Technische Universität Berlin
Institut für Kommunikations- und Softwaretechnik
FG Softwaretechnik
Franklinstr. 28/29
10587 Berlin

Jähnichen
Koch
Bittner

Musterklausur

Softwaretechnik

26. Juli 2000

Zum Bestehen der Klausur sind **35 Punkte** notwendig. Insgesamt sind **100 Punkte** möglich. Die Bearbeitungszeit beträgt **120 Minuten**. In der Klausur sind **sämtliche Hilfsmittel** zugelassen.

Viel Erfolg!

Aufgabe 1 (Fusion)

(Geschätzte Bearbeitungszeit: 40 min.)

Start- und Landeüberwachung eines Flughafens

Ein Flughafen arbeitet mit integrierter Steuerungssoftware zur Abwicklung von Start- und Lande-Vorgängen. Ein Teil dieser Software soll für unseren Flughafen modelliert werden. Unser Flughafen hat die folgenden Komponenten:

- einen *Tower*
- *Bahnen: Landebahnen und Startbahnen*
- *Terminals*
- *Hangars*

Für alle Startvorgänge und für alle Landevorgänge wird pro Bahn jeweils eine *Queue* (Warteschlange) benötigt, deren Elemente die sog. *Slots* sind. Ein Slot enthält neben der Start- bzw. Landezeit die Daten des Flugzeuges, die zugewiesene Bahn, ggf. die Position des Flugzeuges und ggf. den Stellplatz in einem der Hangars.

Die Steuerungssoftware besteht wenigstens aus den autark arbeitenden Komponenten Tower, Landebahnen und Startbahnen. Diese Komponenten kommunizieren untereinander mit Hilfe der Slots.

Die Aufgabe des Towers ist die Verwaltung der Kapazitäten des Flughafens, wie z.B. der Positionen, der Bahnen und der Hangars. Bei der Anmeldung einer Landung durch ein Flugzeug wird geprüft, welche Kapazitäten frei sind. Der Tower reserviert freie Kapazitäten und trägt sie in einen Slot ein, der dem Flugzeug und anderen Steuerungskomponenten mitgeteilt wird. Wünsche des landenden Flugzeuges können berücksichtigt werden.
Unser Tower vergibt bei Landewunsch eines Flugzeuges freie Kapazitäten nach folgenden einfachen Kriterien:

- er wählt eine Landebahn mit der jeweils kürzesten Queue, wobei angenommen wird, dass alle Slots nach aufsteigenden Landezeiten geordnet sind; die vergebene Landezeit ist um einen geeigneten Differenzbetrag höher als die zuletzt in dieser Queue vergebene Zeit,

- er wählt ein Terminal mit der grössten freien Kapazität und vergibt unabhängig von der geplanten Verweildauer des Flugzeuges eine der freien Positionen.

Eine *Position* ist definiert durch Terminal und Gate (s.u.).

Bei Start werden ähnlich wie bei der Landung freie Kapazitäten vergeben. Und ähnlich wie bei der Landung dient der Slot zur Kommunikation zwischen den Beteiligten.

Bei allen Starts und Landungen wird die Verfügbarkeit von Kapazitäten aktualisiert.

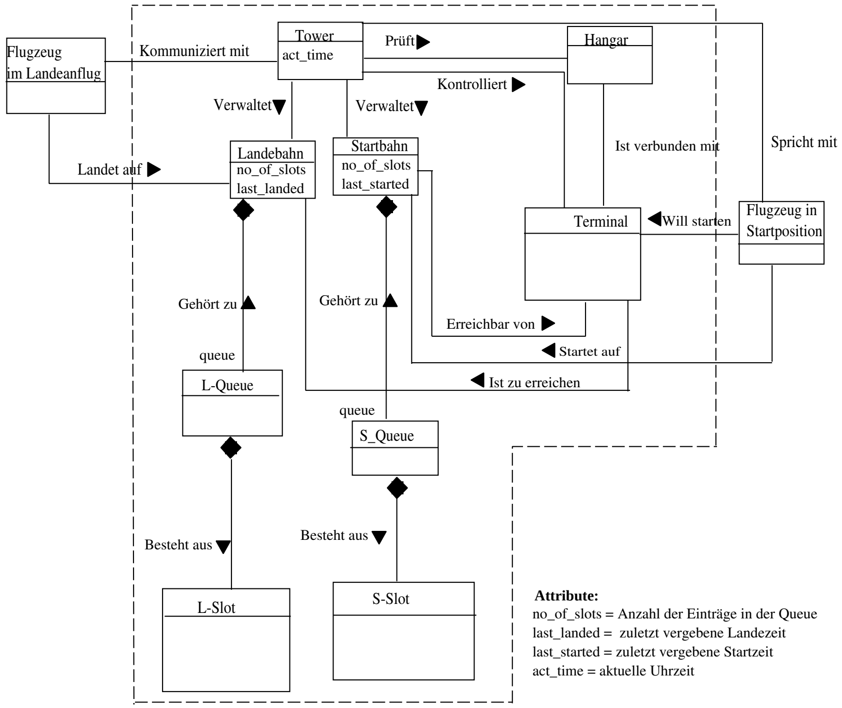
Start- und Landebahnen steuern die vom Tower bestätigten Lande- und Start-Vorgänge mit Hilfe von Slots, die in die Queue der entsprechenden Bahn eingetragen worden sind. Slots der Queue werden sequentiell abgearbeitet und bei erfolgreichem Start bzw. erfolgreicher Landung aus der Queue entfernt.

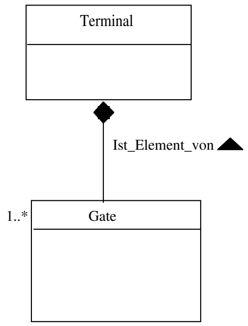
Der Flughafen besteht i. allg. aus mehreren *Terminals*. Ein Terminal verfügt über *Gates*. An einem Gate können sich Flugzeuge befinden,

- um dort nach der Landung anzudocken und zu entladen
- um von dort zu starten oder
- um dort zu parken.

Gates, an denen sich keine Flugzeuge befinden, sollen als frei gelten.

Hangars verwalten nur Stellplätze für längerfristigen Aufenthalt, auf denen Flugzeuge gewartet werden können. Flugzeuge, die in Hangars fahren, kommen aus Positionen und werden aus dem Hangar auch wieder in eine Position gebracht.





Anhand des Textes und des vorgegebenen Klassenmodells sind die nachfolgenden Teilaufgaben zu lösen.

Aufgabenstellung:

a.) Geben Sie alle für Landevorgänge relevanten Attribute der Klasse LSlot an. *3 Punkte*

Lösung:
Landebahn, Landezeit, Terminalposition (Terminal,Gate), Flugzeugdaten.

b.) Geben Sie alle für den Tower wichtigen Attribute der Klasse Terminal an und wichtige Attribute der Klasse Gate. *5 Punkte*

Lösung:
Landebahn, L Terminal:
Terminal_id, Anzahl der Gates für gelandete Flieger, Anzahl der Gates für startende Flieger, Anzahl der Gates für parkende Flieger, maximale Anzahl der Gates, Anzahl freie Gates

Gates:
Gate_id, Belegungsdauer, Flugzeugid, Startflughafen (für landende Flugzeuge), Zielflughafen (für startende Flugzeuge).

c.) Tragen Sie die Systemgrenze in das vorgegebene Klassenmodell ein. *2 Punkte*

d.) Die Systemoperation *will_landen_ohne_parken* veranlasst den Tower, für ein Flugzeug im Landeanflug eine geeignete Landebahn und eine Position zum Andocken zu reservieren, falls dies möglich ist. Diese Reservierung wird in ein Slot eingetragen und das Flugzeug über die Reservierung informiert. Das Flugzeug wird ebenfalls informiert, wenn die Landeerlaubnis verweigert werden muss.
Geben Sie für diese Systemoperation das vollständige Operationsschema an. *20 Punkte*

Lösung:

Operation	=	will_landen_ohne_parken
Description	=	Flugzeug will landen. Der Tower erteilt Landeerlaubnis mit Landbahn und Terminalposition, wenn in der L_Queue noch ein Slot frei ist. Freier Slot heisst, Landebahn und Terminalposition. Andernfalls wird die Landeerlaubnis verweigert.
Input	=	$flugzeug, startflughafen$
Reads	=	$\underline{tower : Tower}$
Changes	=	$\underline{ls : LSlot}$ **new**, $\underline{t : Terminal}$ **with** $(tower, t) \in Kontrolliert$, $\underline{l : Landebahn}$ **with** $(tower, l) \in Verwaltet$
Sends	=	$\underline{: Flugzeug\_im\_Landeanflug} : \{Landeerlaubnis(ls)\}, \{keine\_Landung\_möglich\}$
Pre	=	**true**

Post =

$$
\begin{array}{l}
\textbf{let } freie\_position == \\
\quad \forall\, term : Terminal \mid (tower, term) \in Kontrolliert \bullet \\
\quad \exists\, t : Terminal \mid (tower, t) \in Kontrolliert \wedge \\
\quad t.freie\_positionen \geq term.freie\_positionen \bullet \\
\qquad \exists\, g : Gate \mid (t, g) \in Ist\_Element\_von \wedge \\
\qquad\quad g.status = frei; \\
freie\_landebahn == \\
\quad \forall\, lb : Landebahn \mid (tower, lb) \in Verwaltet \bullet \\
\quad \exists\, l : Landebahn \mid l.anzahl \leq lb.anzahl \\
\quad \bullet \\
\quad (freie\_position \wedge freie\_landebahn) \Rightarrow \\
\qquad t.freie\_positionen' = t.freie\_postionen - 1 \wedge \\
\qquad g.status' = landen \wedge \\
\qquad g.flugzeug\_id' = flugzeug \wedge \\
\qquad first(ls.position') = t \wedge \\
\qquad second(ls.position') = g \wedge \\
\qquad l.no\_of\_slots' = l.no\_of\_slots + 1 \wedge \\
\qquad ls.flugzeug\_id' = flugzeug \wedge \\
\qquad ls.landebahn' = l \wedge \\
\qquad ls.landing\_time \geq tower.act\_time \Rightarrow \\
\qquad\quad ls.landing\_time' = l.last\_landing + landeanflugszeit \wedge \\
\qquad ls.landing\_time < tower.act\_time \Rightarrow \\
\qquad\quad ls.landing\_time' = tower.act\_time + landeanflugszeit \wedge \\
\qquad l.last\_landed' = ls.landing\_time' \wedge \\
\qquad Besteht\_aus' = Besteht\_aus \cup \{(l.queue, ls)\} \wedge \\
\qquad \textbf{is\_sent } \{Landeerlaubnis(ls)\} \wedge \\
\quad \neg\, freie\_position \Rightarrow \\
\qquad \textbf{is\_sent } \{keine\_Landung\_möglich\}
\end{array}
$$

Aufgabe (Object-Z)

(Geschätzte Bearbeitungszeit: 50 min.)

Die Fahrschein-Maschine

Eine Maschine, die Fahrscheine verkauft, besteht aus zwei verschiedenen Aggregaten:

- einem Automaten, der nach Annahme der Einzahlung einer entsprechenden Summe einen gewünschten Fahrschein ausgibt,
- einer Box, die eine maximale Anzahl von Fahrscheinen mehrerer Sorten aufnehmen kann und einen Fahrschein auf Anforderung abgibt, solange noch Fahrscheine vorhanden sind.

Unsere Fahrschein-Maschine wird in Object-Z durch die drei Klassen *Automat*, *Box* und *Fahrschein* modelliert.

Die Klasse *Fahrschein* enthält eine Kennung, in unserem Falle die Kennung *AB* oder *Tageskarte* und jeweils den Preis, der dieser Kennung entspricht.

Die Klasse *Box* unterscheidet zwei Sorten von Waren-Exemplaren, die in unserem Falle Fahrscheine der Kennung AB oder Tageskarte sind. Die Box wird initial mit einer ihrer Aufnahmekapazität entsprechenden charakteristischen maximalen Anzahl von Exemplaren der Sorten 1 und 2 gefüllt und hat die Aufgabe, über die Verfügbarkeit von Exemplaren und über den Wert der verfügbaren Warenmenge Buch zu führen und, wenn möglich, auf Anforderung ein Exemplar einer Sorte abzugeben.

Der *Automat* führt Buch über das eingenommene Geld und den Wert der ausgegebenen Fahrscheine. Er kann seine Aufgabe, Fahrscheine auf Anforderung zu verkaufen, nur dann erfüllen, wenn die Box signalisiert, dass sie noch Fahrscheine liefern kann. Das soll hier immer dann der Fall sein, wenn von *beiden* Fahrscheinsorten noch wenigstens ein Exemplar verfügbar ist. Im Automaten befindet sich bei jeder neuen Füllung der Box kein Geld. Unser einfacher Automat kann nicht wechseln, er kann aber Beträge, die nicht exakt dem Preis eines Fahrscheins entsprechen, wieder an den Kunden des Automaten zurückgeben.

Ein Teil der Modellierung ist vorgegeben. Ergänzen Sie nur die mit der Teilaufgaben-Nummer versehenen Teile.

Aufgabenstellung:

a) Bestimmen Sie die Invariante der Klasse *Fahrschein*. $\boxed{\text{10 Punkte}}$

b) Spezifizieren Sie das INIT-Prädikat der Klasse *Box*.

c) Spezifizieren Sie das INIT-Prädikat der Klasse *Automat*.

d) Die Operation *Auswahl_AB_OK* des Automaten stellt fest, dass der Automat nicht leer ist und der korrekte Betrag eingegeben wird, um einen Fahrschein der Sorte AB zu kaufen.
Spezifizieren Sie die Operation *Auswahl_AB_OK*. $\boxed{\text{10 Punkte}}$

e) Konstruieren Sie die (totalisierte) Operation *Verkauf_AB* in der Klasse *Automat* mit Hilfe der vorgegebenen Operationen *Automat_leer*, *Betrag_AB_nicht_passend*, *Auswahl_AB_OK* aus der Klasse *Automat* und der Operation *AB_abgeben* aus der Klasse *Box*. $\boxed{\text{25 Punkte}}$

f) Welchen Typ hat *rest_Sorte1* aus der Klasse *Box*. $\boxed{\text{5 Punkte}}$

Fahrschein

$Typ ::= AB \mid Tageskarte$

Teilaufgabe a):

$kennzeichen : Typ$
$preis : \mathbb{N}$

$preis \geq 0$
$\exists\, p, q : \mathbb{N} \bullet$
$\quad (kennzeichen = AB \Rightarrow preis = p \land$
$\quad kennzeichen = Tageskarte \Rightarrow preis = q)$

Box

$max\_sorte1, max\_sorte2 : \mathbb{N}$

$max\_sorte1 \geq 1 \land max\_sorte2 \geq 1$

$sorte\_1 : \mathbb{F}\, Fahrschein$
$sorte\_2 : \mathbb{F}\, Fahrschein$
$akt\_wert : \mathbb{N}$
$gesamtwert : \mathbb{N}$
$preis\_sorte1 : \mathbb{N}$
$preis\_sorte2 : \mathbb{N}$
$rest\_sorte1 : \mathbb{F}\, sorte\_1$
$rest\_sorte2 : \mathbb{F}\, sorte\_2$

$gesamtwert = max\_sorte1 * preis\_sorte1 + max\_sorte2 * preis\_sorte2$
$akt\_wert = \#rest\_sorte1 * preis\_sorte1 + \#rest\_sorte2 * preis\_sorte2$
$\#rest\_sorte1 \leq max\_sorte1$
$\#rest\_sorte2 \leq max\_sorte2$
$sorte\_1 = \{x : Fahrschein \mid x.kennzeichen = AB \land x.preis = preis\_sorte1\}$
$sorte\_2 = \{x : Fahrschein \mid x.kennzeichen = Tageskarte \land x.preis = preis\_sorte2\}$

Teilaufgabe b):

INIT

$\#rest\_sorte1 = max\_sorte1$
$\#rest\_sorte2 = max\_sorte2$

AB_abgeben

$\Delta(rest\_sorte1, akt\_wert)$
$fs! : Fahrschein$

$\#rest\_sorte1 > 0$
$fs!.kennzeichen = AB$
$rest\_sorte1' = rest\_sorte1 \setminus \{fs!\}$
$akt\_wert' = akt\_wert - preis\_sorte1$

Tageskarte_abgeben

$\Delta(rest\_sorte2, akt\_wert)$
$fs! : Fahrschein$

$\#rest\_sorte2 > 0$
$fs!.kennzeichen = Tageskarte$
$rest\_sorte2' = rest\_sorte2 \setminus \{fs!\}$
$akt\_wert' = akt\_wert - preis\_sorte2$

Automat

$\upharpoonright(Verkauf\_AB, Verkauf\_Tageskarte)$

$display ::= Automat\ leer \mid Betrag\ nicht\ passend \mid bitte\ Fahrschein\ entnehmen$

$meine\_box : Box$
$eingenommenes\_geld : \mathbb{N}$

$eingenommenes\_geld + meine\_box.akt\_wert = meine\_box.gesamtwert$
$eingenommenes\_geld \geq 0$

Teilaufgabe c):

INIT

$eingenommenes\_geld = 0$
$meine\_box.INIT$

Automat_leer

$d! : display$

$\#meine\_box.rest\_sorte1 = 0 \lor \#meine\_box.rest\_sorte2 = 0$
$d! = Automat\ leer$

Betrag_AB_nicht_passend

$b? : \mathbb{N}$
$retour! : \mathbb{N}$
$d! : display$

$b? \neq meine\_box.preis\_sorte1$
$retour! = b?$
$d! = Betrag\ nicht\ passend$

Betrag_Tageskarte_nicht_passend

$b? : \mathbb{N}$
$retour! : \mathbb{N}$
$d! : display$

$b? \neq meine\_box.preis\_sorte2$
$retour! = b?$
$d! = Betrag\ nicht\ passend$

Teilaufgabe d):

Auswahl_AB_OK

$\Delta(eingenommenes\_geld)$
$b? : \mathbb{N}$
$d! : display$

$\#meine\_box.rest\_sorte1 \neq 0 \land \#meine\_box.rest\_sorte2 \neq 0$
$b? = meine\_box.preis\_sorte1$
$eingenommenes\_geld' = eingenommenes\_geld + b?$
$d! = bitte\ Fahrschein\ entnehmen$

$$
\begin{array}{|l}
\hline
\textit{Auswahl\_Tageskarte\_OK} \\
\Delta(\textit{eingenommenes\_geld}) \\
b? : \mathbb{N} \\
d! : \textit{display} \\
\hline
b? = \textit{meine\_box.preis\_sorte2} \\
\textit{eingenommenes\_geld}' = \textit{eingenommenes\_geld} + b? \\
d! = \textit{bitte Fahrschein entnehmen} \\
\hline
\end{array}
$$

Teilaufgabe e):

$\textit{Verkauf\_AB} \mathrel{\widehat{=}} \textit{Automat\_leer} [] \textit{Betrag\_AB\_nicht\_passend} []$
$\quad (\textit{Auswahl\_AB\_OK} \mathbin{⨟} \textit{meine\_box.AB\_abgeben})$

$\textit{Verkauf\_Tageskarte} \mathrel{\widehat{=}} \textit{Automat\_leer} [] \textit{Betrag\_Tageskarte\_nicht\_passend} []$
$\quad (\textit{Auswahl\_Tageskarte\_OK} \mathbin{⨟} \textit{meine\_box.Tageskarte\_abgeben})$

Teilaufgabe f): **Der Typ der Menge** rest_sorte1 **ist**

$\mathbb{P}\,\mathbb{P}\,\textit{Fahrschein}$

Aufgabe (Statecharts)

(Geschätzte Bearbeitungszeit: 30 min.)

Wisch-Wasch-Anlage für Autos

Automobile verfügen häufig über eine Wisch-Wasch-Anlage, bei der die Betätigung mehrerer verschiedener Wischer-Stufen mit einem Waschvorgang der Windschutzscheibe kombiniert werden kann.

Die Wisch-Wasch-Anlage funktioniert nur bei eingeschalteter Zündung bzw. bei laufendem Motor und besteht aus zwei getrennten Aggregaten:

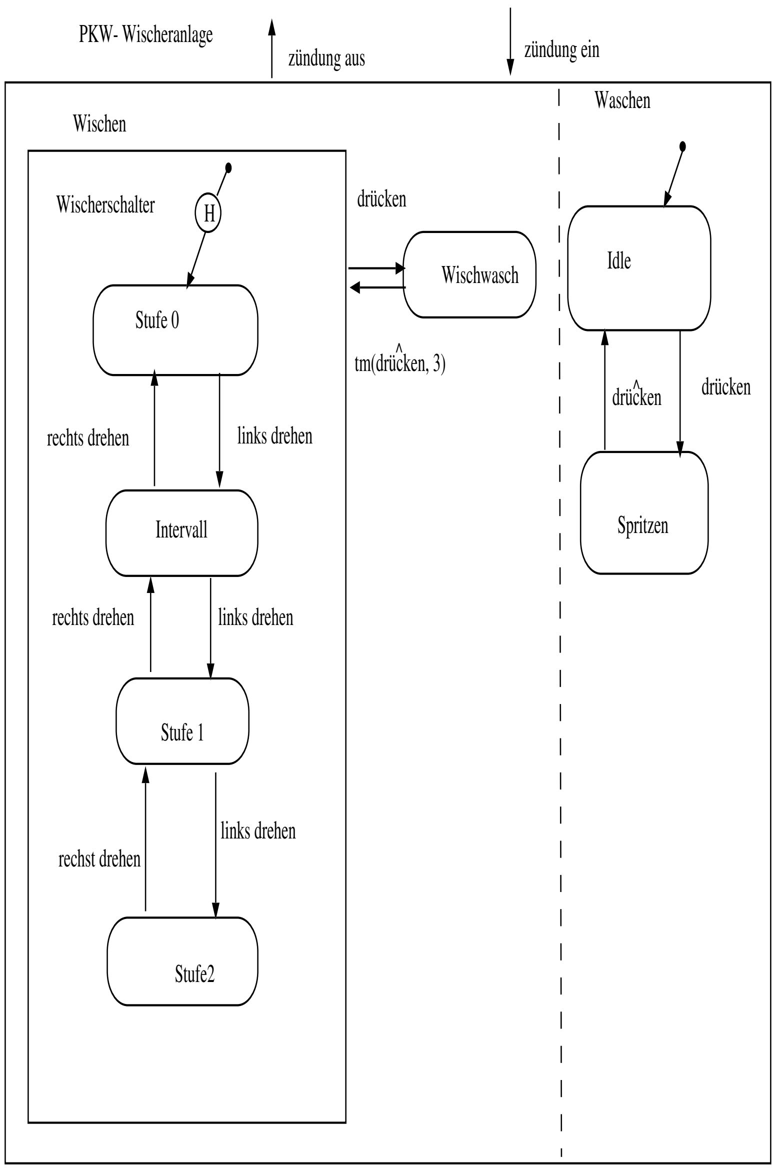
- Der Wischer hat die vier Normalstufen *Ruhelage*, *Intervall*, *Dauer-langsam* und *Dauer-schnell*. Aus der *Ruhelage* kann man diese Stufen nur sequentiell nacheinander durch *Linksdrehung* des Wischerschalters erreichen. Dasselbe gilt für die umgekehrte Richtung durch *Rechtsdrehung* des Wischerschalters bis zur Ruhelage. Jede dieser Grundstufen kann unterbrochen werden durch *Druck* auf den Wischerschalter. Dadurch gelangt man in die Stufe *Wisch-Wasch*. Die Stufe Wisch-Wasch wird 3 Sekunden nach Beendigung des Drucks auf den Wischerschalter mit der Rückkehr in die Ausgangsstufe beendet.

  Hinweis: Ein erneuter Druck auf den Wischerschalter während das System sich noch in der Stufe Wisch-Wasch befindet, hat keinen Effekt.

- Die Waschanlage funktioniert nur gleichzeitig mit dem Eintritt in die Stufe Wisch-Wasch des Wischeraggregates. Sie spritzt so lange Wasser aus Düsen auf die Windschutzscheibe, bis man den Druck auf den Wischerschalter beendet.

Aufgabenstellung:

a) Modellieren Sie die Wisch-Wasch-Anlage mit Hilfe von Statecharts. $\boxed{\textit{20 Punkte}}$


PKW- Wischeranlage
zündung aus
zündung ein
Wischen
Waschen
Wischerschalter
H
drücken
Wischwasch
tm(drûcken, 3)
Idle
drûcken
drücken
Spritzen
Stufe 0
rechts drehen
links drehen
Intervall
rechts drehen
links drehen
Stufe 1
rechst drehen
links drehen
Stufe2