

Problemorientierte Systemunterlagen für das Rechnersystem K1600

Inhaltsverzeichnis

1.	Übersicht	3
1.1.	Einsatzgebiet	3
1.2.	Anwendungsbedingungen	5
2.	Konstruktionsprinzipien der Softwareprodukte für K 1600	6
2.1.	Bestandteile der Softwareprodukte.....	7
2.2.	Die Programmbasis	8
2.3.	Die Datenbasis.....	11
3.	Allgemeiner Aufbau von TESO 1600.....	12
3.1.	Wirkungsweise von TESO 1600	12
3.2.	Übersicht über die Systemkomponenten des TESO 1600	14
4.	Software-Entwicklung mit TESO 1600	17
4.1.	Aufgabenstellung und Randbedingungen.....	17
4.2.	Technologische Vorbereitung der Software-Entwicklung	17
4.3.	Ablauf der Software-Entwicklung	19
5.	Literatur	23

Abbildungsverzeichnis

Abbildung 1: Übersicht zur POST 1600	3
Abbildung 2: Bestandteile der Softwareprodukte	7
Abbildung 3: Das Programm als Modulhierarchie.....	8
Abbildung 4: Standardisierter Modulaufbau	9
Abbildung 5: Kommunikation zwischen Programm- und Datenbasis.....	12
Abbildung 6 : Prinzipielle Wirkungsweise des TESO 1600	13
Abbildung 7 : TESO-Bibliotheken eines POS-Entwicklerkollektivs	19
Abbildung 8 : Beispiel für den Ablauf der Software-Entwicklung mit TESO 1600.....	20
Abbildung 9 : Prinzipbeispiel für TESO-Quellmodul.....	22

Tabellenverzeichnis

Tabelle 1: Bereitstellungsform der POST 1600.....	4
Tabelle 2: Logische Grundstrukturen des TESO 1600	100
Tabelle 3: Logische Grundstrukturen des TESO 1600	111
Tabelle 4: Codiermittel des TESO 1600.....	14
Tabelle 5: TESO- Aktionen	166

1. Übersicht

1.1. Einsatzgebiet

Problemorientierte Systemunterlagen für das Rechnersystem K1600 werden als technologisches System zur Software-Entwicklung bereitgestellt. Dieses im folgendem mit POS-Technologie K1600 (abgekürzt **POST 1600**) bezeichnete Produkt besteht aus manuellen (methodischen) und maschinellen (programmtechnischen) Mitteln (Abb.1) zur rationellen Entwicklung konkreter Anwendersoftware (Softwareprodukte) für den Einsatz des Rechnersystems K1600.

Die manuellen Mittel der POST 1600 sind anwendungstechnisch aufbereitete Methoden und Verfahren. Sie dienen vor allem der technologischen Unterstützung des Entwerfens, Testens und Dokumentierens der Software. Die maschinellen Mittel sind zu einem

Technologischen Programmsystem zur Software-Entwicklung (TESO 1600) zusammengefasst. TESO 1600 unterstützt das Implementieren, Testen und das zugehörige Dokumentieren, Verwalten und Kontrollieren der Software.

Die Erscheinungsform der POST 1600 zeigt Tabelle 1.

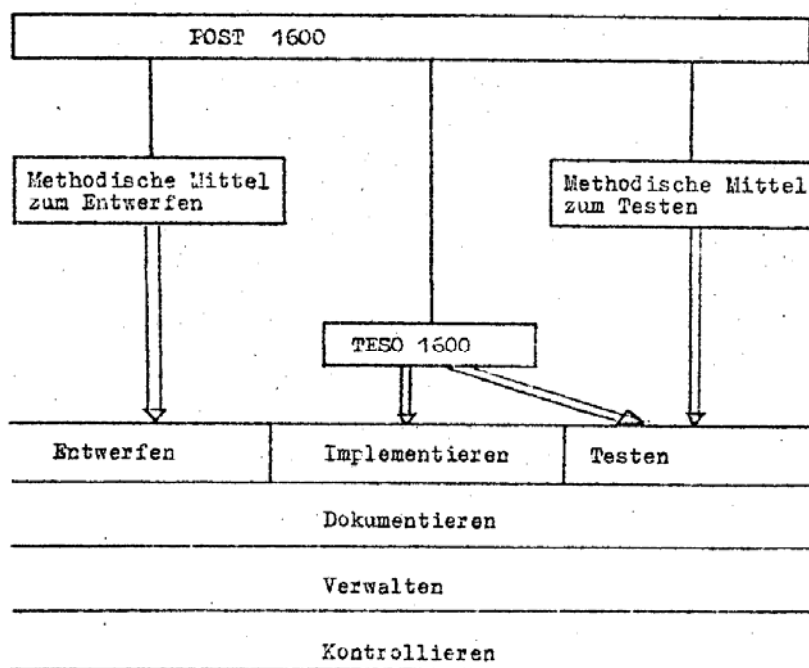


Abbildung 1: Übersicht zur POST 1600

Vertriebs- einheit	Bezeichnung	Bestandteile
1	Konzeption der POS-Technologie und allgemeine methodische Mittel	<ul style="list-style-type: none"> - Konzeption der POS-Technologie - Begriffe - Allgemeine technologische Mittel <ul style="list-style-type: none"> • Funktionsorientierte Entwurfsmethode • Datenorientierte Entwurfsmethode • Dokumentationssystematik für problemorientiertes Entwerfen • Pseudocode • Struktogramme • HIPO Entwurfsmethode - Allgemeine technologische Mittel zum Testen <ul style="list-style-type: none"> • Testmethodik
3	POS-Technologie K1600 - Allgemeine methodische Mittel	<ul style="list-style-type: none"> - Übersicht über POST 1600 - Konstruktionsprinzipien der Software für K1600 - Kurzcharakteristik TESO 1600 - Ablauf des Software-Entwicklungsprozesses mit POST 1600 - Methodische Mittel: Programmorientiertes Entwerfen - Methodische Mittel: Dokumentationssystematik der POST 1600
4	Technologisches Programmsystem zur Software-Entwicklung für Makroassemblersprache TESO 1600 (MAC)	<ul style="list-style-type: none"> - TESO 1600 (MAC) auf Datenträger - Anwendungsbeschreibung - Programntechnische Beschreibung
5 u. weitere	Technologisches Programmsystem zur Software-Entwicklung für höhere Programmiersprachen des K1600 TESO 1600 (XXX)	<ul style="list-style-type: none"> - TESO 1600 (XXX) auf Datenträger - Anwendungsbeschreibung - Programntechnische Beschreibung

XXX - Kurzname der höheren Programmiersprache

Tabelle 1: Bereitstellungsform der POST 1600

Außer für die Makroassemblersprache von K1600 wurde TESO 1600 auch für die höheren Programmiersprachen FORTRAN und COBOL entwickelt.

Durch TESO 1600 unterstützte Lebenszyklen eines Softwareproduktes sind

- Analyse
- Spezifikation
- Entwurf
 - Funktioneller Entwurf
 - Technischer Entwurf
 - Programmentwurf
 - Modulentwurf
- Implementation

- Test
- Fertigstellung
- Einführung
- Dauerbetrieb
- Wartung/Reparatur

TESO 1600 steht zur allgemeinen **Nutzung seit 1984** zur Verfügung.

1.2. Anwendungsbedingungen

TESO 1600 ist auf einem Mikrorechnersystem K1630 ab einer Hauptspeichergröße von mindestens 64 K Worten ablauffähig und setzt das Betriebssystem MOOS 1600, Version 1.2. voraus mit einem Kassettenplattenlaufwerk und einem Bedienterminal als Mindestkonfiguration.

Weiterhin können bedient werden

- weitere Kassettenplattenlaufwerke
- Magnetbandeinheiten
- weitere Terminals
- Parallel- oder Seriendrucker
- Lochband-Lese/Stanz-Einheit
- Magnetbandkassettengeräte
- Folienspeichereinheiten

In Abhängigkeit von der Einsatzkonzeption werden folgende Betriebssystemkomponenten benötigt:

- Steuerprogramm MOEX 1600
- Taskbilder TKB 1600
- Macroassembler MAC 1600
- FORTRAN-Compiler FOR 1600
- COBOL-Compiler COB 1600
- Bibliothekar LBR 1600
- Dateizugriffsroutinen FCS 1600
- Satzzugriffssystem RMS 1600
- Editor EDI 1600
- Dateitransferprogramm PIP 1600
- Dateitransferprogramm SLP
- Lochbandeingabeprogramm PTI 1600
- Dateiumwandlungsprogramm FLX 1600
- Dateiumwandlungsprogramm FEX 1600
- Dateiausgabeprogramm DMP 1600
- Indirektkommandodatei-Prozessor des MOOS 1600
- Systembibliotheken des MOOS 1600

- POST-MORTEM-DUMP-Programm PMD 1600
- Testprogramm DEP 1600 bzw. ODT 1600
- Plattenkopierprogramm DSC 1600

TESO 1600 ist generierungsfähig bezüglich

- Hardware-Konfiguration
- gewünschter technologischer Funktionen
- Installation auf konkreten Nutzersystemen

Die Nutzung von TESO 1600 kann auf verschiedene Weise erfolgen:

1. Installation des TESO 1600 auf dem jeweiligen K1600-Anwendungssystem. Die Aktivierung kann in den Phasen der Programmentwicklung und -wartung erfolgen. Während des Nutzbetriebes des Rechners befindet sich TESO auf dem externen Speicher und belastet die zentrale Verarbeitungseinheit nicht. Diese erste Form realisiert die Forderung der Anwender nach Nutzung der eigenen Anlage zur Software-Entwicklung.

2. Installation des TESO 1600 auf einem speziellen Software-Entwicklungssystem.

Ein solches System wird vorrangig von Institutionen betrieben, die eigenständige Software-Produktionsbereiche für K1600 haben. Dabei wird als Arbeitsmittel für diese Bereiche ein System K1600, das aus Gerätesystem, Betriebssystem und TESO 1600 besteht, für eine Gruppe von 20 - 30 Software-Entwicklern betrieben.

3. Installation des TESO 1600 auf K1600-kompatiblen Rechnern.

TESO 1600 kann auf allen Rechnern ständig oder zeitweise installiert werden, die zu dem Rechnersystem K1600 mit MOS 1600 kompatibel sind. Auf diesen Rechnern kann die Programmentwicklung für den Zielrechner K1600 erfolgen. Diese Form der Nutzung wird insbesondere in der ersten Phase des Einsatzes des Rechnersystems K 1600 eine Rolle spielen. Auf den SKR- Rechnern CM3, CM4 kann damit die Programmentwicklung für K1600 erfolgen.

Beim Vorhandensein eines K1600 mit dem Betriebssystem MOOS 1.2 und mehreren Bildschirmterminals oder einem äquivalenten System ist der Mehrnutzer-Betrieb des TESO 1600 möglich.

2. Konstruktionsprinzipien der Softwareprodukte für K 1600

Ebenso wie z.B. die Technologie des Maschinen- und Gerätebaus abhängt von den Konstruktionsprinzipien der Maschinen und Geräte ist die Software-Technologie in starkem Maße abhängig von den Konstruktionsprinzipien der mit ihrer Hilfe herzustellenden Software. Diese legen fest:

1. aus welchen Bestandteilen die Software besteht und wie die Bestandteile zum Software-Produkt miteinander zu verbinden sind
2. welche Grundregeln bei der Software-Entwicklung gelten.

Die Bestandteile und die Beziehungen zwischen den Bestandteilen der Software existieren sowohl in logischer als auch in physischer Form.

Die physische Form der Bestandteile wird durch Speicherbereiche und Code, die

Beziehungen zwischen den Bestandteilen durch Speicherbereichs- und Codehierarchien einschließlich der bestehenden Verknüpfungen repräsentiert.

Beschrieben werden die Bestandteile der Software durch Daten und Funktionen.

Ziel der Softwareentwicklung ist die Realisierung der physischen Form. Die logische Form wird benutzt, um die Softwareentwicklung zu erleichtern, zu systematisieren und zu verallgemeinern. Während des Software-Entwicklungsprozesses werden häufig verschiedene, der jeweiligen Entwicklungsphase (z.B. Spezifikation, Entwurf, Implementierung) angepasste logische Formen für die Beschreibung der Bestandteile und Beziehungen zwischen den Bestandteilen der Software benutzt, die entsprechend dem Ablauf des Entwicklungsprozesses ineinander und letztlich rechnergestützt in die physische Form überführbar sind.

Ziel der POST 1600 ist es, den Software-Entwickler weitgehend von der Kenntnis der physischen Form zu entlasten und mit logischen Formen der Software zu arbeiten, die an die praktische menschliche Denk- und Arbeitsweise gut angepasst sind.

2.1. Bestandteile der Softwareprodukte

Im allgemeinen Fall besteht ein Softwareprodukt aus den Hauptbestandteilen **Programmbasis** und **Datenbasislösung** (Abb. 2).

Die Programmbasis enthält die programmtechnische Realisierung der Funktionen. Die Datenbasislösung enthält die zum Ablauf der Programmbasis erforderlichen Daten (Datenbasis), die Beschreibung des Aufbaues der Datenbasis und die programmtechnischen Mittel für Speicherung, Zugriff und Pflege der Datenbasis.

Die Programmbasis besteht aus Programmsystemen. Bestandteile eines Programmsystems sind Programme. Ein Programm setzt sich aus Modulen zusammen. Ein Modul besteht aus Strukturblöcken.

Die Datenbasis besteht aus Dateien, eine Datei aus Datensätzen und ein Datensatz aus Daten.

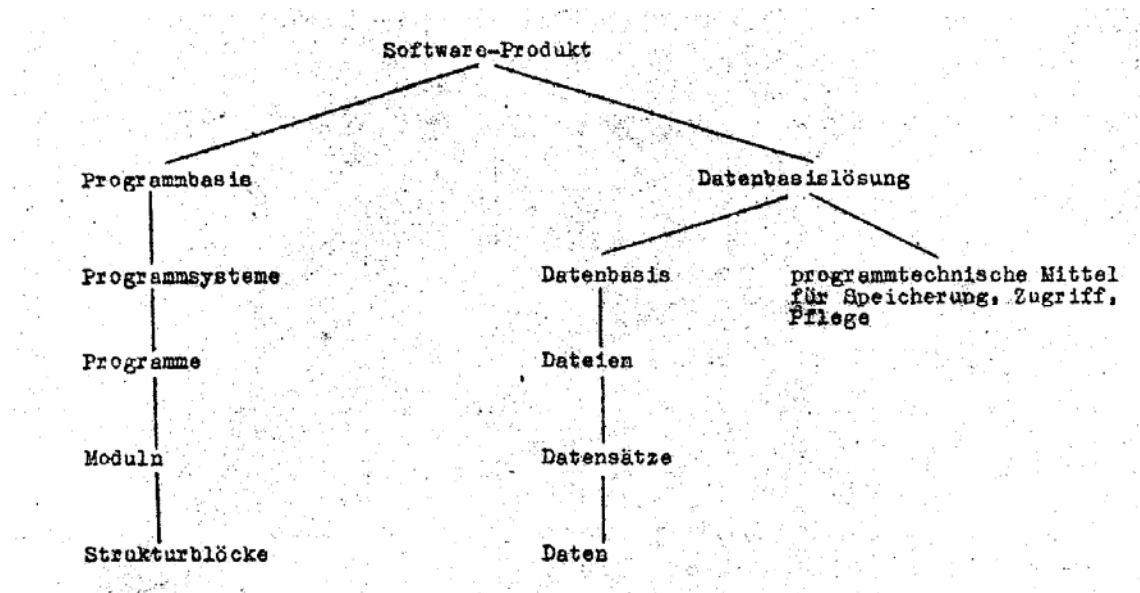


Abbildung 2: Bestandteile der Softwareprodukte

2.2. Die Programmbasis

Die zur Lösung einer gemeinsamen Aufgabe existierenden Programme bilden die Programmbasis. Jedes Programm realisiert eine Gruppe von Funktionen und ist selbstständig durch das Betriebssystem startbar. Verbindungen zwischen den Programmen werden über das Betriebssystem vorgenommen.

Es wird davon ausgegangen, dass ein Programm aus einer **Hierarchie von Moduln** aufgebaut ist (Abb. 3), der eine Baumstruktur zugrunde liegt. Der an der Spitze der Modulhierarchie stehende Modul (**TOP-Modul**) repräsentiert das Programm. Der Steuerfluss innerhalb der Modulhierarchie erfolgt stets von einer Hierarchieebene zur nächsten über- oder untergeordneten Ebene. Die Moduln, die die Blätter der Baumstruktur darstellen, führen Verarbeitungsfunktionen aus, sie werden als **Verarbeitungsmoduln** bezeichnet. Die innerhalb der Hierarchie darüber angeordneten Moduln realisieren Steuerfunktionen und werden als **Steuermoduln** bezeichnet.

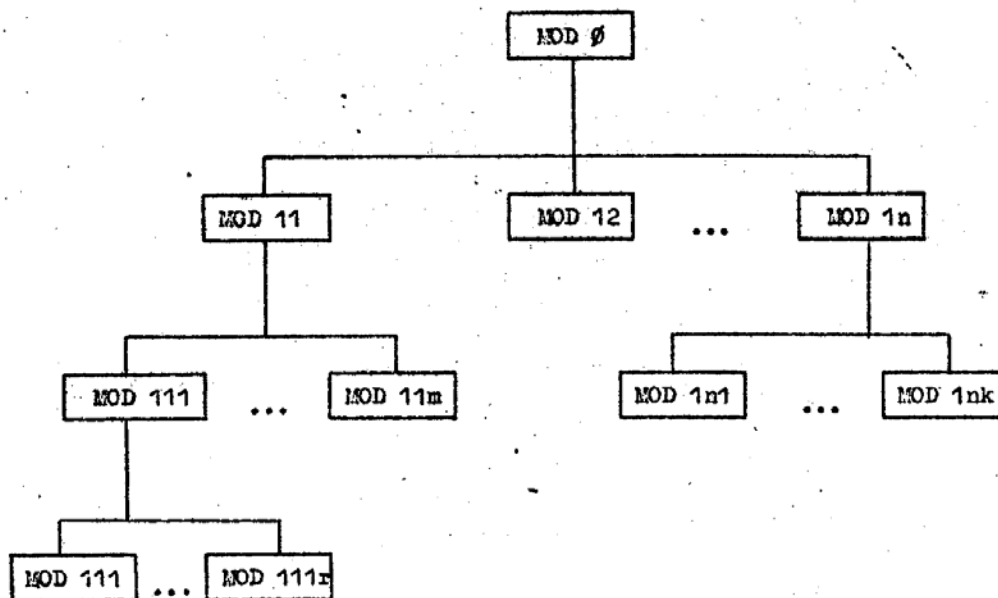


Abbildung 3: Das Programm als Modulhierarchie

Der **Modul** ist eine selbständige programmtechnische Einheit, die eine abgegrenzte Aufgabe erfüllt und einen Namen trägt. Über diesen Namen kann der Modul aufgerufen, übersetzt, verbunden oder verwaltet werden.

Der Modul hat einen standardisierten Aufbau (Abb.4). Die Hauptbestandteile sind

- Dokumentationsteil
- Modulkörper

Dokumentationsteil und Modulkörper sind physisch getrennt verwaltbar.

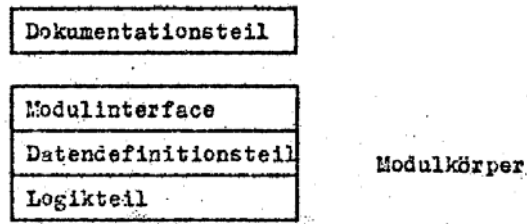


Abbildung 4: Standardisierter Modulaufbau

Der Dokumentationsteil dient der rechnergestützten Dokumentation und besteht aus maschinell auswertbaren Textteilen.

Der Modulkörper wird aus dem Modulinterface, dem Datendefinitionsteil und dem Logikteil gebildet. Über das Modulinterface erfolgt die Kommunikation des Moduls mit seiner Umgebung. Es enthält alle Informationen, die zur Einbindung des Moduls in eine Umgebung notwendig sind. Dazu gehören

- Spezifikation der formalen Parameter
- Spezifikation der im Modul gerufenen Moduln
- Angabe des Modulstatus
- Art der Speicherplatzzuweisung

Der Datendefinitionsteil enthält die Beschreibung der im Modul benötigten Daten auf logischem Niveau.

Der Logikteil besteht aus Strukturblocken.

Der **Strukturblock** realisiert eine abgegrenzte Teilaufgabe. Bei seinem Aufbau werden die Prinzipien der strukturierten Programmierung angewendet. Strukturblocke besitzen einen Eingang und einen Ausgang, ihre Vereinigung ergibt wieder einen Strukturblock.

Strukturblocke können sein

- Grundstrukturen der strukturierten Programmierung
- Aufrufe von Moduln
- Aufrufe von untergeordneten Strukturblocken
- Mengen von Operationen in der Zielsprache (elementare Strukturblocke)

Zielsprache ist die für die Realisierung der Software durch den Entwickler des Softwareproduktes ausgewählte Programmiersprache.

Die für TESO 1600 festgelegten logischen Grundstrukturen sind in Tabelle 2 und 3 dargestellt.

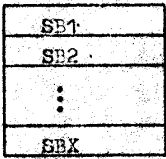
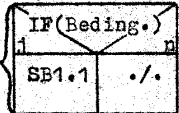
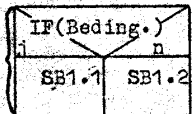
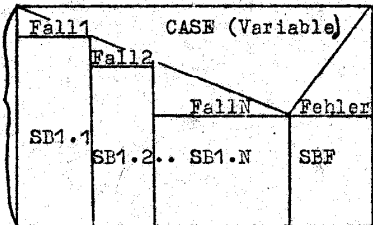
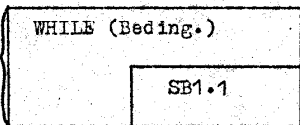
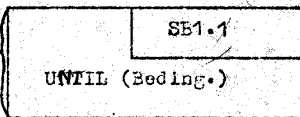
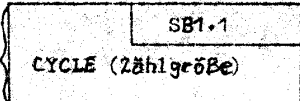
Name	Spezifikation Struktogramm	Kommentar
FOLGE		Die Strukturblöcke werden in der notierten Reihenfolge durchlaufen.
AUSWAHL	ohne ELSE-Zweig 	Bei erfüllten logischen Bedingungen wird SB1.1 abgearbeitet, andernfalls wird SB1 beendet.
	mit ELSE-Zweig 	Bei erfüllten logischen Bedingungen wird SB1.1 einmalig abgearbeitet, sonst SB1.2
	Fallauswahl 	In Abhängigkeit vom Wert der Auswahlvariablen wird zu einem der Strukturblöcke SB1.1,..., SB1.N verzweigt und dieser einmal ausgeführt. Trifft kein definierter Fall zu, erfolgt die Abarbeitung des Strukturblockes SBF. Der Fehlerzweig kann auch entfallen.
WIEDERHOLUNG	mit Bedingungsprüfung am Anfang 	Solange die Bedingung erfüllt ist, wird wiederholt der Strukturblock SB1.1 abgearbeitet. Bei nicht erfüllter Bedingung wird SB1 beendet.
	mit Bedingungsprüfung am Ende 	Der Strukturblock SB1.1 wird wiederholt abgearbeitet, bis die Bedingung erfüllt ist. Bei erfüllter Bedingung wird SB1 beendet.
	mit Zählgröße 	Der Strukturblock SB1.1 wird wiederholt abgearbeitet, bis eine Zählgröße den Endwert überschritten hat. Danach wird SB1 beendet.

Tabelle 2: Logische Grundstrukturen des TESO 1600

Name	Spezifikation Struktogramm	Kommentar
	<p>mit Abbruch</p>	<p>In der Wiederholungsschleife wird eine zusätzliche Bedingungsprüfung (BREAK) an beliebiger Stelle der Folge von inneren Strukturblocken vorgenommen. Bei Erfüllung der Bedingung (z.B. Bedingung B) wird die Schleife durch Abbruch beendet. (Verlassen Strukturblock SB1).</p>
UNBEDINGTER ABBRUCH	<p>Modul</p>	<p>Durch LEAVE erfolgt auf beliebiger Ebene ein unbedingter Abbruch (Verlassen) des Moduls.</p>

Tabelle 3: Logische Grundstrukturen des TESO 1600

Die **Darstellung der Daten und Datenbeziehungen** erfolgt in der Programmbasis mit Hilfe von Relationen. Durch eine Relation wird die Beziehung zwischen einem Informationsobjekt (Entity) und mindestens einem Informationsmerkmal (Attribut) beschrieben. Jede Relation hat mindestens ein Hauptmerkmal, das der eindeutigen Identifikation der jeweiligen Informationsobjektes dient. Jedes Merkmal verfügt über einen bestimmten Wertevorrat (Merkmalswerte). Zusammenhängende Merkmalswerte einer Relation bilden einen Relationensatz.

Jeder Modul kennt nur die für ihn als gültig definierten Datenobjekte.

2.3. Die Datenbasis

Die Datenbasis enthält Datenbestände, die z.B. auf Plattenspeicher, Magnetbandspeicher und/oder anderen Ein-/Ausgabemedien (Lochband, Lochkarte, Drucker oder Schreibmaschinenformular) vorliegen. Diese Datenbestände sind zu Dateien zusammengefasste Sätze von Daten. Zur Arbeit mit der Datenbasis gibt es für das Rechnersystem K 1600 insbesondere folgende Möglichkeiten:

3. Allgemeiner Aufbau von TESO 1600

- Routinen zum physischen Zugriff
- FCS 1600
- RMS 1600
- SAZU 1600
- DABA 1600
- Höhere Programmiersprachen

In TESO 1600 wurden die Anwendungsbedingungen und -konventionen dieser Komponenten übernommen.

Die Kommunikation zwischen einer mit TESO 1600 erzeugten Programmbasis und einer Datenbasislösung erfolgt über einen als Schnittstelle im Hauptspeicher bereitgestellten Datensatz (Abb. 5). Dem Zugriff zu den Daten des Datensatzes dienen relationale Datenbeschreibungen.

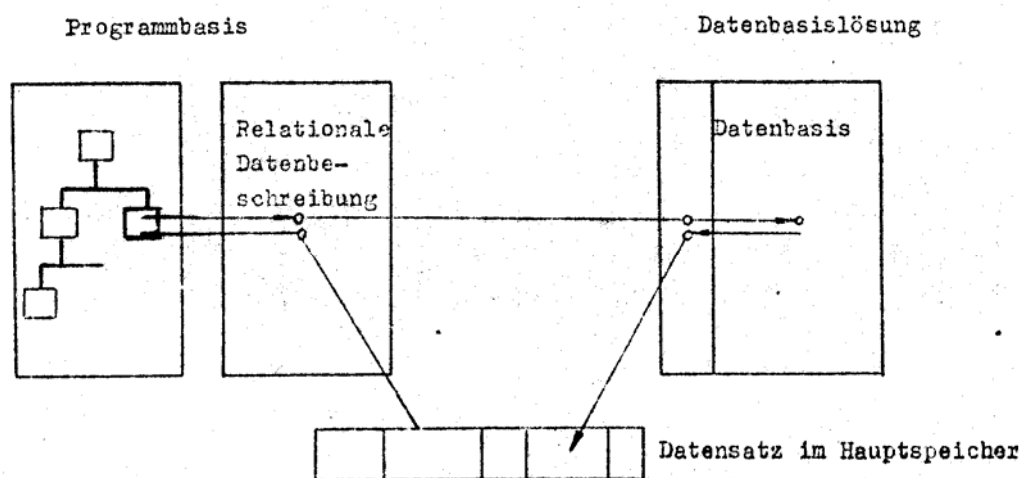


Abbildung 5: Kommunikation zwischen Programm- und Datenbasis

3. Allgemeiner Aufbau von TESO 1600

3.1. Wirkungsweise von TESO 1600

TESO 1600 realisiert folgende prinzipielle Wirkungsweise (Abb. 6):

Ein Eingangsinformationsobjekt I_E wird durch eine technologische Aktion T_A in ein Ausgangsinformationsobjekt I_A überführt. Jede technologische Aktion ist durch ein Kommando K aufrufbar und kann eine Nachricht N erzeugen. Ein- und Ausgangsinformationsobjekte sind Zwischenprodukte und Produkte der Softwareentwicklung in den Phasen Implementieren, Testen, Dokumentieren und Kontrollieren.

Eingangsinformationsobjekte des TESO 1600 sind z.B.:

- Quellcode für programmtechnische Einheiten
- Objektcode für programmtechnische Einheiten
- Textdateien

Ausgangsinformationsobjekte sind z.B.:

- ablauffähige getestete Moduln und Programme

- Dokumentationen
- Protokolle, die die formale technologische Richtigkeit der Software-Produkte nachweisen

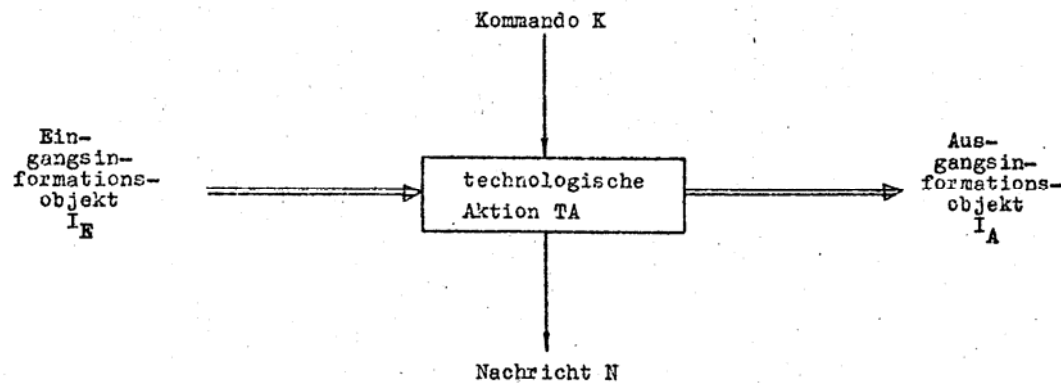


Abbildung 6 : Prinzipielle Wirkungsweise des TESO 1600

Technologische Aktionen sind die Tätigkeiten im Software-Entwicklungsprozess, die der TESO-Nutzer bei der Programmentwicklung durchführt.

Kommandos und Nachrichten sind Mittel zur Kommunikation mit dem Software-Entwicklungssystem.

TESO 1600 hat aus Nutzersicht vier Systemkomponenten:

- TESO-Codiermittel zum Codieren der Software
- TESO-Standardkomponenten für die Realisierung häufig benötigter Funktionen bei der Codierung der Software
- TESO-Aktionen für die Teilprozesse der Software-Entwicklung
- TESO-Dialogmittel zur Kommunikation zwischen Mensch und Rechnersystem bei der Software-Entwicklung

Eine zentrale Rolle hat das TESO-Verwaltungssystem. Es besteht aus den TESO-Bibliotheken und den Mitteln zur Arbeit mit diesen Bibliotheken. Die TESO-Bibliotheken werden in TESO-Systembibliotheken und TESO-Nutzerbibliotheken unterteilt. In den TESO-Systembibliotheken befinden sich die TESO-Systemkomponenten. Die TESO-Nutzerbibliotheken beinhalten Produkte und Zwischenprodukte der Software-Entwicklung. Entsprechend dem Verwaltungsgegenstand werden sie in Quelltextbibliotheken und Objektmodulbibliotheken unterteilt.

Die Quelltextbibliotheken können Software-Komponenten in TESO-Quellform (TESO-Quellmoduln) oder in Zielsprachen-Quellform (Zielsprachen-Quellmoduln) sowie Textdateien enthalten.

TESO 1600 bildet eine leicht handhabbare Software-Schicht über dem Betriebssystem MOOS 1600 und entlastet damit den Software-Entwickler weitgehend von Detailkenntnissen über das Betriebssystem und seine Dienstprogramme.

Auf Grund seiner Kompaktheit wird TESO 1600 auch als

Systemunterlagen-Entwicklungsplatz K1600 (abgekürzt mit **SEP 1600**) bezeichnet.

3.2. Übersicht über die Systemkomponenten des TESO 1600

TESO-Codiermittel

Die Codiermittel des TESO 1600 bilden die Sprache zur Notation der Eingangs-informationsobjekte in TESO-Quellform. Es wurde dafür die **Entwurfssprache SDL/R** definiert, die es gestattet, Algorithmen in Form von Ablaufstrukturen mittels eines Pseudocodes zu beschreiben. SDL/R erzwingt und unterstützt somit die strukturierte und modulare Programmierung. Der SDL/R-Text wird von einem im TESO 1600 vorhandenen Prozessor syntaktisch und semantisch analysiert, strukturiert aufbereitet und automatisch der Zielsprachencode für die Zielsprachen MAC 1600, FORTRAN und COBOL erzeugt.

Zur Realisierung der Konstruktionsprinzipien der Software stehen die in Tabelle 4 als Beispiele aufgeführten Codiermittel zur Verfügung.

Beziehungen zu den Konstruktionsprinzipien	Codiermittel	realisierte Funktion
Modulaufbau Programmaufbau	BEGINN	Modulanfang
	ENDMOD	Modulende
	INTER	Modulinterface
	DECLAR	Definition modulinterner Daten
	CONS	Definition modulinterner Konstanten
Logische Grundstrukturen der strukturierten Programmierung	CALLM	Modulaufruf
	CALLS	Aufruf eines Strukturblockes
	CASE	Fallauswahl
	IF	Alternative
	WHILE	Wiederholung (Bedingungsprüfung am Anfang)
	UNTIL	Wiederholung (Bedingungsprüfung am Ende)
	CYCLE	Laufvariablensteuerung
Datenbasisbeschreibung	BREAK	Abbruch einer Wiederholung
	LEAVE	unbedingter Modulabbruch
	RELATE	Verbindung der Relationenbeschreibung mit dem Modul
	ATTR	Merkmalsbeschreibung Beschreibung des Satzaufbaus

Tabelle 4: Codiermittel des TESO 1600

TESO- Standardkomponenten

TESO-Standardkomponenten unterstützen die Einhaltung der Konstruktionsprinzipien bei der Codierung der Software. Zusammen mit den TESO-Codiermitteln rationalisieren und vereinheitlichen sie den Prozess der Codierung weitgehend.

TESO-Standardkomponenten werden unterteilt in

- TESO-Standardkomponenten zur Kommunikation zwischen Programm- und Datenbasis
- TESO-Standardkomponenten zur Codierung von Testfunktionen
- TESO-Standardkomponenten zur Fehlerbehandlung

Umfang und Realisierung der TESO-Standardkomponenten sind abhängig von der Zielsprache und der Datenbasislösung.

Als TESO-Standardkomponenten stehen z.B. folgende Zugriffsmoduln zur Datenbasis zur Verfügung:

für FCS 1600 die Moduln

TDGETS - Lesen des physisch nächsten Satzes von einer Direktzugriffs- oder sequentiellen Datei eines blockorientierten Gerätes

TDGETD - Lesen eines Satzes von einer Direktzugriffsdatei nach vorgegebener Satznummer

TDGETK - Lesen eines Satzes von einer Direktzugriffsdatei nach einer vorgegebenen Schlüsselnummer

TDPUTS - Ausgabe eines Datensatzes als physisch nächsten Satz auf eine sequentielle oder Direktzugriffsdatei

TDPUN - Zurückschreiben eines vorher gelesenen Datensatzes in eine Direktzugriffsdatei

TDPUTD - Schreiben eines Satzes nach vorgegebener virtueller Satznummer

für RMS 1600 die Moduln

TDREAD – Lesen eines Satzes von dem durch die Dateibeschreibung für externe Eingabe spezifizierten Gerät

TDWRIT – Ausgabe eines Datensatzes auf ein satzorientiertes Gerät

Als TESO-Standardkomponenten für Testfunktionen stehen zur Verfügung:

- statische Testmittel, die bereits bei der Codierung berücksichtigt werden müssen (z.B. Hauptspeicherauszüge mittels der Anweisungen SNAP bzw. SNAPID) oder vor der Testdurchführung erzeugt werden (Pseudomoduln, Testrahmen)
- dynamische Testmittel, die erst während der Testdurchführung ausgewählt und spezifiziert werden, z.B. der Testmonitor

Die vorhandenen Testmittel erzeugen für die Testauswertung folgende Dokumente:

- Traceprotokolle
- Hauptspeicherausdrucke
- Statistikprotokolle
- Backtraceprotokolle
- Protokolle der MOOS-Testhilfen

Als TESO-Standardkomponenten zur Fehlerbehandlung stehen z.B. folgende Moduln zur Verfügung:

TSFEHL – Steuermodul zum Aufbereiten und Ausgeben einer Nachricht (aus MAC)

TSFOFE – Steuermodul zum Aufbereiten und Ausgeben einer Nachricht (zielsprachenunabhängig)

TSNACH – Aufbereiten der Fehlernachricht

TSDRUC – Drucken des Fehlerprotokolls

TSCODE – Ermitteln der Adresse des erforderlichen Fehlertextes

TSAUSB – Definition des Ausgabebereiches der Fehlernachricht

FILE1 ... FILE5 – Nachrichtentexte

Bei der Fehlerbehandlung wird ein standardisiertes Fehlerprotokoll zu Grunde gelegt.

Teilprozeß	technolog. Aktion	Funktion	I _K	I _A
Implementieren	VORÜBERSETZEN	Erzeugung eines Moduls in Zielsprache	TESO-Quellmodul	Zielsprachen-Quellmodul
	EDITIEREN	Editor von Quelltexten (TESO-Quelltext, Zielsprachen-Quelltext)	Programmkomponenten in TESO-Quellform, Zielsprachen-Quellform	Programmkomponenten in TESO-Quellform, Zielsprachen-Quellform
	ÜBERSETZEN	Erzeugung eines Objektmoduls	Zielsprachen-Quellmodul	Objektmodul
	BINDEN	Erzeugung einer ablauffähigen Task aus Moduln	Objektmodul Bindersteuerdatei	Task
Testen	PSEUDOMODUL	Erzeugung eines Pseudomoduls im Dialog	Spezifikation d. Testparameter	Pseudomodul in Objekt- oder Quellform
	TESTSTRAHLEN	Erzeugung eines Testrahmens für Modultest	zu testende Moduln, Parameter	Testrahmen in Objektform
	TESTABLAUF	Steuerung der Taskabarbeitung	Steuerparameter	Protokoll des Testablaufs
	TESTDATENMANIPULATION	Erzeugung, Wartung von Testdaten	Parameter Einzeldaten	Testdaten Listen
Kontrollieren	MODULKONTROLLE	Kontrolle der Einhaltung des standardisierten Modulaufbaus	TESO-Quellmodul	Modulprotokoll
	PROGRAMMKONTROLLE	Kontrolle der Widerspruchsfreiheit der Modulhierarchie	Alle TESO-Quellmoduln eines Programms	Programmprotokoll
Dokumentieren	MODULDOKUMENTATION	Erzeugung der Moduldokumentation	- Textdateien - Spezifikation d. Modulkennblattaufbaus - TESO-Quellmoduln - Zielsprachen-Quellmoduln	- Textdatei - Modulkennblatt - TESO-Quelltext - Zielsprachen-Quelltext - Struktogramm - E/A-Übersicht
	PROGRAMMDOKUMENTATION	Erzeugung der Programmdokumentation	TESO-Quellmoduln des Programms	- Programmprotokoll - E/A-Verknüpfung
Verwalten	EINGEBEN AUSGEBEN KATALOGISIEREN DATEIPFLEGE	Eingabe Ausgabe Katalogisieren Verwalten von Quellcode, Objektcode, Textdateien	Quellmoduln Objektmoduln Textdateien	Katalogisierte Quellmoduln Objektmoduln Textdateien
Systemsteuerung	STEUERUNG	Start der TESO 1600, Steuerung und Ende.	Nutzerspezifikation	Protokolle

Tabelle 5: TESO- Aktionen

TESO- Aktionen

Durch TESO- Aktionen (Beispiele siehe Tabelle 5) werden alle technologischen Arbeitsgänge realisiert, die beim

- Implementieren
- Testen
- Verwalten
- Dokumentieren
- Kontrollieren

der Software ausgeführt werden müssen.

Jede TESO-Aktion ist durch ein Kommando aufrufbar. Durch eine TESO-Aktion werden mehrere technologisch zusammenhängende Arbeitsschritte (STEP's) realisiert, mit denen der TESO-Nutzer nicht in Berührung kommt. Das betrifft auch den Aufruf der TESO-Dienstprogramme und MOOS-Systemprogramme.

TESO- Dialogmittel

TESO- Dialogmittel realisieren die TESO-Kommandoverarbeitung einschließlich der Parametrisierung der TESO-Kommandos, die über den Kommandonamen aufgerufen werden können. Im allgemeinen Falle hat ein Kommando Parameter, die im Dialog mit dem Bediener durch TESO-Dialogmittel gesetzt werden.

4. Software-Entwicklung mit TESO 1600

4.1. Aufgabenstellung und Randbedingungen

Die Software-Entwicklung ist ein komplizierter, mit vielen Iterationen behafteter und von vielen Randbedingungen abhängiger Prozess. Deshalb ist es nicht möglich, einen allgemein gültigen Standardablauf für die Software-Entwicklung festzulegen. Aus diesem Grunde werden technologische Mittel auch nicht in Form geschlossener Systeme, sondern als System von relativ selbständigen Einzelwerkzeugen bereitgestellt.

Zu Beginn der Software-Entwicklung müssen folgende Voraussetzungen erfüllt sein:

- die präzisierte Ziel- und Aufgabenstellung liegt vor
- die technischen Mittel sind vorhanden
- das Entwicklerkollektiv wurde gebildet

Die präzisierte Ziel- und Aufgabenstellung für die Software muss beinhalten

- Aufgaben des Software-Produktes
- Informationen und Daten
- sachliche, zeitliche, örtliche und technische Bedingungen
- Termine und Planungsstufen
- Qualitätsforderungen

Die technischen Mittel des Entwicklungskollektives sind

- Rechnersystem K1630 mit 64KW Hauptspeicher
- Betriebssystem MOOS 1600
- TESO 1600

Das Entwicklerkollektiv setzt sich im Allgemeinen aus mehreren Personen zusammen, zwischen denen eine Arbeitsteilung besteht. Es besteht aus dem Leiter des Kollektivs, einem Bibliotheksverantwortlichen, mehreren Systemprogrammierern und Programmierern. Fachspezialisten können zeitweilig dem Entwicklerkollektiv angehören.

4.2. Technologische Vorbereitung der Software-Entwicklung

Entwicklungstechnologie und Organisation

Die Entwicklungstechnologie ist durch die dem TESO 1600 zugrunde liegende POS-Technologie vorgegeben. Präzisierungen, z.B. zum Umfang der Dokumentation, zu Namenskonventionen u.a., müssen entsprechend den konkreten Anforderungen erfolgen.

Für die Entwicklung ist ein Ablaufplan mit mindestens folgendem Inhalt zu erarbeiten und abzustimmen:

- Liste der Moduln
- Liste der Bearbeiter pro Modul
- Übersicht über Abhängigkeiten der Moduln untereinander
- Zeitplan für die Implementierung der Einzelmoduln
- Zeitplan für die Testung von größeren Modulkomplexen bis zum Systemtest
- Verantwortlichkeit und Zeitplan für die Testdatenbereitstellung
- Festlegungen zu den einzurichtenden Bibliotheken für die Verwaltung der POS-Komponenten

Dazu können die methodischen Mittel des TESO 1600, in den Institutionen vorhandene Ordnungen und Richtlinien sowie in der Literatur vermittelte Erfahrungen genutzt werden.

Einrichten des TESO 1600

TESO 1600 wird entsprechend den konkreten Entwicklungsbedingungen und der Entwicklungstechnologie generiert. Die Einrichtung erfolgt entsprechend der Anleitung für den Systemverantwortlichen des TESO 1600 durch dialoggeführte Generierung. Dabei werden auch die TESO-Bibliotheken des Entwicklerkollektivs eingerichtet.

Bei arbeitsteiliger Entwicklung sind folgende TESO-Nutzerbibliotheken einzurichten (Abb. 7):

- Produktionsbibliotheken
- Entwicklungsbibliotheken
- Archivierungsbibliotheken

Zu diesen Bibliotheksgruppen können jeweils die beiden Bibliotheksarten

- Quelltextbibliotheken mit Bibliotheken für TESO-Quellcode, Zielsprachen-Quellcode, Text für Dokumentationsteile
- Objektmodulbibliotheken

gehören.

Produktionsbibliotheken enthalten immer die neuesten freigegebenen Komponenten der entwickelten Software. Innerhalb dieser Bibliotheken dürfen keine Entwicklungs- und Wartungsarbeiten durchgeführt werden. Lesen innerhalb dieser Bibliotheken ist jedem Entwickler gestattet, schreiben darf nur der Bibliotheksverantwortliche.

Die Entwicklungsbibliotheken enthalten die aktuellen Versionen der in Entwicklung befindlichen Software-Komponenten. Jedem Nutzer (bzw. jeder Nutzergruppe) wird ein Bereich (Nutzerbereich) zugeordnet, der vom Betriebssystem durch einen Nutzeridentifikationscode gekennzeichnet wird.

Archivierungsbibliotheken dienen zur Sicherung der Software-Komponenten. Sie enthalten immer die vorherige bzw. die gerade aktuelle Version der Software-Komponenten der Produktionsbibliotheken.

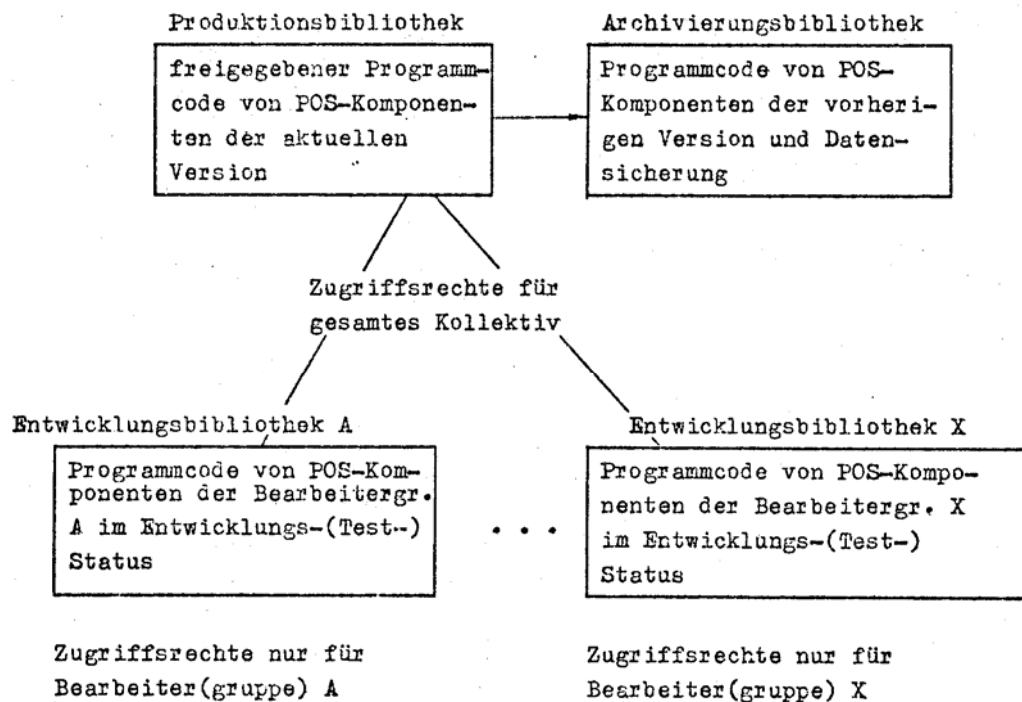


Abbildung 7 : TESO-Bibliotheken eines POS-Entwicklerkollektivs

4.3. Ablauf der Software-Entwicklung

Ein Beispiel für den Ablauf der Software-Entwicklung mit TESO 1600 zeigt Abbildung 8. Das problemorientierte und programmtechnische **Entwerfen** des Systems erfolgt nach Methoden, die in den methodischen Mitteln des TESO 1600 beschrieben sind. Im Ergebnis des Entwerfens liegt der dokumentierte programmtechnische Systementwurf einschließlich der Entwürfe der einzelnen Moduln vor.

Das **Implementieren** beginnt mit dem Codieren der einzelnen Moduln. Dabei kann in der Modulhierarchie entweder **TOP-DOWN** oder **BOTTOM-UP** vorgegangen werden.

Beim **Codieren** finden die TESO-Codiermittel und TESO-Standardkomponenten Verwendung. Es lassen sich damit alle Steuerfunktionen, das Modulinterface, der Datendefinitionsteil der Moduln und alle nicht elementaren Strukturböcke eines Moduls codieren. Elementare Strukturböcke können direkt in der Zielsprache codiert werden. Das Ergebnis sind TESO-Quellmoduln, welche die in Abb.9 dargestellte allgemeine Form haben.

Die TESO-Quellmoduln werden in der eingerichteten Entwicklungsbibliothek mit Hilfe der technologischen Aktionen **EINGEBEN** und **KATALOGISIEREN** abgelegt. Danach wird der Modul mit Hilfe der TESO-Aktion **MODULKONTROLLE** auf Einhaltung der Konstruktionsprinzipien des TESO 1600 und die technologisch korrekte Verwendung der TESO-Codiermittel geprüft. Weist das erzeugte Modulprotokoll keine Fehler aus, kann der TESO-Quellmodul mit Hilfe der Aktion **VORÜBERSETZEN** in einen Zielsprachen-Quellmodul überführt und in der entsprechenden Bibliothek abgelegt werden.

Mit Hilfe der Aktionen zur **MODULDOKUMENTATION** können das Modulkennblatt, ein Struktogramm, eine Übersicht über die Ein- und Ausgangsgrößen des Moduls, eine Übersicht zur Datenverwendung sowie TESO-Quelltextlisten und Zielsprachen-Quelltextlisten erzeugt werden.

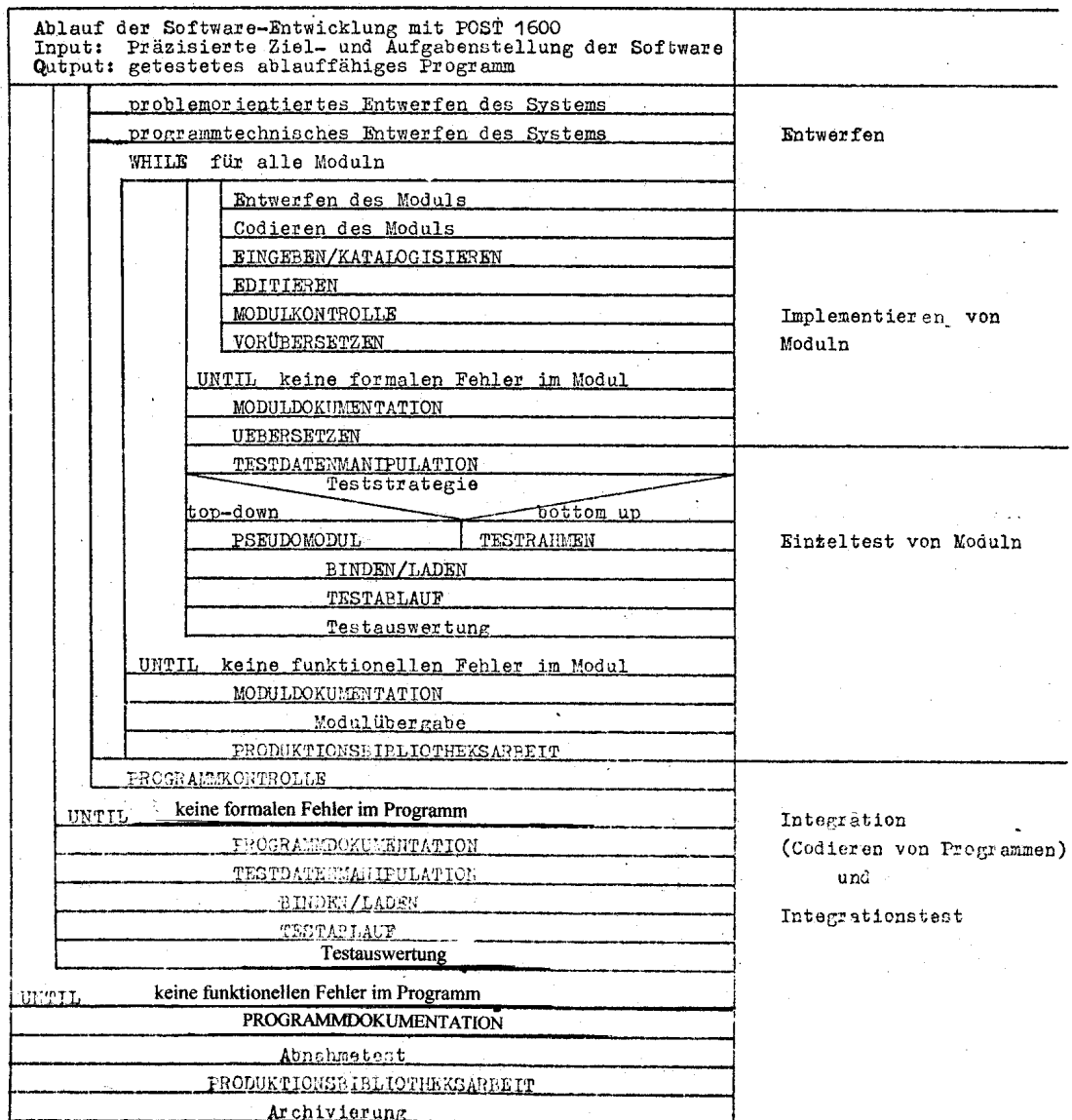


Abbildung 8 : Beispiel für den Ablauf der Software-Entwicklung mit TESO 1600

Schritt haltend mit der Codierung des Moduls können die Textdateien für Abschnitte des Modulkennblattes aufgebaut werden. Die Erzeugung eines Modulkennblattes ist in verschiedenen Phasen der Modulimplementierung und Modultestung mit der Aktion MODULDOKUMENTATION möglich.

Mit der Aktion **UEBERSETZEN** wird aus dem Zielsprachen-Quellmodul ein Objektmodul erzeugt und in der Entwicklungsbibliothek abgelegt. Damit ist die Phase der Implementierung abgeschlossen und es beginnt die Testphase der Moduln.

Für jeden Modul ist zunächst ein **Einzeltest** durchzuführen. Als Vorbereitung darauf wurden bereits beim Codieren in den Moduln bzw. Strukturblocken TESO-Standardkomponenten zur Erzeugung von Speicherausügen und zur Ablaufverfolgung eingefügt. Die Testdaten für den Moduleinzeltest werden durch die Aktion **TESTDATENMANIPULATION** bereitgestellt. Die Erzeugung des erforderlichen Testrahmens beim BOTTOM-UP-Vorgehen erfolgt durch die Aktion **TESTRAHMEN**.

Beim TOP-DOWN-Testen werden mit der Aktion **PSEUDOMODUL** die erforderlichen Pseudomoduln für die noch nicht vorliegenden untergeordneten Moduln des zu testenden Moduls erzeugt.

Die Aktion **BINDEN** stellt dann aus den fertig gestellten Komponenten und Systemmoduln des Betriebssystems ein **ablauffähiges Programm (Task)** her, wobei ein automatisches Binden (Spezifikation typischer Parameter im Dialog) oder Binden mit Bindersteuerdatei möglich ist. Damit kann der Einzeltest des Moduls mit Hilfe der Aktion **TESTABLAUF** durchgeführt werden. Die beim Testlauf entstehenden Speicherauszüge und Protokolle ermöglichen die Beurteilung der Funktion des Moduls auf funktionelle Richtigkeit.

Vorhandene Fehler sind zu analysieren, der Modulentwurf zu berichtigen und alle Phasen bis zum fehlerfreien Einzeltest zu wiederholen.

Fehlerfreie Moduln werden an den Verantwortlichen zur Aufnahme in die Produktionsbibliothek übergeben.

Dazu gehören als Bibliotheksbestände

- die TESO-Quellform
- die Zielsprachen-Quellform
- die Objektform
- die Textdateien für die Moduldokumentation
- das Testbeispiel

In Schriftform sind die vollständigen Moduldokumentationen und die Dokumentation des Testbeispiels zu übergeben.

Die nächste Phase im Entwicklungsprozess ist der **Integrationstest** für die einzelnen Zweige der Modulhierarchie. Die Modulhierarchie kann in einer verketteten Struktur oder als Überlagerungsstruktur (zur Optimierung des Hauptspeicherplatzbedarfs) gebunden werden.

Dazu werden ebenfalls die TESO-Aktionen TESTDATENMANIPULATION, BINDEN und TESTABLAUF benutzt.

Die letzte Phase ist die Integration des Gesamtsystems und der **Systemtest**. Dabei ist zunächst mit der Aktion **PROGRAMMKONTROLLE** die Einhaltung der Konstruktionsprinzipien des TESO 1600 und die technologisch korrekte Verwendung der TESO-Codiermittel zu prüfen. Mit Hilfe der Aktion **PROGRAMMPROTOKOLL** werden das Programmprotokoll und die Übersicht zur Gesamtdatenverwendung erzeugt, die auch zur Unterstützung der Fehleranalyse genutzt werden können. Vorhandene Fehler sind im Entwurf und/oder im TESO-Quelltext zu korrigieren und der Systemtest unter Benutzung der TESO-Aktionen TESTDATENMANIPULATOR, BINDEN und TESTABLAUF zu wiederholen.

Werden beim Systemtest keine Fehler ermittelt, ist die Dokumentation des Programms zu überprüfen und ggf. zu vervollständigen.

Danach kann der **Abnahmetest** erfolgen. Nach dessen erfolgreichem Abschluss wird mit der Aktion **PRODUKTIONSbibliotheksARBEIT** das gesamte Programm bzw. Programmsystem in die Produktionsbibliothek überführt.

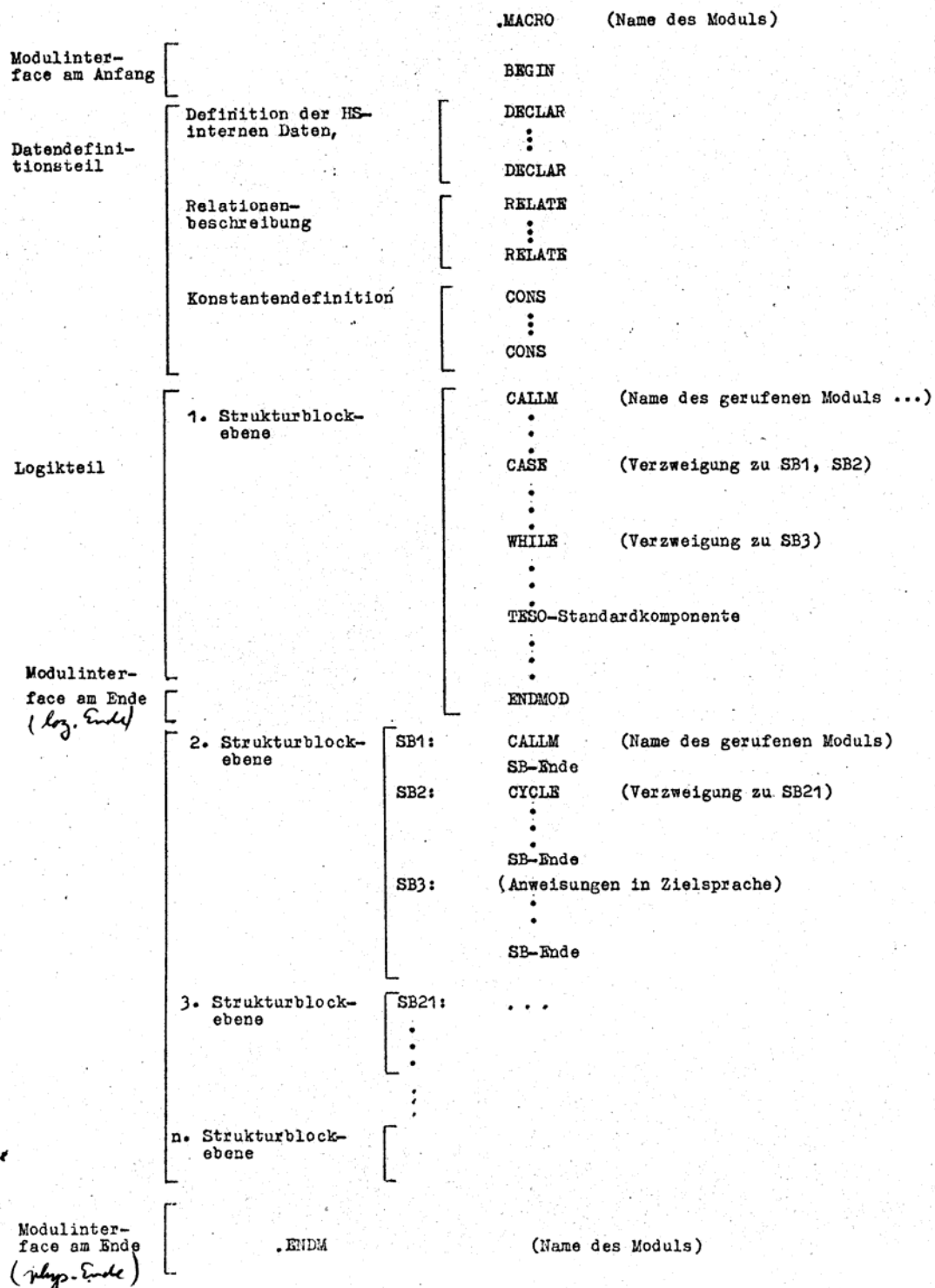


Abbildung 9 : Prinzipbeispiel für TESO-Quellmodul

In den Archivierungsbibliotheken werden vom fertigen Software-Produkt abgespeichert

- TESO-Quellmoduln
- Zielsprachen-Quellmoduln
- Objektmoduln
- Dokumentationsdateien

- Bibliotheksbestände für Wartung (Moduln der Testbeispiele in Quell- und Objektform, Quellmoduln mit Testanweisungen, Testdaten)

Bei notwendigen Wartungsarbeiten sind die betreffenden Bibliotheksbestände aus der Archivierungsbibliothek in die Entwicklungsbibliothek zu übernehmen und die erforderlichen Änderungen einschließlich Tests durchzuführen.

5. Literatur

- [1] Horn, E.; Schmidt, G. : Implementieren von POS für die Mikrorechnerfamilie 1600
rechentechnik / datenverarbeitung, 1980, 3. Beiheft