FACHHOCHSCHULE WEDEL
Prof. Dr. Uwe Schmidt

8. August 1994

Aufgaben zur Klausur **Software Entwicklungs–Methoden** im SS 94 (WI 56)

Zeit: 90 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

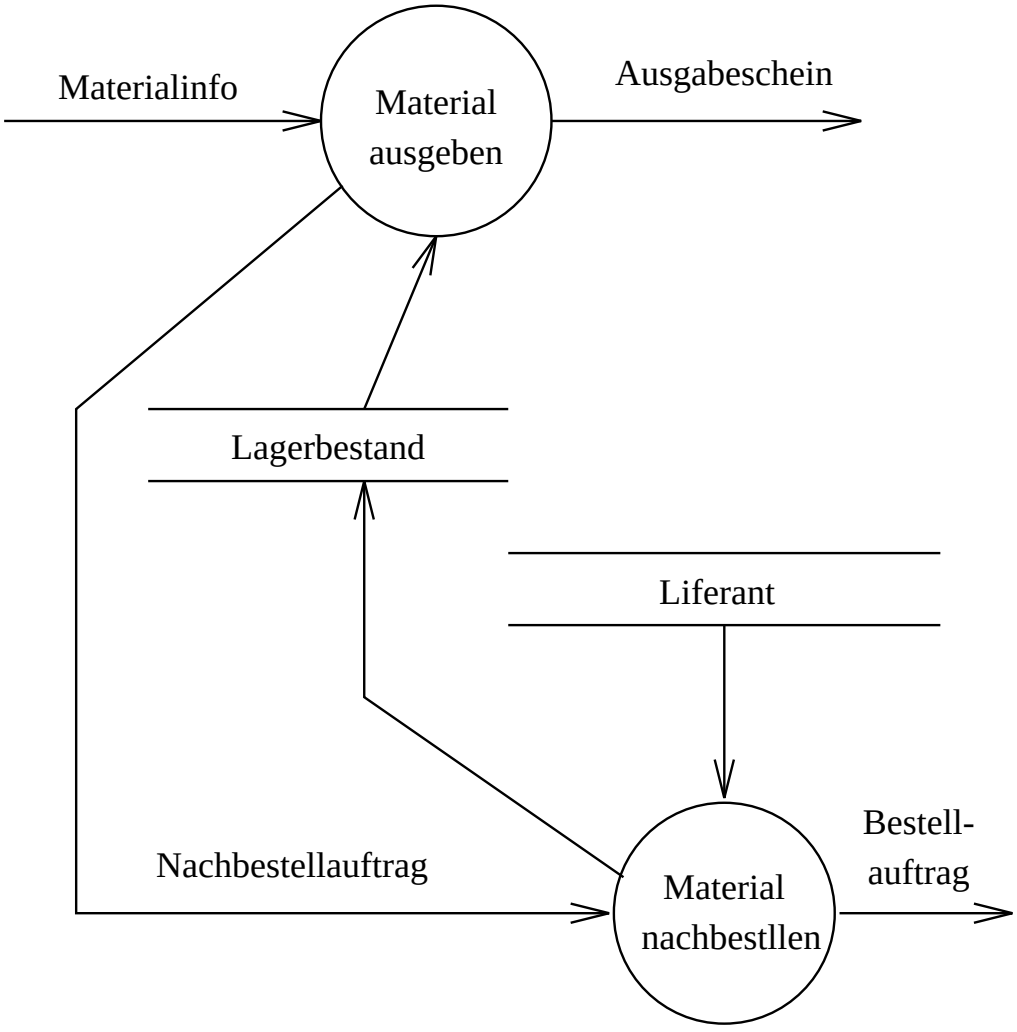
Diese Klausur besteht einschließlich dieses Deckblattes aus 8 Seiten

---

Aufgabe 1:

Ein CASE–tool für die Verarbeitung von SA–Diagrammen.

Im Beispiel sehen sie einen Ausschnitt aus einer SA-Spezifikation.



Ziel der Aufgabe ist es, den Kern eines (sehr vereinfachten) CASE–tools zu entwerfen. Hierfür ist als erstes ein Datenmodell zu erstellen, in dem die Struktur der Objekte aus den Diagrammen festgelegt werden. Ein erster Ansatz für dieses Datenmodell könnte ein gerichteter markierter Graph sein.

Warum ist diese Struktur nicht geeignet:

1) .....

2) .....

3) .....

Aus den eben entwickelten Gründen müssen die Knoten — die Kreise (*bubbles*) und die Speicher (*storages*) — einerseits, und die Pfeile (*arrows*) in getrennten Bereichen modelliert werden.

Die Geometrie der Objekte und ihre Anordnung soll in diesem Modell nicht berücksichtigt werden.

Unter der Annahme, daß jeder Knoten und jede Kante eine eindeutige interne Identifizierung (z.B. eine interne Nummmer) bekommt, kann von dem folgenden noch lückenhaften Modell ausgegangen werden. Bei der Ausarbeitung verwenden sie bitte die unten angegebenen Namen für die einfachen Wertebereiche.

.0 $SaSpezifikation = Knoten \times Kanten$

.1 $Knoten = KnotenId \xrightarrow[\text{m}]{} KnotenBescheibung$

.2 $Kanten = KantenId \xrightarrow[\text{m}]{} KantenBeschreibung$

.3 $KnotenId =$ token

.4 $KantenId =$ token

.5 $KnotenBeschreibung = \ldots$

.6 $BubbleBeschreibung = \ldots$

.7 $StorageBeschreibung = \ldots$

.8 $KantenBeschreibung = \ldots$

.9 $KantenMenge = \ldots$

.10 $DatentypName = Text$

.11 $Speichername = Text$

.12 $BubbleFunktion = Text$

Eine *bubble* besitzt einen Namen für die Funktion die durch die *bubble* beschrieben wird, außerdem sollen alle ein– und ausgehenden Pfeile direkt zugreifbar sein. Ein *storage* hat den gleichen Aufbau, nur besitzt er einen Speichernnamen anstatt einer Funktionsbeschreibung.

Geben Sie ein Datenmodell an für die Bereiche $KnotenBeschreibung$, $BubbleBeschreibung$ und $StorageBeschreibung$

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

Eine Kante wird beschrieben durch Anfangs– und Endpunkt (Vorsicht). Außerdem können Pfeile — die nicht mit Speichern verbunden sind — mit einem $DatentypNamen$ markiert werden. Die mit Speicher verbundenen Kanten brauchen nicht mit einem $DatentypNamen$ markiert werden.

Geben Sie ein Datenmodell für den Bereich $KantenBeschreibung$ an.

1) .....

2) .....

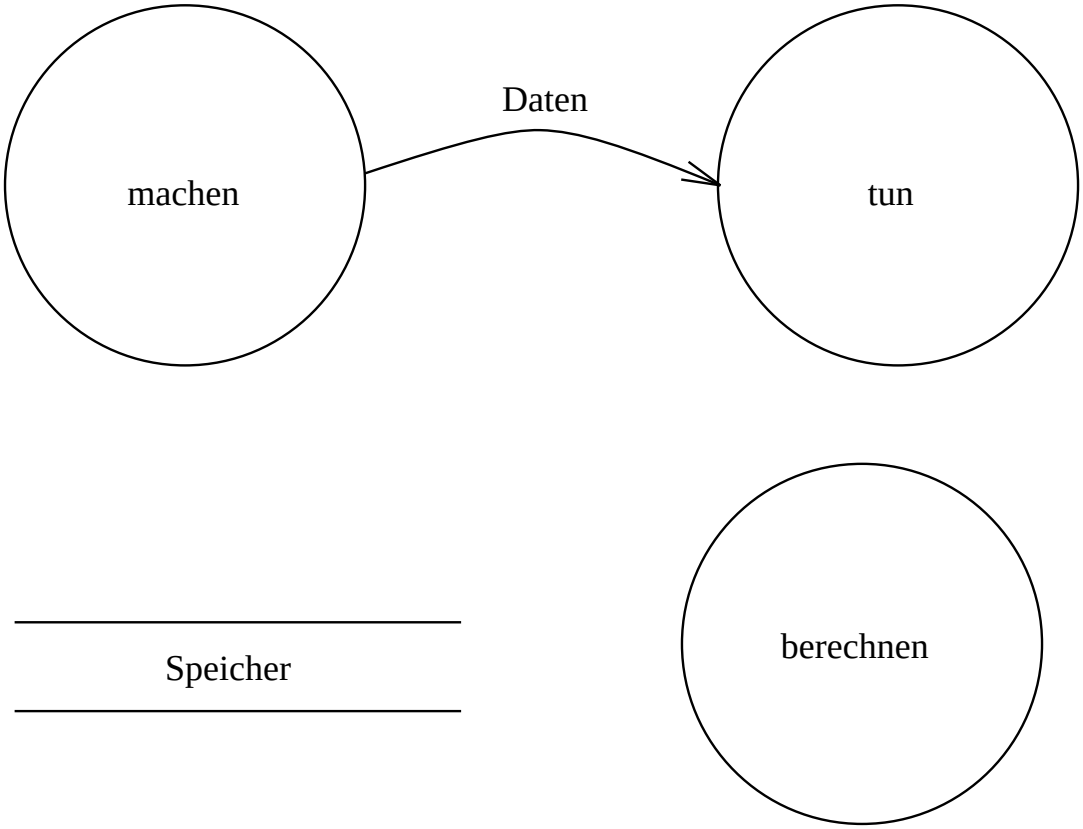
3) .....

4) .....

5) .....

In diesem Datenmodell sind noch Diagramme zugelassen, die keine syntaktisch richtige SA–Spezifikation bilden.

Schließen Sie durch eine geeignete Invariante, die folgenden Situationen aus.



inv-$SaSpezifikation((knoten, kanten)) \triangleq$

. . . . . .

. . . . . .

. . . . . .

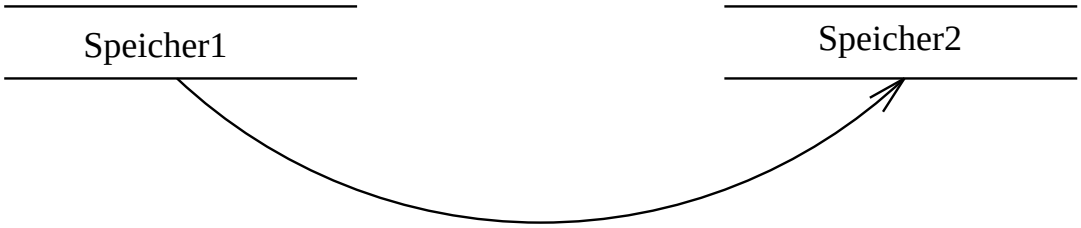
. . . . . .

. . . . . .

. . . . . .

. . . . . .

Schließen Sie durch eine zweite Invariante die folgende Situation aus.



$SaSpezifikation_1 \;=\; SaSpezifikation$

inv-$SaSpezifikation_1((knoten, kanten)) \triangleq$

. . . . . .

. . . . . .

. . . . . .

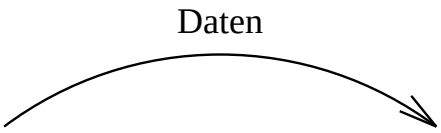
. . . . . .

. . . . . .

. . . . . .

. . . . . .

Schließen Sie durch eine dritte Invariante die folgende Situation aus.



$SaSpezifikation_2 \;=\; SaSpezifikation_1$
inv-$SaSpezifikation_2((knoten, kanten)) \triangleq$

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Zu einer SA–Spezifikation gibt es einen sogenannten Datenkatalog (*datadictionary*). Dieser Katalog ist ähnlich wie eine abstrakte Syntax aufgebaut. Hier soll der Aufbau eines *datadictionary* mit Hilfe eines Datenmodells beschrieben werden. Ein *datadictionary* besteht aus einer Menge von Gleichungen, die linke Seite legt den Namen des Datentyps fest, die rechte Seite die Struktur.

Folgendes Beispiel zeigt einen Ausschnitt aus einem *datadictionary*.

$$
\begin{array}{lll}
.0 & Flug & = @Flugnr + @Flugdatum + Startzeit + Zielzeit \\
.1 & Flugauskunft & = [\ \text{“}nichtbekannt\text{“} \mid \text{“}ausgebucht\text{“} \mid Preis\ ] \\
.2 & Flugliste & = 2\{Flugeintrag\}10 \\
.3 & Startzeit & = Zeit
\end{array}
$$

Wir betrachten hier vier Arten von Einträgen.

Die erste Art beschreibt die Zeile einer Tabelle, d.h. einen Record, bei dem einige Felder als Schlüsselkandidaten gekennzeichnet sind. Vereinfachend soll angenommen werden, daß als Felder nur wieder *DatentypNamen* auftreten.

Die zweite Art bescheibt Alternativen, dabei tauchen auf der rechten Seite entweder Konstanten (“*nichtbekannt*“) oder *DatentypNamen* auf.

Der dritte Eintrag beschreibt Listen. Eine Listenbeschreibung setzt sich aus einem Element–*DatentypNamen* und der minimalen und maximalen Länge der Liste zusammen. Diese Grenzen können sowohl einzeln als auch gemeinsam fehlen, so daß dann keine Einschränkung nach unten oder oben an die Listenlänge gemacht wird.

Die vierte Art erlaubt es Aliasnamen für Wertebereiche festzulegen.

Entwerfen Sie eine abstrakte Syntax zur Beschreibug eines Datenkatalogs. Verwenden Sie dabei die folgenden Bereichsnamen: $DatenKatalog$, $DatentypName$, $TypBeschreibung$, $RecordBeschreibung$, $AlternativenBeschreibung$, $ListenBeschreibung$, $AliasBeschreibung$ und $KonstantenName$.

1) .....

2) .....

3) .....

4) .....

5) .....

6) .....

7) .....

8) .....

9) .....