Informatik II: Modellierung

Prof. Dr. Martin Glinz

Kapitel 5

Funktionsmodellierung I: Steuerflussmodelle

Universität Zürich
Institut für Informatik

Inhalt

5.1 Motivation

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
  - Entscheidungslogik
- Der Datenfluss (dataflow) zwischen den Funktionen eines Systems bzw. innerhalb einer Funktion
- Arbeitsprozesse sind ebenfalls mit Flussmodellen beschreibbar (wird später behandelt)

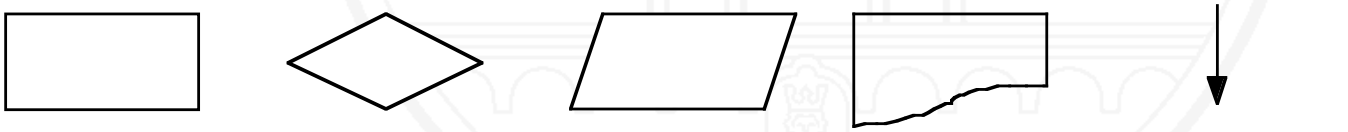
5.2 Modellierung von Ablaufstrukturen

- Programmablaufpläne (flow charts)
- Strukturierte Modellierung von Programmabläufen
  - Jackson-Diagramme
  - Nassi-Shneiderman-Diagramme
  - Aktigramme, Pseudocode
- Aktivitätsmodelle

Programmablaufpläne (flow charts)

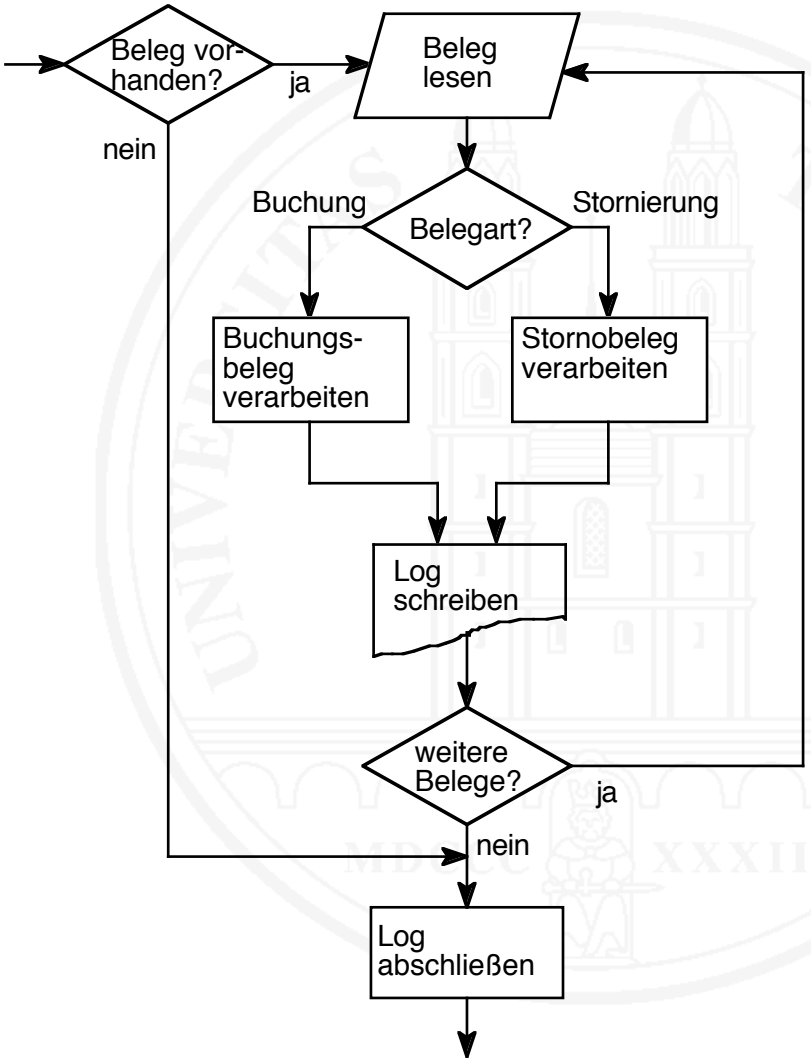
- Ältestes Mittel zur graphischen Visualisierung von Programmabläufen (Goldstine und von Neumann 1947)
- Beliebige Strukturen können modelliert werden
- Gefahr der Modellierung von „Spaghetti“-Strukturen
- Ist veraltet und sollte nicht mehr verwendet werden

- Symbole in Programmablaufplänen:



Aktion Entscheidung Eingabe Ausgabe Steuerfluss

Beispiel eines Programmablaufplans



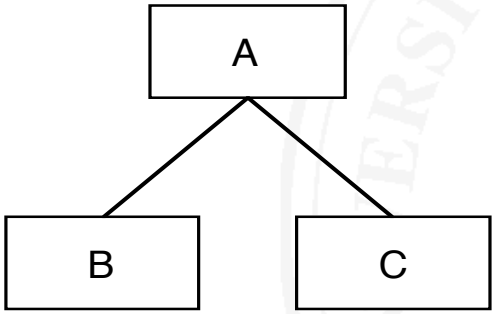
Strukturierte Modellierung von Programmabläufen

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

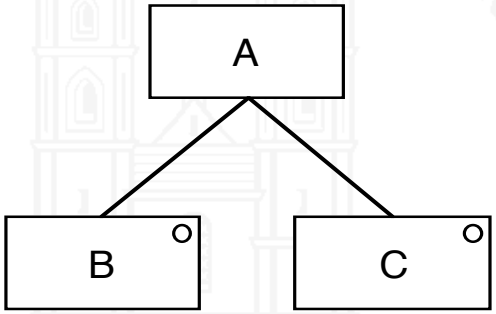
  zusammengesetzt werden (Böhm und Jacopini 1966).
- Dies ist die Grundlage der strukturierten Programmierung.
- Gebräuchliche Notationen zur graphischen Modellierung strukturierter Abläufe:
  - Jackson-Diagramme
  - Nassi-Shneiderman-Diagramme
  - Aktigramme (oder Aktionsdiagramme, action diagrams)

Jackson-Diagramme

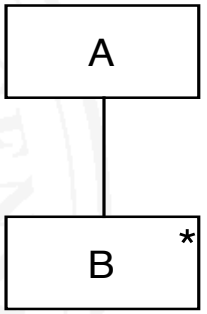
- Jackson-Diagramme (Jackson 1975) bestehen aus drei Grundelementen. Jedes Rechteck steht für eine Aktion. Die obenstehende Aktion wird durch die untenstehenden Komponenten-Aktionen definiert.



Sequenz:
A ist B gefolgt von C



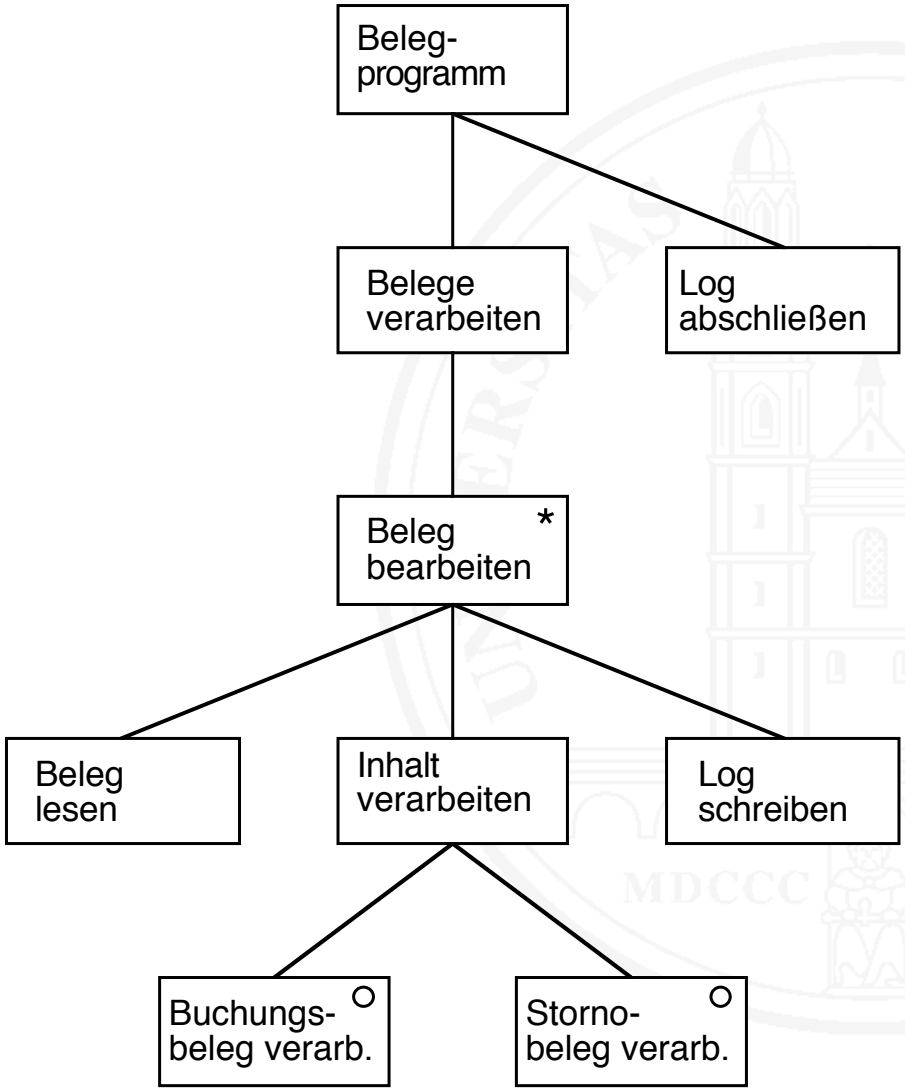
Alternative:
A ist entweder B oder C



Iteration:
A ist mehrfache Wiederholung von B

- Diagramme entstehen durch Verschachtelung: Jede Komponenten-Aktion kann durch ein Grundelement ersetzt werden. Dieser Vorgang ist beliebig oft wiederholbar.

Beispiel eines Jackson-Diagramms



Hinweis:

In der Programmentwicklungs-methode von Jackson (Jackson Structured Programming, JSP) werden zunächst die Datenstruk-turen, die ein Programm verarbei-ten soll, mit den gleichen Mitteln modelliert. Daraus wird eine Pro-grammstruktur 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

Aufgabe 5.1

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.

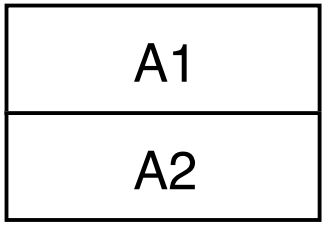
Modellieren Sie die Funktionalität eines einzelnen Buches als Jackson-Diagramm.

Nassi-Shneiderman-Diagramme

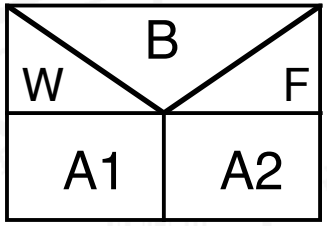
- Nassi-Shneiderman-Diagramme, auch Struktogramme genannt, (Nassi und Shneiderman, 1973) sind konzeptionell eng mit den Jackson-Diagrammen verwandt.
- Sie verwenden jedoch eine andere Notation für die Grundelemente und stellen die Verschachtelung von Aktionen graphisch auch als solche dar.
- Ferner gibt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

Grundelemente von Nassi-Shneiderman-Diagrammen

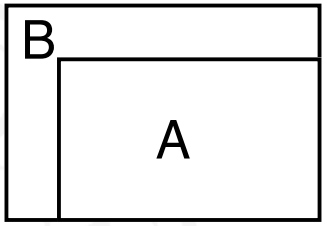
Grundsymbole:



Sequenz:
A2 folgt auf A1



Alternative:
Wenn B dann
A1 sonst A2



Iteration:
Solange B wahr
ist, führe A aus

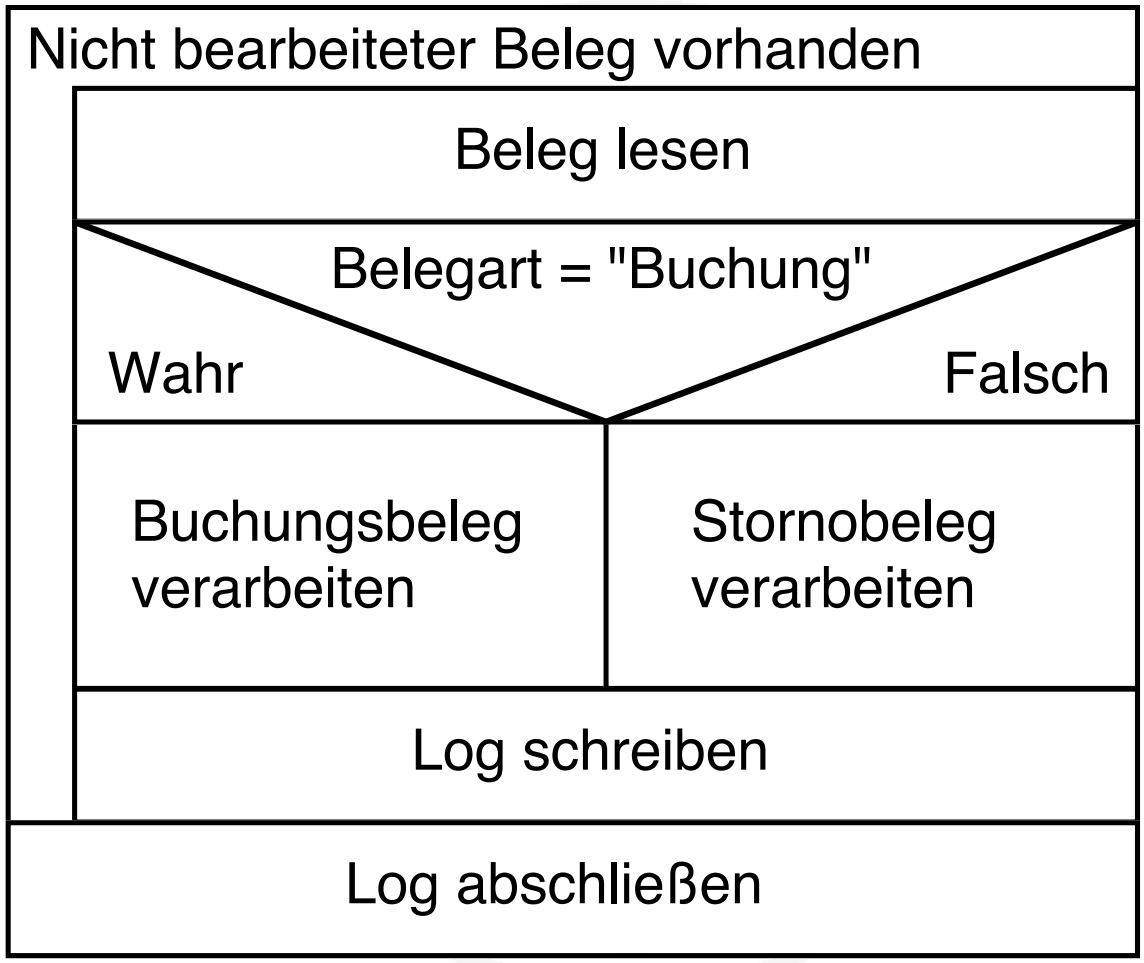
Zusatzsymbole:

Verlassen einer
Schleife

Verlassen einer
Prozedur

Diagramm einer
Prozedur

Beispiel eines Nassi-Shneiderman-Diagramms



Aufgabe 5.2

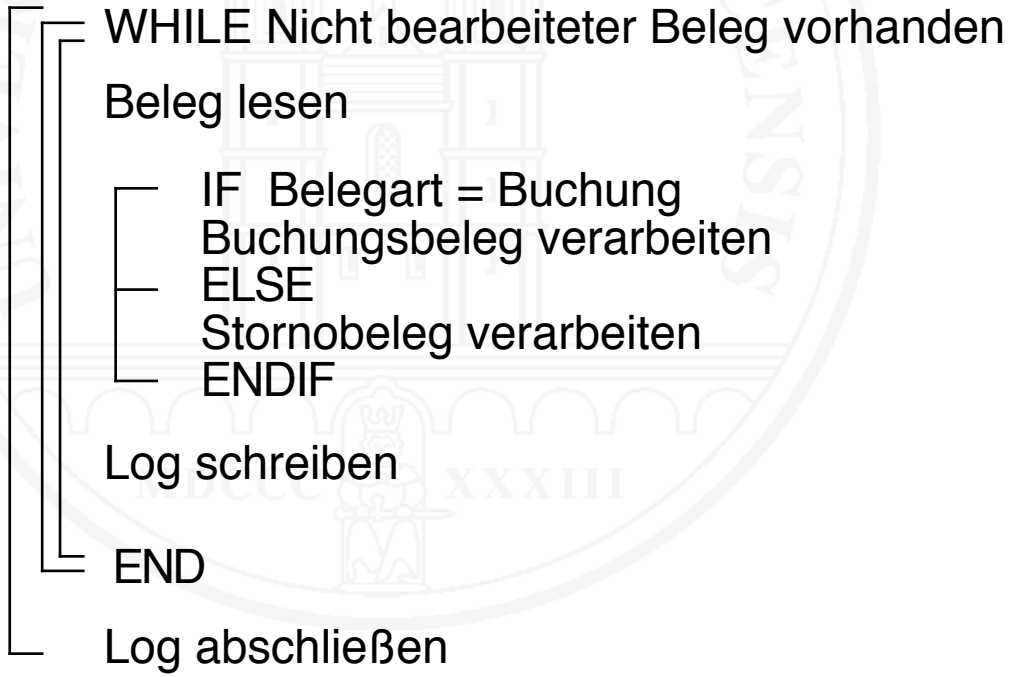
a) Beschreiben Sie die modelltheoretischen Konzepte, welche den Nassi-Shneiderman-Diagrammen zugrunde liegen.

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

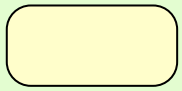
Klassischer Ansatz: **Schrittweise Verfeinerung** (stepwise refinement)
(Wirth 1971, Dijkstra 1976, Wirth 1983)

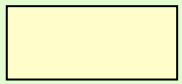
1. Wähle eine Aktion, welche den gesamten Ablauf repräsentiert
2. Zerlege diese Aktion entsprechend der Problemstellung in eine Sequenz, eine Alternative oder ein Iteration
3. Wiederhole Schritt 2 für jede beim Zerlegen neu entstandene Aktion, bis das Modell hinreichend detailliert ist

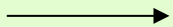
- Zerlegungsfehler zu Beginn sind später kaum mehr reparierbar
- Funktioniert nur für kleine Probleme ➪ sonst besser:
  - Teilprobleme identifizieren
  - Alle Teilprobleme durch schrittweise Verfeinerung modellieren
  - Teilmodelle durch Verschachtelung zusammensetzen

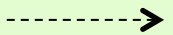
Aktivitätsmodelle

- UML beschreibt Aktivitäten und deren Ablauf in so genannten **Aktivitätsdiagrammen** (activity diagrams)
- Notation (UML 2):


Aktivität


Datenobjekt


Steuerfluss

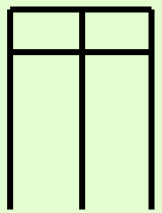

Beeinflussung

Start


Ende


Verzweigungs-konnektor

Parallelitäts-konnektor


Verantwort-lichkeits-bereiche

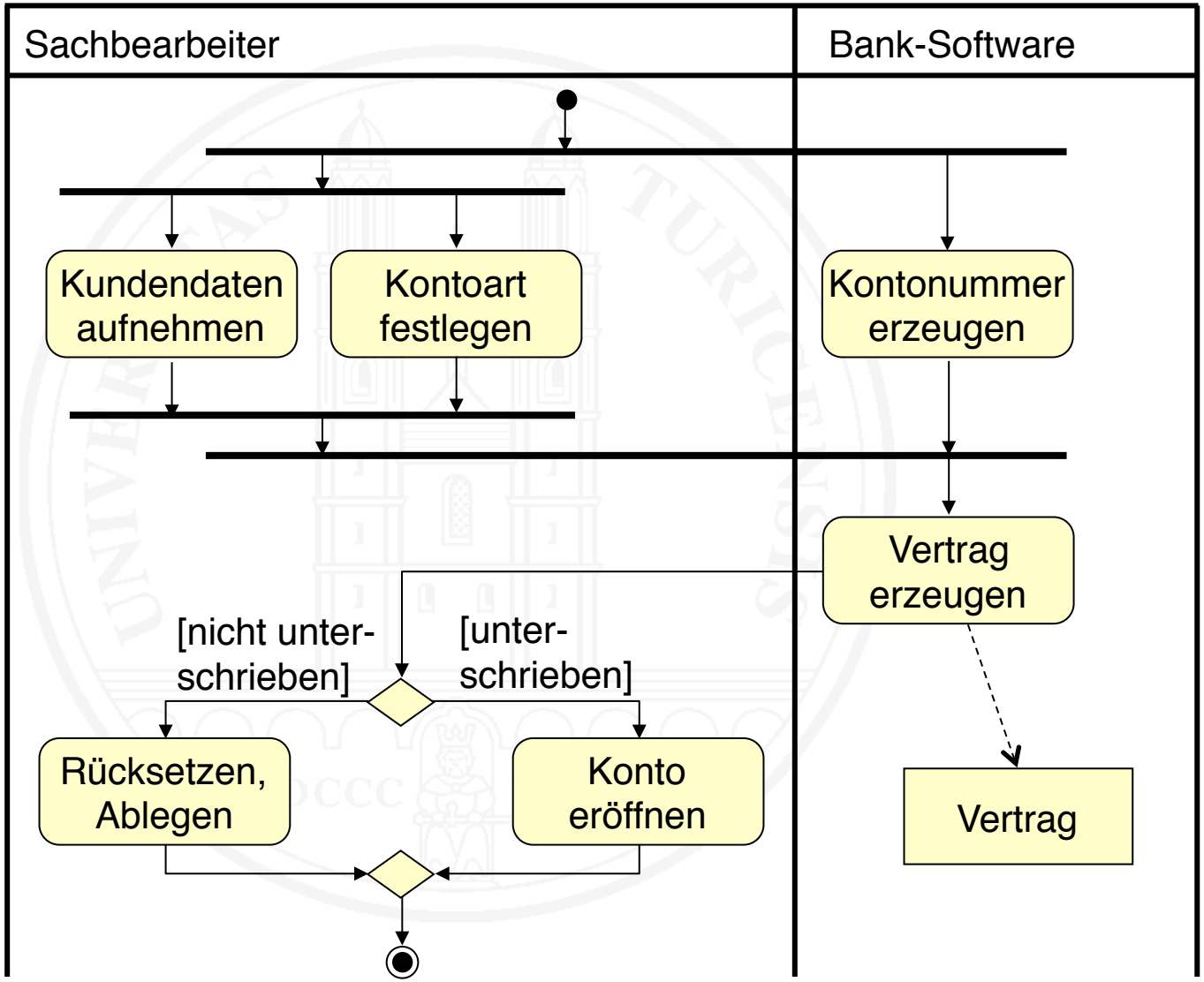
Hinweis: UML Aktivitätsdiagramme umfassen eine Reihe weiterer Sprachkonstrukte, welche grafische Programmierung ermöglichen. Eine genaue Beschreibung würde den Rahmen dieser Vorlesung sprengen.

Eignung

- UML-Aktivitätsdiagramme eignen sich zur Modellierung von Abläufen aller Art:
  - Programmabläufe
  - Prozessabläufe
    - in Arbeitsprozessen
    - in technischen Prozessen
- Saubere Struktur (mit Sequenz, Alternative, Iteration) wird nicht erzwungen: „Spaghetti“-Modelle möglich
- Parallelität ist modellierbar
- Verantwortlichkeitsbereiche sind abgrenzbar

Beispiel: Modell eines Prozessablaufs

Prozess zur Eröffnung eines Kontos:



Aufgabe 5.3

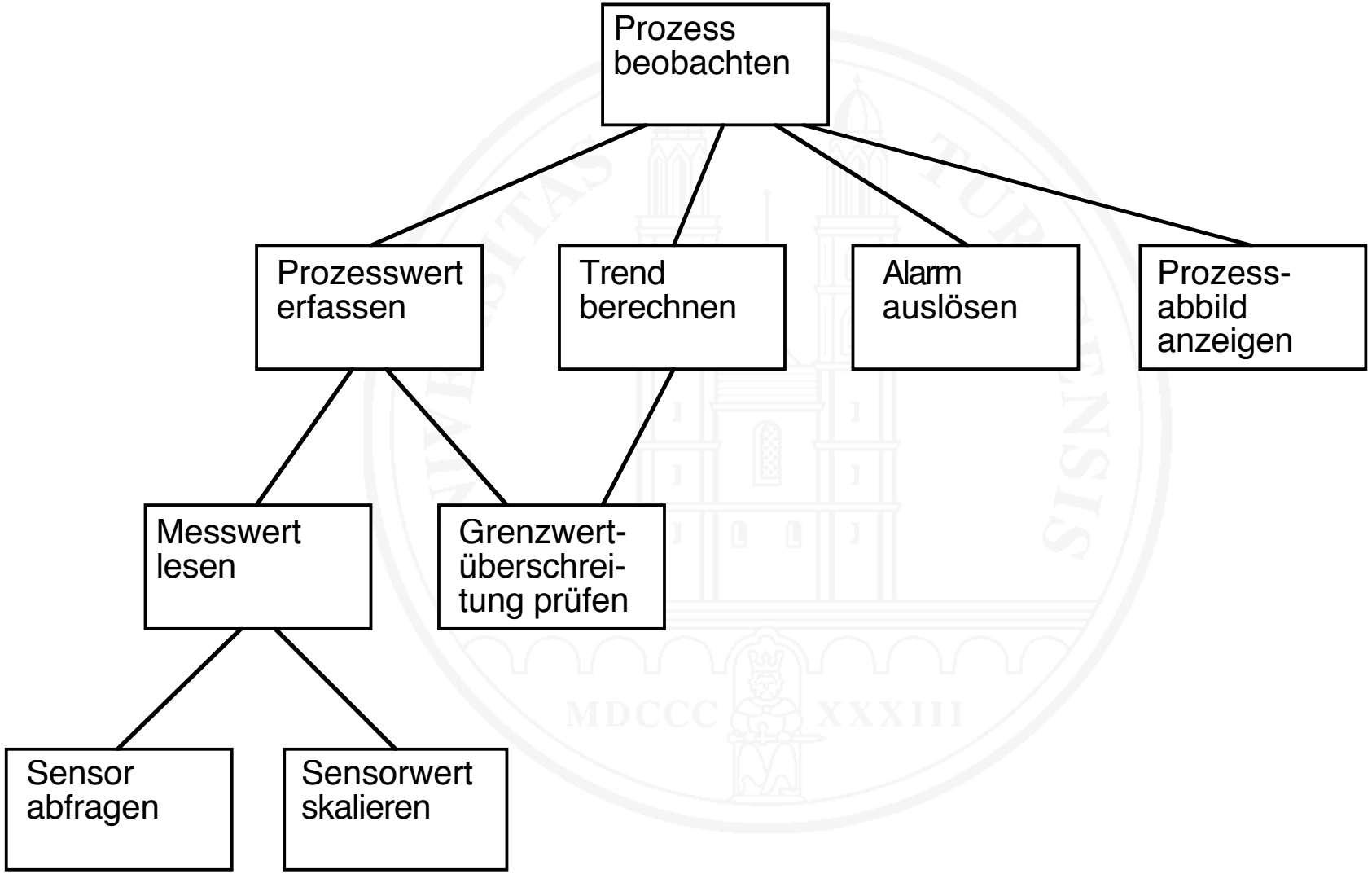
Dieter Dollmaier geht jeden Morgen als erstes zum Kaffeeautomaten. Wenn dieser noch ausgeschaltet ist, schaltet er ihn ein. Während der Automat aufwärmt, spült er seine Tasse und schäkert mit seiner Kollegin Claudia Kussmaul. Dann lässt er seinen Kaffee heraus, nimmt sich Milch und Zucker, verabredet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

Modellieren Sie diesen Ablauf mit einem UML-Aktivitätsdiagramm.

5.3 Modellierung von Aufrufstrukturen

- **Prozeduraufrufgraphen** (structure charts, call graphs) modellieren die statische Aufrufhierarchie der Prozeduren (Unterprogramme) eines Programms.
- Das zugrunde liegende Konzept ist, nur die Prozedurnamen und die Aufrufbeziehungen zwischen den Prozeduren zu modellieren.
  - Von den Prozedurrümpfen wird abstrahiert.
  - Je nach Notation werden zusätzlich Prozedurparameter modelliert.
- Prozeduraufrufgraphen sind ein Mittel, um die statische 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.
- Dynamische Bindung in objektorientierter Software ist nicht modellierbar.

Beispiel eines Prozeduraufrufgraphen



5.4 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

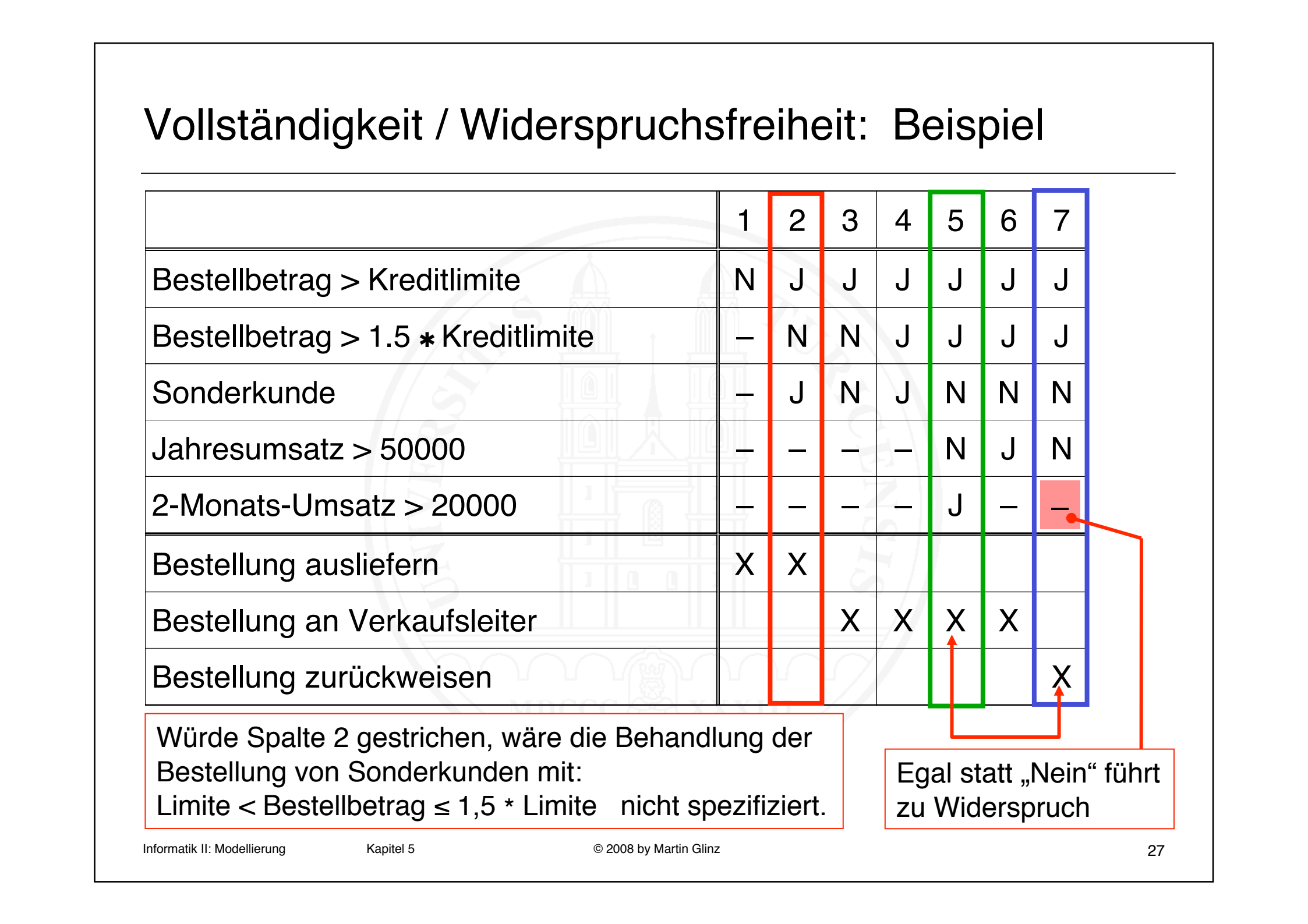
	1	2	3	4	5	6	7
Bestellbetrag > Kreditlimite	N	J	J	J	J	J	J
Bestellbetrag > 1.5 ∗ Kreditlimite	–	N	N	J	J	J	J
Sonderkunde	–	J	N	J	N	N	N
Jahresumsatz > 50000	–	–	–	–	N	J	N
2-Monats-Umsatz > 20000	–	–	–	–	J	–	N
Bestellung ausliefern	X	X					
Bestellung an Verkaufsleiter			X	X	X	X	
Bestellung zurückweisen	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

Vollständigkeit / Widerspruchsfreiheit: Beispiel

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



Würde Spalte 2 gestrichen, wäre die Behandlung der Bestellung von Sonderkunden mit:
Limite < Bestellbetrag ≤ 1,5 * Limite nicht spezifiziert.

Egal statt „Nein“ führt zu Widerspruch

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 5.4

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 Prüfung denkt, geht sie nicht zur Vorlesung.
- Wenn sie keine Lust auf Lernen hat, aber an die Prüfung 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.

Literatur

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.

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

Goldstine, H.H., J. von Neumann (1947). *Planning and Coding of Problems for an Electronic Computing Instrument*. Institute for Advanced Study, Princeton N. J.

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

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

Oestereich, B. (2004). *Objektorientierte Softwareentwicklung: Analyse und Design mit der UML 2.0*. 6. Auflage. München: Oldenbourg Verlag.

OMG (2007). *Unified Modeling Language: Superstructure,* version 2.1.1. OMG document formal/2007-02-05

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

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.

