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

Santen
Pfannenstiel
Sokenou

Musterlösung zur

Klausur

Softwaretechnik

21. Februar 2001

Prüfen Sie zunächst, ob dieses Exemplar vollständig ist (16 Blätter ohne Deckblatt).

Tragen Sie auf diesem Deckblatt und darüber hinaus auf allen Blättern, die Sie für Ihre Niederschrift verwenden, Ihren Namen und Ihre Matrikelnummer ein.

Zum Bestehen der Klausur sind **35 Punkte** notwendig. Insgesamt sind 95 Punkte möglich. Die Bearbeitungszeit beträgt 120 Minuten. In der Klausur sind **außer Telekommunikationseinrichtungen sämtliche Hilfsmittel** zugelassen.

Viel Erfolg!

Name, Vorname: ______________________________

Matrikelnummer: ______________

Studienrichtung/Prüfungsordnung: ______________
SWT-Übungen habe ich im Semester ________besucht.

Aufgabe	1	2	3	Gesamt
Maximal:	40	35	20	95
Erreicht:				

Aufgabe 1 (Fusion) - Das Gerichtssystem

(Geschätzte Bearbeitungszeit: 50 min.)

Es soll ein Softwaresystem für ein Gericht entwickelt werden, das hilft, bei Gericht anfallende Dokumente zu verwalten. Das Gericht wickelt sogenannte Fälle ab, wobei folgende Abläufe des Gerichts auftreten und durch das System unterstützt werden sollen.

- Ein Anwalt kann eine Anklage erheben, in dem er elektronisch eine Anklageschrift beim Gericht *einreicht*. Das Gericht informiert hierüber automatisch einen Richter.
- Der Richter *entscheidet* darüber, ob die Anklage zur Verhandlung *zugelassen* oder abgewiesen wird. Diese Entscheidung wird dem Anwalt elektronisch vom Gericht mitgeteilt.
- Der Anwalt kann eine zugelassene Anklage jederzeit *zurückziehen*, solange noch kein Urteil gefällt wurde. Der mit dem Fall betraute Richter wird vom Gericht hierüber online informiert.
- Eine zugelassene Anklage führt zu einer oder mehreren *Verhandlungen*.
- Der Richter lädt die Angeklagten und den Anwalt zur Verhandlung ein, indem er postalisch Einladungen verschicken läßt.
- Ein Gerichtsdiener führt für jede *Verhandlung* ein Protokoll, das er während der Verhandlung direkt in das System eingibt.
- Wird die Anklage nicht zurückgezogen, fällt der Richter nach dem Ende der Verhandlungen schließlich ein *Urteil*, das er dem Gericht elektronisch übermittelt.
- Jedes Urteil erfordert eine *Begründung*, die der Richter spätestens drei Wochen nach Verkündung des Urteils online an das Gericht schicken muß.

Hinweis: Die Verwaltung von Adressen und von persönlichen Daten beteiligter Personen (insbesondere Anwälte und Angeklagte), sowie der postalische Verkehr wird durch die Verwaltungsabteilung des Gerichts übernommen.

Die Entwicklung des Systems soll mit Hilfe der Fusion-Methode durchgeführt werden. Betrachtet werden sollen *nur* die genannten Funktionalitäten *Einreichen*, *Zulassungsentscheidung*, *Zurückziehen*, *Verhandeln*, *Urteilen* und *Begründen*.

1. Erstellen Sie ein UseCase-Modell, das die oben genannten Anforderungen realisiert und nennen Sie die Akteure. *5 Punkte*
2. Zeichnen Sie in das gegebene Klassendiagramm die Systemgrenze ein. *5 Punkte*

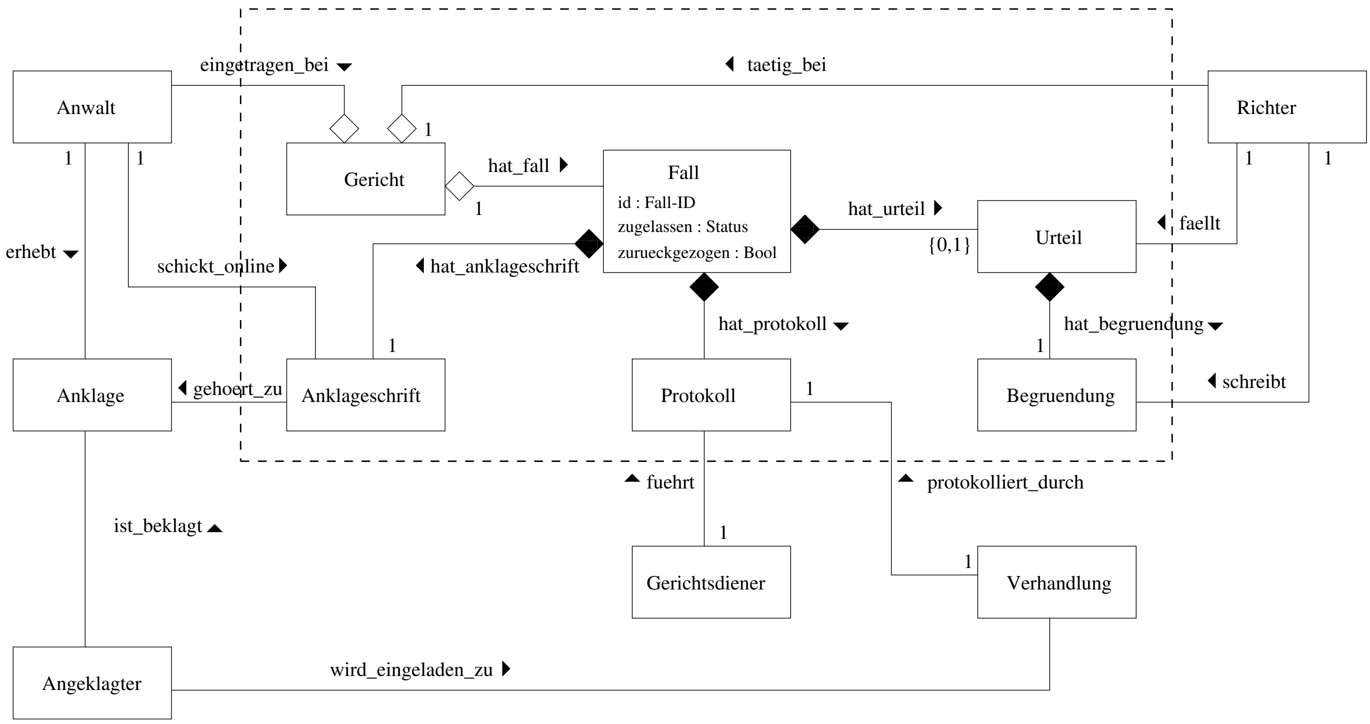
3. Geben Sie zu den gefundenen UseCases Timeline-Diagramme an, die konkrete Szenarien im Zusammenhang mit *Einreichen*, *Zulassungsentscheidung* und *Zurückziehen* aufzeigen. Betrachten Sie beim *Einreichen* und der *Zulassungsentscheidung* nur den jeweiligen Normalfall. Behandeln Sie beim *Zurückziehen* von Klagen typische Fehlerfälle. *10 Punkte*

4. Vervollständigen Sie das Operationsmodell für die Systemoperation *anklage_zurückziehen*. Geben Sie *nur* die Teile *Name, Description, Input, Reads, Changes, Sends* und *Pre* an, diese jedoch vollständig. Nehmen Sie dabei an, daß die von Ihnen behandelten Fehler *alle* möglichen Fehler seien. *8 Punkte*

5. Geben Sie für die Systemoperation *anklage_zurückziehen* den Objekt-Interaktionsgraphen an, jedoch nur für den Normalfall und **einen** der behandelten Fehlerfälle. Gehen Sie dabei also davon aus, daß alle anderen Fehler nicht auftreten können. *7 Punkte*

6. Stellen Sie das Referenzierungsmodell *nur* für den von Ihnen aufgestellten Objekt-Interaktionsgraphen von *anklage_zurückziehen* auf. *5 Punkte*

2. Systemgrenze

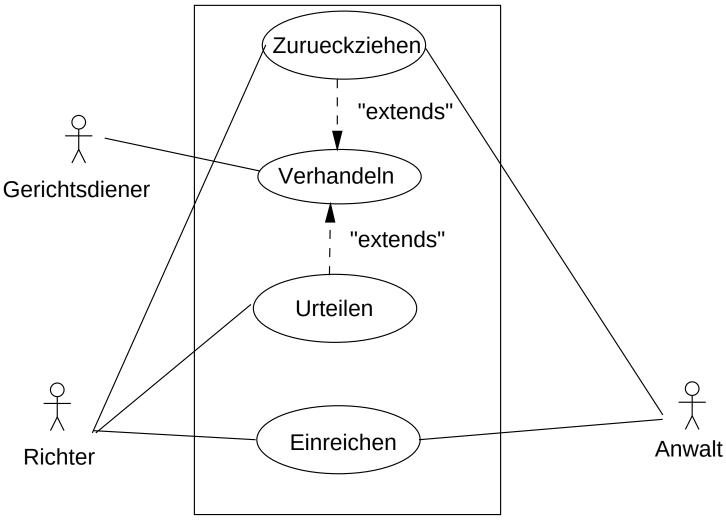


Status = eingereicht | zugelassen | abgewiesen

1. Akteure:

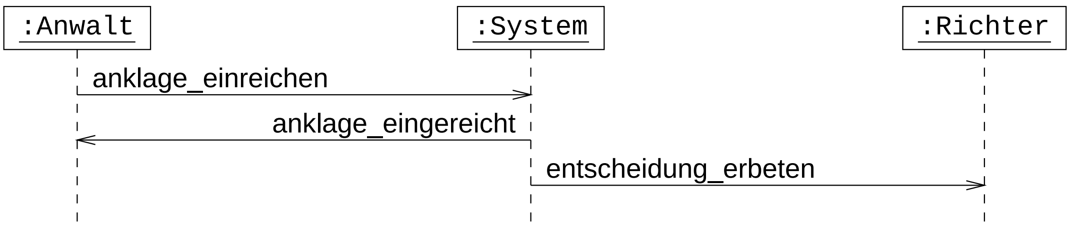
Anwalt, Richter, Gerichtsdiener

1. UseCase-Modell:

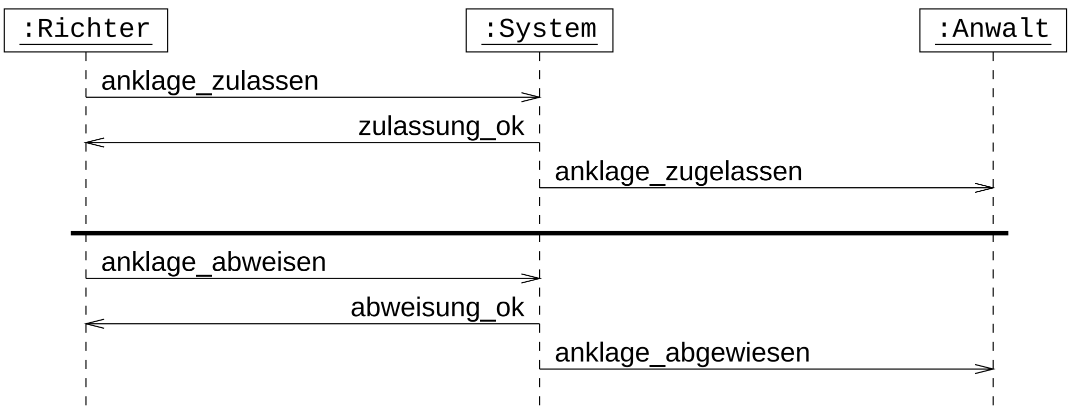


3. Timeline-Diagramme:

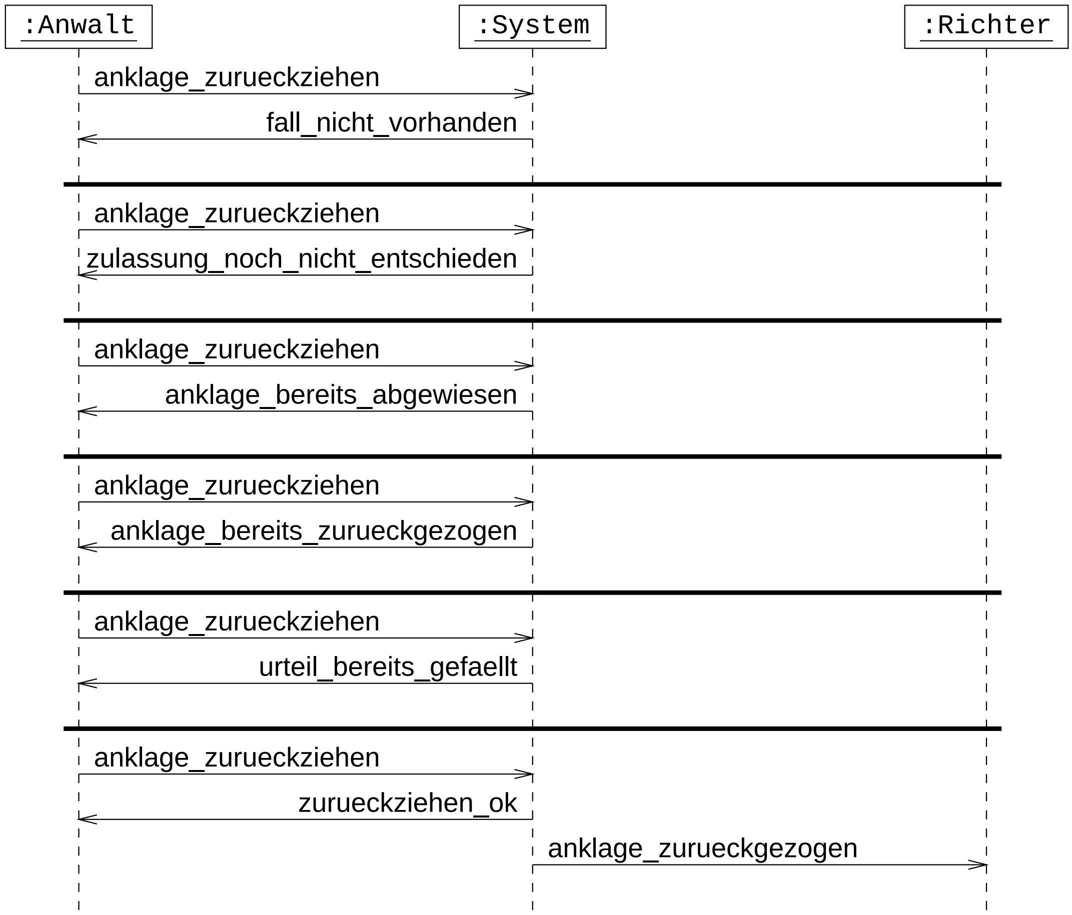
Anklage_einreichen



Zulassungsentscheidung



Anklage Zurückziehen



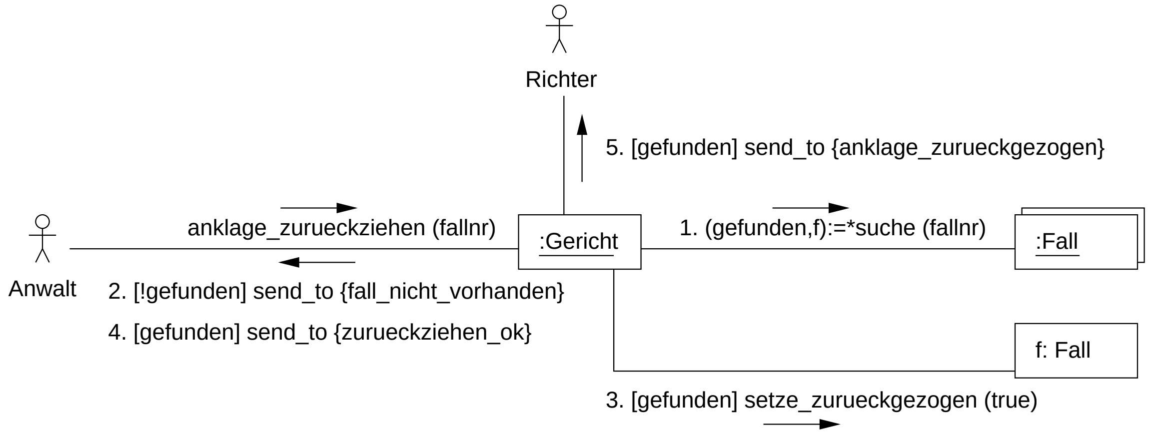
4. Operationsmodell:

Name: anklage_zurueckziehen
Description: Der Anwalt zieht eine Anklage zurück.
Input: $fallnr$: *Fall-ID*
Reads:
Changes:
Sends: $Richter\{anklage\_zurueckgezogen\}$
PRE:

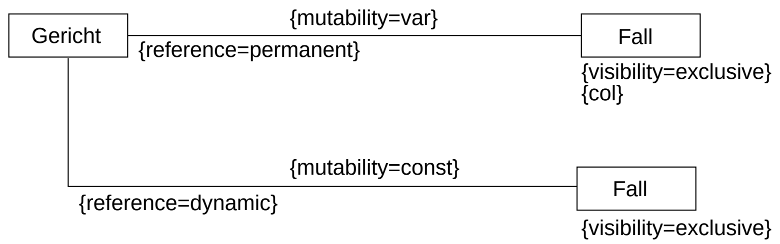
Name: anklage_zurueckziehen
Description: Der Anwalt zieht eine Anklage zurück.
Input: $fallnr$: *Fall-ID*
Reads: u : *Urteil*, hat_urteil
Changes: f : *fall*
Sends: $Richter\{anklage\_zurueckgezogen\}$
$Anwalt\{fall\_nicht\_vorhanden, zulassung\_noch\_nicht\_entschieden,$
$anklage\_bereits\_abgewiesen, anklage\_bereits\_zurueckgezogen,$
$urteil\_bereits\_gefaellt, zurueckziehen\_ok\}$

PRE: $true$

5. Objekt-Interaktionsgraph (anklage_zurueckziehen)

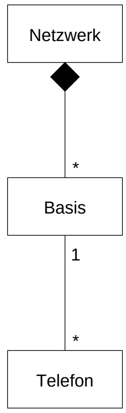


6. Referenzierungsmodell (nur anklage_zurueckziehen)



Aufgabe 2 (Object-Z)

(Geschätzte Bearbeitungszeit: 40 min.)



In Object-Z soll ein einfaches Modell des sog. *Short Message Service* (SMS) für Mobiltelefone spezifiziert werden. Dieses System besteht aus einer Anzahl von Mobiltelefonen, die bei Basisstationen eines Netzwerks eingebucht sind (siehe Klassendiagramm). Die Basisstationen empfangen Nachrichten von Telefonen, können diese gemäß der Topologie des Netzwerks an andere Basisstationen weiterleiten und schließlich an die Zieltelefone der Nachrichten senden.

Gegeben seien die Basistypen SMS von Nachrichten und $TelNr$ von Telefonnummern, über die Telefone identifiziert werden.

$[SMS, TelNr]$

Ein *Telefon* hat eine konstante Telefonnummer nr. Es speichert in seiner *inbox* die Menge der bereits empfangenen Nachrichten. In seinem Zustand hält es außerdem die Information, mit welcher Basisstation es verbunden ist.

Ein Telefon bietet zwei Operationen an:

- Es sendet eine eingegebene Nachricht und die Telefonnummer des Zieltelefons an seine Basisstation.
- Es empfängt eine Nachricht und eine Telefonnummer. Der Empfang findet nur statt, falls die empfangene Nummer mit der eigenen Nummer des Telefons identisch ist.

Eine *Basisstation* (Klasse $Basis$) verwaltet die Menge der bei ihr eingebuchten Telefone und eine Zuordnung der von ihr empfangenen aber noch nicht weitergesendeten Nachrichten zu den Nummern der Zieltelefone, an die diese Nachrichten zu senden sind. Alle bei einer Basisstation eingebuchten Telefone müssen mit genau dieser Basisstation kommunizieren.

Eine Basisstation bietet drei Operationen an:

- Sie kann eine Nachricht mit zugehöriger Zieltelefonnummer *empfangen*.
- Sie kann eine zuvor empfangene Nachricht mit der zugehörigen Zieltelefonnummer *senden*. Diese Operation dient sowohl der Kommunikation mit anderen Basisstationen als auch als Hilfsoperation zum Zustellen einer Nachricht an ein Telefon (siehe nächste Operation).
- Sie kann eine Nachricht an ein bei ihr eingebuchtes Telefon *zustellen*. Hierbei sendet sie die Nachricht an das Telefon und fordert das Telefon zum Empfangen der Nachricht auf.

Das *Netzwerk* aggregiert die vorhandenen Basisstationen. In der *Topologie* des Netzwerks sind die möglichen direkten Verbindungen zwischen je zwei Basisstationen gespeichert.

Das Netzwerk bietet folgende Operationen an:

- Die Operation *weiterleiten* wählt nichtdeterministisch zwei verbundene Basisstationen aus und überträgt eine Nachricht von der einen zur anderen Station. Diese Operation wird als Hilfsoperation zum Bearbeiten von Nachrichten verwendet.
- Die Operation *bearbeiten* fordert entweder nacheinander eine Reihe von Basisstationen auf, Nachrichten an Telefone zuzustellen, oder sie leitet eine Nachricht zwischen zwei Basisstationen weiter.

Aufgabenstellung:

Die oben beschriebene Funktionalität des SMS-Systems soll in den drei Klassen $Telefon$, $Basis$ und $Netzwerk$ spezifiziert werden. **Initialisierungen** der Objekte brauchen **nicht** betrachtet zu werden! Sie können auch davon ausgehen, daß alle versendeten **Nachrichten voneinander verschieden** sind.

Betrachten Sie zunächst die Klasse $Telefon$. Gegeben ist die Zustandsbeschreibung und die Spezifikation der Operation $SMSsenden$.

$Telefon$

$nr : TelNr$

$inbox : \mathbb{P}\, SMS$
$verbunden\_mit : Basis$

$SMSsenden == verbunden\_mit.empfangen$

a.) Spezifizieren Sie die Operation $SMSempfangen$. *6 Punkte*

$SMSempfangen$

Betrachten Sie nun die Klasse $Basis$. Hier ist die Spezifikation der Operation $an\_tel\_zustellen$ vorgegeben.

$Basis$

b.) Spezifizieren Sie den Zustand der Klasse $Basis$. Geben Sie insbesondere eine geeignete Invariante an. *7 Punkte*

$eingebucht:$

$nachrichten:$

$an\_tel\_zustellen == []\![t : eingebucht] \bullet (senden \mathbin{;} t.SMSempfangen)$

c.) Spezifizieren Sie die Operationen $empfangen$ und $senden$. Beachten Sie hierbei die Einschränkungen, die sich aus der Verwendung der Operation in $senden$ in $an\_tel\_zustellen$ ergeben! *12 Punkte*

$empfangen$

$senden$

Schließlich sei der Zustand der Klasse $Netzwerk$ wie folgt spezifiziert:

$Netzwerk$

$stationen : \mathbb{P}\, Basis$
$topologie : Basis \leftrightarrow Basis$

$stationen = \mathsf{dom}\ topologie \cup \mathsf{ran}\ topologie$

d.) Spezifizieren Sie die Operationen $weiterleiten$ und $bearbeiten$ unter Verwendung geeigneter Operatoren des **Schemakalküls**. *10 Punkte*

$weiterleiten ==$

$bearbeiten ==$

Musterlösung

$[TelNr, SMS]$

Telefon

$nr : TelNr$

$inbox : \mathbb{P}\, SMS$
$verbunden\_mit : Basis$

SMSempfangen

$n? : SMS$
$ziel? : TelNr$

$ziel? = nr$
$inbox' = inbox \cup \{n?\}$

$SMSsenden == verbunden\_mit.empfangen$

Basis

$eingebucht : \mathbb{P}\, Telefon$
$nachrichten : SMS \pfun TelNr$

$\forall\, t : eingebucht \bullet t.verbunden\_mit = self$

empfangen

$\Delta(nachrichten)$
$n? : SMS$
$ziel? : TelNr$

$nachrichten' = nachrichten \oplus \{n? \mapsto ziel?\}$

senden

$\Delta(nachrichten)$
$n! : SMS$
$ziel! : TelNr$

$n! \mapsto ziel! \in nachrichten$
$nachrichten' = nachrichten \setminus \{n! \mapsto ziel!\}$

$an\_tel\_zustellen == [\!]\,[t : eingebucht] \bullet (senden \fatsemi t.SMSempfangen)$

Netzwerk

$$
\begin{array}{|l}
\hline
stationen : \mathbb{P}\, Basis \\
topologie : Basis \leftrightarrow Basis \\
\hline
stationen = \mathrm{dom}\, topologie \cup \mathrm{ran}\, topologie \\
\hline
\end{array}
$$

$weiterleiten == \Box[ba, bz : Basis \mid ba \mapsto bz \in topologie] \bullet (ba.senden \mathbin{;} bz.empfangen)$

$bearbeiten == (\mathbin{;}[b : stationen] \bullet b.an\_tel\_zustellen) \Box weiterleiten$

Aufgabe 3 (Statecharts)

(Geschätzte Bearbeitungszeit: 30 min.)

An einer Zapfsäule einer Tankstelle kann Benzin getankt werden. Die Zapfsäule hat eine *Anzeige*, die während der Betriebsbereitschaft der Säule den Preis pro Liter Benzin anzeigt. Sobald Benzin abgegeben wird, zeigt die Zapfsäule die Menge des gezapften Kraftstoffs an.

Der Tankwart hat eine *Statuslampe*, die anzeigt, ob eine Zapfsäule betriebsbereit oder gerade belegt ist. Als dritte Möglichkeit kann sich ein Warnlicht einschalten, wenn eine Zapfsäule länger als 10 Minuten am Stück belegt bleibt.

Der *Tankvorgang* läuft wie folgt ab: Der Kunde zieht den Zapfhahn der betriebsbereiten Zapfsäule aus der Halterung (und führt ihn in die Tanköffnung seines Wagens ein). Die Statuslampe des Tankwarts zeigt an, daß die Säule belegt ist. Sobald der Zapfhahn nicht mehr in der Halterung der Zapfsäule steckt, reagiert die Zapfsäule auf das Drücken des Abzugs, und der Zapfvorgang beginnt. Solange der Abzug gedrückt ist, pumpt die Zapfsäule Benzin. Wird der Abzug losgelassen, stoppt die Abgabe sofort. Der Abzug kann während eines Zapfvorgangs beliebig oft gedrückt werden. Während der Abzug gedrückt ist, läßt sich der Zapfhahn nicht zurück in die Zapfsäule stecken. Der Zapfvorgang wird beendet, indem der Kunde den Zapfhahn zurück in die Halterung steckt. Bevor die Zapfsäule nicht wieder betriebsbereit ist, kann kein Benzin mehr gezapft werden. Ist die abgegebene Benzinmenge gleich 0, so ist die Zapfsäule automatisch wieder betriebsbereit. Im Normalfall muß der Kunde das abgezapfte Benzin beim Tankwart bezahlen. Ist das geschehen, beendet der Tankwart den Tankvorgang, indem er die Zapfsäule durch einen Knopfdruck freigibt und sie damit wieder betriebsbereit ist. Dauert ein Tankvorgang länger als 10 Minuten, stoppt die Benzinzufuhr automatisch, und die Warnlampe des Tankwarts leuchtet. Der Tankwart muß dann die Säule zur erneuten Betriebsbereitschaft durch Knopfdruck freigeben.

Modellieren Sie das Verhalten des Zapsfäulensystems nach obiger Beschreibung mit Hilfe von StateCharts!

1. Dokumentieren Sie die externen Ereignisse, auf die das System reagiert. *3 Punkte*
2. Geben sie den Zustandsautomaten inklusive all seiner Teile an. *14 Punkte*
3. Geben Sie interne Ereignisse an, die durch das System erzeugt werden und dokumentieren Sie diese *kurz*. *3 Punkte*

Hinweise

- Der **genaue Inhalt (Wortlaut)** der Zapfsäulenanzeige ist nicht relevant.
- Benennen Sie alle Zustände und Ereignisse sinnvoll! Vergessen Sie nicht, die Gesamt-Struktur des Automaten anzugeben.
- Es ist hilfreich, den Entwurf so zu zerlegen, daß „unabhängige“ Teile geeignet getrennt voneinander modelliert werden.

1. Externe Ereignisse

- freigeben – Knopfdruck des Tankwarts zum Freigeben der Zapfsäule.
- hahn_ziehen – Der Zapfhahn wird gezogen.
- einhängen – Der Zapfhahn wird wieder eingehängt.
- drücken – Der Abzug wird gedrückt.
- drückenˆ – Der Abzug wird losgelassen.

3. Interne Ereignisse

- ag – Die Anzeige springt auf „Abgabemenge" um.
- ppl – Die Anzeige wechselt zu „Preis pro Liter".
- f – Die Statuslampe springt auf „betriebsbereit" um.
- b – Die Statuslampe wechselt zu „belegt".
- w – Die Statuslampe springt auf „Warnung" um.

2. StateChart