Vorlesung Software Engineering

Foliensatz Nr. 7 (30.11.11)

Arbeitsgruppe Software Engineering
Prof. Elke Pulvermüller

Universität Osnabrück
Institut für Informatik, Fachbereich Mathematik / Informatik
Raum 31/318, Albrechtstr. 28, D-49069 Osnabrück

elke.pulvermueller@informatik.uni-osnabrueck.de

http://www.inf.uos.de/se

Sprechstunde: mittwochs 14 – 15 und n.V.

Inhalt

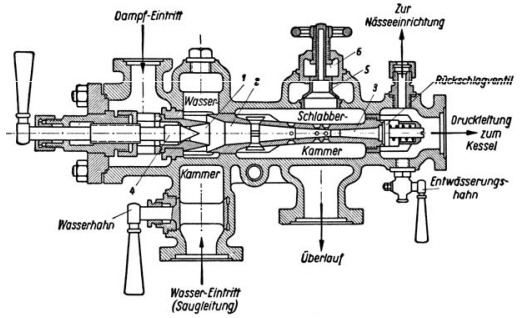
5 Software-Modelle

Inhalt

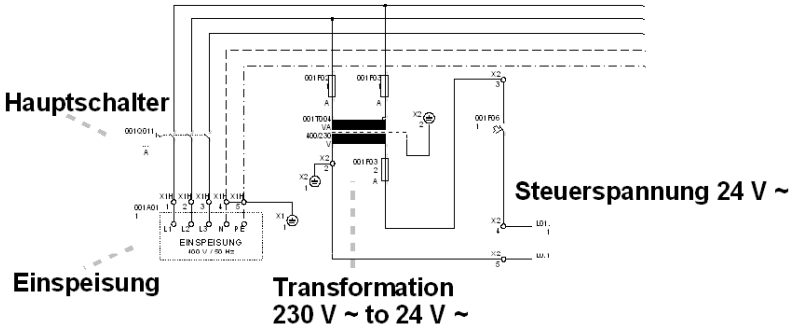
Was sind Modelle?

- **Modelle sind Abstraktionen der realen Welt**

- **Beschreiben reale, greifbare und sichtbare Objekte (z.B. Plan eine Hauses, Grundriss, Bauplan von mechanischen Elementen)**



- **Beschreiben reale, aber nicht greifbare, bzw. nicht sichtbare Objekte (z.B. elektrische Schaltpläne, physikalische Phänomene wie Kraft und Impuls)**



- **Beschreiben statischer Zusammenhänge und Abhängigkeiten (z.B. Funktionshierarchien, objektorientierte Klassendiagramme) oder dynamischer Abläufe und Abhängigkeiten (z.B. Geschäftsprozesse, Fertigungsprozesse)**

- **Modelle unterstützen (deskriptiv oder präskriptiv) die Visualisierung, Spezifikation, Konstruktion und Dokumentation der Konstruktionselemente**

5.1 Grundlagen und Modelltypen: Sichten

Arten von Modellierungssprachen (für Modelle in den verschiedenen Phasen im Software Lebenslauf) bzw. Sichten

- **Algorithmenorientierte Modellierung**, Kontrollstrukturen z.B. Programmablaufplan, Struktogramm, Pseudocode
- **Regelbasierte Modellierung**, Wenn-Dann-Beziehungen z.B. Entscheidungstabelle, Regel
- **Zustandsorientierte Modellierung**, Zustand, Nebenläufigkeit z.B. Zustandsübergangsdiagramm, Petri-Netz
- **Funktionsorientierte Modellierung**, Funktionshierarchie z.B. Funktionsbaum
- **Datenflussorientierte Modellierung**, z.B. SA (Structured Analysis) von DeMarco (1978/79)
- **Datenstrukturorientierte Modellierung**, z.B. ERM (Entity Relationship Model) von Chen (1976), Syntaxdiagramm

5.1 Grundlagen und Modelltypen: Sichten

Arten von Modellierungssprachen (für Modelle in den verschiedenen Phasen im Software Lebenslauf) bzw. Sichten

- **Szenariobasierte Modellierung**, Interaktionsstrukturen z.B. Interaktionsdiagramme
- **Objektorientierte Modellierung**, Klassenstrukturen z.B. Klassendiagramm mit UML (Unified Modeling Language) von Booch, Rumbaugh, Jacobson (1996)
- **Geschäftsprozessorientierte Modellierung**, z.B. z.B. ARIS (Architektur integrierter Informationssysteme) von Scheer (1988) oder Workflow-Modelle wie BPEL

- **Formale Modellierung**, z.B. Automaten, Petri-Netze, Algebraische Spezifikation, Z, Temporale Logik, Prozessalgebra
- **Spezielle Modellierungssprachen**, z.B. für spezielle Aufgaben wie Echzeitsysteme (z.B. Realtime-UML) oder unternehmensspezifisch

5.1 Grundlagen und Modelltypen: Einsatz

Einsatz der verschiedenen Modelltypen / Sichten:

- **In verschiedenen konkreten Modellierungssprachen (teils überlappend; die Sichten an sich sind bereits teils überlappend)**

- **In verschiedenen Phasen der Softwareentwicklung (Definition, Entwurf, Implementierung, …)**

- **Auswahlkriterien:**
  - Verständlichkeit, Erlernbarkeit
  - Präzision, Eindeutigkeit
  - Textuelle und graphische Darstellung
  - Inkrementelle Veränderung
  - Partieller Entwurf

5.1 Grundlagen und Modelltypen: Gründe der Vielfalt

UNIVERSITÄT OSNABRÜCK

Gründe für die Vielfalt der Modellierungskonzepte

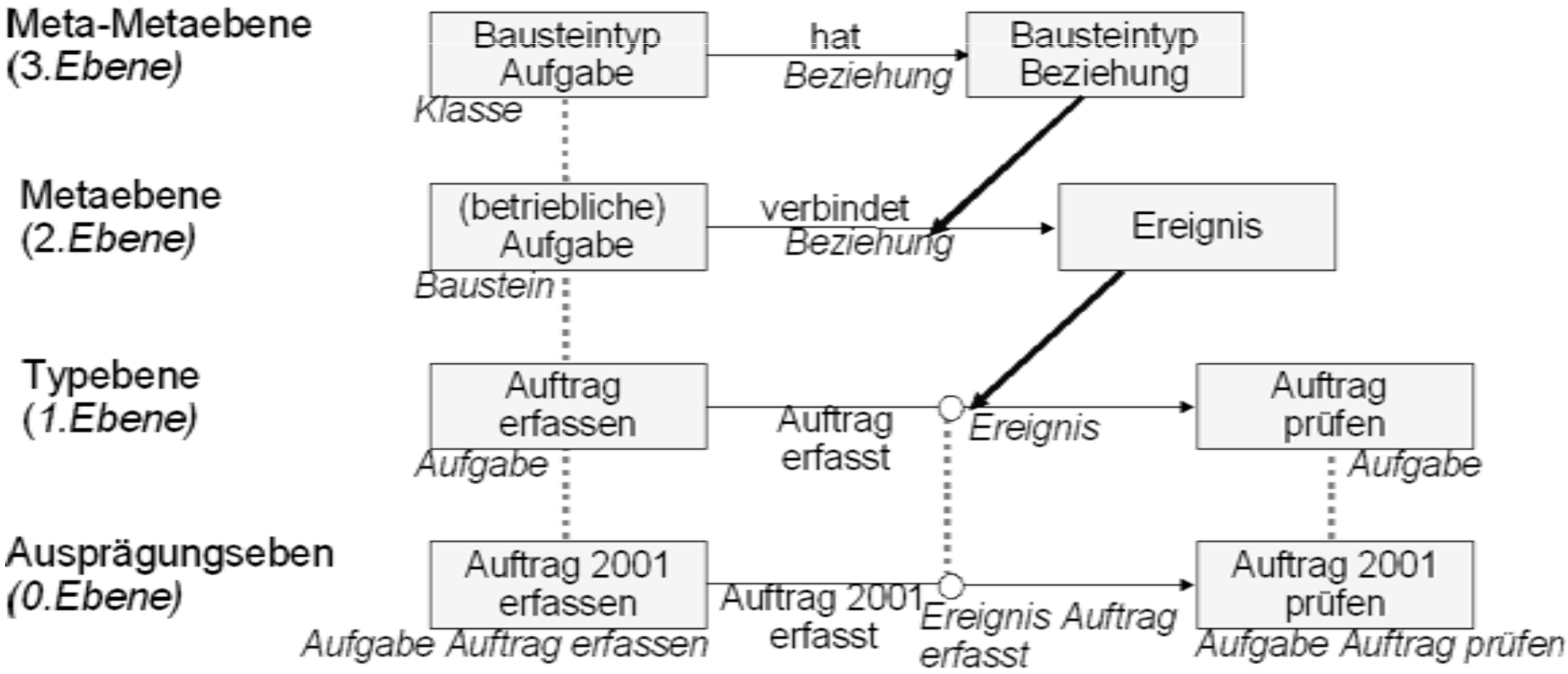
- **Spezialisierung auf Domänen des Betrachtungsgegenstands (z. B. Software, Anwendungssystem, IT-Gesamtsystem, Unternehmen, Unternehmensverbünde)**
- **Spezialisierung auf Untersuchungsaspekte (z. B. Daten, Funktionen, Organisationen, Geschäftsprozesse)**
- **unterschiedliche Aufgabenstellungen/Zielsetzungen im Entwicklungsprozess (z. B. Analyse, Gestaltung, Optimierung/Reengineering, Customizing)**
- **Unterschiede im angestrebten Lösungsverfahren (z.B. Standardsoftware, objektorientierte Software, datenbankgestützte Software, Workflow-basierte Software)**

Eine einzelne Sicht beschreibt das System im Allgemeinen nicht vollständig.

Probleme mit der Vielfalt der Modellierungskonzepte

- unterschiedliche Modelle mit “ähnlicher” Semantik und z.T. ähnlicher Notation aber nur bedingt kompatibel
- Modelle unterschiedlicher Abstraktionsstufen sind inkonsistent
- Inkonsistenz von Modellen zur Beschreibung unterschiedlicher Systemaspekte
- unklare Semantik im Modell und uneindeutige Bezeichnungen
- Missverständnisse in der Kommunikation (Gesprächspartner denken in zueinander abweichenden Modellen)

Abstraktionsgrad der Modellierungskonzepte (Hierarchie)



5.2 Programmablaufplan

Programmablaufplan (PAP) = Flussdiagramm (Flow Chart) = Ablaufdiagramm

- **sind endliche gerichtete knotenmarkierte Graphen, die**
- **den Kontrollfluss zwischen den Verarbeitungsschritten graphisch darstellen**

Einsatz:

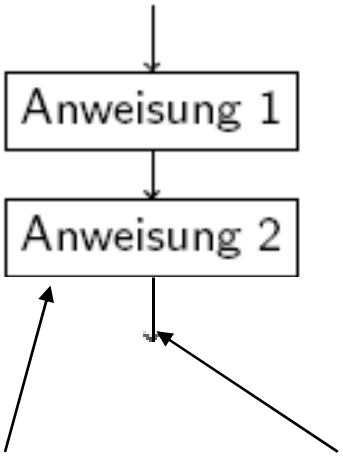
- **graphische Darstellung zur Umsetzung eines Algorithmus in einem Programm**
- **beschreibt die Folge von Operationen zur Lösung einer Aufgabe**
- **werden auch zur Darstellung von Prozessen und Tätigkeiten allgemein verwendet**

Norm (1983): DIN 66001, ISO 5807

5.2 Programmablaufplan: Notation

Notationselemente:

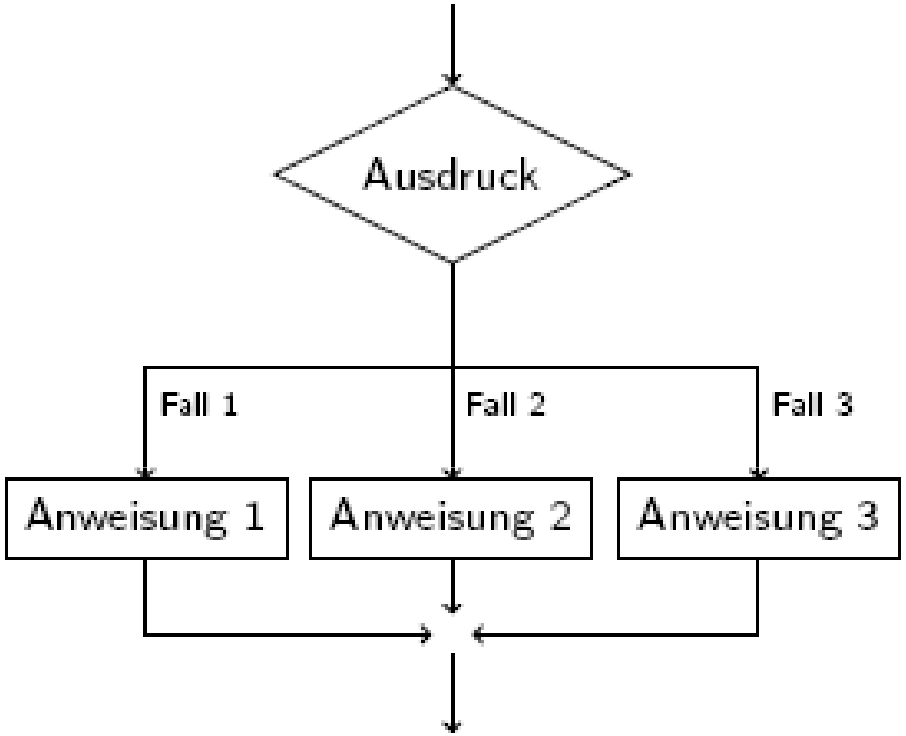
Sequenz:



Operation, Verarbeitung (Process)

Kontrollfluss, Ablauflinie

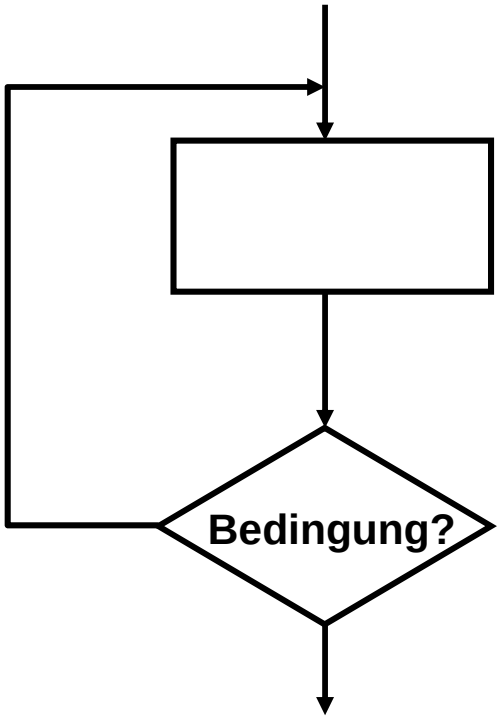
Auswahl, Verzweigung (Decision):



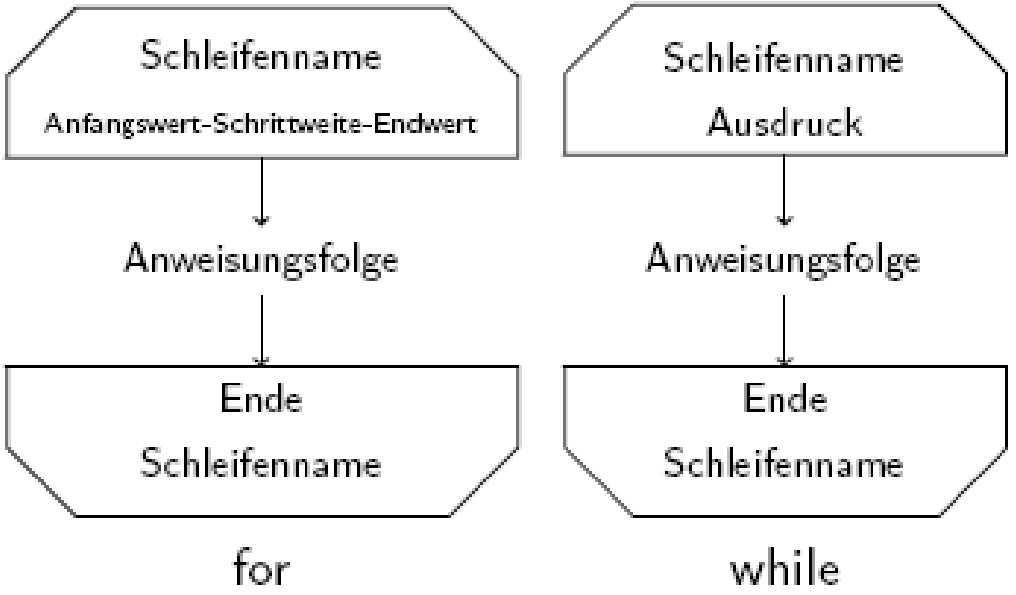
5.2 Programmablaufplan: Notation

Notationselemente:

Schleifen:



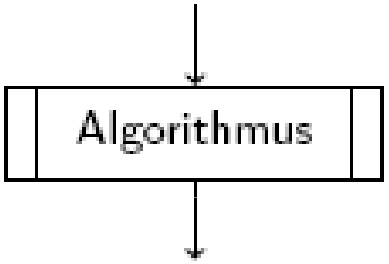
Manchmal auch: (nicht bei uns!)



5.2 Programmablaufplan: Notation

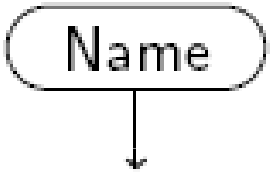
Notationselemente:

Algorithmus-Anwendung, Unterprogramm-aufruf (Predefined Process)

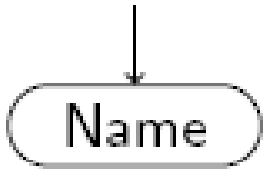


Ein- / Ausgabe (-anweisung, Data)

Anfang



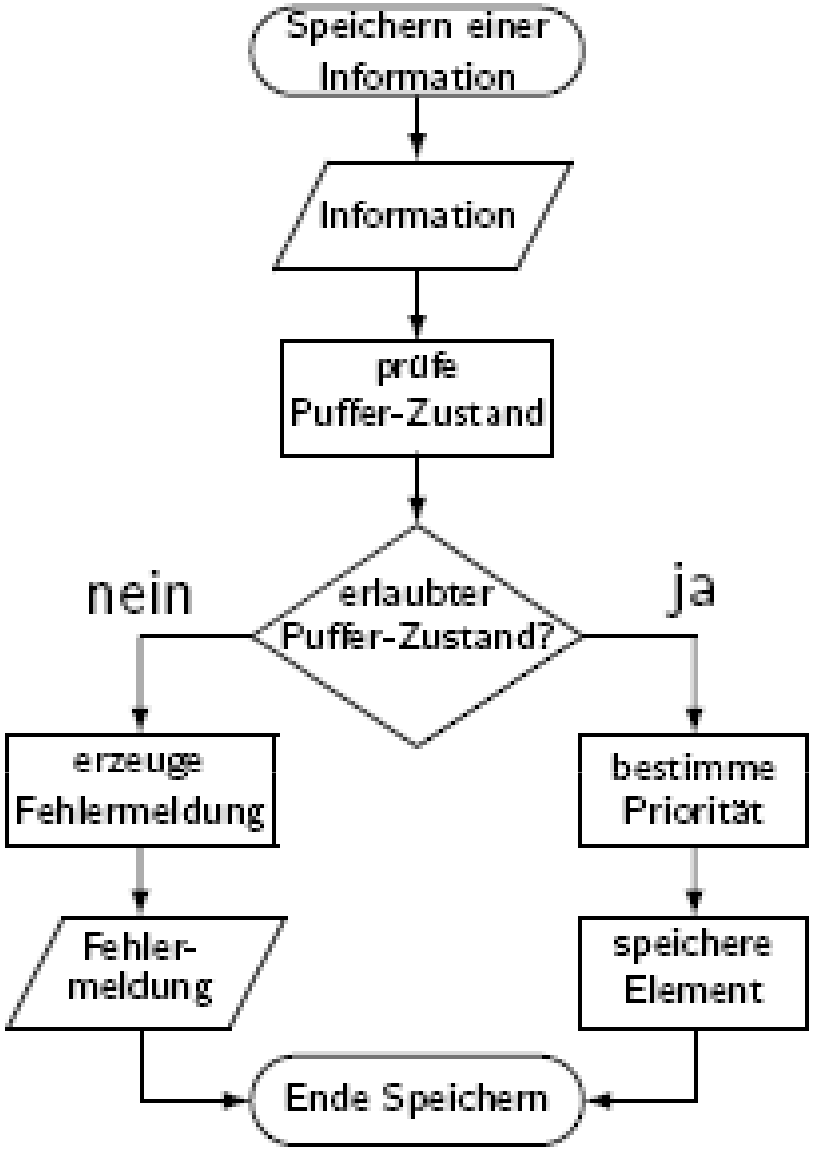
Ende



Terminator, Grenzstelle

5.2 Programmablaufplan: Beispiel

Beispiel:



5.2 Programmablaufplan: Bewertung

Nachteile:

- **Ausschliesslich Ablauforientierung („wie“ statt „was“)**
- **Keine differenzierte Darstellung von Datenstrukturen**
- **Wenig methodische Unterstützung (Verfeinerung)**
- **Ungeeignet für große Systeme (fehlende Abstraktions- und Modularisierungskonzepte, zu detaillierte Darstellung)**

Struktogramme (= Nassi-Shneidermann-Diagramme) ermöglichen eine graphische Darstellung von Kontrollstrukturen

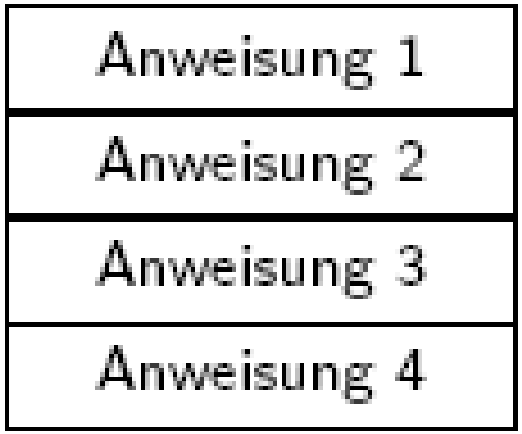
- Beschreibungsmöglichkeit für strukturorientierte Sprachkonstrukte (Programmstrukturen; ohne beliebige Programmsprünge)
- programmiersprachenunabhängige Logikdarstellung
- Top-Down Zerlegung des Gesamtproblems
- Norm: DIN 66261

5.3 Struktogramm: Notation

Anweisung:

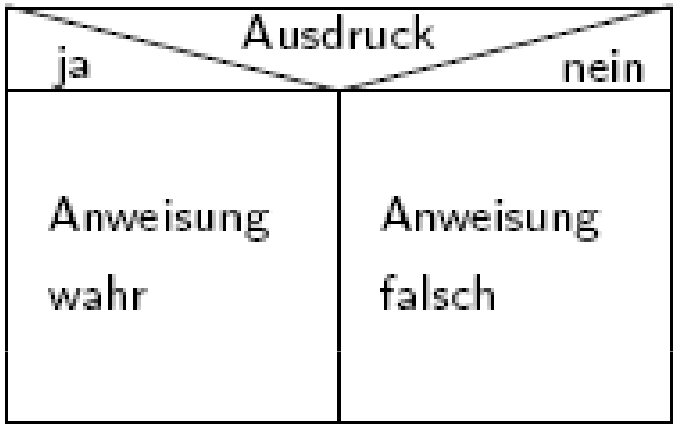
Aufruf:

Sequenz, linearer Ablauf:

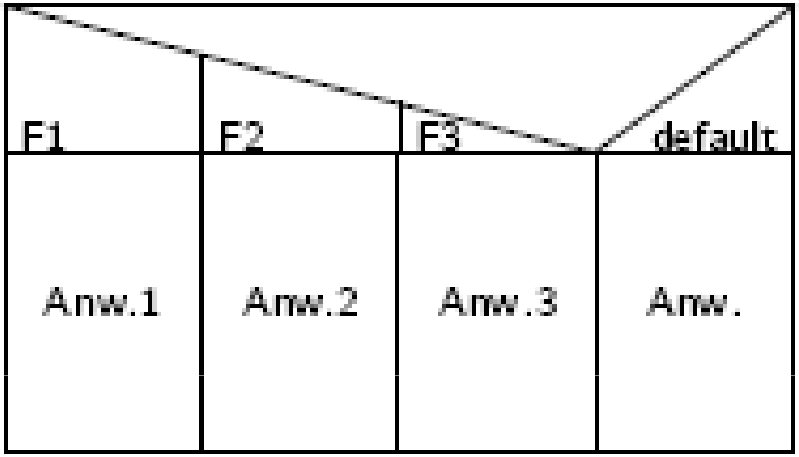


5.3 Struktogramm: Notation

Bedingte Anweisung (bedingte Auswahl, alternative Verzweigung):



Fallunterscheidung, Fallauswahl:

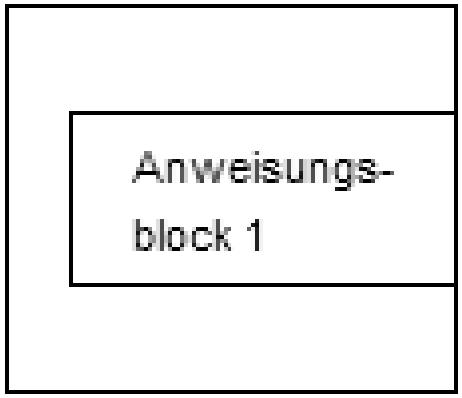


5.3 Struktogramm: Notation

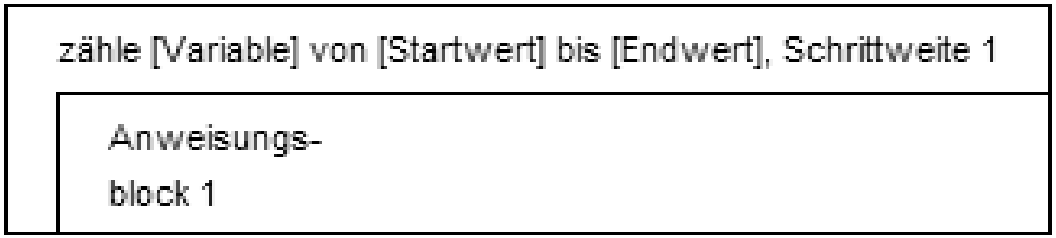
Schleife mit Schleifeneintrittsprüfung (abweisende / kopfgesteuerte Schleife):

Schleife mit Schleifenaustrittsprüfung (nicht abweisend / fußgesteuerte Schleife):

Endlosschleife:



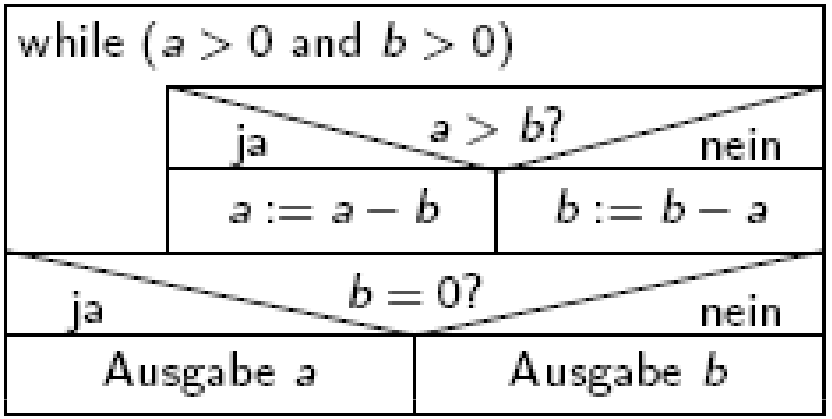
zählergesteuerte Schleife:



Beispiel:

```
WHILE a>0 and b>0 DO
   IF a>b THEN
      a:=a-b
   ELSE
      b:=b-a;
IF b=0 THEN
   WriteLn(a);
ELSE
   WriteLn(b);
```

Name des Struktogramms



Die Notationselemente können ineinander geschachtelt werden.

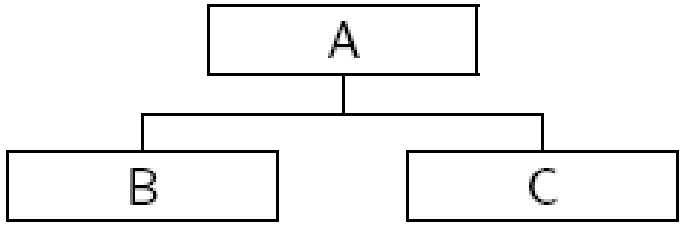
5.3 Struktogramm: Bewertung

- Ermöglicht gute graphische Darstellung von linearen Kontrollstrukturen – (beliebige) Sprünge sind nicht vorgesehen
- Sind manuell aufwendig zu zeichnen und zu ändern (Einsatz existierender Struktogramm-Generatoren)
- Besser lesbar als Programmablaufpläne

5.4 Funktionsbaum

Ein Funktionsbaum (Funktionshierarchie) strukturiert die Systemfunktionen in allgemeinere Funktionen und spezifische Teilfunktionen
(Analyse: “besteht aus”, Entwurf: “ruft auf”)

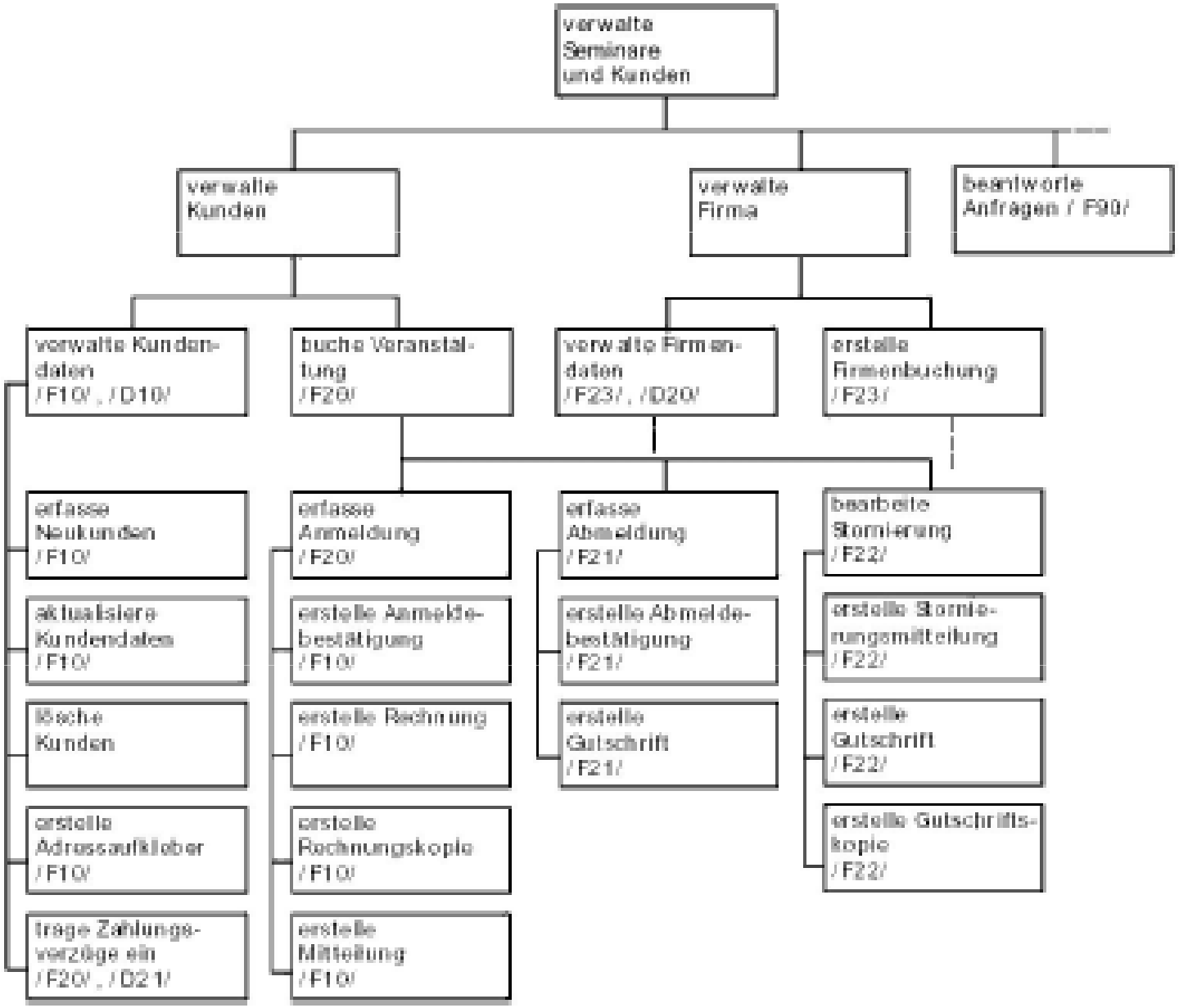
Notation:



Analyse: A besteht aus B und C.

Entwurf: A ruft B und C auf.

5.4 Funktionsbaum: Beispiel



5.4 Funktionsbaum: Bewertung

UNIVERSITÄT OSNABRÜCK

Funktionsbäume

- dienen zur systematischen Gliederung einer Vielzahl von Funktionen
- geben erste Hinweise für Implementierung und Dialoggestaltung

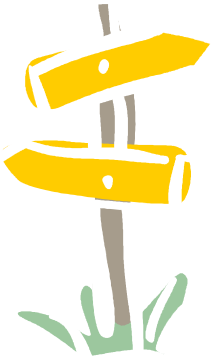
5 Software-Modelle

Zusammenfassung und Ausblick

1 Software-Krise und Software Engineering
2 Grundlagen des Software Engineering
3 Projektmanagement
4 Konfigurationsmanagement
5 Software-Modelle
6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle
7 Qualität
8 … Fortgeschrittene Techniken

- 5.1 Grundlagen und Modelltypen (Modellbegriff, Modellarten/Sichten, Einsatz, Modellvielfalt, Abstraktionsebenen)
- 5.2 Programmablaufplan
- 5.3 Struktogramm
- 5.4 Funktionsbaum

bekannte Modelle bzw. Modellierungs-sprachen



→ **Wege im Umgang mit der Software-Krise und Umsetzung der Grundlagen und Prinzipien**

Einsatz von Modellen
Weitere bekannte Modelle für verschiedene Sichten