Informatik II: Modellierung

Prof. Dr. Martin Glinz

Kapitel 3

Funktionsmodelle

Universität Zürich
Institut für Informatik

3.1 Motivation, Einsatz von Funktionsmodellen

Funktionsmodelle beschreiben und strukturieren die Funktionalität eines Systems („das, was ein System kann“)

Folgende Aspekte können modelliert werden:

- Der Steuerfluss (control flow) in einem System oder in einer Funktion eines Systems
  - Ablaufstrukturen
  - Aufrufstrukturen
- Der Datenfluss (dataflow) zwischen den Funktionen eines Systems bzw. innerhalb einer Funktion
- Der Arbeitsfluss (workflow) in einem Arbeitsprozess

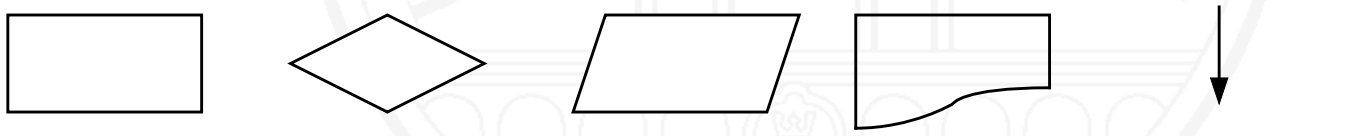
3.2 Steuerflussmodelle

- Programmablaufpläne (flow charts)
- Strukturierte Modellierung von Programmabläufen
  - Jackson-Diagramme
  - Nassi-Shneiderman-Diagramme
  - Aktigramme, Pseudocode
- Prozeduraufrufgraphen
- Entscheidungstabellen
- UML-Aktivitätsdiagramme (hier nicht behandelt, siehe Kapitel 3.4)

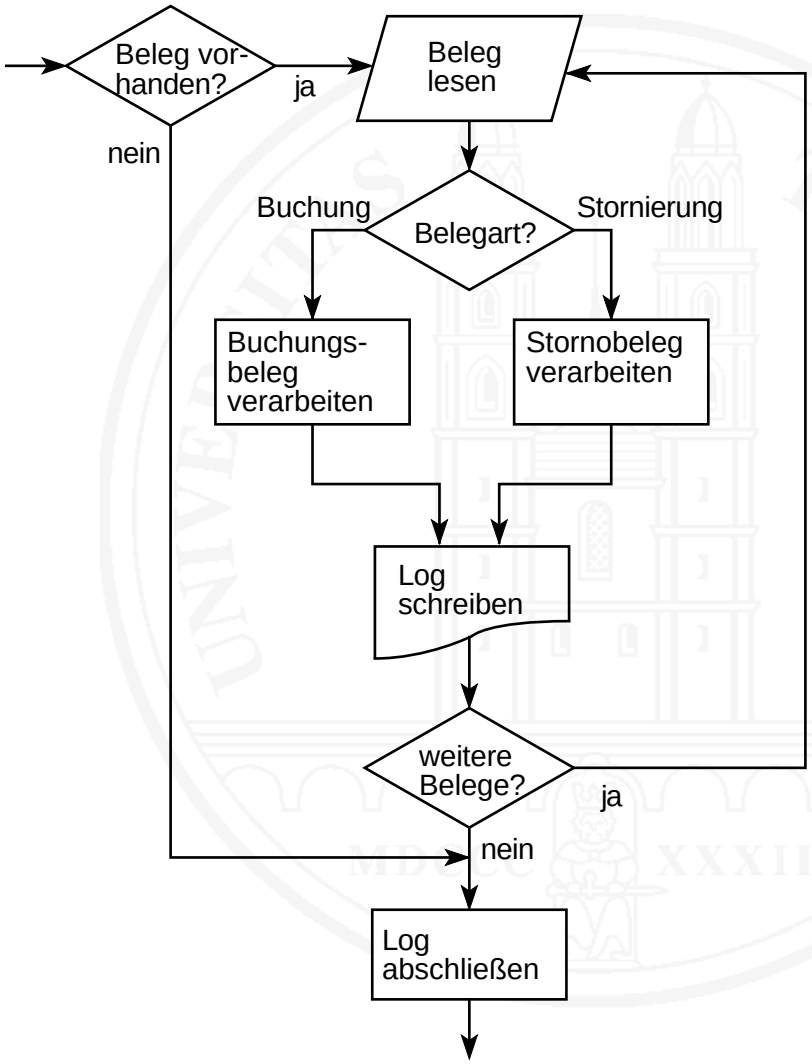
Programmablaufpläne (flow charts)

- Ältestes Mittel zur graphischen Visualisierung von Programmabläufen.
- Beliebige Strukturen können modelliert werden
- Gefahr der Modellierung von “Spaghetti”-Strukturen
- Ist veraltet und sollte nicht mehr verwendet werden

- Symbole in Programmablaufplänen:



Beispiel eines Programmablaufplans



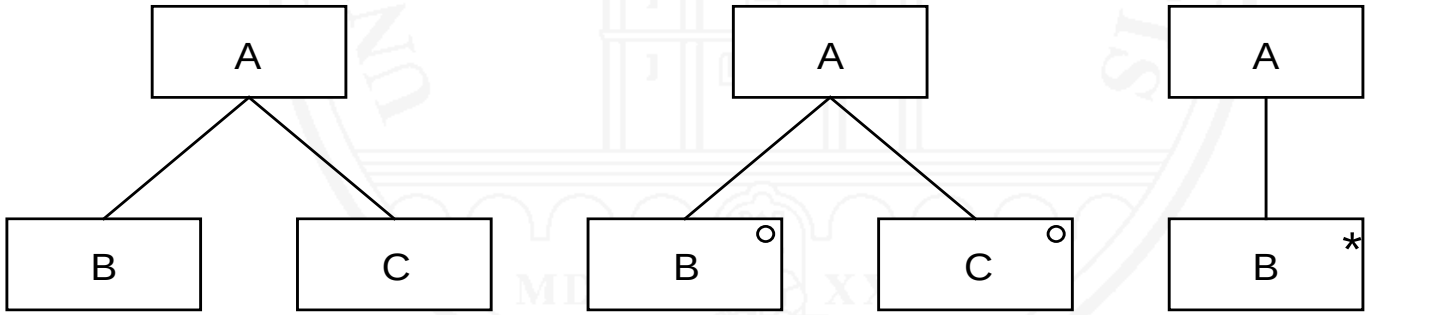
Strukturierte Modellierung von Programmabläufen

- Jedes sequentielle Programm kann aus
  - Anweisungssequenzen,
  - Alternativen (Fallunterscheidungen) und
  - Iterationen

  zusammengesetzt werden (Böhm und Jacopini 1966).
- Dies ist die Grundlage der strukturierten Programmierung.
- Gebräuchliche Notationen zur graphischen Modellierung strukturierter Abläufe:
  - Jackson-Diagramme
  - Nassi-Shneiderman-Diagramme
  - Aktigramme (oder Aktionsdiagramme, action diagrams)

Jackson-Diagramme

- *Jackson-Diagramme* (Jackson 1975) bestehen aus drei Grundelementen. Jedes Rechteck steht für eine Aktion. Die obenstehende Aktion wird durch die untenstehenden Komponenten-Aktionen definiert.
- Diagramme entstehen durch Verschachtelung: Jede Komponenten-Aktion kann durch ein Grundelement ersetzt werden. Dieser Vorgang ist beliebig oft wiederholbar.

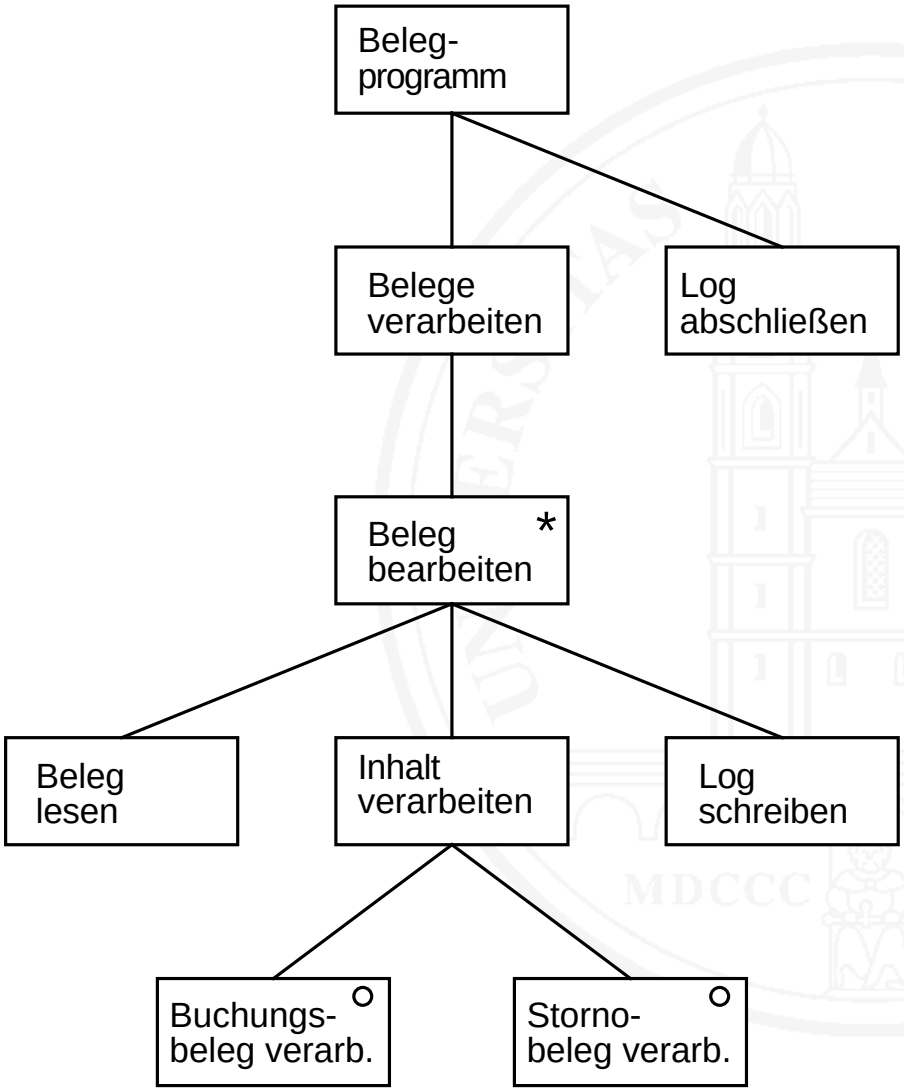


Sequenz:
A ist B gefolgt von C

Alternative:
A ist entweder B oder C

Iteration:
A ist mehrfache Wiederholung von B

Beispiel eines Jackson-Diagramms



Hinweis:

In der Programmentwicklungsmethode von Jackson (Jackson Structured Programming, JSP) werden zunächst die Datenstrukturen, die ein Programm verarbeiten soll, mit den gleichen Mitteln modelliert. Daraus wird eine Programmstruktur abgeleitet, die den zu verarbeitenden Datenstrukturen entspricht (Jackson 1975).

Modelltheoretische Konzepte in Jackson-Diagrammen

Modellelement	**Modelltheoretisches Konzept**
Aktion	Individuum
Iterationssymbol	Attribut
Alternativsymbol	Attribut
Komponenten-Beziehung	Attribut
Zu schreibendes Programm	Original
Abbildung der Programmstruktur	Abbildungsmerkmal
Keine Programmdetails	Verkürzungsmerkmal
Granularität der Darstellung	Pragmatisches Merkmal
Rechteck, Linie, Stern, ...	Notation

Nassi-Shneiderman-Diagramme

- Nassi-Shneiderman-Diagramme, auch Struktogramme genannt, (Nassi und Shneiderman, 1973) sind konzeptionell eng mit den Jackson-Diagrammen verwandt.
- Sie verwenden jedoch eine andere Notation für die Grundelemente und stellen die Verschachtelung von Aktionen graphisch auch als solche dar.
- Ferner gibt es zusätzliche Elemente, mit denen Prozeduren, das Verlassen von Prozeduren und das Verlassen von Schleifen modelliert werden können.

Aufgabe 3.1

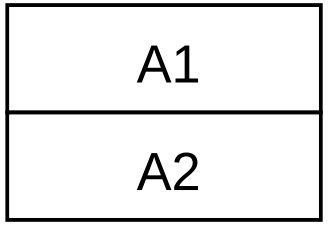
In einer Bibliothek werden Bücher wie folgt behandelt: Ein Buch wird beschafft und dann katalogisiert. Anschließend steht es im Lesesaal zum Betrachten und Ausleihen bereit.

Ein Buch kann beliebig oft betrachtet oder ausgeliehen werden. Ist ein Buch ausgeliehen, so muss es zurückgegeben werden, bevor es erneut betrachtet oder ausgeliehen werden kann.

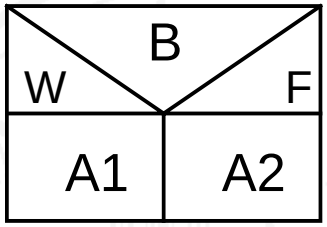
Modellieren Sie die Funktionalität eines einzelnen Buches als Jackson-Diagramm.

Grundelemente von Nassi-Shneiderman-Diagrammen

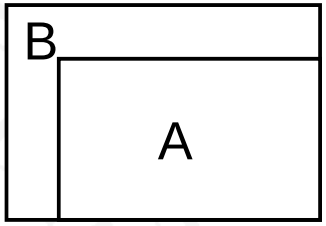
Grundsymbole:



Sequenz:
A2 folgt auf A1



Alternative:
Wenn B dann
A1 sonst A2



Iteration:
Solange B wahr
ist, führe A aus

Zusatzsymbole:

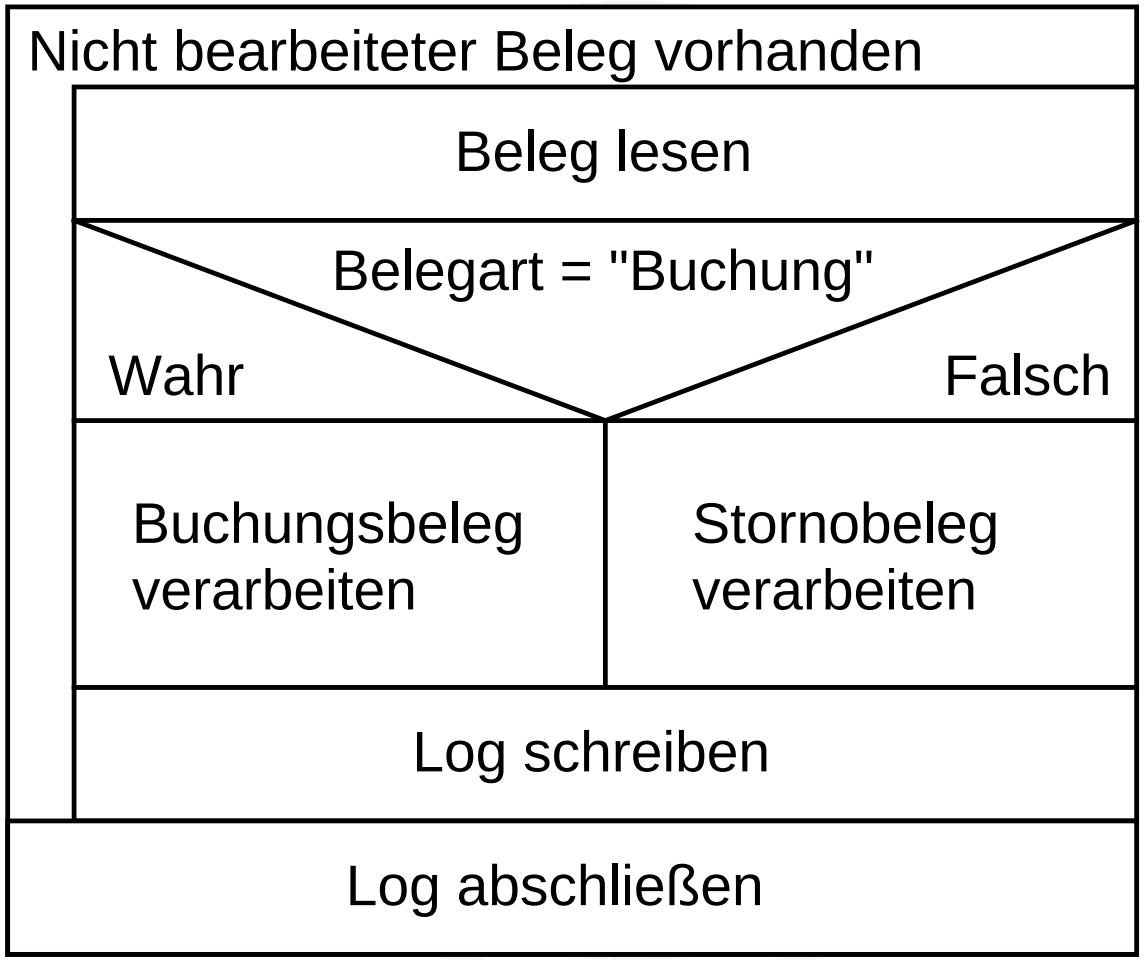
Verlassen einer
Schleife

Verlassen einer
Prozedur

name

Diagramm einer
Prozedur

Beispiel eines Nassi-Shneiderman-Diagramms



Aufgabe 3.2

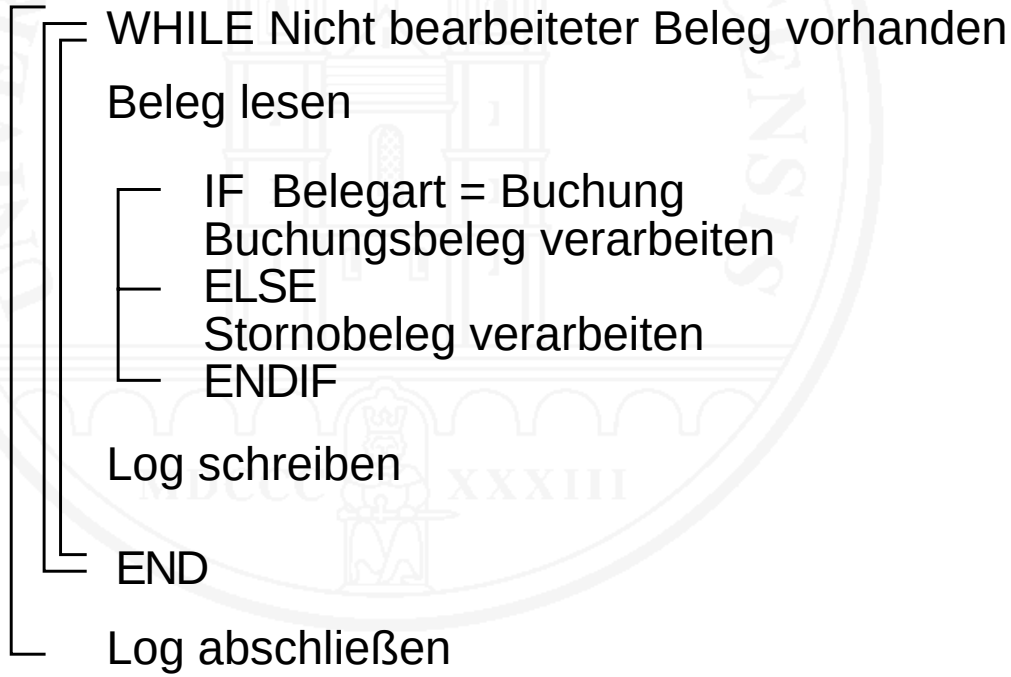
a) Beschreiben Sie die modelltheoretischen Konzepte, welche den Nassi-Shneiderman-Diagrammen zugrunde liegen.

b) Sind Jackson-Diagramme und Nassi-Shneiderman-Diagramme äquivalente Modellierungsmittel?

Aktigramme, Pseudocode

- Aktigramme (action diagrams) sind eine Mischung aus einer programmartigen Text-Notation, unterstützt durch graphische Elemente.
- Konzeptionell sind sie mit Jackson-Diagrammen und Nassi-Shneiderman-Diagrammen vergleichbar

Beispiel:



Erstellung strukturierter Ablaufmodelle

Klassischer Ansatz: **Schrittweise Verfeinerung**
(Wirth 1971, Dijkstra 1976, Wirth 1983)

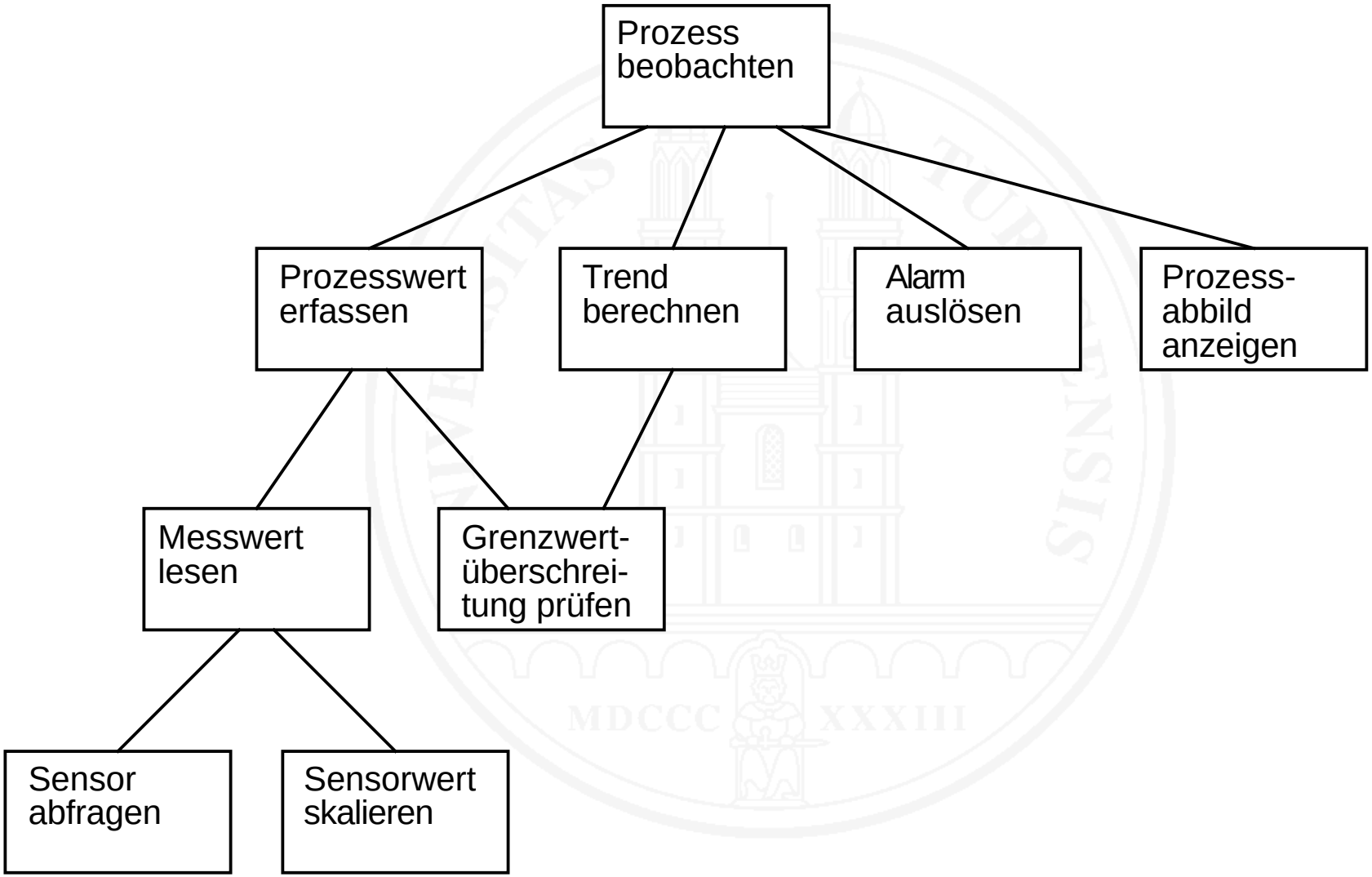
1. Wähle eine Aktion, welche den gesamten Ablauf repräsentiert
2. Zerlege diese Aktion entsprechend der Problemstellung in eine Sequenz, eine Alternative oder ein Iteration
3. Wiederhole Schritt 2 für jede beim Zerlegen neu entstandene Aktion, bis das Modell hinreichend detailliert ist

- Zerlegungsfehler zu Beginn sind später kaum mehr reparierbar
- Funktioniert nur für kleine Probleme ➪ sonst besser:
  - Teilprobleme identifizieren
  - Alle Teilprobleme durch schrittweise Verfeinerung modellieren
  - Teilmodelle durch Verschachtelung zusammensetzen

Prozeduraufrufgraphen

- **Prozeduraufrufgraphen** (structure charts, call graphs) modellieren die statische Aufrufhierarchie der Prozeduren (Unterprogramme) eines Programms.
- Das zugrundeliegende Konzept ist, nur die Prozedurnamen und die Aufrufbeziehungen zwischen den Prozeduren zu modellieren. Von den Prozedurrümpfen wird abstrahiert. Je nach Notation werden auch die Prozedurparameter noch modelliert.
- Prozeduraufrufgraphen sind ein Mittel, um die Struktur von Programmen begrenzter Größe zu visualisieren.
- Sie sind automatisch aus dem Programm-Code erzeugbar und können daher auch beim Reverse-Engineering bestehender Software eingesetzt werden.

Beispiel eines Prozeduraufrufgraphen



Entscheidungstabellen

- **Entscheidungstabellen** (decision tables) modellieren komplexe Entscheidungsabläufe in tabellarischer Form
- Eine Entscheidungstabelle besteht aus
  - einem Bedingungsteil (mögliche Bedingungskombinationen)
  - und einem Aktionsteil (auszuführende Aktion(en))
- Einsatz: Anschauliche tabellarische Darstellung komplexer Entscheidungsstrukturen mit vielen Bedingungen

Beispiel einer Entscheidungstabelle

	1	2	3	4	5	6	7
Bestellbetrag > Kreditlimite	N	J	J	J	J	J	J
Bestellbetrag > 1.5 $\star$ Kreditlimite	–	N	N	J	J	J	J
Sonderkunde	–	J	N	J	N	N	N
Jahresumsatz > 50000	–	–	–	–	N	J	N
2-Monats-Umsatz > 20000	–	–	–	–	J	–	N
Bestellung ausliefern	X	X					
Bestellung an Verkaufsleiter			X	X	X	X	
Bestellung zurückweisen							X

Vereinfachung von Entscheidungstabellen

- Bei n binären Bedingungen hat die volle Tabelle 2^n Spalten
- Durch Zusammenfassung von Bedingungskombinationen, die gleiche Aktionen auslösen, wird die Tabelle kompakter.
- Beim Zusammenfassen muss sichergestellt werden, dass
  - es keine Bedingungskombination gibt, für die mehr als eine Spalte zutrifft (Widerspruchsfreiheit)
  - für jede Bedingungskombination mindestens eine Spalte zutrifft (Vollständigkeit)

Methodik der Erstellung von Entscheidungstabellen

- Alle Bedingungen (und deren mögliche Werte) ermitteln
- Vollständigen Bedingungsteil (ohne „egal“-Werte) aufbauen; Bedingungswerte systematisch variieren
- Aktionsteil aufbauen: Zu jeder Bedingungskombination die zu treffende(n) Aktion(en) notieren
- Tabelle durch Zusammenfassen von Spalten mit gleichem Aktionsteil vereinfachen

Aufgabe 3.3

Franziska Freitag hat folgendes Problem mit der Vorlesung Informatik II am Dienstag Nachmittag:

- Wenn sie Lust auf Lernen hat und ihr Freund Abendschicht hat, besucht sie die ganze Vorlesung.
- Wenn sie Lust auf Lernen hat und ihr Freund keine Abendschicht hat, besucht sie die Vorlesung, geht aber in der Pause.
- Wenn sie keine Lust auf Lernen hat und nicht an die Vorprüfung denkt, geht sie nicht zur Vorlesung.
- Wenn sie keine Lust auf Lernen hat, aber an die Vorprüfung denkt, besucht sie die die ganze Vorlesung, wenn ihr Freund Abendschicht hat. Hat er keine Abendschicht, bleibt sie nur bis zur Pause.

Modellieren Sie das Verhalten von Franziska Freitag als Entscheidungstabelle. Vereinfachen Sie die Tabelle so weit wie möglich.

3.3 Datenflussmodelle

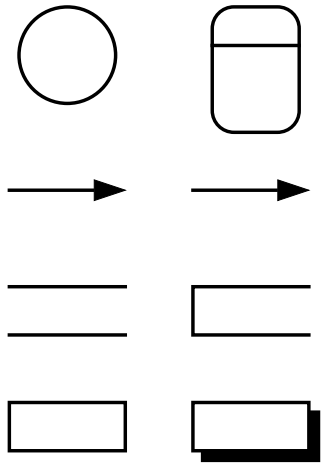
Datenflussmodelle beschreiben die Funktionalität eines Systems durch Aktivitäten und die Datenflüsse zwischen diesen Aktivitäten.

- Basis: Konzept der datengesteuerten Verarbeitung:
  - Eine Aktivität arbeitet, wenn ihre benötigten Datenflüsse eintreffen
  - Sie erzeugt bei ihrer Arbeit neue Datenflüsse
  - Diese steuern entweder andere Aktivitäten an oder verlassen das System als Ergebnis
- Der Steuerfluss ist implizit und wird nicht modelliert
- Ein reines Datenflussmodell modelliert nur Datentransport und Datenbearbeitung, nicht aber Datenspeicherung
- In der Praxis hinderliche Einschränkung ➪ Hinzufügen von Mitteln zur Modellierung von Datenspeicherung
- Bekanntester Vertreter: Datenflussdiagramm

Datenflussdiagramme

Ein **Datenflussdiagramm** (dataflow diagram, DFD) modelliert den Transport, die Bearbeitung und die Speicherung von Daten.

Zwei Notationen sind gebräuchlich:



Aktivität (Prozess; activity, process)

Datenfluss (dataflow)

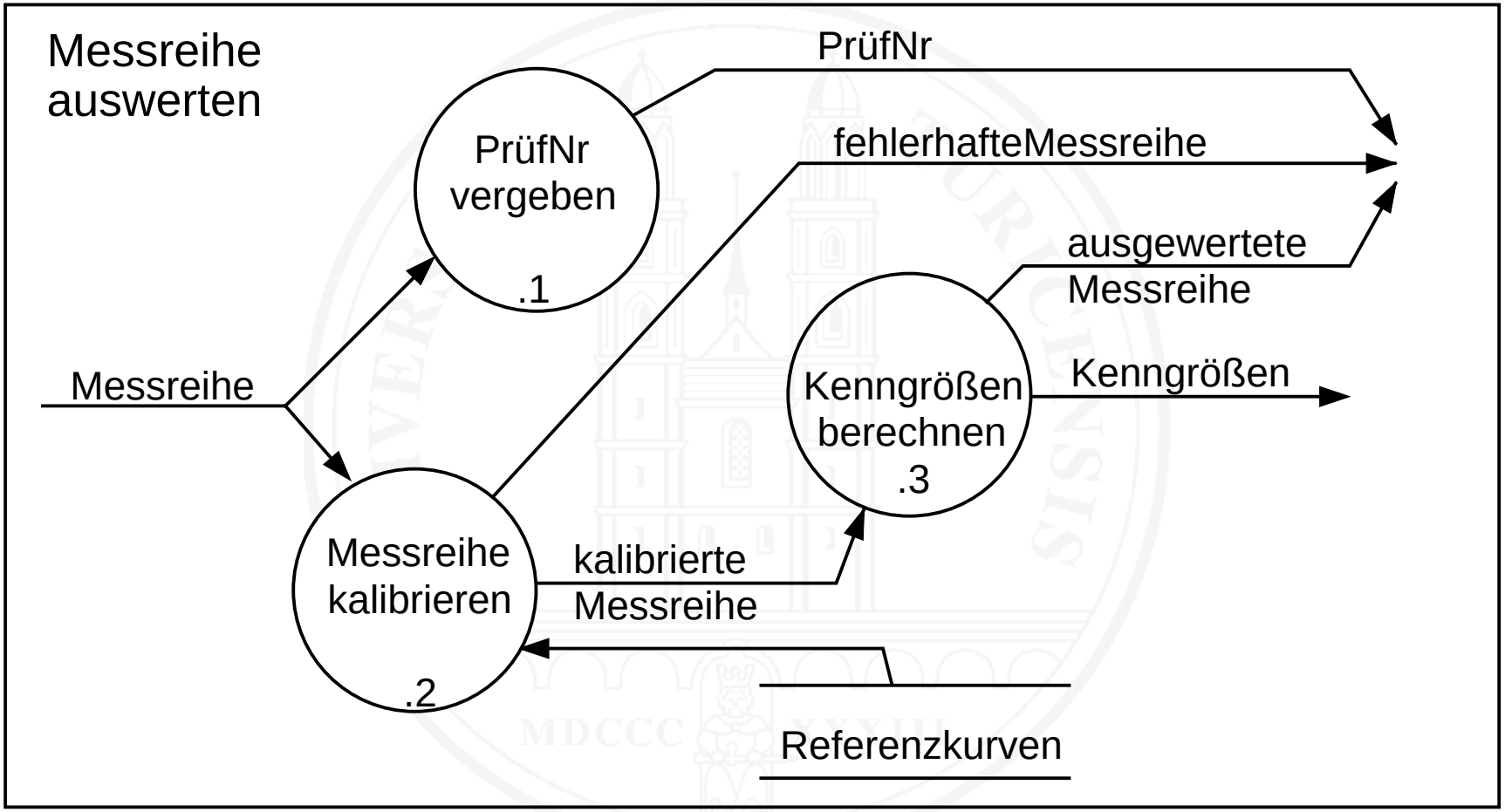
Speicher (store, file)

Endknoten (terminator, terminal, external)

Interpretation von Datenflussdiagrammen

- **Datenflüsse** transportieren Daten*pakete*, die von Aktivitäten oder Endknoten produziert bzw. konsumiert werden.
- **Aktivitäten** arbeiten nur dann, wenn alle von ihnen benötigten Eingabe-Datenflüsse vorliegen. Die Aktivität konsumiert die Daten, bearbeitet sie und produziert Ausgabe-Datenflüsse. Sie kann dabei zusätzlich Speicherinhalte lesen oder schreiben.
- **Speicher** modellieren Datenbehälter. Ihr Inhalt kann gelesen werden (ohne den Speicher zu verändern) und geschrieben werden (dabei wird der alte Inhalt zerstört).
- **Endknoten** sind Aktivitäten in der Systemumgebung.

Beispiel eines Datenflussdiagramms

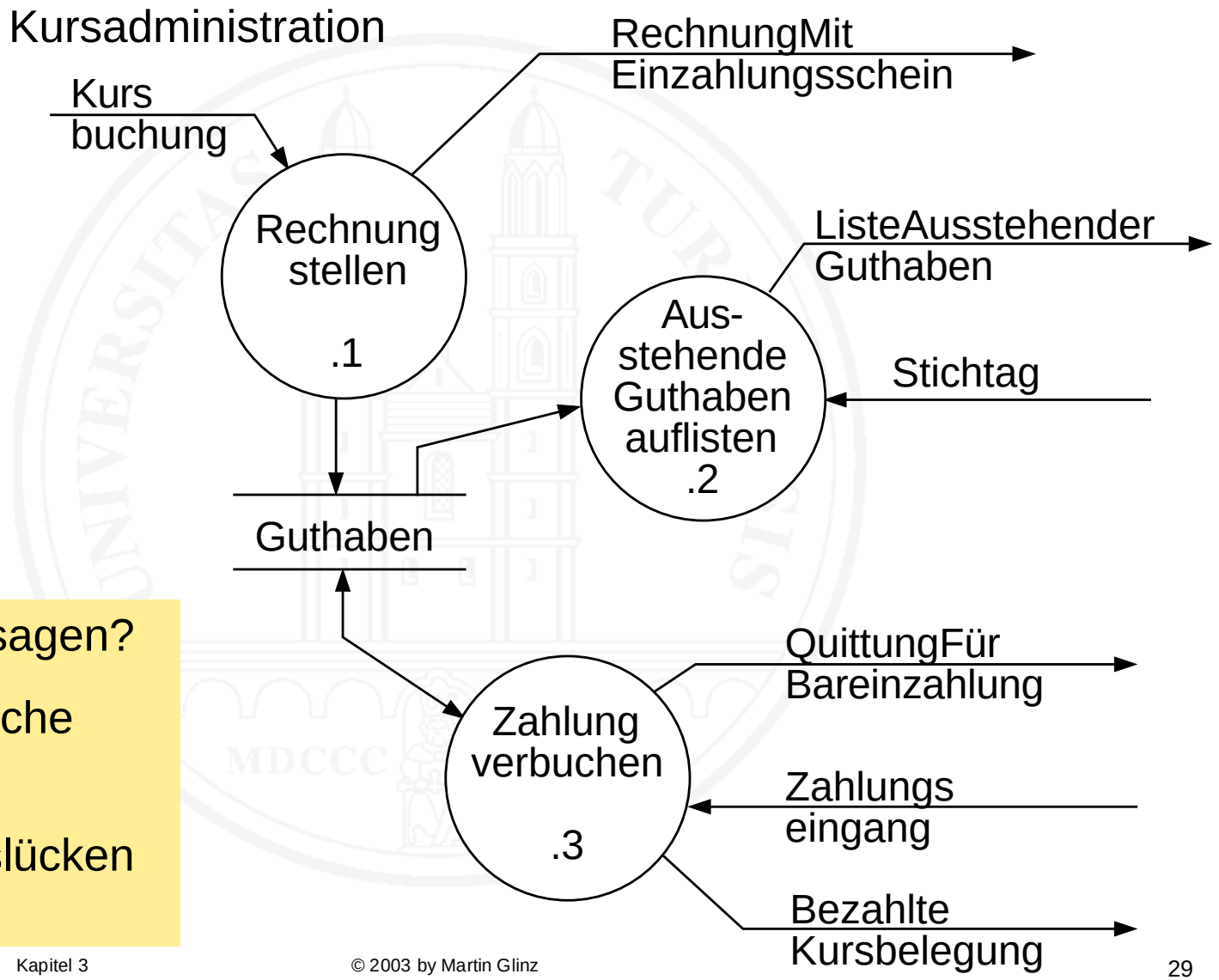


Aufgabe 3.4

Das nachstehende Datenflussdiagramm zeigt einen Ausschnitt aus dem Modell der Kursadministration einer Volkshochschule.

Fassen Sie die Aussagen dieses Datenflussdiagramms in Worte. Wo können Sie (fast) sichere Aussagen machen, was ist nur wahrscheinlich, was wissen Sie nicht? Wie könnten Sie die Wissenslücken schließen?

Aufgabe 3.4 (Fortsetzung)



- Sichere Aussagen?
- Wahrscheinliche Aussagen?
- Wie Wissenslücken schließen?

Notwendige Ergänzungen

- Datenflussdiagramme sind Übersichtsmodelle. Zur Präzisierung müssen
  - die Namen aller Datenflüsse und Speicher definiert werden
  - die Funktionalität jeder Aktivität beschrieben werden

- Eine Aktivität kann wiederum durch ein Datenflussdiagramm beschrieben werden →DFD-Hierarchie
- Es braucht eine Notation zur Beschreibung elementarer, nicht zerlegter Aktivitäten

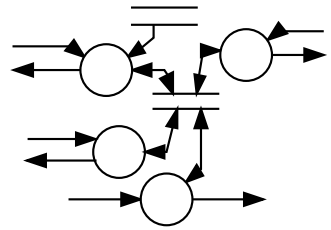
Strukturierte Analyse

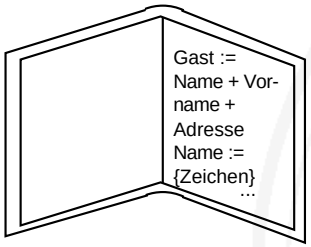
Strukturierte Analyse (structured analysis) ist das klassische Beispiel einer datenflussorientierten Methode
(DeMarco 1978, Gane und Sarson 1979, McMenamin und Palmer 1984, Yourdon 1989)

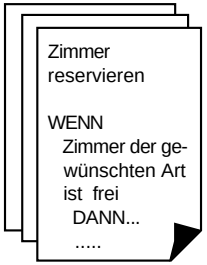
SA verwendet vier Konzepte:

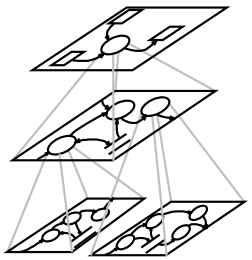
- Datenflussdiagramme als zentrales Modellierungsmittel
- Ein Datenlexikon zur Datendefinition
- Mini-Spezifikationen zur Beschreibung elementarer Aktivitäten
- Hierarchische Zerlegung der DFD zur Strukturierung des Modells

Grundkonzepte der Strukturierten Analyse









Datenfluss-
Diagramm(e)

Datenlexikon

SA

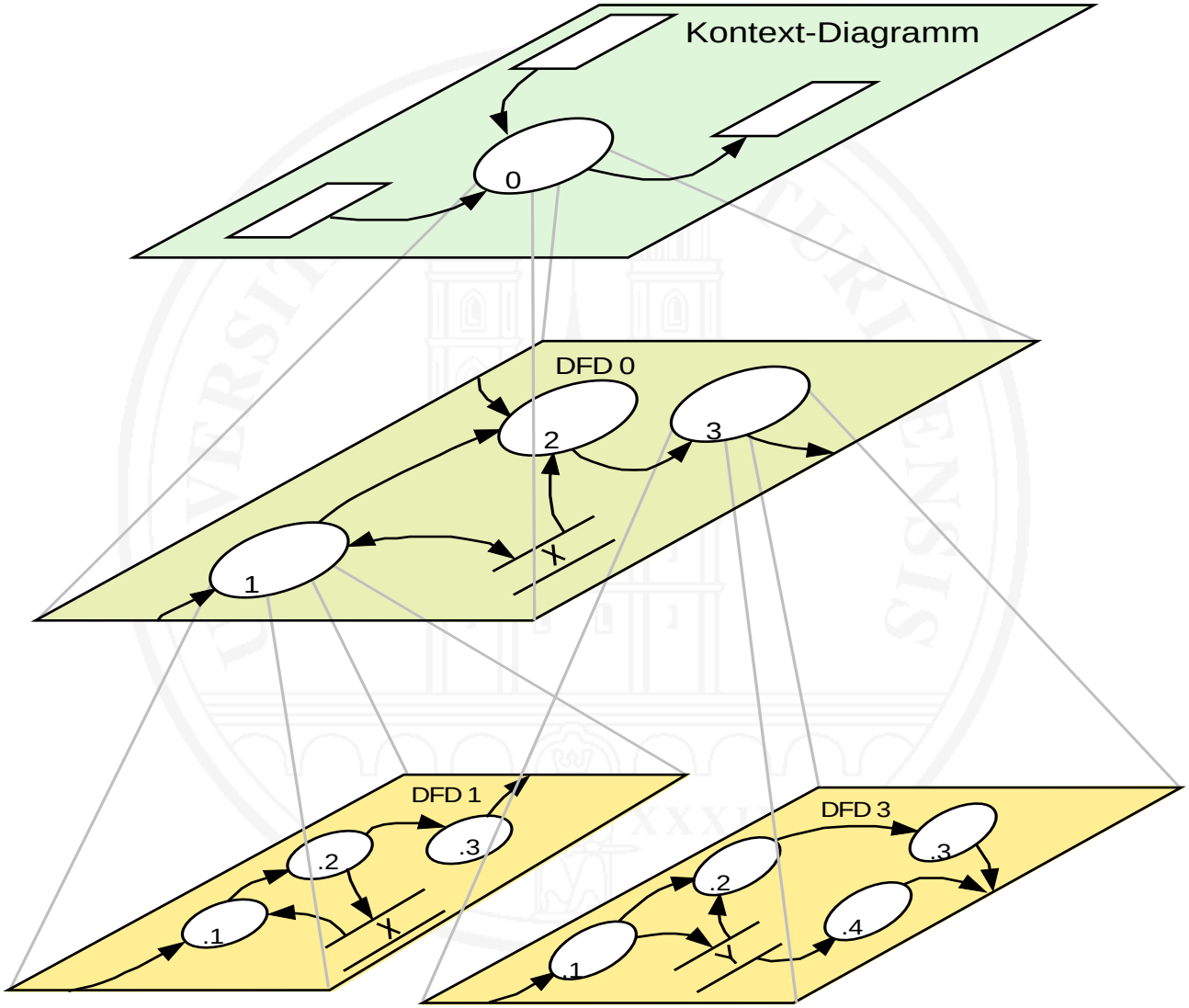
Mini-
Spezifikationen

Dekomposition

Hierarchische Zerlegung

- Die Datenflussdiagramme eines Systems können hierarchisch in Schichten angeordnet werden
- Jede Ebene fasst die Datenflussdiagramme der darunterliegenden Ebenen zu je einer Aktivität zusammen
- In der Regel wird ein dazu passendes hierarchisches Nummerierungs-schema für Aktivitäten und DFD verwendet
- Bei geeigneter Wahl der Zerlegung kann so ein komplexes Modell schrittweise in einfachere Teilmodelle zerlegt werden

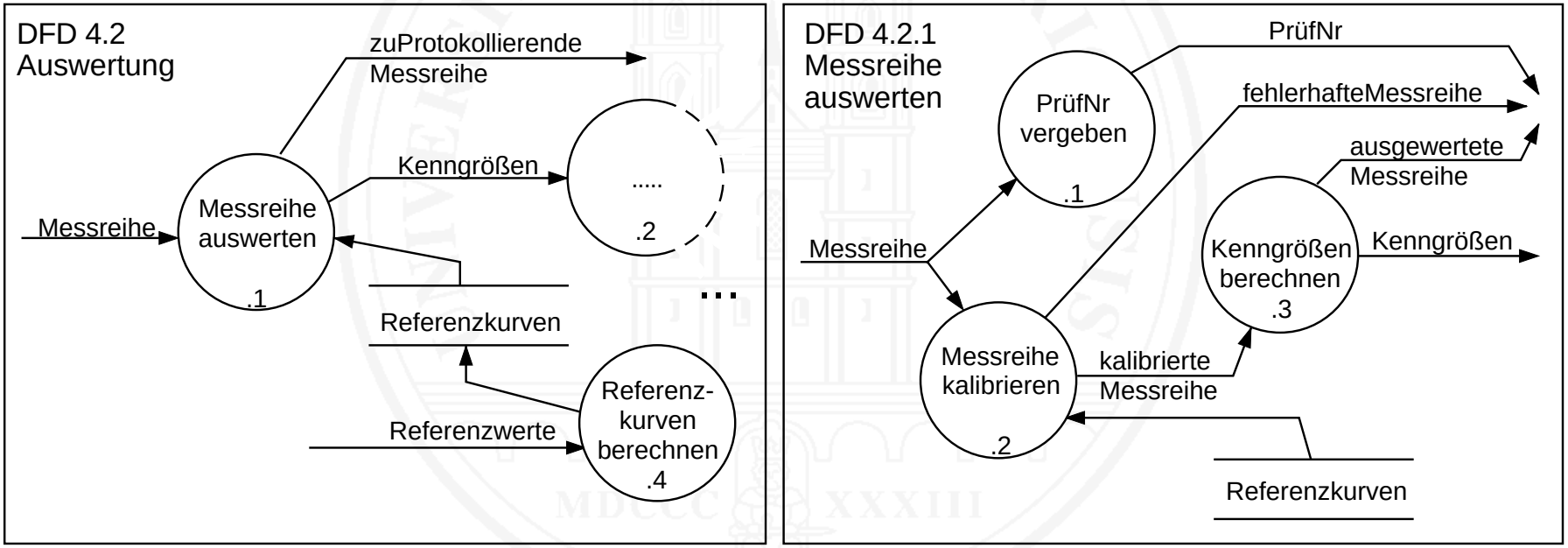
Beispiel einer DFD-Hierarchie in SA



Konsistenz der Hierarchieebenen

In jeder DFD-Hierarchie müssen die Diagramme auf den verschiedenen Ebenen miteinander konsistent sein.

Beispiel: zwei konsistente DFD:



Auszug aus dem Datenlexikon: zuProtokollierendeMessreihe := PrüfNr + [ausgewerteteMessreihe | fehlerhafteMessreihe]

Modelltheoretische Konzepte in SA

Modellelement	**Modelltheoretisches Konzept**
Aktivität, Speicher, Endknoten	Individuum
Datenfluss	Attribut
Datenlexikon-Eintrag	Attribut
Mini-Spezifikation	Attribut
Problembereich	Original
Abbildung der Flussstruktur	Abbildungsmerkmal
(beispielsweise) keine Beziehungen zwischen Speichern	Verkürzungsmerkmal
Auswahl und Granularität von Aktivitäten und Speichern	pragmatisches Merkmal
Kreis, Doppellinie, Pfeil ...	Notation

Methodik der Modellerstellung

Klassisches Vorgehen (event partitioning, McMenamin und Palmer 1984)

1. Erstelle ein Kontext-Diagramm (vgl. Kapitel 6)
2. Erstelle eine Ereignisliste (Ereignis = Anstoß aus der Außenwelt, der eine Reaktion des Systems erfordert)
3. Zeichne ein DFD mit einer Aktivität für jeden Vorgang (= Verarbeitung eines Ereignisses und Erzeugung der erwarteten Reaktionen)
   - Schließe interaktiv die dabei auftretenden Informationslücken
   - Definiere die verwendeten Daten
   - Skizziere Mini-Spezifikationen
4. Restrukturiere dieses (in der Regel sehr große) DFD in eine Hierarchie
5. Vervollständige das Modell

Methodik der Modellerstellung

Das klassische Vorgehen hat bei großen Modellen erhebliche Schwächen

➪ **Besseres Vorgehen:**

2' Gliedere die Aufgabe in Teile, die (soweit wie möglich) in sich geschlossen sind

- Erstelle die Ereignisliste
- Bilde Unter-Ereignislisten für jede Teilaufgabe

3' Arbeite gemäß 3 für jede Teilaufgabe; stimme dabei das Neue mit schon vorhandenen Teilmodellen ab

4' Arbeite gemäß 4 für jede Teilaufgabe (falls nötig)

5' Arbeite gemäß 5 und synthetisiere die oberste(n) Zerlegungsebene(n) aus den Teilmodellen

Aufgabe 3.5

Ein Kurs-Informationssystem für eine Volkshochschule ist zu spezifizieren.

Folgende Informationen sind Ihnen bekannt:

Meldet sich jemand zu einem Kurs an, so werden der belegte Kurs sowie Name und Adresse der angemeldeten Person erfasst. Das Kursgeld kann entweder direkt bar bezahlt oder mit einem bei der Anmeldung erzeugten Einzahlungsschein eingezahlt werden. Eingehende Zahlungen werden verbucht; nur Bareinzahlungen werden quittiert. ...
Bis eine Woche vor Kursbeginn kann eine angemeldete Person sich wieder abmelden. Hat sie das Kursgeld bereits bezahlt, wird der Betrag von der Buchhaltung zurückerstattet. Die Abmeldung wird schriftlich bestätigt. ...
Eine Woche vor Kursbeginn wird automatisch eine Liste aller Kursteilnehmer erzeugt. ...

a) Erstellen Sie aufgrund dieser Informationen eine Ereignisliste.

Aufgabe 3.5 (Fortsetzung)

b) Modellieren Sie mit Hilfe der Ereignisliste das Problem der Kursadministration als Datenflussdiagramm. Stimmen Sie Ihr Modell mit dem Modell der Buchhaltung (Aufgabe 3.4) ab.

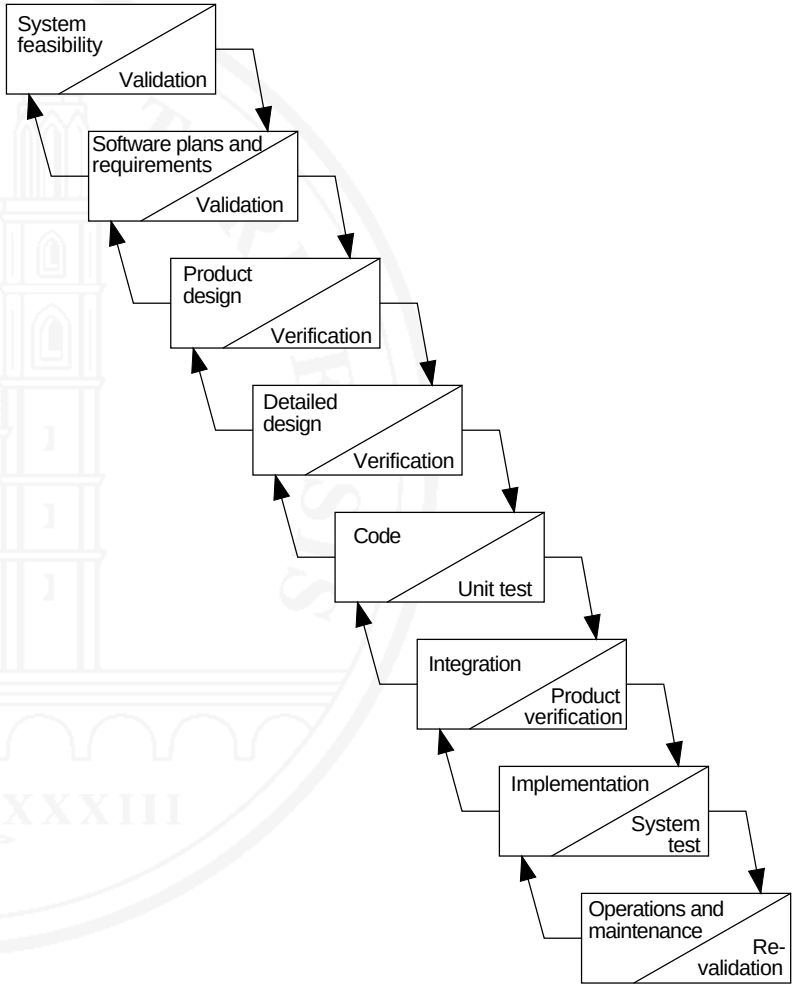
3.4 Arbeitsflussmodelle

Arbeitsflussmodelle (workflow models) dienen zur Beschreibung von Arbeitsprozessen. Sie modellieren den Ablauf von Arbeitsschritten, die den Ablauf steuernden Ereignisse, die beteiligten Stellen bzw. die Rollen der beteiligten Personen und die verwendeten bzw. erzeugten Materialien.

- Einfache Arbeitsflussmodelle modellieren nur eine Folge von Arbeitsschritten
- Vollständige Arbeitsflussmodelle beschreiben alle Elemente eines Arbeitsprozesses

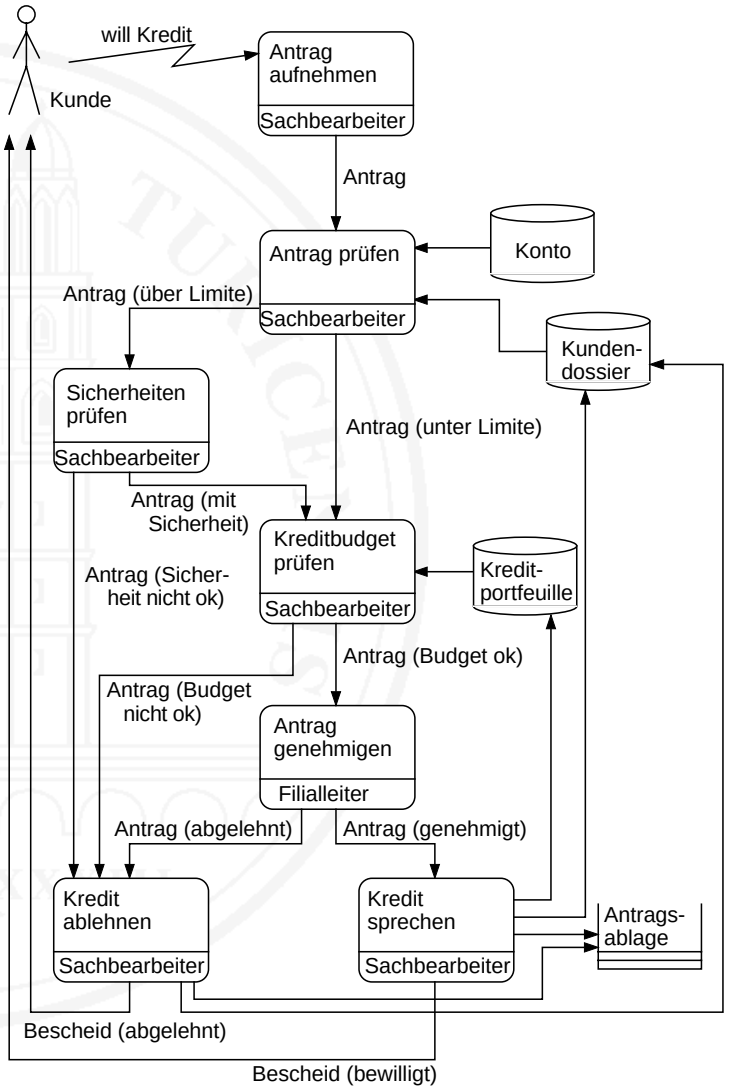
Beispiel eines einfachen Arbeitsflussmodells

Das sogenannte Wasserfallmodell
für die Entwicklung von Software



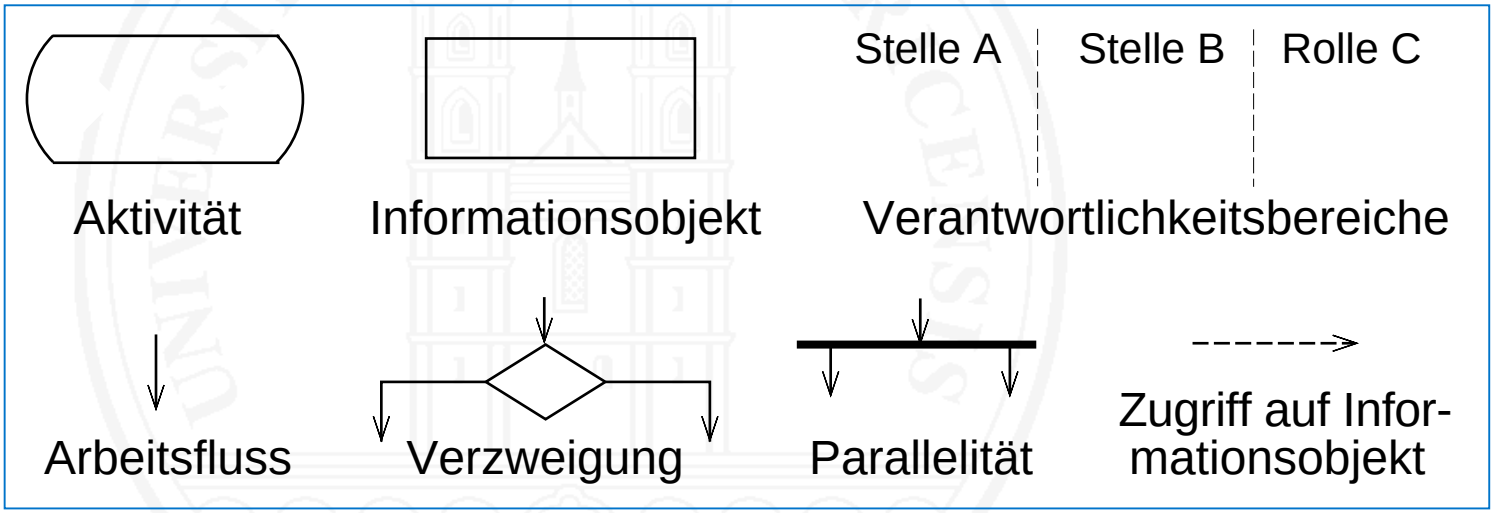
Notationen für Arbeitsflussmodelle

- Generische Notation, siehe Beispiel rechts (Bearbeitung eines Kreditantrags)
- UML-Aktivitätsdiagramme
- Ereignis-Prozess-ketten-Diagramme (in dieser Vorlesung nicht behandelt)
- Sprachen zur Verhaltens-modellierung (zum Beispiel Statecharts oder Petrinetze, vgl. Kapitel 4)



UML-Aktivitätsdiagramme

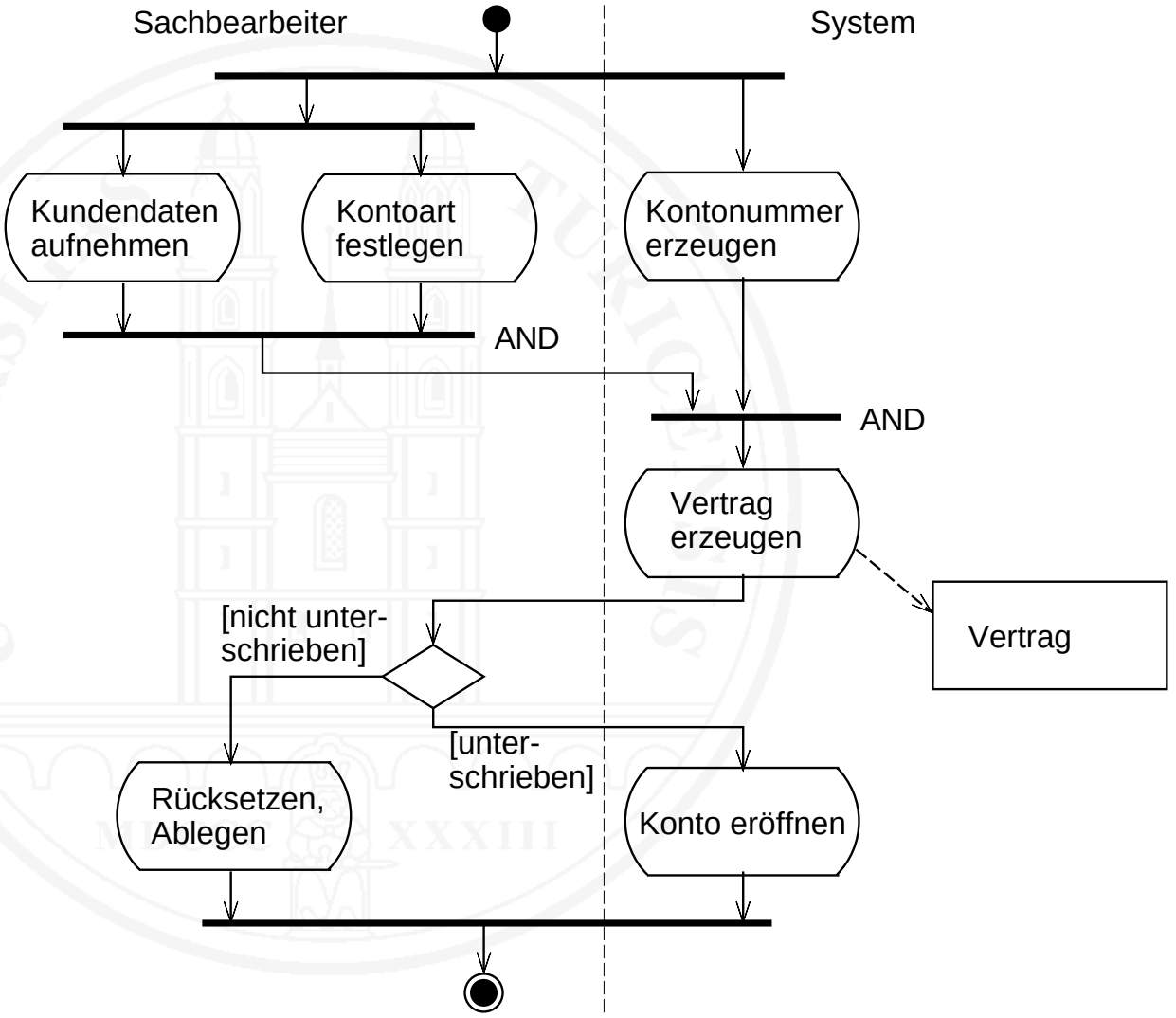
- In der Sprache UML (Unified Modeling Language) [Rumbaugh, Jacobson, Booch 1999] gibt es sogenannte **Aktivitätsdiagramme** (activity diagrams)
- Notation:



- UML-Aktivitätsdiagramme eignen sich als Notation für
  - Arbeitsflussmodelle
  - Steuerfluss- und Interaktionsmodelle

Beispiel eines Arbeitsflussmodells mit einem UML-Aktivitätsdiagramm

Prozess zur Eröffnung eines Kontos:



Aufgabe 3.6

Dieter Dollmaier geht jeden Morgen als erstes zum Kaffeeautomaten. Wenn dieser noch ausgeschaltet ist, schaltet er ihn ein. Während der Automat aufwärmt, spült er seine Tasse und schäkert mit seiner Kollegin Claudia Kussmaul. Dann lässt er seinen Kaffee heraus, nimmt sich Milch und Zucker, verabredet sich gleichzeitig mit Claudia Kussmaul zum Abendessen und kehrt anschließend beschwingt in sein Büro zurück.

Modellieren Sie diesen „Arbeitsprozess“ mit einem UML-Aktivitäts-diagramm.

Wie abstrakt ist dieses Arbeitsflussmodell im Vergleich zu den weiter oben gezeigten Modellen?

Literatur

Boehm, B. (1981). *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice Hall.

Böhm, C. G. Jacopini (1966). Flow Diagrams, Turing Machines and Languages With Only Two Formation Rules. *Communications of the ACM* **9**, 5 (May 1966). 366-371.

Chvalovski, V. (1983). Decision Tables. *Software Practice and Experience* **13**, 5 (Mai 1983). 423-429.

DeMarco, T. (1979). *Structured Analysis and System Specification*. New York: Yourdon Press.

Dijkstra, E.W. (1976). *A Discipline of Programming*. Englewood Cliffs, N.J.: Prentice Hall.

Gane C., T. Sarson (1979). *Structured Systems Analysis: Tools and Techniques*. Englewood Cliffs, N.J.: Prentice Hall.

Jackson, M. (1975). *Principles of Program Design*. New York: Academic Press.

Literatur – 2

McMenamin, S.M., J.F. Palmer (1984). *Essential Systems Analysis*. New York: Yourdon Press.

Nassi, I., B. Shneiderman (1973). Flowchart Techniques for Structured Programming. *SIGPLAN Notices* August 1973. 12-26.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Reading, Mass. : Addison-Wesley.

Yourdon. E.N., L.L. Constantine (1978). *Structured Design*. Englewood Cliffs, N.J.: Prentice Hall.

Yourdon, E. (1989). *Modern Structured Analysis*. Englewood Cliffs, N.J.: Prentice Hall.

Wirth, N. (1971). Program Development by Stepwise Refinement. *Communications of the ACM* **14**, 4 (April 1971). 221-227.

Wirth, N. (1983). *Systematisches Programmieren*. Stuttgart: Teubner.