Einige Hilfsmittel zur Erstellung und Pflege von modularen Softwaresystemen

Heiner Halbach, SOPTIM Ingenieurbüro GmbH
Roermonder Straße 139
5100 Aachen

1. Einleitung und Überblick

In diesem Bericht wird ein Programmsystem vorgestellt, das im Hinblick auf Erstellung und Pflege von modular aufgebauter Software eine ganze Reihe von Hilfsmitteln und Techniken zur Verfügung stellt.
Nach einigen Angaben zur erforderlichen Systemumgebung schließen sich im nächsten Kapitel - quasi als Motivation für das Folgende - einige Bemerkungen zur modularen Programmerstellung an, aus denen sich Mindestanforderungen für eine realisierungsbegleitende Programmdokumentation ableiten lassen. Die Funktionen der angebotenen Hilfsmittel auf Quell- und Grundspracheebene werden in zwei weiteren Abschnitten dargestellt und anhand eines kleinen Beispiels erläutert.
Das gesamte Programmsystem ist ausnahmslos in FORTRAN geschrieben und in der SIEMENS-Implementierung auf einer Maschine R 30 E an den Compiler FTN-300 angepaßt. Die Ablaufsteuerung des Systems erfolgt unter dem gegenwärtig eingesetzten Betriebssystem AMBOSS-3 mit Hilfe von Kommandoprozeduren. Eine vergleichbare Steuerung unter dem ORG-300 PV ist unter Verwendung des Monitors möglich.
Zur Vervollständigung sei noch vermerkt, daß dieses Programmsystem ebenfalls auf kommerziellen Großrechnern sowie auf einem Mikro-Computer mit CP/M-Betriebssystem implementiert wurde.

2. Bemerkungen zur modularen Programmerstellung

Bei der Formulierung von Anforderungen an eine realisierungsbegleitende Dokumentation von Programmen, die soweit wie möglich automatisch erstellbar sein soll, ist es unerläßlich, eine klare Definition der zu dokumentierenden Einheiten zu geben. Hier existiert eine Spannweite, die vom einzelnen Befehl bzw. vom einzelnen Datum - und seiner Bedeutung im Gesamtablauf - bis zum kompletten Programm reicht. Es erweist sich als sinnvoll, jeweils eine Reihe von Befehlen bzw. Daten zusammenzufassen zu einer kleinsten Dokumentationseinheit "Modul" und die folgende Festlegung zu treffen :

Definition : Ein Modul ist ein Programmstück, welches dem Benutzer über eine wohldefinierte Schnittstelle eine oder mehrere wohldefinierte Funktionen zur Verfügung stellt.

Hiermit ist klar, daß ein Modul definiert ist durch seine Funktionen - nicht aber dadurch, wie die Funktionen realisiert sind. Genau die interne Realisierung wird vor dem Benutzer durch die Aufrufschnittstelle verborgen und tritt völlig in den Hintergrund. Die frühzeitige Dokumentation der angebotenen Funktionen und Schnittstellen ist aufgrunddessen unabdingbar, speziell falls mehrere Anwender parallel arbeiten wollen.

Der Dokumentationsschritt vom einzelnen Modul hin zum gesamten Programm wird beschrieben in der folgenden

Definition : Die Struktur eines Programmes ist die Zusammenfassung aller Benutzt-Relationen zwischen einzelnen Moduln dieses Programmes.

Wenn man voraussetzt, daß ein Programm zusammengesetzt ist aus einer Anzahl wohldefinierter Moduln, so kann die Programmarchitektur anhand der Aufrufstruktur als wesentlicher Bestandteil der Dokummentation und als Unterlage für die Programmpflege in automatisierter Form erstellt werden.

Zusammen mit den Funktionsbeschreibungen der Einzelmoduln ergibt sich daraus ein guter Überblick über den Leistungsumfang des erstellten Programmes.

Führt man zusätzlich noch eine Hierarchisierung in die Programmfunktionen ein, so gelangt man zu der Architektur in Abbildung 1.

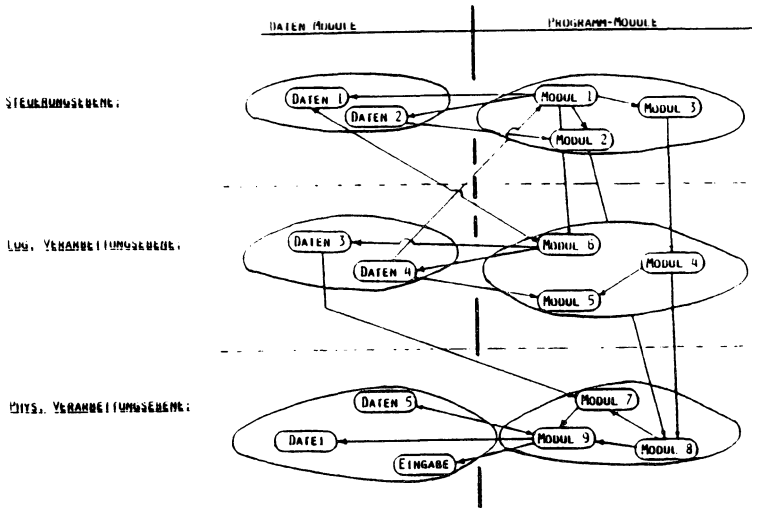

Abb.1: Einfacher modularer (hierarchischer) Programmaufbau

Hier wird ein Drei-Ebenen-Modell angenommen mit Daten-und Verarbeitungsmoduln auf jeder Stufe. Die Benutzt-Relationen können bestehen sowohl zwischen beliebigen Moduln der gleichen Ebene als auch zu den übrigen Ebenen. Ein Datenaustausch von Verarbeitungsmoduln ist über die Aufrufschnittstelle und über Datenmoduln (z.B. COMMON-Blöcke) möglich. Auch hier ist eine geeignete Dokumentation der Benutzt-Relationen erforderlich um die komplexen Beziehungen zwischen den Einzelmoduln transparent zu machen.

3. Hilfsmittel auf Quellprogramm-Ebene

Bei der üblichen Programmerstellung mit Hilfe von Editor, Übersetzer und Binder werden automatisch lediglich Compilations- und Binderprotokolle erstellt, die aber lediglich einen geringen Dokumentationswert besitzen. Aus diesem Grunde werden auf Quellprogramm-Ebene einige Hilfsmittel zur Verfügung gestellt, die die Erstellung modularer FORTRAN-Programme geeignet unterstützen.

In einem ersten Schritt kann Funktion und Aufrufschnittstelle der Einzelmoduln dokumentiert werden durch eine spezielle gegliederte Druckaufbereitung bestimmter Kommentarzeilen, sodaß einerseits die vollständige Dokumentation innerhalb der Programmquelle erfolgt und andererseits die Modulbeschreibung automatisch in Textbibliotheken und/oder als Druckausgabe in normierter Form zur Verfügung gestellt werden kann.
Durch ein anderes Werkzeug ist es möglich, die in den einzelnen Verarbeitungsmoduln benötigten Datenbereiche (COMMON-Blöcke) auszulagern, zentral zu pflegen und erst unmittelbar vor dem Übersetzungsvorgang zur Komplettierung des Quellenrumpfes an definierten Stellen einzufügen, wodurch sichergestellt ist, daß diese Datenmoduln stets einheitlich sind.
Es zeigt sich, daß bei "großen" Programmen mit ca. 70 Verarbeitungsmodulen das generelle Einfügen aller COMMON-Vereinbarungen problematisch ist, da der Compiler FTN-300 ggfs. eine Reihe von überflüssigen Zeigerfeldern generiert, sodaß ein unbenutzter Overhead von 10-15 KW keine Seltenheit ist.
An dieser Stelle setzt eine weitere Programmsystemkomponente an, die den Quellencode analysiert und in einer normierten Form hinterlegt. Hieraus kann bei der Einfügeverarbeitung erkannt werden, ob einzelne COMMON-Blöcke in einem Verarbeitungsmodul benutzt werden, sodaß beträchtliche Programmverkürzungen durch Nichteinfügung entstehen können. -
Neben der standardisierten Analyse aufgrund einer vorgebbaren Sprachsyntaxdefinition können eine Reihe von Reorganisationsmaßnahmen durchgeführt werden, die zu einer Standardisierung und besseren Lesbarkeit der Programmquelle führen :
- Einrücken von Folgeanweisungen bei IF- oder DO-Anweisungen
- Neunumerierung von Marken und Sammeln von FORMAT-Anweisungen
- Standardisierung der Notation der einzelnen Anweisungen

Schließlich wird aufgrund der Quellen-Analyse durch eine weitere Komponente eine quellenorientierte Programmstruktur aufgebaut, welche eine klar gegliederte Darstellung liefert für
- die Aufrufrelationen zwischen den Verarbeitungsmoduln und
- die Benutzung von COMMON-Blöcken und Variablen.

Ausgehend von der Programmquelle kann somit ein erster Ansatz zur automatischen Dokumentation der Softwarearchitektur gemacht werden. Dabei können allerdings Module, die nicht als Quellencode vorliegen oder aus anderen Programmiersprachen erzeugt werden, nicht vollständig erfaßt werden. -
Aufgrund der normierten Quellen-Analyse können schließlich - parallel zur Einfügung von COMMON-Blöcken - gezielt spezielle DEBUG-Anweisungen (Haltepunkte, Testausdrucke usw.) in den Modulrumpf eingebracht werden, die als Testhilfe auf FORTRAN-Ebene trotz ihrer Einfachheit ein erhebliches Hilfsmittel darstellen, da sie an symbolischen Adressen (Marken, Unterprogrammsprüngen) orientiert sind.

4. Hilfsmittel auf Grundsprache-Ebene

Die Dokumentation der Einzelmodule tritt bei den folgenden Betrachtungen völlig in den Hintergrund; denn hier kann es nur darum gehen, die Unzulänglichkeiten in der Darstellung der Quellprogrammstruktur durch eine Analyse der Grundsprache-Moduln, -Bibliotheken und Bindersteuerkarten stufenweise abzubauen um so zu der vollständigen Darstellung der Programmarchitektur zu gelangen.
Analog zur Quellen-Analyse wird durch eine zusätzliche Komponente des Programmsystems aufgrund der X- und A-Karten in der Grundsprache eine normierte Beschreibung der Benutzt-Relationen zwischen den einzelnen Moduln erstellt, aus der dann ein vollständig(er)es Abbild der Programmstruktur erzeugt werden kann. Es ist dabei parametrierbar, wieviele Stufen der Bibliothekshierarchie bearbeitet werden sollen, sodaß die Beschreibungstiefe an dieser Stelle variabel ist.
Die Abbildung 2 enthält eine schematische Darstellung des Programmsystems mit allen in diesem Bericht erwähnten Komponenten für die Verarbeitungen auf Quell- und Grundspracheebene, wobei die Datenübergabe zwischen den Einzelprogrammen über Bibliothekselemente (im Sinne von BIBEAS) bzw. Dateien erfolgt.

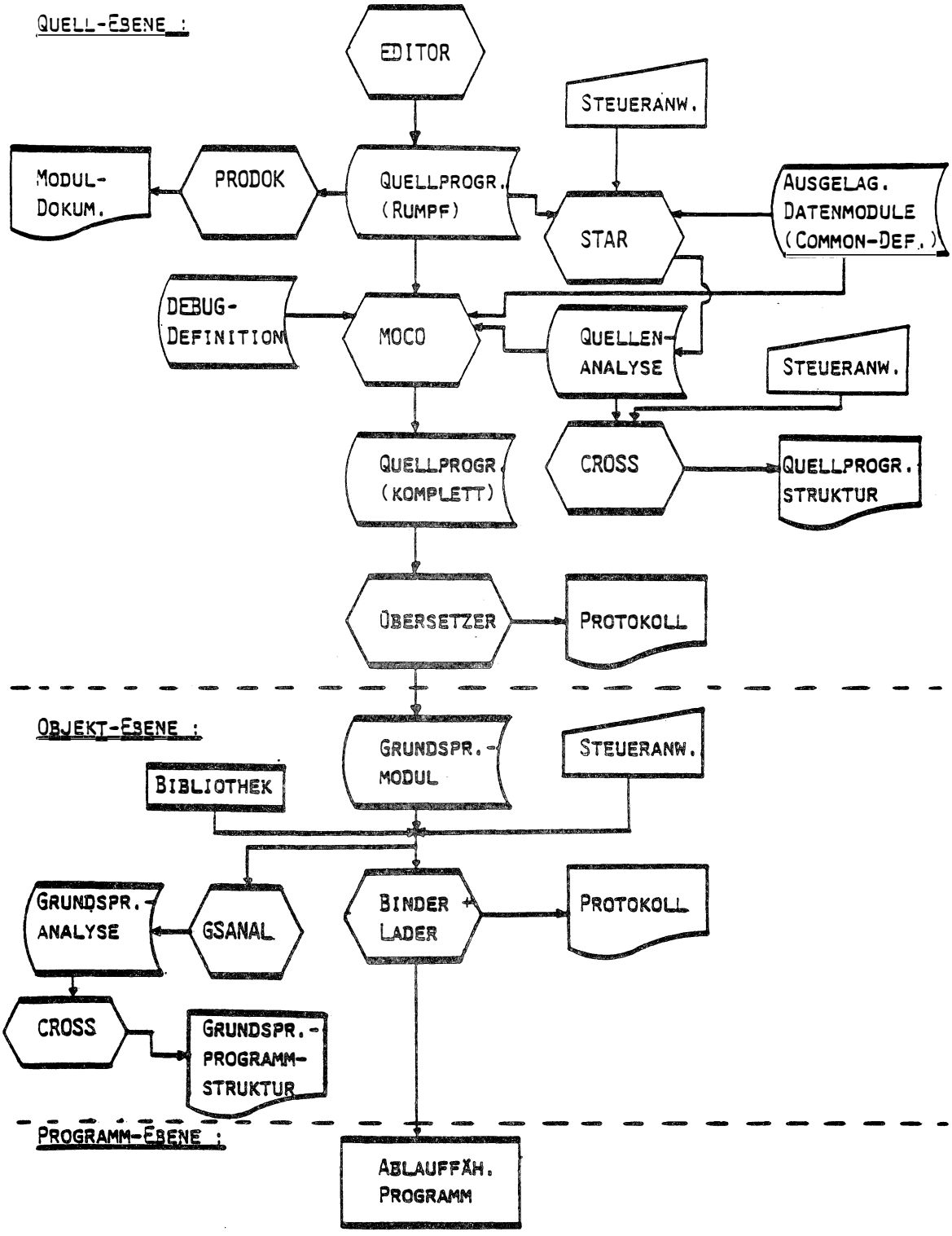

Abb. 2: Darstellung der Werkzeuge im Programmerstellungsprozeß

5. Erläuterungen anhand eines Beispiels

Im Anhang sind zwei Druckausgaben der hier vorgestellten Dokumentationsmittel beigefügt. Es handelt sich dabei um die Programmstruktur eines einfachen FORTRAN-Programmes aufgrund der Quellen-Analyse (Abb.3) sowie einen Auszug aus der Grundsprache-Analyse (Abb.4). Die Angabe eines '*' besagt, daß der entsprechende Modul bereits früher in der Struktur aufgetreten ist, sodaß auch alle von ihm benutzten Moduln bereits ausgegeben wurden. Neben den Strukturen werden außerdem jeweils Referenzlisten über die rufenden und gerufenen Moduln sowie über die Benutzung von COMMON-Blöcken ausgegeben.
Es ist klar, daß in der quellen-orientierten Programmstruktur wesentliche Aufrufrelationen fehlen, die nur über die Grundsprache-Analyse abgeleitet werden können. Andererseits kann die Vielzahl beispielsweise der FORTRAN-Laufzeitroutinen zu einer sehr unübersichtlichen Struktur führen, sodaß gegebenenfalls einzelne Bibliotheken aus der Grundsprache-Analyse ausgeschlossen werden müssen.

6. Zusammenfassung

Das in diesem Bericht beschriebene Programmsystem stellt speziell für den FORTRAN-Programmierer eine Reihe von Werkzeugen bereit, die eine geeignete Dokumentation der Funktionen und der Architektur von modular aufgebauter Software in einer automatisierten Form ermöglicht und die getrennte Pflege von Verarbeitungs- und Datenmoduln (COMMON-Blöcke) stark unterstützt, sodaß eine definierte Datenschnittstelle einheitlich für alle Teile eines Programmes existiert.

7. Anhang

```
UNTERPROGRAMMSTRUKTUR GRUDOK

ZEILEN-REF.    PROGRAMMNAMEN (HIERARCHIEORDNUNG ==> )

     1          GRUDOK
     2            .   BBEROE
     3            .   BDANAL
     4            .   BDINF
     5            .   GDLESE
     6            .       .   BBENDE
     7            .       .   BBLESE
     8            .   GDSCHR
     9 /   6      .       .   BBENDE   *
    10 /   7      .       .   BBLESE   *
    11            .       .   BBSCHR
```

Abb. 3: Programm-Struktur aus Quellen-Analyse

```
UNTERPROGRAMMSTRUKTUR GRUDOK

ZEILEN-REF.    PROGRAMMNAMEN (HIERARCHIEORDNUNG ==> )

     1          GRUDOK
     2            .   AFOA
     3            .   AFOI
     4            .   AFOZ
     5            .   BBEROE
     6            .       .   BBFEHL
     7            .       .       .   ADRE
     8 /   3      .       .       .   AFOI     *
     9 /   4      .       .       .   AFOZ     *
    10            .       .       .   BIBFTN
    11            .       .       .       .   INFO
    12            .       .       .       .   LBIA
    13            .       .       .       .   PARA
    14            .       .       .   EFOZ
    15            .       .       .   FGIK
    16            .       .       .   FORM
    17            .       .       .   IWFE
    18            .       .       .   MIFO
    19 /  13      .       .       .   PARA     *
    20            .       .       .   STOP
    21 /  10      .       .       BIBFTN   *
    22 /  13      .       .       PARA     *
    23            .       .       XCH1
    24            .       BDANAL
    25 /   7      .       .       ADRE     *
    26 /  11      .       .       INFO     *
```

Abb. 4: Programm-Struktur aus Grundsprache-Analyse