Ein modulares, universelles Dialogsystem programmiert in PEARL

von Dr.-Ing. Horst Weber und Ing. (grad.) Karl-Dieter Egner, Frankfurt am Main

Zusammenfassung

Im Battelle-Institut e.V., Frankfurt, wird die Betriebssoftware für ein umfangreiches Automatisierungssystem - im Endausbau bestehend aus 8 Prozeßrechnern und 65 Mikroprozessoren - entwickelt [2].

Wie bei ähnlichen Vorhaben heute allgemein üblich, bestehen die Mensch-Maschine-Schnittstellen für Überwachung und Steuerung aus Videoterminals mit Tastaturen.

Zum Betrieb der Terminals werden Dialogprogramme realisiert, an die vielfältige Anforderungen gestellt werden. Es wird gezeigt, wie ausgehend von diesen Anforderungen ein modulares, universelles Dialogsystem spezifiziert und entwickelt wird. Die Entwicklungsschritte bestehen aus funktionalem Entwurf mit "top-down"-Zergliederung, Zuordnung von Programmodulen zu den Funktionsmodulen, Entwicklung der Programmodule, "bottom-up"-Integration und Inbetriebnahme.

Da die auftretenden Probleme sich generell mehr oder weniger ausgeprägt in allen Dialogsystemen ergeben und als Implementierungssprache Basis-PEARL gewählt wurde, kann das entstandene System leicht nach Portierung und Feinanpassung der Software in anderen Projekten eingesetzt werden.

Schlüsselwörter: modulares Dialogsystem, PEARL, Bildschirmmasken, Bildschirmformate, anpaßbar an Nutzerwünsche.

Summary

In the Battelle-Institut e.V., Frankfurt/M., the user software for an extensive automation system, composed of 8 realtime computers and 65 microprocessors, is being developed.

As is nowadays common, the man-machine-interface for supervision and control consists of video terminals with keyboards.

To operate the terminals, dialog programs are being produced which must meet manifold requirements. It will be shown in the following pages how, starting from these requirements, a modular and universal dialog system has been defined and developed. The steps in the development consist of: functional design with top down structure, association of program modules with functional modules, development of the program modules, bottom up integration and installation.

Because the problems which arise are more or less those which are met with in any dialog system, and the system has been written in Basis-PEARL, it can easily be used, after transfer and adaption, in other projects.

Keywords: modular dialog system, PEARL, video screen masks, video screen arrangements, adaptable to user requirements.

1. Nutzerwünsche

Grundlegend für die Entwicklung der Dialogsoftware sind die Anforderungen, die die späteren Nutzer der Anlage formulieren. Der entstehende Katalog ist die Basis für die Systemspezifikation.

Die Nutzerwünsche sind gegliedert in generelle Anforderungen und mehr spezielle Detailwünsche zu einzelnen Komponenten der Anlage.

1.1 Allgemeine Nutzerwünsche

Hierzu gehört, daß die Terminals von nicht-EDV-geschultem Personal bedient werden sollen. Da jedoch auch unmittelbare Steuerbefehle mittels Dialog eingegeben werden können, ist weitgehende Unempfindlichkeit der Programme gegen vorsätzliche oder irrtümliche Fehlbedienungen zu gewährleisten.

Alle notwendigen Manipulationen sollen möglichst einfach erlernbar, plausibel und anschaulich sein. Die Dialoge sollen in engen Grenzen geführt werden.

1.2 Spezielle Nutzerwünsche

Die speziellen Nutzerwünsche wurden in enger Zusammenarbeit mit dem Nutzer während der Ist-Analyse formuliert, wobei natürlich schon einige Aspekte der möglichen Realisierung Verwendung fanden.

Hierzu gehört die Strukturierung der Dialogführung auf verschiedene Arbeitsplätze mit wahlweiser Zuordnung von Aufgaben zu diesen Arbeitsplätzen. Die Aufgaben sollen sich noch in Unteraufgaben teilen lassen.

Die Dialoge sollen maskenunterstützt ablaufen, wobei die Masken leicht erstellt und geändert werden können. Den Ein- und Aus-

gabefeldern innerhalb der Masken sollen verschiedene Attribute zugeordnet werden können. Für die eingegebenen Daten wird verlangt, daß Typ- und Plausibilitätskontrollen variabel zugeordnet werden können. Schnittstellen zur Datenhaltung und zu den übrigen Prozessen sind festzulegen und zu realisieren. Die Software soll anpassungsfähig und übertragbar sein, um gegebenenfalls Änderungen von Komponenten und Anforderungen vornehmen zu können.

Zur Konkretisierung der Nutzerwünsche wurden off-line insgesamt 9 verschiedene Bildschirmmasken entworfen, wie man sie für den späteren Betrieb als notwendig erachtete.

Bild 1 zeigt einen solchen Entwurf mit einem Kommandofeld oben links und vier Eingabebereichen für Flugzeugdaten, Flugzieldaten, Funkkontakte und Übungsdaten. Mit Hilfe der Entwürfe konnten auch die gewünschten Reaktionen beim Auslösen von Cursor-Manipulationstasten festgelegt werden.

Weiterhin wurde definiert, wie das Verhalten bei Abbruch einer laufenden Eingabe und bei Ausgabe wichtiger Informationen über Prozesse sein soll.

2. Funktionsgliederung

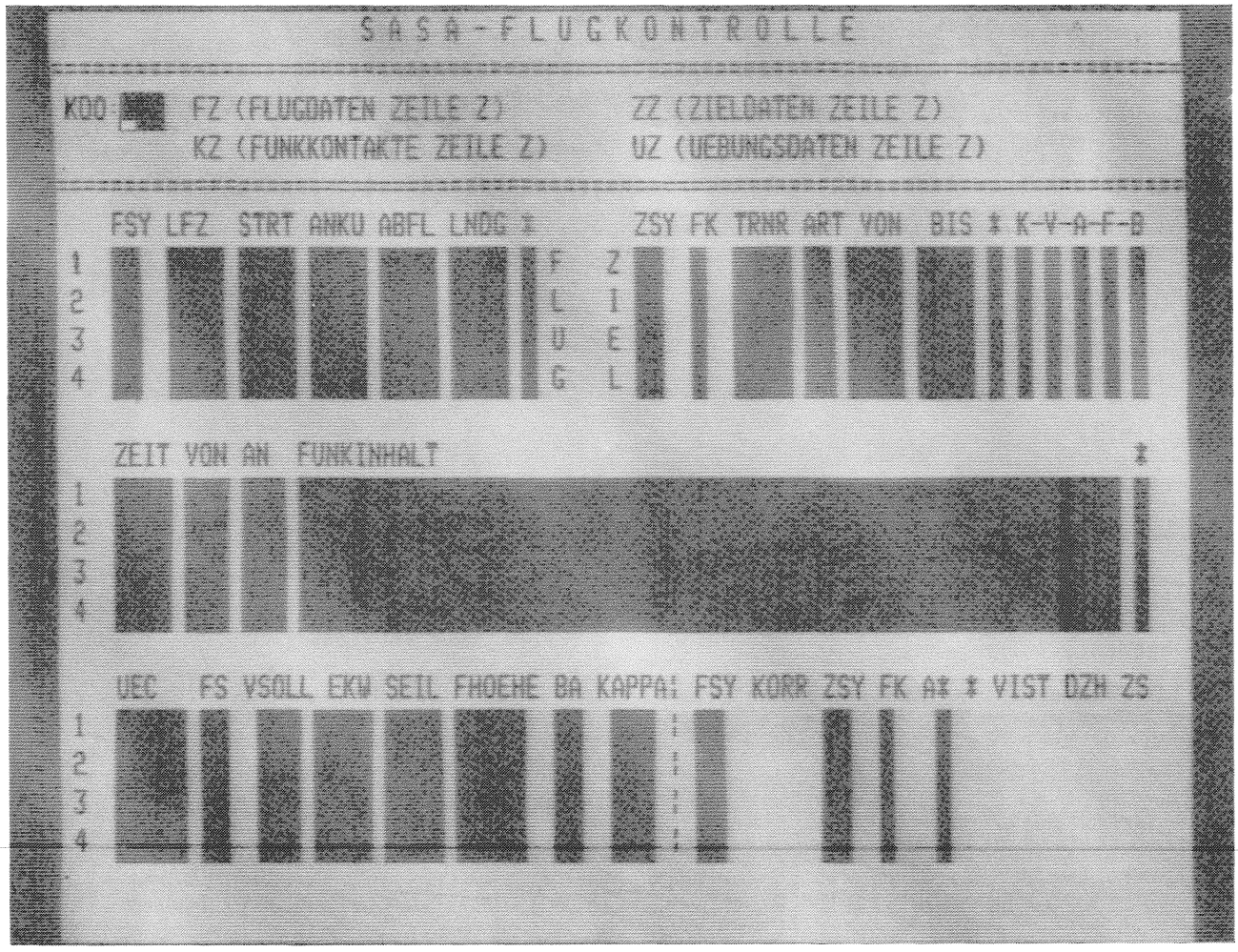
Ausgehend von den gesammelten Nutzerwünschen wurden die Funktionen des Dialogsystems strukturiert.

Bei dem zu realisierenden Projekt handelt es sich um die Automatisierung einer Schießplatzanlage und die in den Bildern verwendeten Begriffe stammen aus diesem Einsatzgebiet [2]. Die auftretenden Probleme sind jedoch genereller Natur.

2.1 Entwurf der Funktionsstruktur

Zunächst wurde die Aufteilung auf Arbeitsplätze, Aufgaben und Unteraufgaben vorgenommen (B i l d 2), die formal die Hierarchielevel 3, 4 und 5 erhielten. Level 1 ist die Funktion der Gesamtanlage und Level 2 die Interaktionen zwischen Bedienern und den ihnen zugeordneten Anlagenteilen. Auf gleichem Level befinden sich z. B. auch die Sicherheitsüberwachung und weitere Prozesse.

Die dem Personal der übenden Truppenteile zugeordneten Anlagenteile sind im Endausbau fünffach vorhanden. Die zur Realisierung der Funktionen notwendige Hardware ist auf einer weiteren Abbildung (B i l d 3) skizziert.



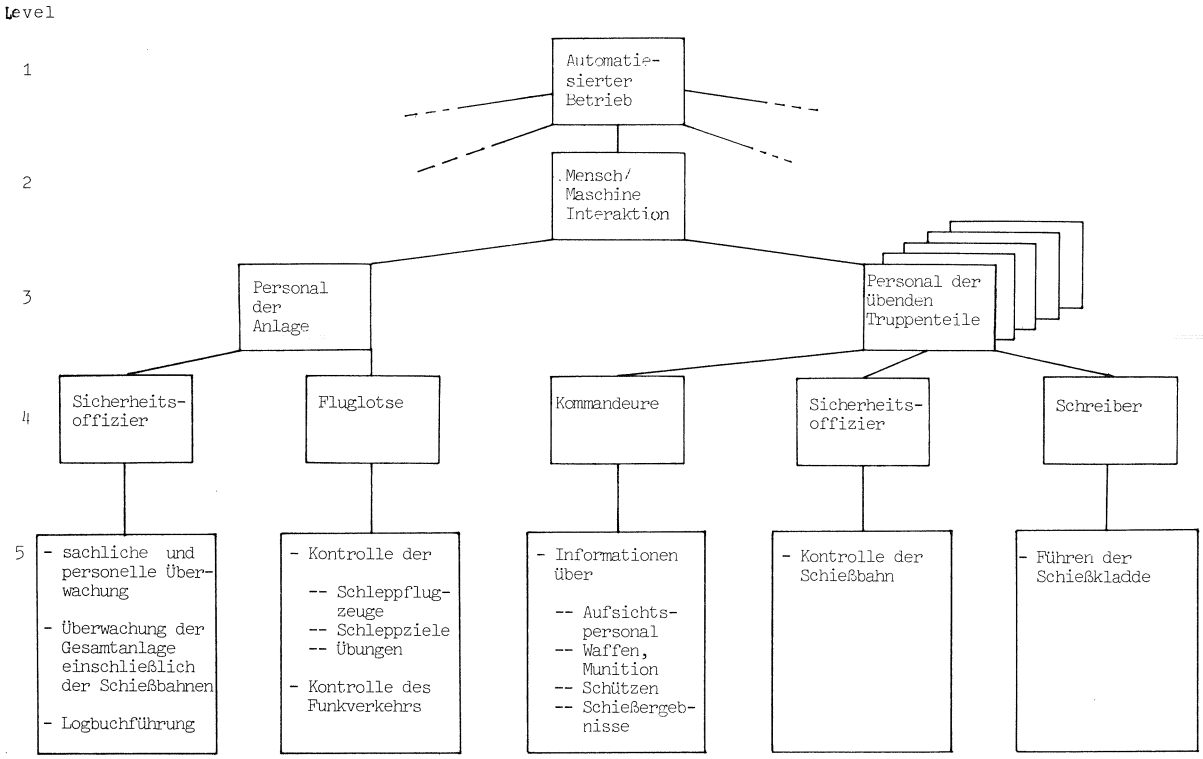


Bild 2: Funktionsstruktur

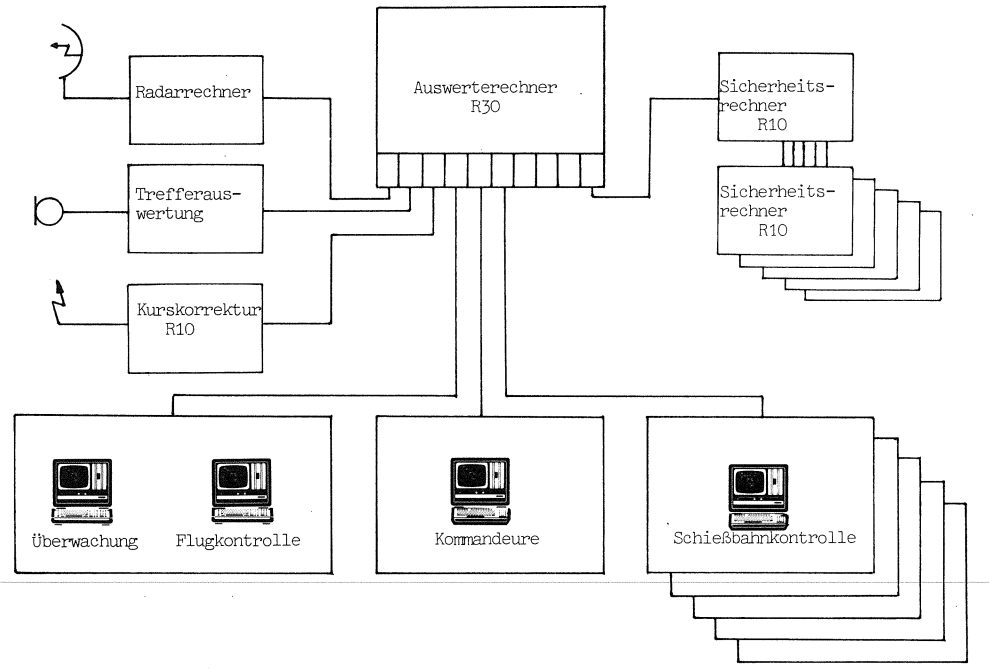


Bild 3: Hardware-Komponenten der Anlage

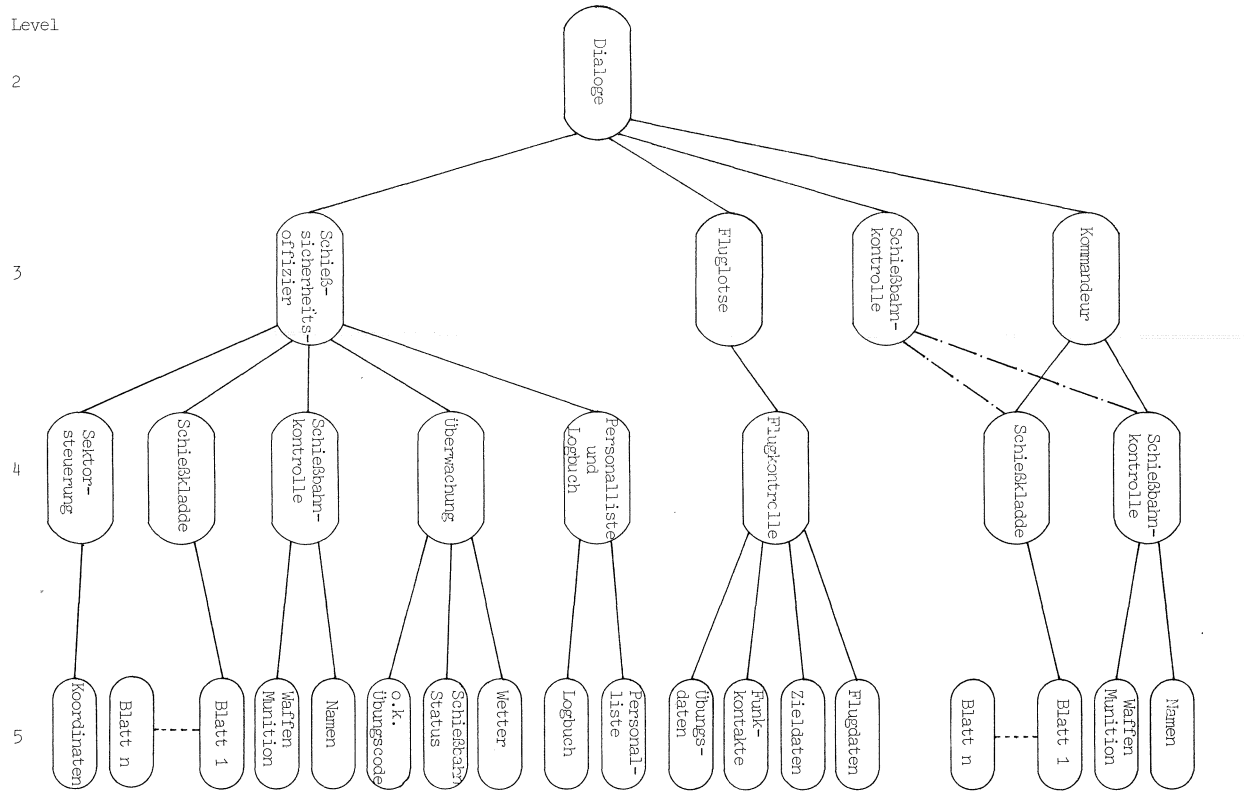


Bild 4: System-Hierarchie

2.2 Systementwurf

Aus der Funktionsstruktur ist die Systemhierarchie mit ihren "besteht-aus"-Relationen direkt ableitbar:

- Der Mensch-Maschine-Dialog findet an verschiedenen Arbeitsplätzen statt
- Einem Arbeitsplatz ist eine bestimmte Person oder Personengruppe und eine Reihe von Aufgaben (Formaten) sowie ein Bildschirmterminal zugeordnet
- Aufgaben bestehen aus verschieden vielen Unteraufgaben (Subformaten). B i l d 4 zeigt die Hierarchie des Systems:

  Level 3 entspricht den Arbeitsplätzen
  Level 4 entspricht den Formaten
  Level 5 entspricht den Subformaten

2.2.1 Datenorientierter Systementwurf

Nutzerwünsche, Funktionsstruktur und Systemhierarchie beschreiben die logisch zusammenhängenden, der Aufgabe entsprechenden Datenbestände und deren Bearbeiter. Diese sind Kandidaten für Komponenten und Module. Komponenten bestehen aus Modulen. Ein Modul kann ein Datenmodul oder ein Dateimodul sein [3]:

- Datenmodule beinhalten Operationen und Datenstrukturdefinitionen. Maskendatei, Abbilddatei und Dialog-Kontrolldateien (B i l d 5) sind über solche Datenmodule an das System angeschlossen
- Dateimodule realisieren den Anschluß von Dateien, auf die nur lesend oder schreibend zugegriffen wird. Sie erfordern lediglich Angaben über die Zugriffsart und den Satzaufbau. Über ein Dateimodul ist die Datenhaltung (B i l d 6) an das System angeschlossen

2.2.2 Anschluß an die Datenhaltung

Die logische Struktur der Datenhaltung beschränkt mögliche Integritätsverletzungen auf lokale Bereiche und ermöglicht die Zugriffskontrolle der Benutzer, denen eigene Bereiche zugewiesen sind (B i l d 7).

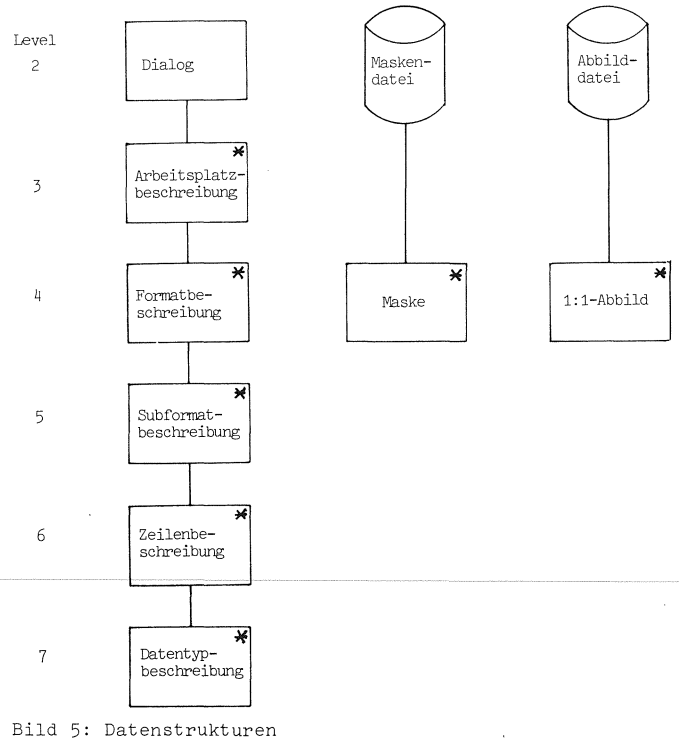


Bild 5: Datenstrukturen

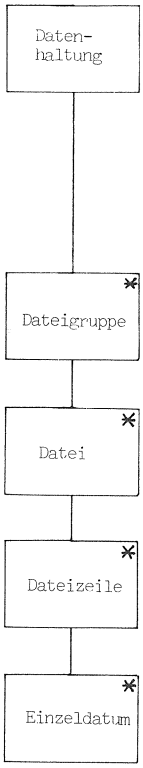


Bild 6: Datenhaltung

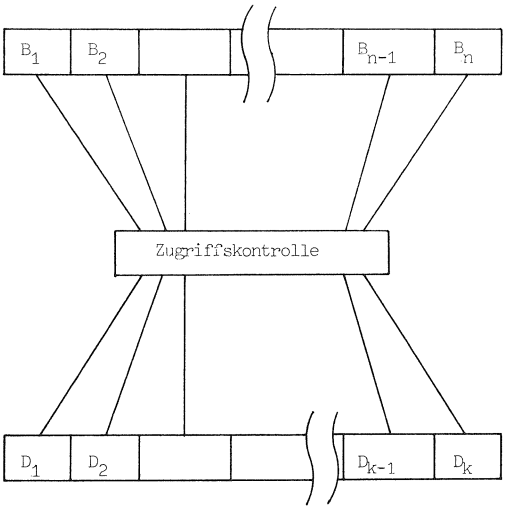


Bild 7: Anschluß der Datenhaltung

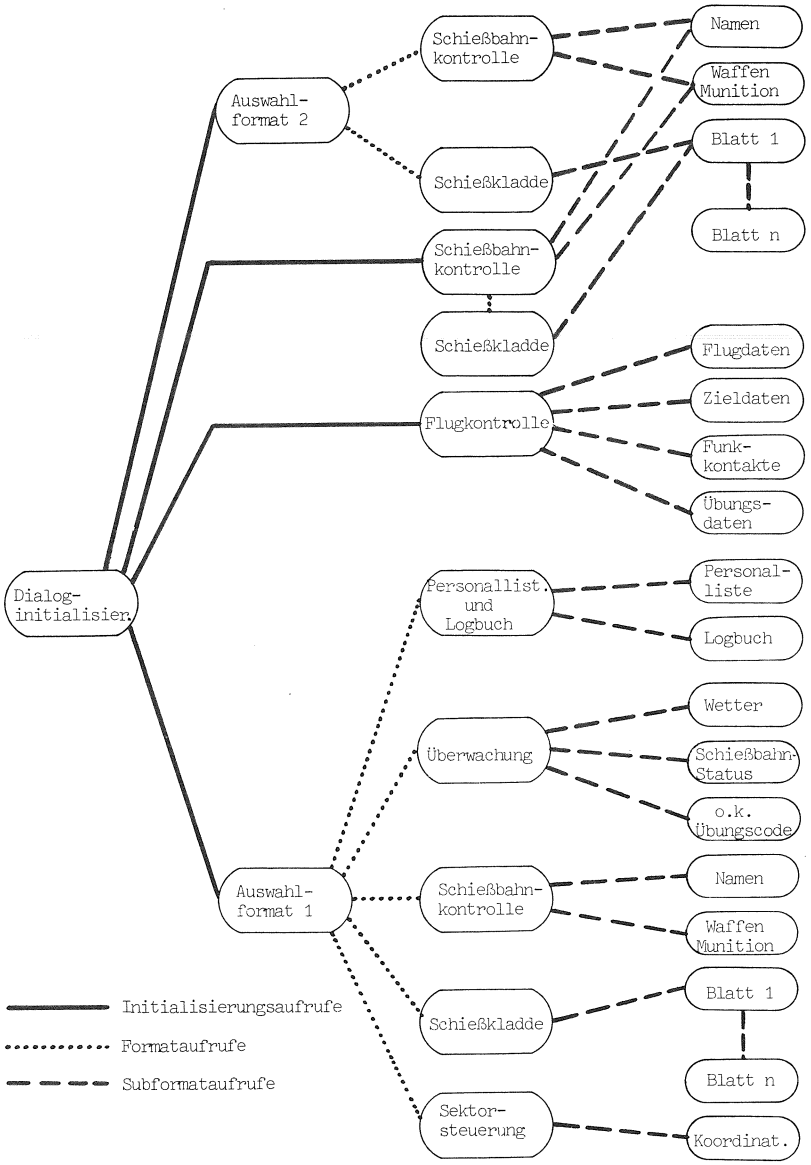


Bild 8: Aufrufstruktur

2.2.3 Übergangsorientierter Systementwurf

Man gewinnt den übergangsorientierten Systementwurf, indem man die in der hierarchischen Darstellung (B i l d 4, L e v e l 3) angegebenen Funktionen durch Modulbezeichnungen ersetzt. Diese Module stehen für die Formatauswahl, die an den so modifizierten Systemknoten erforderlich ist. Im übergangsorientierten Systementwurf werden die Komponenten- und Modulbeziehungen als "ruft-auf"-Relationen [4] betrachtet. Der übergangsorientierte Systementwurf (B i l d 8) zeigt drei Kommandolevel, in denen Initialisierungs- oder Auswahl-Aufrufe erfolgen:

- Initialisierungsaufrufe
  Level 2 (Dialog) ruft Level 3 (Auswahlformat 1, Auswahlformat 2) bzw. Level 4 (Flugkontrolle, Schießbahnkontrolle)

- Formataufrufe
  Level 3 ruft Level 4

- Subformataufrufe
  Level 4 ruft Level 5

2.2.4 Generalisierung und Moduldefinition

Die Verallgemeinerung des Systementwurfs erlaubt, den identifizierten Modulen generelle Eigenschaften zuzuordnen:

- Arbeitsplatz-Task
  Die Task erlaubt die Bearbeitung beliebiger Formate, sie ist an allen Arbeitsplätzen gleich

- Format-Prozedur
  Die Prozedur organisiert die Bearbeitung eines beliebigen Formates. Das Format ist ausschließlich durch Daten beschrieben. Die Formatbeschreibung enthält Angaben über die in diesem Format möglichen Subformate

- Subformat-Prozedur
  Die Prozedur organisiert die Bearbeitung eines beliebigen Subformates. Ein Subfor-

mat ist durch Daten beschrieben. Die Subformatbeschreibung enthält Angaben darüber, welche Zeilentypen in diesem Subformat verwendet werden

- Zeilen-Prozedur
  Die Prozedur organisiert die Bearbeitung eines beliebigen Zeilentyps. Die Zeilenbeschreibungen geben der Prozedur Kenntnis von der Struktur einer gelesenen Dateizeile und benennen die Datentypen

- Dateieingabe-Prozedur
  Die Prozedur kontrolliert die Eingabe eines der vorgesehenen Datentypen entsprechend der Datentypbeschreibung. Sie bedient sich dabei der "Ein-Zeichen-Eingabe"

- "Ein-Zeichen-Eingabe"-Prozedur
  Diese Prozedur erlaubt, ein Zeichen einzugeben, ohne zusätzliche Eingabe eines Dateneingabe-Steuerzeichens (DÜ, DÜM, DÜZ). Dadurch ist es möglich, jedes einzelne Zeichen der Kontrolle eines in PEARL geschriebenen Programmes zu übergeben

2.3 Konfigurierungsmittel

Die das Verhalten des Dialog-Systems bestimmenden Daten (Formatbeschreibungen, Subformatbeschreibungen, Zeilen- und Datentypbeschreibungen) sowie die Bildschirmmasken können mit Hilfe von Editoren erzeugt bzw. geändert werden.

2.4 Initialisierung

Die Initialisierung legt bei Arbeitsbeginn leere 1:1-Abbildungen an. Während der Bearbeitung eines Formates wird sein Abbild aktualisiert.

3. Modulspezifikationen

3.1 Formatmodul

Die Eingangsdaten des Formatmoduls (B i l d 9), die Formatbeschreibung, spezifizieren den Kommandoumfang, der in dem entsprechenden Format zugelassen ist. Daneben sind für jedes gültige Kommando die Reaktionen angegeben, die auf das Kommando zu erfolgen haben.

Die Reaktionen umfassen drei Gruppen:

- Kommandos wählen ein Subformat aus und leiten dessen Bearbeitung ein. Soll die Bearbeitung eines Subformates in einer bestimmten Zeile begonnen werden, so ist diese Zeilen-Nummer Bestandteil des Kommandos

- Kommandos stoßen das "blättern" in einem Datenbestand an, die "Blatt"-Nummer ist Bestandteil des Kommandos

- Kommandos lösen einen Formatwechsel aus. Die Kennzeichnung des neuen Formates ist als Format-Nummer oder Format-Kurzbezeichnung im Kommando enthalten

3.2 Subformatmodul

Die Eigenschaften eines Subformates sind in der zugehörigen Beschreibung festgelegt, die Subformatbeschreibung bestimmt das Verhalten des Subformatmoduls (B i l d 1 o):

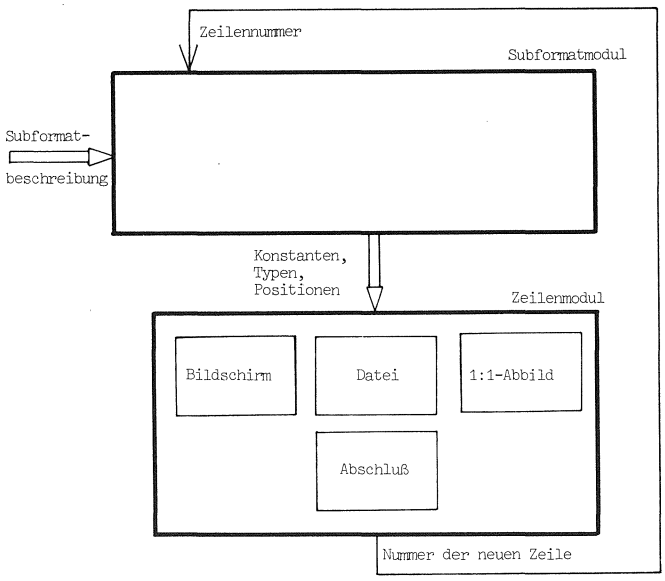


Bild 1o: Subformatmodul

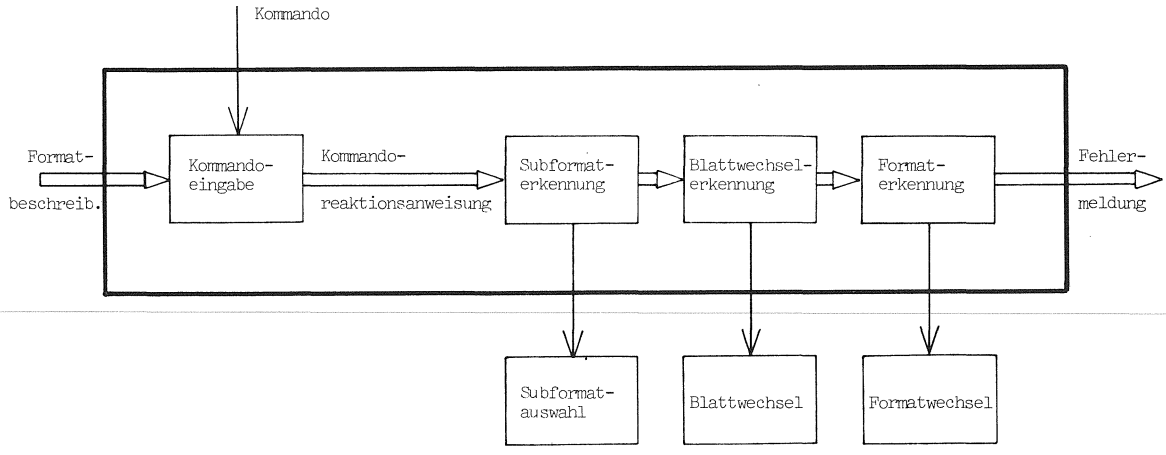


Bild 9: Formatmodul

- Subformatposition
  Einem Subformat (Unteraufgabe) wird ein Bildschirmausschnitt zugewiesen, es wird die Position des ersten Zeichens der ersten Zeile angegeben. Alle weiteren Positionsangaben sind relativ, sie beziehen sich auf diese Angabe

- Dateizugriff
  Die Parameter für den Zugriff auf die für dieses Subformat zuständige Datei werden beschrieben. Benötigt ein Subformat mehr als eine Datei als Datensenke, kann dies durch Aufspalten in verschiedene Subformate und Überlagern von im Prinzip beliebig vielen Subformaten realisiert werden

- Fensterbeschreibung
  Jedes Subformat und seine Repräsentation als Bildschirmausschnitt kann als verschiebbares Fenster auf einem Datenbestand angesehen werden. Die "Fensterbreite" und die Schrittweite bei Verschieben des Fensters werden hier angegeben

- Zeilenbeschreibung
  Im Rahmen der Subformatbeschreibung werden für jede Zeile zwei Angaben gemacht: Zeilenposition und Zeilentyp. Die Zeilenposition ist relativ, sie bezieht sich auf die absolute Positionsangabe für die erste Zeile. Die Aufeinanderfolge der Zeilenbeschreibungen entspricht der Bearbeitungsreihenfolge. Durch die Angabe der Zeilenposition ist die Bearbeitungsreihenfolge nicht an die Darstellungsreihenfolge gebunden. Der Zeilentyp ist ein Verweis auf die Zeilenbeschreibung, für gleich aufgebaute Zeilen existiert nur eine Beschreibung

3.3 Initialisierungsmodul

Der Systemstart ist mit der Aktivierung der den Arbeitsplätzen zugeordneten Tasks verbunden. Durch ein Initialisierungsmodul werden unmittelbar nach der Aktivierung leere 1:1-Abbildungen der Formatbildschirme angelegt. Die 1:1-Abbildungen ermöglichen einen schnellen Wechsel des Bildschirminhaltes bei Formatwechsel. Sie erlauben einen einfachen Zugriff, wenn von anderen Bildschirmen die Bearbeitung einer Aufgabe kontrolliert werden soll.

4. Programmentwurf

Dem Programmentwurf wurden die im Systementwurf definierten Strukturierungen zugrunde gelegt. Die ebenfalls dort festgelegten Datenstrukturen wurden verwendet.

4.1 Zuordnung von Programmmodulen zu Funktionsmodulen

Den einzelnen Arbeitsplätzen wurde jeweils eine Task zugeordnet, den Formaten eine Prozedur und den niederhierarchischen Funktionsmodulen weitere Prozeduren.

Dementsprechend wurden für Sicherheitsoffizier, Fluglotse und Kommandeure je eine Task geplant und für das Personal der übenden Truppe eine weitere Task, die bis zu fünfmal reproduziert werden kann.

Aufgrund der Entwurfsmethode konnten die Prozeduren auf gleichem Level auch gleich entworfen werden, das heißt, ein Dialog läßt sich jeweils aus gleichen Bausteinen konfigurieren. Dies erleichtert neben allen übrigen Aspekten auch wesentlich die vom Auftraggeber geforderte Dokumentation.

4.2 Einzelprogrammentwurf

Da die Datenstrukturen nach Jackson [1] dokumentiert worden waren, wurde das Programm dementsprechend entworfen. Ein Beispiel für einen solchen Entwurf befindet sich auf B i l d 1 1 .

Durch die weitgehende Strukturierung und wegen der sehr einfachen Datenstrukturen war es in vielen Fällen möglich, den Programmentwurf unmittelbar durchzuführen.

Im Hinblick auf die Dokumentation wurde das Programm unter Verwendung der strukturierenden Sprachelemente aus PEARL in Pseudocode entworfen. Die Anweisungen wurden durch Klartextzeilen beschrieben und durch Kommentare so ergänzt, daß sie identisch im Quellencode verwendet werden können.

Weiterhin gehört zur kompletten Dokumentation einer Prozedur die graphisch dargestellte Programmumgebung, bestehens aus rufenden und gerufenen Prozeduren, verwendeten Dateien und eventuellen Test- und Simulationsprogrammen.

5. Codierung

Nach den in den vorhergehenden Kapiteln beschriebenen Schritten stellte die Codierung keine Schwierigkeit dar. Sie wurde mit Hilfe eines Editors durchgeführt, wobei die Klartextzeilen durch PEARL-Anweisungen ersetzt wurden.

Die Bezeichner wurden weitestgehend im Hinblick auf Selbstdokumentation gewählt. Leider ist dies auf Modulebene nicht möglich, da bei dem verwendeten PEARL-System die Referenzen auf Modulebene durch einen Ladebinder aufgelöst werden, der nur 6-Zeichenlange Namen verarbeiten kann.

6. Erfahrungen, Wertung, Ausblick

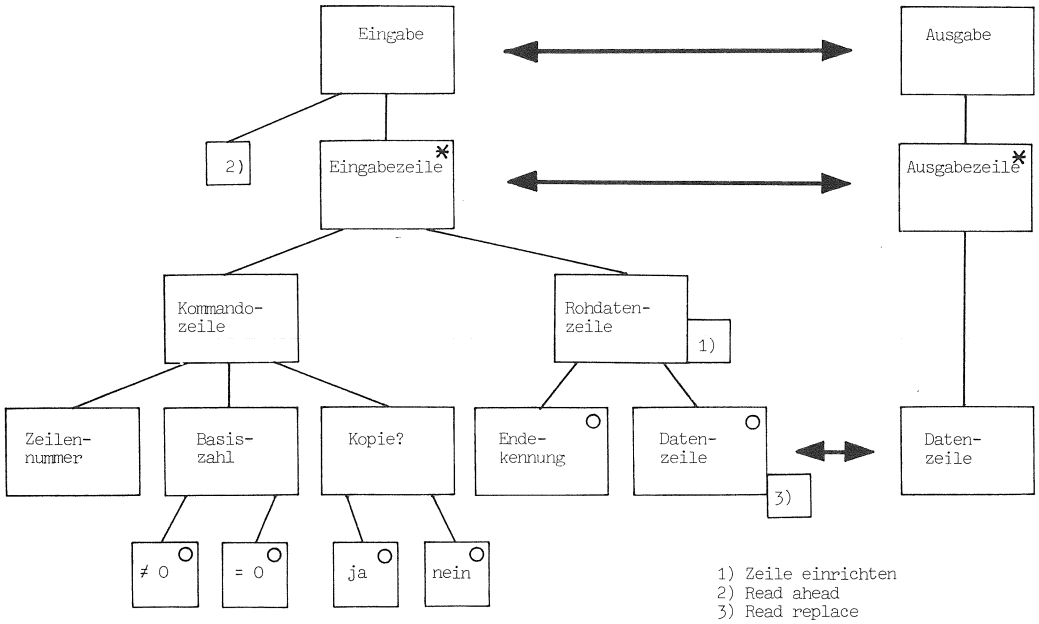
Die geschilderte Vorgehensweise ergab automatisch eine Strukturierung der Funktionen und im weiteren Verlauf ein modulares System von Tasks und Prozeduren.

In einem ersten Ansatz waren ausschließlich zugeschnittene Dialoge vorgesehen. Das realisierte Konzept ist dagegen wesentlich übersichtlicher und einfacher, demzufolge auch billiger in Erstellung und Wartung.

Das entstandene Dialogsystem ist portabel in dem Maße, wie auf anderen Zielmaschinen PEARL-Compiler und ähnliche Terminals zur Verfügung stehen. Lediglich das Assemblerprogramm für 1-Zeichen-Betrieb müßte ersetzt werden.

Falls Terminals eingesetzt werden, die zum Beispiel das Auslesen der Cursor-Position erlauben, oder "intelligente" Terminals mit Mikrocomputern, so läßt sich das Dialogsystem adaptieren, wenn entsprechende Module ausgewechselt werden.

Für das Dialogsystem kann man sich eine




```
/* INDAT
 MODULE;
 SYSTEM;
 /*        GERAETEVEREINBARUNGEN                              ** 0010 */
 PROBLEM;
 /*        GLOBALE SPEZIFIKATIONEN                            ** 0020 */
 /*        GLOBALE VEREINBARUNGEN                             ** 0030 */
 EINLES: TASK;
 /*        LOKALE  VEREINBARUNGEN                             ** 0040 */
 /*        ANFANGSZUWEISUNGEN                                 ** 0050 */

 /*        DATEI ERÖFFNEN                                     ** 0060 */
 /*        AUSGABE DER INFOTEXTE AUF DEM BILDSCHIRM           ** 0070 */
    WHILE TRUE REPEAT;
       /*       EINLESEN VON ZEILENNR., BASIS, KOPIE?        ** 0080 */
       IF KOPIE EQ JA THEN
          /*        ZEILE KOPIEREN                           ** 0090 */
       ELSE
          IF BASIS NE 0 THEN                                 ** 0100 */
             /*        ZEILE EINLESEN                        ** 0110 */
          ELSE
             /*        ZEILE VORBESETZEN                     ** 0120 */
          FIN;
          /*        ZEILE AUF DEM BILDSCHIRM DARSTELLEN      ** 0130 */
          /*        CURSOR POSITIONIEREN                     ** 0140 */
          /*        BILDSCHIRMZEILE EINLESEN                 ** 0150 */
          IF ( ENDEKENNUNG IN DER ZEILE  )   THEN            ** 0160 */
             /*        DATEI SCHLIESSEN                      ** 0170 */
             /*        BILDSCHIRM LOESCHEN                   ** 0180 */
             TERMINATE;  /*  TASK EINLES BEENDEN     */
          FIN;
          /*        ZEILE ABSPEICHERN                        ** 0190 */
       FIN;
    END;    /*  WHILE SCHLEIFE    */
 END;   /*  ENDE VON EINLES  */
 MODEND;
/*
```

Bild 11: Beispiel für einen Programmentwurf

Vielzahl weiterer Einsatzgebiete vorstellen, wie Datenerfassung an Ein- und Mehrplatzsystemen und verteilte Prozeßkontrolle.

In dem geschilderten Einsatzfall war der Bedienungskomfort bei der Off-line-Maskenerstellung und der Zuordnung der Attribute zu den Eingabefeldern ausreichend. Müssen die Eingabemasken aber häufig geändert werden, so empfiehlt es sich, ein Programm zu entwickeln, das die Zuordnung der Attribute dadurch unterstützt, daß das Maskenlayout unmittelbar herangezogen wird. Die Attribute könnten dann interpretativ zugeordnet werden.

Schrifttum

[1] J a c k s o n, Michael: Principles of Program Design. London: Academic Press (1975)

[2] W e b e r, Horst: Einsatz von PEARL bei der Software-Entwicklung für eine Fla-Schießplatz-Automatisierung. PEARL-Rundschau (198o) Nr. 4, S. 26/31

[3] D e n e r t, Ernst: Software-Modularisierung. Informatik-Spektrum (1979) Nr. 2, S. 2o4/218

[4] P a r n a s, D. L.: On a ´buzzword´: Hierarchical Structure. IFIP (1974), S. 336/339

Anschriften der Autoren

Dr.-Ing. Weber, Horst
Battelle-Institut e.V.
Abt. Technische Informatik
Am Römerhof 35
6ooo Frankfurt am Main 9o
Tel.: (o611)79o8-25o8

Ing. (grad.) Egner, Karl-Dieter
Battelle-Institut e.V.
Abt. Technische Informatik
Am Römerhof 35
6ooo Frankfurt am Main 9o
Tel.: (o611)79o8-2867