wichtig ist dabei der "Faktor der Gleichartigkeit". Er sagt aus, daß gleiche Elemente zu Gruppen zusammengefaßt werden.

AARTS, H.F.M., HINDRIKS, K. DE VISSER, C.L.M., 1986
Automatisierungsmöglichkeiten in der Anbauberatung
in: Agrarinformatik Bd.10, S.71-79
Hrsg.: L. Reiner, H. Geidel
Stuttgart: Ulmer, 1986

BALZERT, H., 1982
Die Entwicklung von Software-Systemen
in: Reihe Informatik /34
Mannheim: B.I.-Wissenschaftsverlag

BARTELS, M., 1982
Untersuchungen zur Darstellung von Menüs
Stuttgart: Studienarbeit Uni Stuttgart
Institut für Industrielle Fertigung

BAUER, F.L., WöSSNER, H., 1984
Algorithmische Sprache und Programmentwicklung, 2.Aufl.
Berlin: Springer, 1984

Abbildung 30: Faktor der Gleichartigkeit (Beispiel Literaturangabe)

Abbildung 30 zeigt den visuellen Effekt dieses Faktors in Form einer Literaturzusammenstellung. Als Gruppen bilden sich dabei die einzelnen Literaturangaben heraus, da sie alle gleichartig aufgebaut sind.

Des Weiteren umfaßt dieses Gesetz in Analogie zum Prägnanzsatz den "Faktor der Nähe" und den "Faktor der Symmetrie". Diese beiden Faktoren drücken nichts anderes aus, als daß beieinander liegende Elemente oder symmetrische Elemente sich zu Gruppen zusammenschließen (siehe auch dazu Abbildung 27 auf Seite 50). Abbildung 31 auf Seite 54 zeigt dies am Beispiel zweier Bildschirmseiten. Aus der Anordung, d.h. der Nähe der Informationen zueinander, ergibt sich im ersten Bild eine spaltenweise, im zweiten Bild dagegen eine zeilenweise Strukturierung.

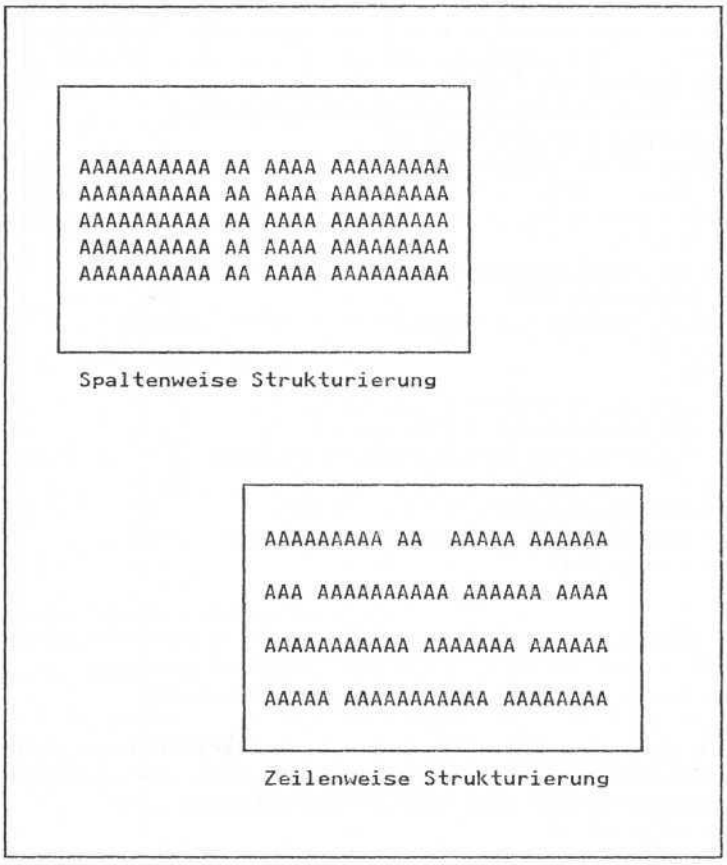

Abbildung 31: Faktor der Nähe: spaltenweise und zeilenweise Strukturierung

Weitere grundsätzliche Hinweise zur Gestaltung von Texten findet man bei Langer (1974).
Diese Grundsätze besagen zum schnelleren und besseren Textverständnis:

- Satzbau und Wortwahl sollen einfach sein, da Einfachheit sehr stark mit Verständlichkeit korreliert.

- Sätze sind folgerichtig aufeinander zu beziehen, um die Information in einer sinnvollen Reihenfolge anzubieten. Eine optimale Gliederung (Verwendung von Absätzen, Überschriften und Hervorhebungen) kann dabei das Verständnis des Inhalts unterstützen.

- Texte sind weder zu knapp noch zu weitschweifig zu fassen. Das Optimum in Bezug auf gute Verständlichkeit liegt in der Mitte.

Das erste Beispiel in Abbildung 32 auf Seite 55 zeigt, wie der Text in einer Maske nicht gestaltet sein sollte. Eng aneinandergehängt, ohne Strukturierung erschwert er das Lesen und damit das Verständnis. Der Benutzer benötigt in dieser

Maske zu viel Zeit, um das Programm weiter bedienen zu können. Anders dagegen im zweiten Beispiel: Hier erkennt er mit einem Blick, daß er nach der Eingabe der Schlagnummer und Betätigen der Datenfreigabe-Taste (Symboldarstellung) einen Schlag erfassen kann. Schon eingegebene Schläge mit zusätzlichen Angaben sind zur Orientierung im unteren Teil der Maske eingeblendet.

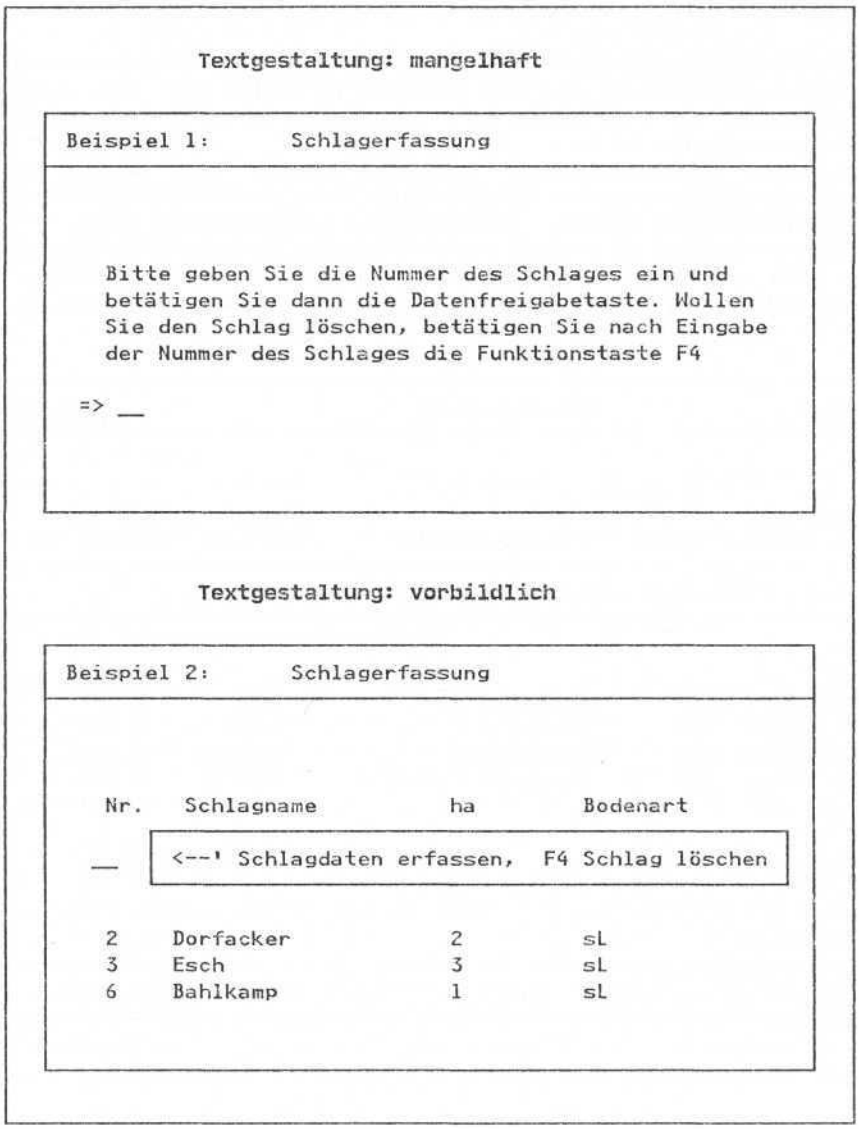

Abbildung 32: Beispiele der Gestaltung von Texten in Bildschirmmasken

4.2.3.2 Praktische Vorgehensweise zur Maskenerstellung

Um die Gestaltungsprinzipien systematisch bei der Erstellung der Masken berücksichtigen zu können, ist folgende Vorgehensweise bei der Erstellung sinnvoll (Siemens, 1983).

1. Aufgaben-Gestaltung (Festlegen des Maskeninhalts)

   Dieser Arbeitsschritt trägt zunächst die Arbeitsinformationen zusammen, d.h. die Informationen, die das Programm im Dialog ein- oder ausgeben soll. Er analysiert dann, wie diese am günstigsten auf unterschiedliche Masken zu verteilen sind.
   Hinsichtlich pflanzenbaulicher Programme ergeben sich häufig unterschiedliche Gliederungsmöglichkeiten, die jedoch unter Inkaufnahme einer Mehrfachdarstellung gleicher Informationen zu kombinieren sind.
   Die Datenerfassung innerhalb eines Schlagkartei-Programms könnte zum einen nach unterschiedlichen Arbeitsmaßnahmen wie Saat, Düngung und Pflanzenschutz gegliedert werden. Abbildung 33 und und Abbildung 34 auf Seite 57 zeigen dazu jeweils ein Beispiel (Pohlmann, 1987 D).

Schlagkartei		Pflanzenschutz			14.06.87
Schläge	Zeitpunkt Sta Datum	Frucht	Pflanzenschutz	Menge kg l/ha	Schlepper Geräte
__/_	<---' Maßnahme erfassen, F4 Maßnahme löschen				
1/0	20 07.04	WW	Herbizid-X	4,50	5 10 2
2/0	20 07.04	WW	Herbizid-X	4,50	5 10 3
1/0	23 27.04	WW	Fungizid-X	0,50	5 10 2
2/0	23 27.04	WW	Fungizid-X	0,50	5 10 3
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
ESC Auswahlmenü					

Abbildung 33: Erfassen von Pflanzenschutzmaßnahmen (PC-Schlagkartei)

Schlagkartei	Mineralische Düngung	14.06.87

Schläge	Zeitpunkt Sta	Datum	Frucht	Mineraldünger	Menge kg l/ha	Schlepper Geräte		Akh
__/_	<---' Maßnahme erfassen, F4 Maßnahme löschen							
1/0	13	23.03	WW	N-Teilgabe	50,0	5	12	3
2/0	13	23.03	WW	N-Teilgabe	50,0	5	12	4
1/0	55	19.06	WW	N-Teilgabe	60,0	5	12	3
2/0	55	19.06	WW	N-Teilgabe	60,0	5	12	4
.		.	.	.	.			
.		.	.	.	.			
.		.	.	.	.			
.		.	.	.	.			

ESC Auswahlmenü

Abbildung 34: Erfassen von Düngungsmaßnahmen (PC-Schlagkartei)

Diese Gliederung der Datenerfassung gibt dem Benutzer jederzeit einen nach unterschiedlichen Arbeitsmaßnahmen geordneten Überblick aller bisher durchgeführten Arbeiten.

Häufig kombiniert aber der Landwirt mehrere Maßnahmen, wie Pflanzenschutz und Düngung, in einem Arbeitsgang und trägt sie in seinem Notizbuch schlagbezogen ein. Zur Eingabe der Daten ins Programm müßte er sie jedoch nach Arbeitsmaßnahmen getrennt (Düngung, Pflanzenschutz etc.) ordnen.

Mit anderen Worten:
Das Programm richtet sich nicht nach den speziellen Anforderungen des Benutzers, sondern der Benutzer muß sich an die nicht praxisgerechte Programmvorgabe halten!

Die folgende alternative Erfassungsmöglichkeit kann diesen Nachteil beseitigen (Abbildung 35 auf Seite 58, Pohlmann (1987 D)).

Schlagkartei	Arbeitsmaßnahmen	14.06.87

Aussaatdaten | Org.Düngung | Min.Düngung | Pflanzenschutz | Maschineneinsatz

Nr	Schlagname	Zeitpunkt	Frucht	Maßnahme
1/0	Dorfacker	10.10.86		Min. Düngung
2/0	Kreuzung	10.10.86		Min. Düngung
1/0	Dorfacker	15.10.86	Winterweizen	Saat
2/0	Kreuzung	15.10.86	Winterweizen	Saat
.	.	.	.	.
.	.	.	.	.

ESC Auswahlmenü

Abbildung 35: Erfassen von Arbeitsmaßnahmen (PC-Schlagkartei)

Es erscheint auf dem Bildschirm ein Arbeitsblatt, über das der Landwirt als Programmbenutzer alle Arbeitsmaßnahmen ungeordnet, d.h. so, wie sie tagsüber angefallen sind, eintragen kann. Er wählt zunächst die zu erfassende Arbeitsmaßnahme über einen horizontal verschiebbaren Lichtbalken aus und erhält dann auf dem Bildschirm die entsprechende Eingabezeile mit dazugehörender Legende.

Das Programm übernimmt bei dieser Art der Gliederung die inhaltliche Ordnung der eingegebenen Maßnahmen.
Unter Inkaufnahme einer Redundanz, d.h. einer mehrfachen oder zumindest doppelt vorhandenen Darstellungsweise gleicher Daten, kann die inhaltliche Gliederung der Masken an den speziellen Betriebsablauf angepaßt werden.

2. Funktionale Gestaltung

Nachdem im ersten Arbeitsschritt der Inhalt der jeweiligen Masken festgelegt ist, beschäftigt sich der zweite mit der funktionalen Gestaltung. Hier ist zu entscheiden, wie der Benutzer den Dialog mit dem Programm führen soll. Neben der Programmsteuerung über Auswahl-Menüs für Unerfahrene bietet sich bei Erfahrenen auch die gezielte Auswahl über die Eingabe eines Kommandos an.

Bei vielen Programmen ist eine Ebene von Auswahlmenüs nicht ausreichend. Aus Gründen der Übersichtlichkeit sollte der Programmentwickler allerdings nicht mehr als zwei Ebenen einsetzen.

"Neben-Menüs" in Auswahlseiten sind in der Lage, sonst notwendige längere Einträge zu vermeiden. Abbildung 36 zeigt dazu ein Beispiel (Pohlmann, 1987 D).

Die Breite eines Menüs sollte dabei wiederum 5 bis 7 Einträge - das entspricht etwa der Kapazität des menschlichen Kurzzeitgedächtnisses - nicht überschreiten (Bartels, 1982).

CIPP-PC	Auswahlmenü		14.06.87
Optimierung	Schlagkartei	Prognose	Simulation
Düngeplanung	Schlagdaten	Septoria	Weizen
Herbizidauswahl	Bod.untersuchung	Epipre	Gerste
Tankmischungen (Pflanzenschutz)	Zwischenfrüchte		
	Ernte		
	Best.beobachtung		
	Arbeitsmaßnahmen		
ESC Ende			

Abbildung 36: Alternative Strukturierung von Auswahlmenüs

3. Visuelle Gestaltung (Abbildung 37 auf Seite 60)

Bei der Gestaltung von Bildschirmmasken ist es sinnvoll, den verschiedenen Informationsklassen feste Bereiche auf dem Bildschirm zuzuordnen (so auch Ziegler, 1985, S.60).

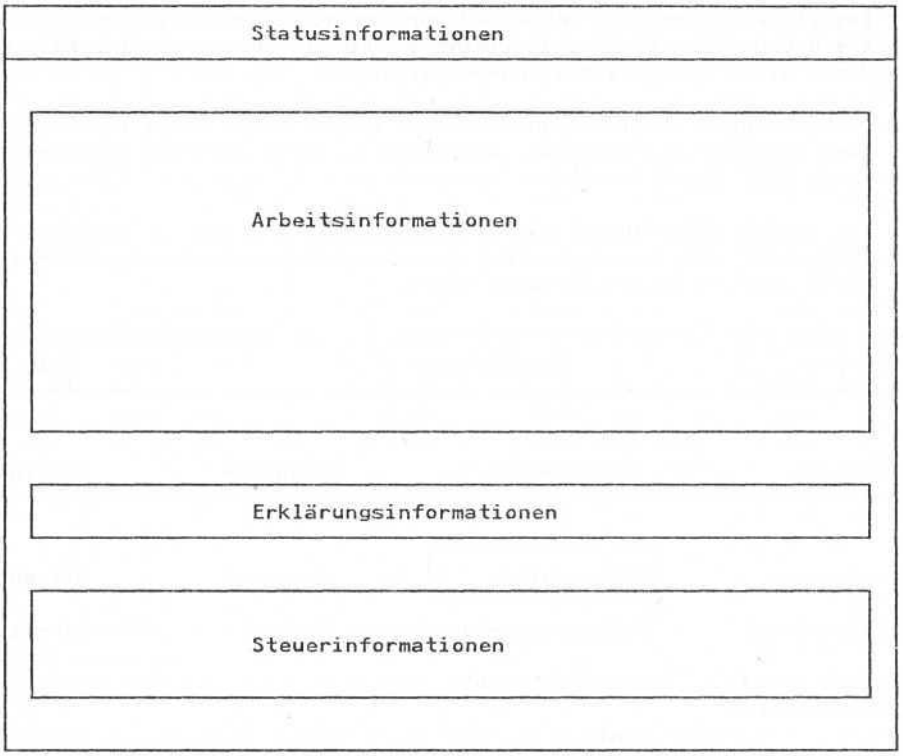

Abbildung 37: Mögliche Grob-Gestaltung einer Bildschirmmaske

Die Statusinformationen bilden den "Kopf" der Maske und geben dem Benutzer Auskunft darüber, an welcher Stelle er sich im Programm befindet. Die Arbeitsinformationen stellen den Bildschirmbereich dar, der die zu bearbeitenden Daten umfaßt (Abbildung 37).
Zu den Erklärungsinformationen zählen Fehlermeldungen und Benutzerhilfen des Programmes.
Die Steuerinformationen geben Auskunft über vorhandene Möglichkeiten zur Steuerung des Programmes (z.B. über Funktionstasten).
Die Abbildung 37 zeigt eine mögliche Anordnung der Informationsklassen. Wichtiger allerdings ist es, eine einmal gewählte Bildschirmaufteilung im Gesamtprogramm beizubehalten. Ein ständiger Wechsel innerhalb der Maske führt zu Irritationen und damit zu Bedienungsschwierigkeiten beim Benutzer.

Eine Fein-Gestaltung kann den Bereich der Arbeitsinformationen nach der Erstellung der Grob-Gestaltung weiter untergliedern (Abbildung 38 auf Seite 61).

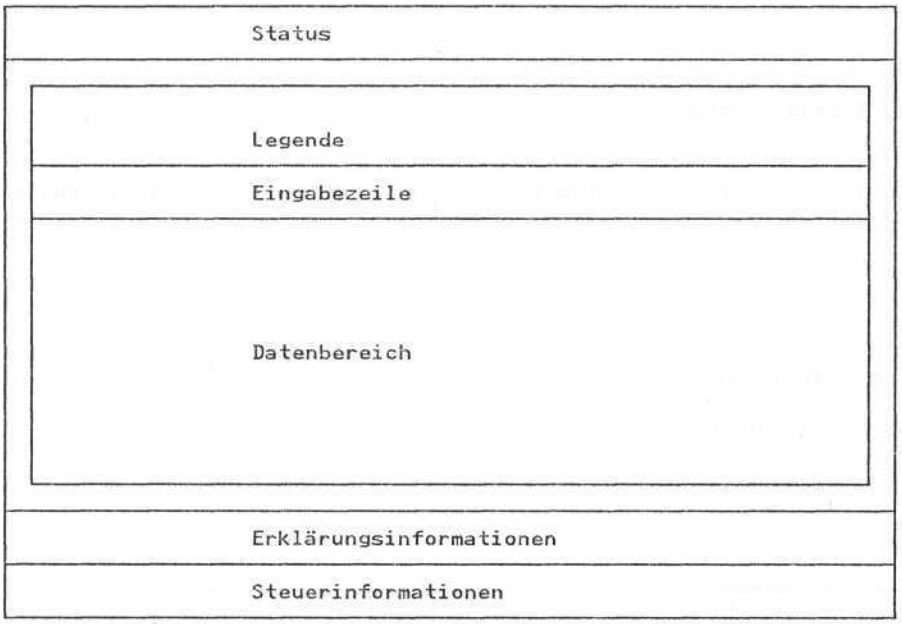

Abbildung 38: Mögliche Fein-Gestaltung einer Bildschirmmaske

Der Bereich der Arbeitsinformationen besteht in diesem Beispiel aus einer Legende für die einzugebenden Daten, einer Eingabezeile und einem Datenbereich.

Bei der systematischen Anwendung dieser drei Prinzipien der Maskenerstellung ist eine effektive Entwicklung der Benutzeroberfläche gewährleistet.[25]

---

[25] Eine umfangreiche Sammlung von Gestaltungsrichtlinien für Dialogsysteme mit Regeln, Begründungen und Gestaltungsbeispielen enthält Smith (1984).

4.2.4 Eignung verschiedener Dialogtechniken für den landwirtschaftlichen Einsatz

4.2.4.1 Begriffsbestimmung

Interaktive Systeme (Dialogsysteme) unterscheiden sich von Stapelverarbeitungssystemen grundsätzlich durch die Möglichkeit eines wechselseitigen Austausches von Nachrichten zwischen Benutzer und System.

Daher ist in Dialogsystemen eine zweiseitige Kommunikation zwischen Benutzer und System realisiert.

Je nachdem, welcher der beiden "Partner" die dominante Rolle oder Führung im Dialog übernimmt, unterscheidet Fähnrich (1985, S.30)

- rechnergeführte oder
- benutzergeführte

Dialogtechniken.

Benutzergeführte Dialogtechniken	Rechnergeführte Dialogtechniken
+ hohe Flexibilität in der Dialogsteuerung	+ gute Benutzerführung
+ hohe Flexibilität durch genaue und effiziente Dialogsprache	+ wenig Einarbeitungszeit bzw. Ausbildung
+ geringere Rechnerkosten	+ wenig Fehler bzw. Fehlerquellen
- höherer Schwierigkeitsgrad	- Dialog häufig umfangreicher und dadurch evtl. ineffizienter
- längere Ausbildungszeit	- geringe Flexibilität bei der Programmsteuerung
- höheres Fehlerrisiko	- geringe Handlungsautonomie des Benutzers

Abbildung 39: Vor- und Nachteile unterschiedlicher Dialogtechniken

Beide weisen Vor- und Nachteile auf, die in Abbildung 39 nach Carlson (1983, S.65-87) und Fähnrich (1985, S.33) zusammengestellt sind.

Vergleicht man jeweils die Vor- und Nachteile beider Dialogtechniken, so zeigt sich, daß rechnergeführte Techniken für den untrainierten Benutzer geeignet sind, während benutzergeführte Techniken vom trainierten, erfahrenen Anwender erfolgreich einzusetzen sind.

Die gute und sichere Benutzerführung mit geringen Fehlermöglichkeiten ist beim rechnergeführten Dialog für den untrainierten Benutzer wesentlich wichtiger als ein etwas längerer und unflexiblerer Dialog. Im Gegensatz dazu beherrscht der trainierte Benutzer sehr schnell ein Programm und kann die durch die benutzergeführten Dialogtechniken gebotene Flexibilität der Programmsteuerung effizient einsetzen.
Der zunächst höhere Schwierigkeitsgrad und Einarbeitungsaufwand zum Bedienen des Programms spielt bei einer späteren häufigen Benutzung des Programms eine unwesentliche Rolle.

Da innerhalb der Landwirtschaft vorwiegend der untrainierte Benutzer anzutreffen ist, wie im Kapitel "Benutzerprofil: der Landwirt und Berater als Programmanwender" auf Seite 46 beschrieben ist, eignen sich somit für ihn rechnergeführte Dialogtechniken am besten.

Mittlerweile haben sich verschiedene standardisierte Dialogformen herausgebildet, die den zuvor gegeneinander abgegrenzten Dialogtechniken zugeordnet werden können (Abbildung 40).

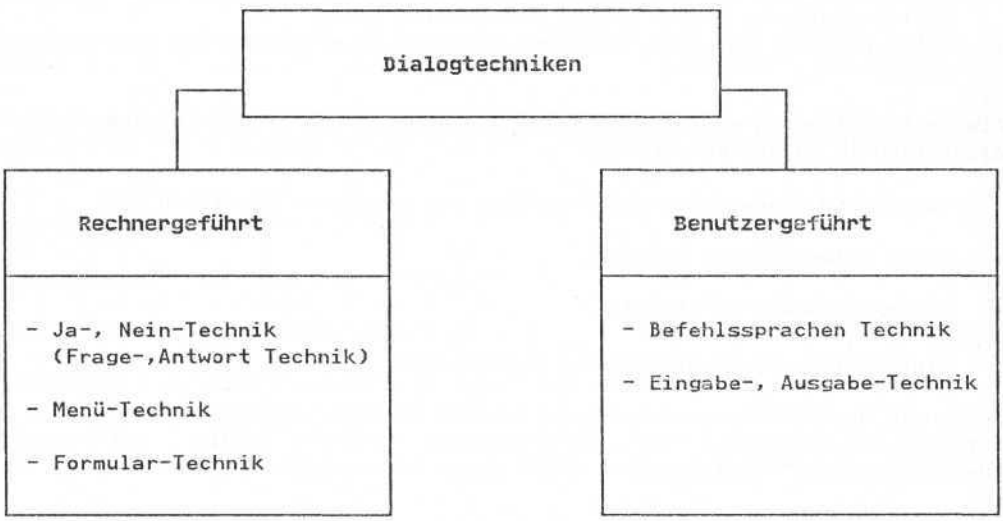

Abbildung 40: Dialogformen und Dialogtechniken

Die folgenden Kapitel beschreiben an Hand von pflanzenbaulichen Beispielen und durch die Diskussion der Vor- und Nachteile ihr effektives Einsatzgebiet.

4.2.4.2 Ja-, Nein-Technik (Frage-, Antwort-Technik)

Die Frage-, Antwort-Technik ermöglicht einen engen Dialog zwischen Benutzer und Programm. Dieses übernimmt dabei den aktiven Teil des Dialogs und führt sozusagen den Benutzer an der Leine.

Abbildung 41 auf Seite 65 zeigt den Ablauf eines Programmes zur Herbizid-Auswahl, in diesem Beispiel unter Verwendung der Frage-, Antwort-Schnittstelle.
Das Programm führt hier den Benutzer mit gezielten Fragen. Dieser trägt jeweils seine Antwort hinter dem Eingabepfeil "=>" ein. Anschließend setzt das Programm den Dialog mit einer weiteren Frage fort.

Die Schnittstelle bietet folgende Vorteile:

- Strenge und sichere Benutzerführung
- Sichere Bedienung des Programms auch für ungeübte Benutzer
- Vollkommener Verzicht auf zusätzliche Funktionstasten für den Programmablauf
- Reduktion des Eingabeaufwands von Daten bei sehr verzweigten Problemen (abhängig von der Beantwortung bestimmter Schlüsselfragen werden bestimmte, nicht zur Problemlösung notwendige Daten nicht mehr erfragt)
- Optimal bei Alternativfragen

Als Nachteile sind zu nennen:

- Bei vielen Fragen ergibt sich ein zu unübersichtlicher Programmablauf.
- Es besteht keine Möglichkeit, den Programmablauf persönlich modifizieren zu können.
- Die Lösung von "was wäre, wenn"-Problemen ist zu zeitaufwendig, da in diesen Fällen sämtliche Fragen neu zu beantworten sind.
- Bei Fragen mit mehreren möglichen Antworten sind zusätzliche Dialogtechniken nötig.

Als Schlußfolgerung ergibt sich folgendes Anforderungsprofil für Programme, die diese Technik nutzen können:

- Vorwiegende Verwendung des Programms von ungeübten Benutzern
- Keine umfangreichen Eingaben
- Nicht permanent im Einsatz
- Viele Alternativfragen bei der Dateneingabe

Innerhalb des CIPP-Konzeptes (siehe Kapitel "Begriffsbestimmung" auf Seite 103) erfüllen die Programme "Septoria-Prognose", "Epipre" (Rößler, 1987) und die "Herbizidauswahl" (Hainzlmaier, 1987) diese Anforderungen.

4.2.4.3 Menü-Technik

Die Menü-Technik basiert auf dem Prinzip, dem Benutzer über die Bildschirmmaske unterschiedliche Auswahlmöglichkeiten anzubieten, von denen er jeweils eine auswählen kann.

```
    Programm zur Auswahl von Herbiziden

 1) Besteht Nachtfrostgefahr?
        (J/N)
=> J

 2) Welche Temperatur besteht zur Zeit?
    (1=kalt, 2=normal, 3=warm)
=> 2

 3) Welche Feuchtigkeit besteht zur Zeit?
    (1=naß, 2=normal, 3=trocken)
=> 2

 4) Um welchen Boden handelt es sich?
    (1=leicht, 2=mittel, 3=schwer)
=> 2

 5) Wie hoch ist der Humusgehalt?
    (1=<2%, 2=2-4%, 3=>4%)
=> 2

 6) Welche Fruchtart soll behandelt werden?

=> Winterweizen
```

Abbildung 41: Frage-, Antwort-Schnittstelle am Beispiel der Herbizidauswahl

Dadurch bestimmt er nach persönlichen Gesichtspunkten den Programmablauf. Danach, wie er seine Auswahl trifft, unterscheiden sich folgende Typen von Auswahlmenüs:

- Menü mit Auswahl über Symbole
- Menü mit Auswahl über Funktionstasten
- Menü mit Auswahl über Cursortasten

Abbildung 42 auf Seite 66 zeigt für jeden Menü-Typ ein Beispiel:

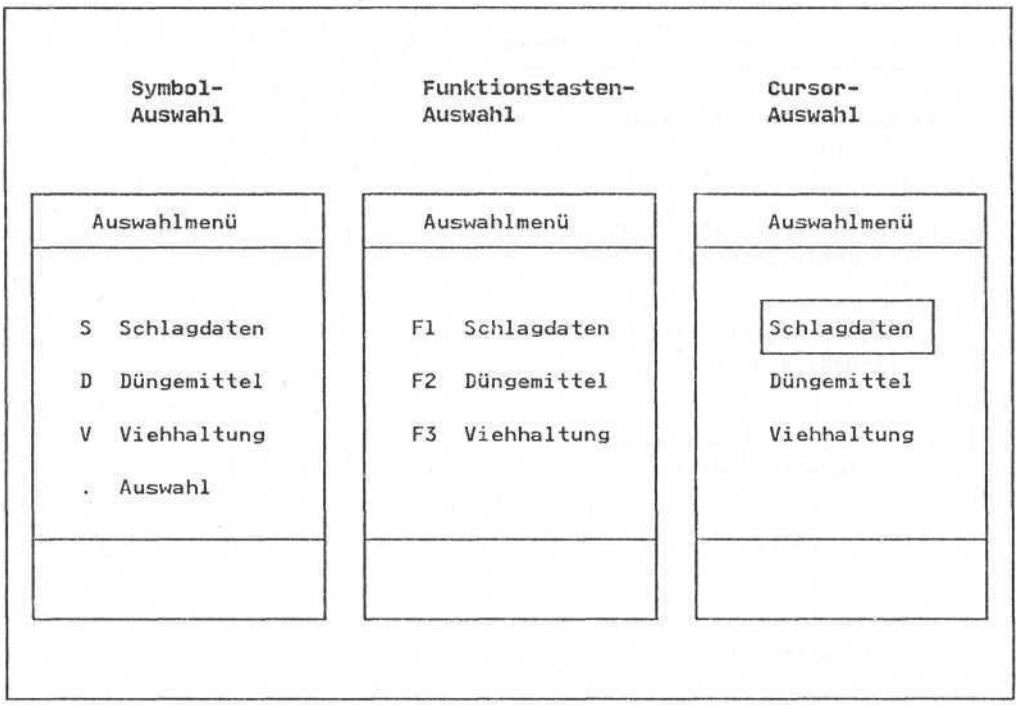

Abbildung 42: Menü-Typen mit unterschiedlicher Auswahlmöglichkeit

Beim Menü-Typ mit Symbolauswahl gibt der Benutzer ein alphabetisches oder numerisches Symbol ein. Mit den Auswahleinträgen korrespondierende alphabetische Symbole (in Abbildung 42 z.B. S für Schlagdaten) haben sich gegenüber numerischen Symbolen wegen ihrer besseren mnemotechnischen Eigenschaften bewährt. Der Vorteil dieses Menütyps liegt in der für den Benutzer geringen Fehleranfälligkeit, sein Nachteil in der zur Auswahl erforderlichen Übersetzung vom Menü-Eintrag in das jeweilige Auswahl-Symbol.[26]

Beim Menü-Typ mit Auswahl über Funktionstasten sind eine Reihe von Funktionstasten mit Menü-Einträgen verknüpft. Beim Betätigen der jeweiligen Taste führt das Programm den mit der Taste virtuell verbundenen Auswahlschritt aus. Der Vorteil dieses Typs liegt

- in seiner sehr schnellen Auswahlmöglichkeit und
- in seiner flexiblen Tastenbelegung.

---

[26] Vgl. auch zur praktischen Anwendung dieses Menü-Typs das landwirtschaftliche Programm COMAK in Gooijer (1986, S.57-60).

Nachteile ergeben sich aus der räumlichen Trennung von Funktionstaste und Menü-Eintrag und durch Fehlermöglichkeiten, bedingt durch falsche Zuordnungen zwischen gewünschter Auswahl und betätigter Funktionstaste.

Beim Menü-Typ über Cursor-Auswahl positioniert der Benutzer einen über die Cursor-Tastatur beweglichen, auf dem Bildschirm invers dargestellten Lichtbalken auf den gewünschten Menü-Eintrag und löst dann die Auswahl durch Betätigen der "Datenfreigabe-Taste" aus.
Der Vorteil dieses Menüs liegt in seiner leichten Erlernbarkeit und in der direkten, sichtbaren Verbindung zwischen Menü-Eintrag und Auswahlinstrument (Lichtbalken).
Nachteilig ist, daß der Benutzer bei umfangreichen Auswahlmenüs viele Tastenanschläge (Cursortastatur) zum Positionieren des Lichtbalkens benötigt.

Während das rechte Beispiel in Abbildung 42 auf Seite 66 typisch für die Anwendung dieses Menü-Typs ist und daher häufig in Programmen vorkommt, zeigt Abbildung 43 eine weitere komfortable Anwendung (Pohlmann, 1987 C).

PC-Düngeplanung Auswahl Düngergruppe

Nr.	Mineraldünger	Preis DM/dt	N %	P2O5 %	K2O %	MgO %	CaO %

N	P	K	<->	Düngergruppe auswählen
N-P	N-K	P-K	<---'	Düngergruppe anzeigen
N-P-K	Ca			

Abbildung 43: Menü-Typ mit Cursor-Auswahl

Um die Erfassung von Düngemitteln zu erleichtern, steht im Hintergrund eine Datei bereit, die sämtliche (ca. 100) in der Bundesrepublik angebotenen Düngemittel mit Namen und Nährstoffen enthält. Aus dieser Düngemittelliste können über Funktionstasten die für den Betrieb beziehbaren Düngemittel in die Erfassungs-

maske kopiert werden. Um die hundert Mittel schneller überblicken zu können, hat der Benutzer zusätzlich die Möglichkeit, mit Hilfe dieses Menü-Typs (Abbildung 43) Düngemittel mit speziellen Nährstoffen aus der Liste zu selektieren. Dies geschieht wiederum über einen mit der Cursor-Tastatur gesteuerten Lichtbalken und die anschließend betätigte Datenfreigabe-Taste.

Grundsätzlich sind alle Menü-Techniken zur Steuerung des Programmablaufs für den Einsatz in landwirtschaftlichen Programmen hervorragend geeignet.
Bei vorwiegend untrainierten Benutzern erfordern sie zur Programmsteuerung ausschließlich eine Gedächtnisleistung, die auf die Wiedererkennung von Menü-Einträgen ausgerichtet ist. Im Englischen wird diese Form des beanspruchten Gedächtnisses als "recognition memory" bezeichnet.
Andere, benutzerinitiierte Techniken, beispielsweise über eine Kommando-Eingabe, erfordern dagegen ein sogenanntes "recall memory", eine weitaus anstrengendere Gedächtnisleistung, die sich auf das "Zurückrufen" von einzugebenden Befehlen konzentrieren muß.

Während in den ersten PC-Programmen unserer Arbeitsgruppe ausschließlich der Menü-Typ mit Auswahl über Funktionstasten eingesetzt wurde, liegt in den heutigen vernetzten Programmen der Schwerpunkt vollkommen auf dem Menü-Typ mit Cursor-Auswahl.

Während der Entwicklung integrierter Programme sind Menü-Einträge unter Umständen besonders häufig zu ändern. Das betrifft zum einen den logisch oder semantisch bedingten Wechsel in der Reihenfolge der Menü-Einträge, zum anderen aber auch das Streichen oder die Erweiterung von Einträgen.

Im Falle solcher Änderungen müssen unterschiedliche Funktionstasten - im übrigen auch bei der Verwendung der Symbol-Auswahl mit Zahlen - den Menü-Einträgen neu zugeordnet werden. Bei bisherigen Benutzern dieser Programme führt das zu schwerwiegenden Folgen dadurch, daß sie ihre Auswahl schon automatisiert, d.h. ohne eine erneute, kontrollierte Zuordnung von Menü-Einträgen und Funktionstasten treffen. Eine Änderung in der Zuordnung bedingt daher ein zunächst extremes Ansteigen der Fehlbedienungen von Programmen.

Hinzu kommt, daß die Funktionstasten durch den Einsatz für die Programmauswahl in zwei unterschiedlichen Ebenen Verwendung finden, der

- Auswahlebene und der
- Ebene genormter Programmfunktionen (wie "Satz löschen", "ein Bild zurück" etc.).

Diese Doppelbelegung der Tasten erfordert vom Benutzer eine ständige, unterbewußt ablaufende Überprüfung, welche der beiden Ebenen im jetzigen Programmschritt durch Betätigen der Tasten angesprochen wird.

Seit der zunehmenden Entwicklung IBM-kompatibler Personal-Computer zeigt sich, daß hinsichtlich der Tastaturgestaltung heute verschiedene Varianten auf dem Markt sind. Die Funktionstasten sind dabei sowohl auf der linken Seite der Tastatur in Zweierreihe als auch horizontal über der eigentlichen Tastatur angeordnet.

Dies erfordert vom Benutzer bei Verwendung des Menü-Typs mit der Auswahl über Funktionstasten bei wechselnden Tastaturen und Geräten einen zusätzlichen Suchprozeß zur Identifizierung der jeweils richtigen Funktionstaste.

Festzuhalten bleibt, daß der Vorteil der Menü-Technik in der

- schnellen, individuellen und sicheren Programmsteuerung,
- der übersichtlichen Möglichkeit zur Programmverzweigung,
- in der benutzerfreundlichen Auswahlmöglichkeit über verschiebbare Lichtbalken und
- in der sich daraus ergebenden Unabhängigkeit von verschieden aufgebauten PC-Tastaturen

liegt.

Sein Nachteil:

- Die Technik ist nur bei Programmverzweigungen einzusetzen.
- Sie setzt häufig Kenntnisse der Struktur des Gesamtprogramms beim Benutzer voraus.
- Sie kann in der Regel höchstens einmal geschachtelt werden, um den Gesamtüberblick beim Benutzer zu erhalten (Ein Menü ruft nach der Benutzerauswahl ein weiteres Menü auf).

4.2.4.4 Formular-Technik

Techniken zur Menü-Auswahl können gut mit Formular-Techniken in Programmen verknüpft werden.

Gibt der Benutzer seine Eingaben über eine Folge von Bildschirmformularen ein, so spricht man von einer Formulareingabe. Abbildung 44 auf Seite 70 zeigt dazu ein Beispiel in Form der Schlagdatenerfassung in der Düngeplanung am Personal-Computer.

```
| Dungplan | Schlagdaten | 14.06.87 |

Schlagnummer ../.
Schlagname  .....                    Größe ..... ha
Hauptfrucht  ....                 Ertragsziel ..... dt /ha


Ernterückstände/Zwischenfrucht
Art ....  Menge .... dt/ha          Art ....  Menge ..... dt /ha


Wirtschaftsdünger
Art ....  Menge .... dt/ha  Monat ..  Art ....  Menge .... dt/ ha  Monat ..
    ....        .... dt/ha        ..      ....        .... dt/ ha  Monat ..


Bodenuntersuchung (nach CAL)
Bodenart ..  N-Faktor ...  pH-Wert ...     CaO(kg/ha) ...  P2O5 ... K2O ...
oder Bedarfswerte (z.B. nach EUF) in kg/ha          N ...  P2O5 ... K2O ...

ESC Auswahlmenü                                  <----¬ Daten erfassen
```

Abbildung 44: Formular-Technik am Beispiel der Erfassung von Schlagdaten

Diese Technik gehört zu den klassischen Dialogtechniken dadurch, daß man schon sehr bald beim Einsatz der EDV versuchte, bisher bestehende Formulare oder Karteikarten auf dem Bildschirm abzubilden. Auf diese Weise wollte man den Übergang zur "elektronischen Datenerfassung" für den Benutzer erleichtern.

Als Vorteil dieser Technik sind zu nennen:

- Gute Anordnung und Codierung von Informationen
- Einfaches Erlernen
- Gute Implementierungsmöglichkeit

Nachteile können sich über schlecht bzw. unzweckmäßig gestaltete Bildschirmformulare ergeben (siehe dazu Kapitel "Gestaltung von Bildschirmseiten" auf Seite 50).

Diese Technik ist innerhalb landwirtschaftlicher Programme am häufigsten vorzufinden und wegen ihrer genannten Vorteile für die Erfassung vieler sinngemäß zusammenhängender Daten zu empfehlen.

4.2.4.5 Befehlssprachen-Technik

Während der dominante Dialogpartner in der "Frage-, Antwort-Technik" das Programm war, steuert in der "Befehlssprachen-Technik" der Benutzer die Funktion des Programms und seinen Ablauf. Mit Hilfe einer normierten und damit vom System verstandenen Befehlssprache teilt er seine Wünsche diesem mit.

Abbildung 45 zeigt diese Technik. Die Eingabe des Benutzers ist mehrzeilig hinter dem Eingabezeichen "=>" aufgeführt.

```
=>  SELECT Sorte, Jahr, Ertrag
    FROM   S-Tab
    WHERE  (Ertrag > 40)
           AND   ( Jahr IN(70,78))
           AND   ( Ort = 'BAYERN')
    ORDER  BY Ertrag DESC;
```

Abbildung 45: Befehlssprachen-Technik in Form einer SQL-Anweisung

Bei der verwendeten Befehlssprache handelt es sich um die Datenbankabfragesprache SQL (Structured Query Language), die heute als standardisierte Abfragesprache in verschiedenen relationalen Datenbanken, wie DB/2 (IBM Systemliteratur), Oracle (Oracle Systemliteratur, 1985) und Ingres (Date, 1987), Verwendung findet.

Diese Befehlssprache ist der englischen Umgangssprache ähnlich und ermöglicht nach ihrer Beherrschung, vollkommen flexible und auf die Bedürfnisse des Benutzers zugeschnittene Anfragen an die Datenbank zu stellen.

Mit der Anweisung aus Abbildung 45 liest das Programm alle Sorten, Jahre und Erträge aus der Datenbank, wenn sie die folgenden Bedingungen erfüllen:

- Der Ertrag muß größer als 40 dt sein.
- Es sollen nur Sorten der Jahre 1970 und 1978 gelesen werden.
- Nur Sorten aus Bayern kommen in Betracht.
- Der Ergebnisausdruck soll die Sorten nach dem Ertrag aufsteigend aufgelistet enthalten.

Als Ergebnis erscheint die in Abbildung 46 auf Seite 72 gezeigte Tabelle:

Sorte	Jahr	Ertrag
Sorte A	78	62,0
Sorte B	78	61,0
Sorte C	78	60,0
Sorte B	70	60,0
Sorte C	70	59,0
Sorte A	70	50,0

Abbildung 46: Ergebnisausgabe einer Befehlssprachen-Anweisung

Die Bedeutung dieser Technik liegt darin, daß der Benutzer bestimmt, was das Programm leisten soll, nicht aber wie es seine Aufgabe lösen soll.

SQL ist als Beispiel für die Befehlssprachen-Technik ein sehr mächtiges und ausgereiftes Instrument. Für einfachere Anwendungen sind durchaus Befehlssprachen mit kleinerem Umfang vorstellbar, die aber ebenfalls eine Flexibilität der Programmfunktionen ermöglichen würden.

Zusammenfassend ergeben sich folgende Vorteile für diese Technik:

- Eine komplexe und flexible Programmnutzung ist möglich.
- Bei richtiger Gestaltung der Befehlssprache ist diese leicht zu erlernen.
- Diese Technik kann unterschiedliche Benutzerwünsche erfüllen.

Als Nachteile sind zu nennen:

- Nur bei häufiger Programmbenutzung ist die Aneignung einer Spezialsprache sinnvoll.
- Eine spezielle Befehlssprache muß erlernt werden.
- Die Technik ist nicht für den EDV-Laien geeignet.
- Der Benutzer muß selbst mit Hilfe der Befehlssprache die vom Programm durchzuführende Arbeit formulieren.
- Sie verlangt umfassende Kenntnisse der zugrunde gelegten Datenbank.
- Für Einzelprogramme ist sie als Dialogtechnik ungeeignet.

4.2.4.6 Ein-, Ausgabe-Technik

Diese Technik beruht auf der Vorstellung, daß ein Benutzer das von ihm gewünschte Format der Programmausgabe am Bildschirm eingibt. Dabei bestimmt er neben diesem auch die Anzahl und Art der gewünschten Daten.

Diese Dialogtechnik ist vom Ergebnis ähnlich der Befehlssprachen-Technik. Es besteht jedoch zwischen beiden der große Unterschied, daß der Benutzer keinen umfangreichen Befehlssatz lernen muß, sondern mit ein paar Befehlen interaktiv seine Wünsche auf den Bildschirm zeichnet.
Diese Technik wird zur Zeit in kaum einem Programm eingesetzt, da sie für den Benutzer gewöhnungsbedürftig ist und einen großen Aufwand bei ihrer Programmierung erfordert.

4.3 Verknüpfung der Medien Großrechner und Personal-Computer

4.3.1 Datenbankaustausch zwischen Großrechner und Personal-Computer

4.3.1.1 Begriffsbestimmung

Bisher benutzte man ausschließlich Datenbanken am Großrechner, um große Datenmengen programmunabhängig und flexibel zu speichern. Diese Entwicklung setzt neuerdings auch im Bereich der Personal-Computer ein, die durch ihre Leistungsfähigkeit heute schon in der Lage sind, relationale Datenbanken zu verwalten. Um mittelfristig nicht gleiche Daten sowohl für Personal-Computer als auch für Großrechenanlagen mit Redundanz abspeichern zu müssen, ist über eine Vernetzung ein Datenaustausch zwischen Datenbanken von Personal-Computern und Großrechenanlagen sinnvoll und erforderlich.

Eine aus der Benutzersicht als Einheit gesehene Datenbank ist grundsätzlich aus Strukturen und Daten aufgebaut (Abbildung 47).

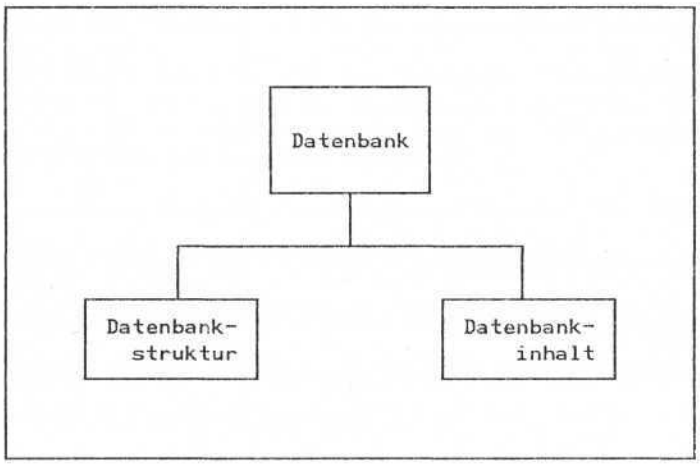

Abbildung 47: Grundsätzlicher Aufbau einer Datenbank

Die Datenbankstruktur stellt dabei das Gerüst oder den formalen Aufbau der Datenbank dar, der Datenbankinhalt umfaßt dagegen die gespeicherten Daten.

Durch diese notwendige Differenzierung ergeben sich verschiedene Interpretationen des Begriffs "Datenbankaustausch".

1. Der Datenbankaustausch kann sich nur auf den Austausch von Datenbankstrukturen beschränken.

Das wäre z.B. dann sinnvoll, wenn parallel zu einem am Großrechner bestehenden Datenbanksystem ein dazu korrespondierendes am Personal-Computer aufgebaut werden soll.

2. Der Austausch beinhaltet eine reine Datenübertragung zwischen zwei Datenbanksystemen.

   Diese Form des Austausches hat innerhalb einer Vernetzung von Medien Bedeutung (siehe dazu Kapitel "Medialer Support: Entscheidungsunterstützung durch die Vernetzung von Großrechner, Personal-Computer und Bildschirmtext" auf Seite 156), wenn es darum geht, dezentral am Personal-Computer erfaßte Daten zentral über eine Datenbank am Großrechner zur weiteren Auswertung zur Verfügung zu haben. In diesem Fall bestehen in beiden Medien identische Datenbankstrukturen, es werden lediglich die Daten ausgetauscht.

3. Der Austausch kann sowohl wechselseitig als auch einseitig von einer Datenbank zur anderen erfolgen.

Das Kapitel "Entwickelte Software zur Durchführung des Datenbankaustausches" auf Seite 76 zeigt exemplarisch die Übertragung einer Datenbank (Struktur und Dateninhalt) vom Großrechner zum Personal-Computer mit Hilfe entwickelter Software.

Bevor mit dem Austausch jedoch generell begonnen werden kann, sind zunächst die Softwarevoraussetzungen zu prüfen.

4.3.1.2 Vorgegebene Datenbanken

Nachdem uns am Großrechner der Bayerischen Staatsministeriums für Ernährung, Lanwirtschaft und Forsten als relationales Datenbanksystem DB/2 zur Verfügung stand, entschieden wir uns auf der Ebene des Personal-Computers für das dazu kompatible Datenbanksystem Oracle.

Da beide Systeme mit der standardisierten Datenbanksprache "Structured Query Language (SQL)" arbeiten, ergeben sich hinsichtlich eines Datenbankaustauschs erhebliche Vorteile.

Die Sprache "Structured Query Language" (SQL) steuert bei beiden Systemen

- den Aufbau von Datenbankstrukturen,
- die Datenmanipulation, sowohl interaktiv als auch eingebettet in Quellenprogramme unterschiedlicher Programmiersprachen,
- die Zugriffskontrolle auf Daten

und ermöglicht dadurch sowohl den Austausch von Daten als auch von Datenbankstrukturen.

Prinzipiell wäre auch ein Austausch zwischen IMS-Datenbanken am Großrechner und relationalen PC-Datenbanken möglich gewesen. Dies hätte aber für die Durchfüh-

rung einen erheblichen softwaretechnischen Aufwand bedeutet. Deshalb ist es auch für zukünftige Vernetzungsmöglichkeiten günstiger, bestehende IMS-Datenbanken in relationale Datenbanken umzuwandeln.[27]

4.3.1.3 Entwickelte Software zur Durchführung des Datenbankaustausches

In einer Umgebung, in der Großrechenanlagen und Personal-Computer miteinander vernetzt sind, ist es wichtig, einen Datenbankaustausch häufig und benutzerfreundlich durchführen zu können.

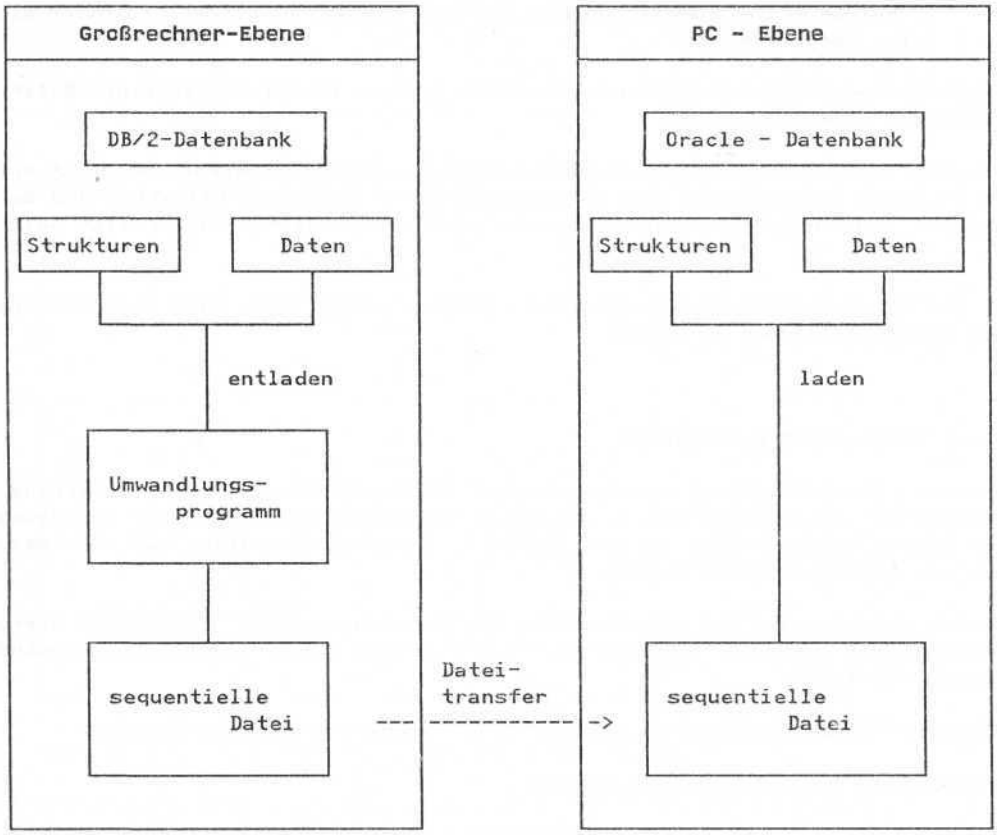

Abbildung 48: Vorgehensweise für den Transfer von Datenbanken zwischen Großrechner und Personal-Computer

---

[27] Um mit bestehenden IMS-Calls auf eine DB/2 Datenbank zugreifen zu können, siehe zur dazu notwendigen Schnittstelle Date (1986).

Voraussetzung ist allerdings Software, die diesen Prozeß aktiv unterstützt bzw. automatisieren kann.

Abbildung 48 auf Seite 76 zeigt zunächst prinzipiell die Vorgehensweise, um eine Datenbank vom Großrechner auf den Personal-Computer übertragen zu können.

Ein Umwandlungsprogramm

- entlädt die Strukturen und enthaltenen Daten der DB/2-Datenbank,
- erzeugt auf Grund dieser Informationen geeignete Befehlssätze zum Aufbau einer deckungsgleichen Datenbank am Personal-Computer und
- speichert die Befehle in einer sequentiellen Datei ab.

Nachdem diese Datei über einen sogenannten "Datei-Transfer" auf den Personal-Computer übertragen worden ist, erstellt Oracle über die enthaltenen Befehle die Datenbankstruktur und lädt anschließend die Daten in diese ein.

Sowohl Strukturen als auch Daten sind innerhalb relationaler Datenbanken in Tabellenform gespeichert. Einen Auszug aus einer solchen "Strukturtabelle" zeigt Abbildung 49.

Name	Datentyp	Beschreibung
NAME	VARCHAR(18)	Variablenname in einer Tabelle
TBNAME	VARCHAR(18)	Tabellenname
TBCREATOR	CHAR(8)	Name des Tabellenerstellers
COLNO	SMALLINT	Stellung der Variablen in der Tabelle
COLTYPE	CHAR(8)	Datentyp (integer,float, usw.)
LENGTH	SMALLINT	Länge der Variablen
SCALE	SMALLINT	Stelle des Dezimalpunktes
NULLS	CHAR(1)	Behandlung der Nullwerte
....		
....		

Abbildung 49: Auszug aus einer Strukturtabelle

Die Strukturtabelle gibt Auskunft über

- den Tabellennamen,
- die vorhandenen Variablennamen (Spaltenbezeichnungen),
- die Stellung der Variablen in der Tabelle,
- den Datentyp, die Länge der Variablen und
- den Namen des Tabellenerstellers.

Das Umwandlungsprogramm kann diese Informationen aus der Strukturtabelle lesen und daraus die SQL-Anweisungen generieren, die zum Aufbau deckungsgleicher Tabellen in einer neuen Datenbank erforderlich sind (Abbildung 50):

```
CREATE TABLE MH.MHTTBETR

  (BETRNR      CHAR(9) NOT NULL,
   BUNDL       CHAR(2) NOT NULL,
   NAME        VARCHAR(22),
   VNAME       VARCHAR(22),
   ORT         VARCHAR(33)
                        )
   IN MHDBTSK.MHTTSK01;
```

Abbildung 50: SQL-Befehl zum Erstellen einer Datenbanktabelle

Diese Anweisung erstellt eine Tabelle (TABLE) zum Abspeichern von Betriebsdaten, wie Betriebsnummer, Bundesland und Adresse. Würde der Benutzer interaktiv eingegebene Daten aus dieser Tabelle abfragen, hätten sie am Bildschirm folgende Darstellungsform (Abbildung 51):

BETRNR	BUNDL	NAME	VORNAME	ORT
111111111	BY	Müller	Alois	Freising
111111112	BY	Schmidt	Werner	Rosenheim
111111113	BY	Kammhuber	Franz	Obing
.	.	.	.	.
.	.	.	.	.

Abbildung 51: Ergebnisausgabe einer Datenbanktabelle

Sobald das Umwandlungsprogramm aus den Informationen der Strukturtabelle die für den Aufbau der Tabellen notwendigen SQL-Anweisungen generiert hat, liest es die

vorhandenen Daten satzweise ein und erstellt wiederum für jeden Satz eine SQL-Anweisung, mit der Oracle die Daten tabellengerecht speichern kann.

Dabei muß jeder Wert eines gelesenen Datenbanksatzes entsprechend der Variablendefinition (siehe Abbildung 49 auf Seite 77) in eine SQL-ISRT-Anweisung eingefügt werden.
Diese ISRT-Anweisungen sind notwendig, um vorhandene Daten in die aufgebauten Datenbanktabellen einbauen zu können.

Ein Beispiel für solche, vom Umwandlungsprogramm erstellten Anweisungen zeigt Abbildung 52.

```
INSERT INTO BETR VALUES (
'444444444','SH','DIESEL','54 ','102','1',' 60',' 53',' 2',' 6.8',
'Schmidt             ','Borg                ',
'Kook 1                        ','1111','Polder                    ',
'                ',' 1',' 2',' 3','1','2','3','4',
'8');

INSERT INTO BETR VALUES (
'333333333','BY','HASE  ','44 ','502','1',' 34',' 30',' 14',' 8.5',
'Müller              ','Karl                ',
'Mauernerstr. 1                ','2222','Mauern                    ',
'08764/111       ',' 1',' 2',' 3','1','2','3','4',
'8');

INSERT INTO BETR VALUES (
'222222222','BY','BAYERN','42 ','489','1',' 75',' 35',' 7',' 8.7',
'Meyer               ','Johann              ',
'Dorfstraße 1                  ','3333','Mannheim                  ',
'08764/1111      ',' 1',' 2',' 3','1','2','3','4',
'8');

INSERT INTO BODE VALUES (
85,'111111111','BY',10,'x',' ','sL','20',' ',' ','30','65',' 1.3',
'10','12','01','01','5.8',' ',' ',' ',' ',' ',' ');
INSERT INTO BODE VALUES (
85,'111111111','BY',20,'x',' ',' L','22',' ',' ','33','60',' 1.5',
'12','10','01','01','5.9',' ',' ',' ',' ',' ',' ');
```

Abbildung 52: Vom Umwandlungsprogramm erzeugte ISRT-Anweisungen

Das Umwandlungsprogramm legt zum Abschluß sämtliche SQL-Anweisungen in einer sequentiellen Datei ab (siehe Abbildung 48 auf Seite 76). Nach ihrer Übertragung auf den Personal-Computer baut Oracle dann mit den darin enthaltenen Anweisungen eine entsprechende Datenbank auf.

Penger (Lehreinheit Ackerbau und Versuchwesen, Weihenstephan) hat innerhalb eines BMFT-Forschungsprojektes das Umwandlungsprogramm dialogorientiert pro-

grammiert.[28] Über die Auswahlmaske (Abbildung 53 auf Seite 80) kann der Benutzer die verschiedenen Routinen zum Datenbanktransfer starten und das Ergebnis in der erwähnten sequentiellen Datei erhalten.

```
                Datenbanktransfer  Host - PC

      Diese Prozedur entlädt eine DB/2-Datenbank und
      erzeugt SQL-Anweisungen, mit denen Oracle auf
      dem Personal-Computer eine deckungsgleiche
      Datenbank aufbauen kann.

      Die Ausgabedaten befinden sich auf: 'MH.P.DB2.OUT'.

          1  zurück

          2  Erstellen und Installieren der Entladeprogramme

          3  Starten des Entlade- und Aufbereitungsprogramms

          4  Statusabfrage  (MHPBDB2)

          5  Jobsysout am Schirm ansehen (OUT-Kommando)

   Auswahl: .
```

Abbildung 53: Auswahlmaske des Umwandlungsprogramms "Datenbanktransfer"

Erste Praxisversuche haben am Beispiel der Übertragung einer DB/2-Datenbank vom Großrechner zum Personal-Computer gezeigt, daß diese mit Hilfe des Programms einfach zu realisieren ist.

In einer Ausbaustufe sind folgende Erweiterungen des Programms geplant:

- Vollständige Dynamisierung der Übertragung,
- Übertragungsmöglichkeit von Teildatenbanken,
- Übertragung von PC-Datenbanken auf den Großrechner.

---

[28] Vgl. dazu Pohlmann, Mangstl, Penger (1987 A).

4.3.2 Programmaustausch zwischen Großrechner und Personal-Computer

4.3.2.1 Vorgehensweise

Neben den Großrechenanlagen kommen seit 1981 verstärkt Personal-Computer zum Einsatz. Da sie zunehmend leistungsfähiger geworden sind, übernehmen sie schon heute mit Erfolg typische Großrechneraufgaben.

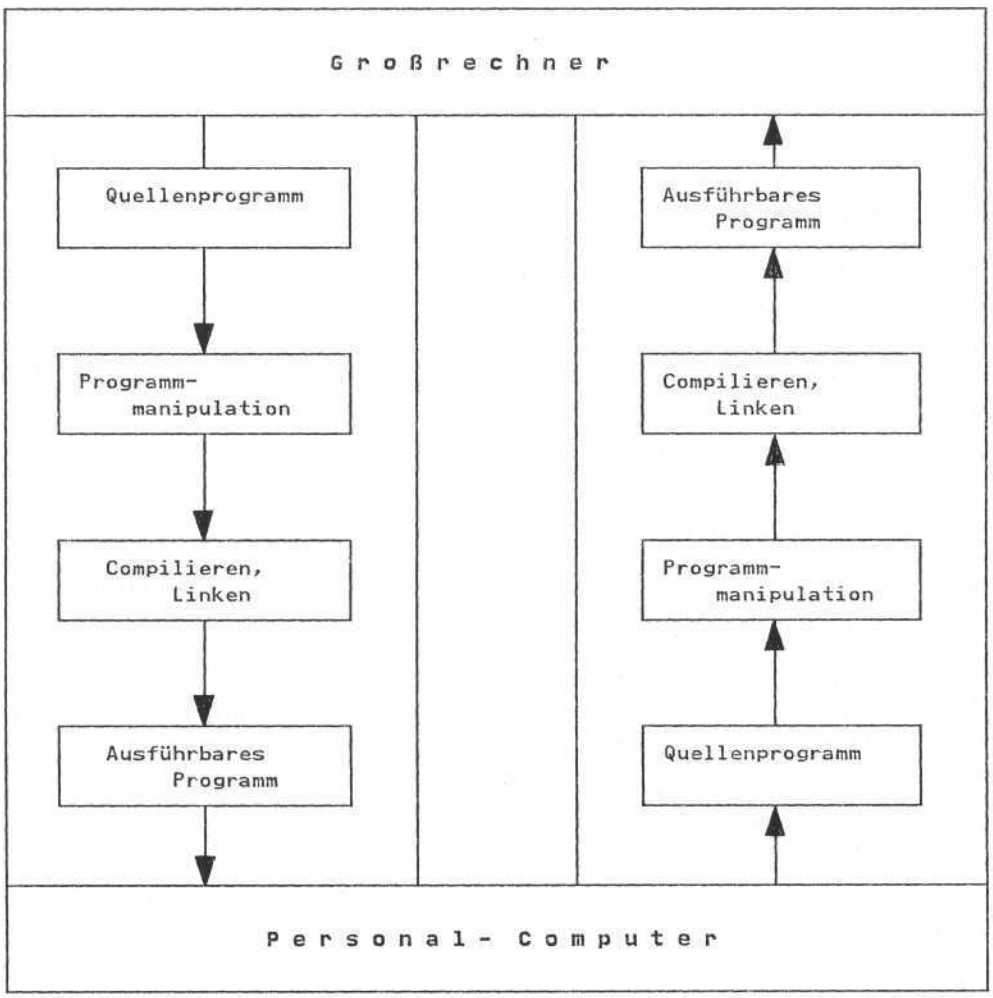

Abbildung 54: Programmaustausch zwischen Großrechner und Personal-Computer

Da in Zukunft eine Doppelentwicklung von Programmen für Personal-Computer und Großrechner ökonomisch nicht tragbar ist, kann über einen Programmaustausch zwischen den Medien diese vermieden werden.

Großrechnersysteme und Personal-Computer sind allerdings von der Hardwareseite her unterschiedlich aufgebaut, so daß ein einfaches übertragen von Programmen auf das jeweils andere System zunächst nicht zum gewünschten Erfolg führt.

Unter Austausch soll deswegen im folgenden das in Abbildung 54 auf Seite 81 skizzierte Vorgehen verstanden werden.

Ein "Übertragungsprogramm" modifiziert ein bestehendes Quellenprogramm so, daß es nach dem Compile- und Link-Schritt auf dem jeweils anderen Medium einsetzbar ist.
Zur Verwirklichung dieses Vorhabens bedarf es zunächst einiger Voraussetzungen, die auf der Softwareseite zu erfüllen sind.

4.3.2.2 Softwarevoraussetzungen

Beim Programmaustausch kommt der Software, konkret der Programmiersprache mitsamt den für den Personal-Computer und Großrechner vorhandenen Compilern, die wichtigste Bedeutung zu.

Die Programmiersprache (incl. Compiler) muß den folgenden Anforderungen genügen:

- Ein Compiler für die gewählte Programmiersprache muß auf beiden Systemen vorhanden bzw. zu beschaffen sein.
- Die Programmiersprache muß Schnittstellen zu Datenbanksystemen am Großrechner und Personal-Computer bieten.
- Der vom Compiler auf beiden Systemen unterstützte Sprachumfang sollte gleich sein, um übersetzungsschwierigkeiten zu vermeiden.

Unter unseren Voraussetzungen erfüllte Cobol die oben genannten Anforderungen. Die Cobol-Compiler am Großrechner (IMS/VS Cobol) und am Personal-Computer (Realia, 1986) sind sprachlich voll kompatibel und Sprachschnittstellen sind zu den relationalen Datenbanksystemen DB/2 (Großrechner) und Oracle (Personal-Computer) vorhanden.

Weiterhin ist zu prüfen, welche Teile eines Quellenprogramms auf beiden Systemen gleich und welche verschieden gestaltet sind.

Eine unter diesem Gesichtspunkt vorgenommene Analyse zeigt Abbildung 55 auf Seite 83 in stark vereinfachter Form.

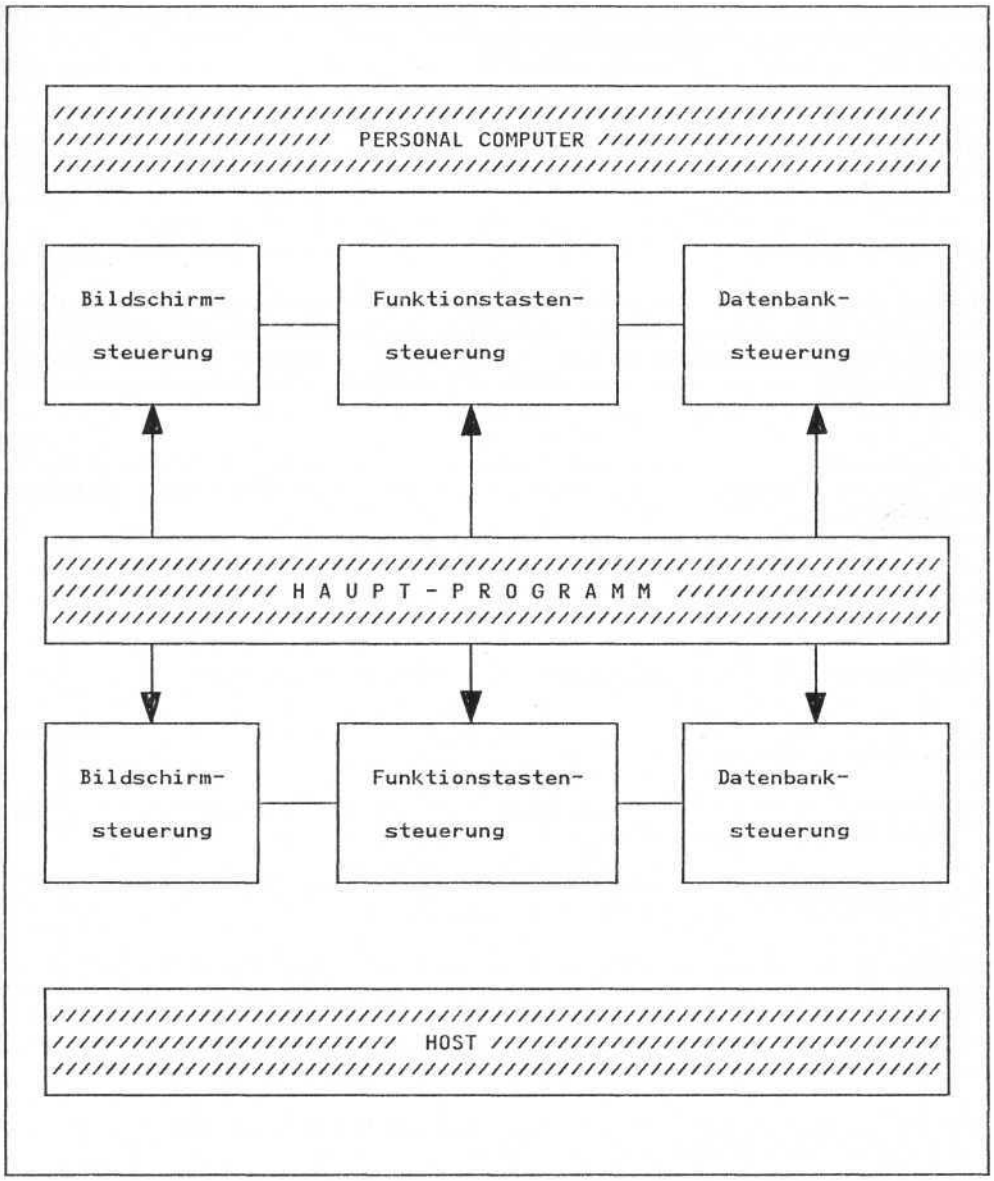

Abbildung 55: Zerlegung eines Programms unter dem Gesichtspunkt eines Programmaustausches zwischen Großrechner und PC

Aus der Abbildung 55 auf Seite 83 ist zu ersehen, daß das Quellenprogramm in ein Hauptprogramm mit drei Teilprogrammen (Module)

- zur Bildschirmsteuerung,
- zur Funktionstastensteuerung und
- zur Datenbanksteuerung

zerlegt werden kann.

Während das Hauptprogramm für den Personal-Computer und den Großrechner gleich aufgebaut und nach geringfügigen Änderungen zu übertragen ist, muß bei den Teilprogrammen eine Anpassung an die jeweils andere Systemumgebung vorgenommen werden. Das Ziel war es, diese Anpassung mit Softwareunterstützung durchzuführen.

Um den Funktionsmechanismus und die Aufgaben der entwickelten Software zu erläutern, ist es zunächst sinnvoll, kurz die Programmbausteine beider Systeme zu beschreiben.

4.3.2.3 Entwickelte Software zur Durchführung des Austausches

Wie die Analyse der Teilprogramme zur Datenbank-, Bildschirm- und Tastatursteuerung gezeigt hat, wäre eine manuelle übertragung von Großrechnerprogrammen zum Personal-Computer wegen der zahlreichen anzupassenden Teilprogramme zeitaufwendig, fehlerträchtig und nur von Spezialisten durchzuführen. Verschiedene notwendige Arbeiten lassen sich jedoch mit Hilfe der in dieser Arbeit entwikkelten Software unterstützen.

Zu diesen Arbeiten zählen

- eine automatische Maskengenerierung,
- eine Generierung des Programmmoduls für die Bildschirmsteuerung und
- ein Programm zur Anpassung der jeweiligen Hauptprogrammstruktur.

Die einzelnen Softwarepakete werden zentral über die folgende Auswahlmaske am Personal-Computer aufgerufen (Abbildung 56):

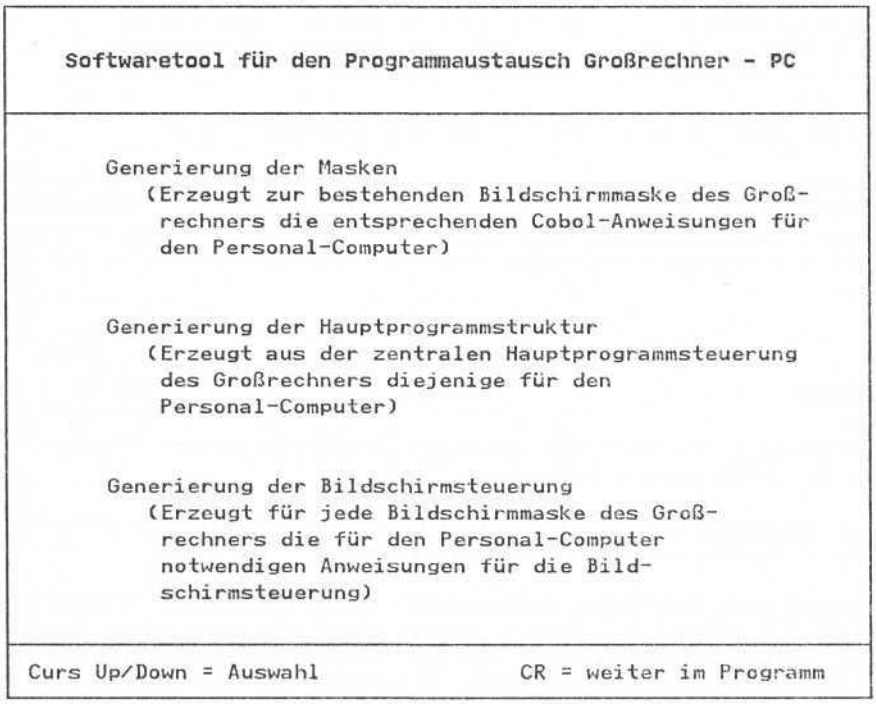

Abbildung 56: Hauptauswahlmenü zum Programmtransfer

Über einen mit der Cursor-Tastatur verschiebbaren Lichtbalken kann der Benutzer das gewünschte Umwandlungsprogramm aufrufen. Als zweite Maske erscheint nach jeder Auswahl die in Abbildung 57 auf Seite 86 dargestellte Input/Output-Steuerung.

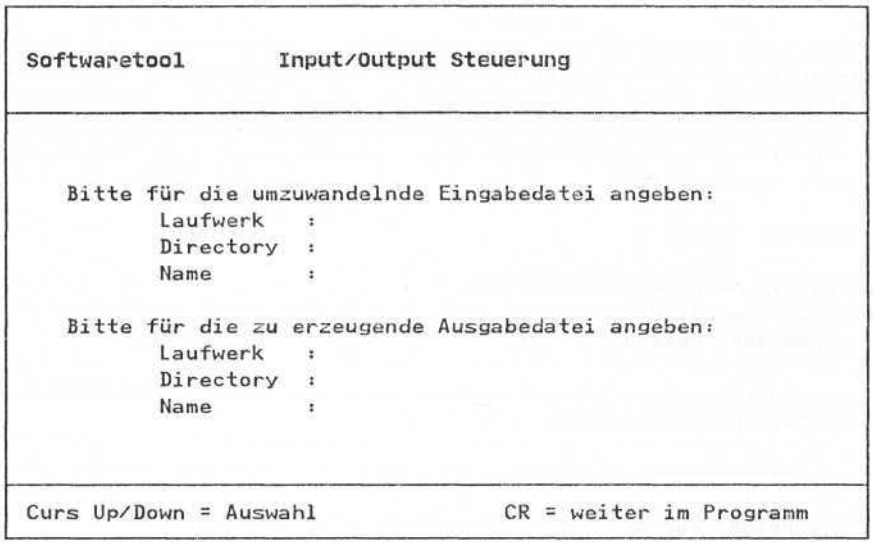

Abbildung 57: Ein-, Ausgabesteuerung für den Programmtransfer

Mit seiner Hilfe ist es möglich, vom Großrechner kommende Programmteile mit frei wählbarem Namen von beliebigen Speicherorten (Festplatte incl. beliebiger Verzeichnisse oder Diskettenlaufwerke) zu verarbeiten. Der erzeugte, manipulierte Programmteil kann wiederum mit gewünschtem Namen versehen an jedem Ort gespeichert werden.

Maskengenerator

Wie schon erläutert, erstellt und verwaltet das Hilfsprogramm "Message Format Service (MFS)" die Masken am Großrechner.

Der Personal-Computer baut jedoch die Maske über Cobol-Statements auf und benötigt dazu kein dem MFS vergleichbares Hilfsprogramm.

Die Maskengenerierung über MFS am Großrechner erfolgt über ein MFS-Programm (Job) (Abbildung 58 auf Seite 87). Dieses enthält neben den Job-Control-Anweisungen (Anweisungen an das Betriebssystem) für den Aufruf des MFS-Programms die zu erstellende Maske im Full-Screen-Format. Der Beginn von Ein- und Ausgabefeldern ist durch das Symbol "¬", die Art und Länge der Felder durch unterschiedliche Ziffern und das Symbol "-" gekennzeichnet.

```
         PRINT NOGEN
         ALPHA 'abcdefghijklmnopqrstuvwxyzüöä.-_ß,:()/%'
MHJA10   FMT
         DEV   FEAT=IGNORE,TYPE=3270-A4,DSCA=X'00A0',PFK=(FUN01,'01',
               '02','03','04','05','06','07','08','09','10','11','12',
               '13','14','15','16','17','18','19','20','21','22','23',
               '24')
         DIV
         DPAGE CURSOR=((5,15,CUR))
§BALIS-TUM Weihenstephan                §
§+++++++++++++++++++++++++++++++++++++++§
§   Schlagdatenerfassung Betrieb        §

Schlagnr.    :¬2- / ¬2  Schlaggröße:¬2----
Schlagname   :¬2--------------

Hauptfrucht :¬2---      Zw.frucht
 - Ertr.ziel:¬2------    - Art     :¬2---
Vorfrucht    :¬2---      - Menge   :¬2--
 - Rückst.   :¬2

 Organ. Düngung          Bodenwerte
 Art  dt/qm Monat     Art :¬2-  ¬2--------
¬2--- ¬2-- ¬2-        pH-W:¬2--
¬2--- ¬2-- ¬2-        P2O5:¬2-   K2O :¬2-
¬2--- ¬2-- ¬2-        MgO :¬2- BUJahr:¬2-
S_Nr : ¬2- / ¬2 ++++++++++ Auswahl =>¬A
¬1--------------------------------------
¬1--------------------------------------
6 << Frucht---- Codes -----Bodenart >> 7
8 << OD.Codes        nächster Schlag >> 1
A << Auswahlmenue(S)   Auswahlmenue >> 0


         FMTEND
         EJECT
MHJA100  MSG   TYPE=OUTPUT,SOR=(MHJA10,IGNORE),OPT=1,NXT=MHJA10I
SEG01    SEG
         MFLD  CUR,LTH=004
001031   L
         MSGEND
         EJECT
MHJA10I  MSG   TYPE=INPUT,SOR=MHJA10,OPT=1,NXT=MHJA100
SEG01    SEG
         MFLD  FUN01,LTH=002
001029   L
         MSGEND
         EJECT
```

Abbildung 58: Auszug aus einem MFS-Job zur Maskenerzeugung

Die Maske wird im Full-Screen-Modus, d.h. nach dem Löschen sämtlicher Job-Control und MFS-Anweisungen zum Personal-Computer übertragen. Ohne Maskengenerator müßte nun die Position aller Maskenzeichen ausgezählt und in Cobol-Statements der Data Division eingebaut werden. Dies ist eine langwierige und fehlerträchtige Arbeit. Deshalb übernimmt ein Maskengenerator diese Aufgabe, wobei der Benutzer ihn über das in Abbildung 56 auf Seite 85 dargestellte Auswahlmenü aufrufen kann.

Der Generator erzeugt nach der Angabe der Eingabe- und Ausgabedatei in Abbildung 57 auf Seite 86 die für den Personal-Computer passenden Cobol-Anweisungen und legt diese an einem beliebig wählbaren Ort (siehe Angabe der Ausgabedatei in Abbildung 57 auf Seite 86) ab. Die über die erzeugten Cobol-Anweisungen aufgebaute Maske kann nun auf Wunsch am Bildschirm dargestellt werden (Abbildung 59), um visuell zu überprüfen, ob die Maske richtig übertragen bzw. vom Maskengenerator richtig übersetzt worden ist. Die Ein- und Ausgabefelder sind dabei zur besseren Identifizierung invers dargestellt (in der Abbildung 59 nicht zu sehen).

```
 BALIS-TUM Weihenstephan
 ++++++++++++++++++++++++++++++++++++++
        Schlagdaten Betrieb

Schlagnr.    : .. /  .  Schlaggröße: .....
Schlagname   : ..............

Hauptfrucht : ....      Zw.frucht
 - Ertr.ziel: .......    - Art     : ....
Vorfrucht    : ....      - Menge   : ...
 - Rückst.   : .

 Organ. Düngung          Bodenwerte
 Art  dt/qm Monat     Art : ..    ........
 ....  ...  ..        pH-W: ...
 ....  ...  ..        P2O5: ..    K2O : ..
 ....  ...  ..        MgO : .. BUJahr: ..
S_Nr :  .. /  . ++++++++++ Auswahl => .
 ....................................
 ....................................
6 << Frucht---- Codes -----Bodenart >> 7
8 << OD.Codes        nächster Schlag >> 1
A << Auswahlmenü(S)      Auswahlmenü >> 0
```

Abbildung 59: Vom Maskengenerator erzeugte Bildschirmmaske am Personal-Computer

In einem weiteren Programmschritt hat der Benutzer durch Aufruf eines Editors die Möglichkeit, die zur Maske erstellten Cobol-Anweisungen zu modifizieren. Zur abschließenden Kontrolle erscheint dann nochmals die sich durch die vorgenommenen Änderungen ergebende Maske am Bildschirm.