1. SoSy - Übung

Aufgabe 1

a) • „mistake": Irrtum, der dazu führt, dass ein „fault" entsteht

• „error": Fehlerhafter Zustand (zB Variable hat falschen Wert)

• „fault": Fehler im Code

• „failure": Versagen, Programm macht nicht die geplante Aktion

b) „mistake" (Irrtum): Falsche Indizes beim Array

„fault" (Fehler): Der Code, der dadurch entsteht

„error" (Fehlerhafter Zustand): Zählvariable hat falschen Wert

„failure": Obiges Programm wird ausgeführt und wirft Exception

c) Produktfehler: Coder review, Debugging

Error: Fehlerbehandlung schreiben, Rollback

Irrtum: Denkfallen bewusst machen

Versagen: Redundanz (Fehlertoleranz)

Aufgabe 2

a) „fault": Negative Zahlen führen zur Endlosschleife

„mistake": Nicht dran gedacht ??

„error": Fehlerhafter Zustand für $n \leq 0$

„failure": Endlosschleife (→ Stack Overflow)

„fault": fak(0) = 0 → müssten n == 0 abfragen

„error": Bei n = 0 fehlerhafter Zustand

„fault": int zu klein (man sollte größeren Rückgabewert nehmen)

„failure": ab n = 13 → Überlauf

„error": sobald int n überläuft

b) „fault": b=b-a müsste im else-case stehen

„mistake": Vertippt, Leichtsinnsfehler

„error": sobald einmal der else-Teil ausgeführt wurde

„failure": kein o. falsches Ergebnis

29.10.2012
SoSy-Ü
(1)

2. SoSy-Übung

Aufgabe 1

a) `if (x4 < x3) {`
`    min34 = x3;` // fehler, das sollte x4 sein?
`}`

b) Form: x1, x2, x3, x4
(1) 1, 2, 3, 4 (alles, wo $x4 \geq x3$)

(2) 1, 2, 5, 4 ($x3 > x4$, aber $min12 < min34$)

(3) 4, 3, 3, 0 ($x3 > x4$, $min12 \geq min34$)

c) (i) Verifikation: Überprüfung ob Programm mit Programmspezifikation übereinstimmt (Programm testen)
(Modultest → Integrationstest → Systemtest → Akzeptanztest)

Validierung: Ist die Spezifikation richtig? Überprüfung, ob sie dem entspricht, was der Kunde will.
Systembetrieb ist auch Validierung

(ii) Unzureichende Validierung

(iii) Bla bla
- Problem ist entstanden zwischen {intentions} und [requirement specification]

d) (i) Verifikation ist fehlgeschlagen, das Programm tut nicht was die Spezifikation besagt

(iii) Implementierungsfehler

Klausuraufgabe 2.10.2009 (Aufgabe 1)

Welchen Anforderungen muss eine Spezifikation genügen?
- Realisierbarkeit (Anforderung ist unter gegebenen Randbedingungen realisierbar [illegible])
- Eindeutigkeit
- Konsistenz (→ Widerspruchsfreiheit)
- Vollständigkeit (Für jede mögl. Eingabe muss Ausgabe spezifiziert sein)
- Nachweisbarkeit
- Korrektheit (Entspricht Wünschen des Auftraggebers)
- Verfolgbarkeit

a) 1. Vollständigkeit: Nein, bei 2. u. 3. ist unterschiedliche Ausgabe für gleiche Eingabe spezifiziert

2. Konsistenz: 2. und 3. ist ein Widerspruch weil Summe ≠ Produkt

3. Realisierbarkeit: Nicht realisierbar wegen Widerspruch

→ Verlesen

a) 1. Vollständigkeit: Ist gegeben, weil für jede Eingabe eine Ausgabe spezifiziert ist

2. Konsistenz: Widerspruch bei 1+2*3

3. Realisierbarkeit: Nicht realisierbar weil Widerspruch

b) In 3.: P(A) = Produkt ∧ P(B) = Produkt
statt Syntaxfehler

Oder in 2.: P(z) = Produkt

Oder 2. komplett löschen

5.11.2012
SoSy-Ü
(1)

3. SoSy-Übung

Aufgabe 4: Anforderungen

a) 1. Vollständigkeit: gegeben, alle Fälle sind abgedeckt
2. Konsistenz: nicht gegeben, Widerspruch für einelementiges x, in dem s enthalten ist
3. Realisierbarkeit: nicht gegeben, weil Widerspruch

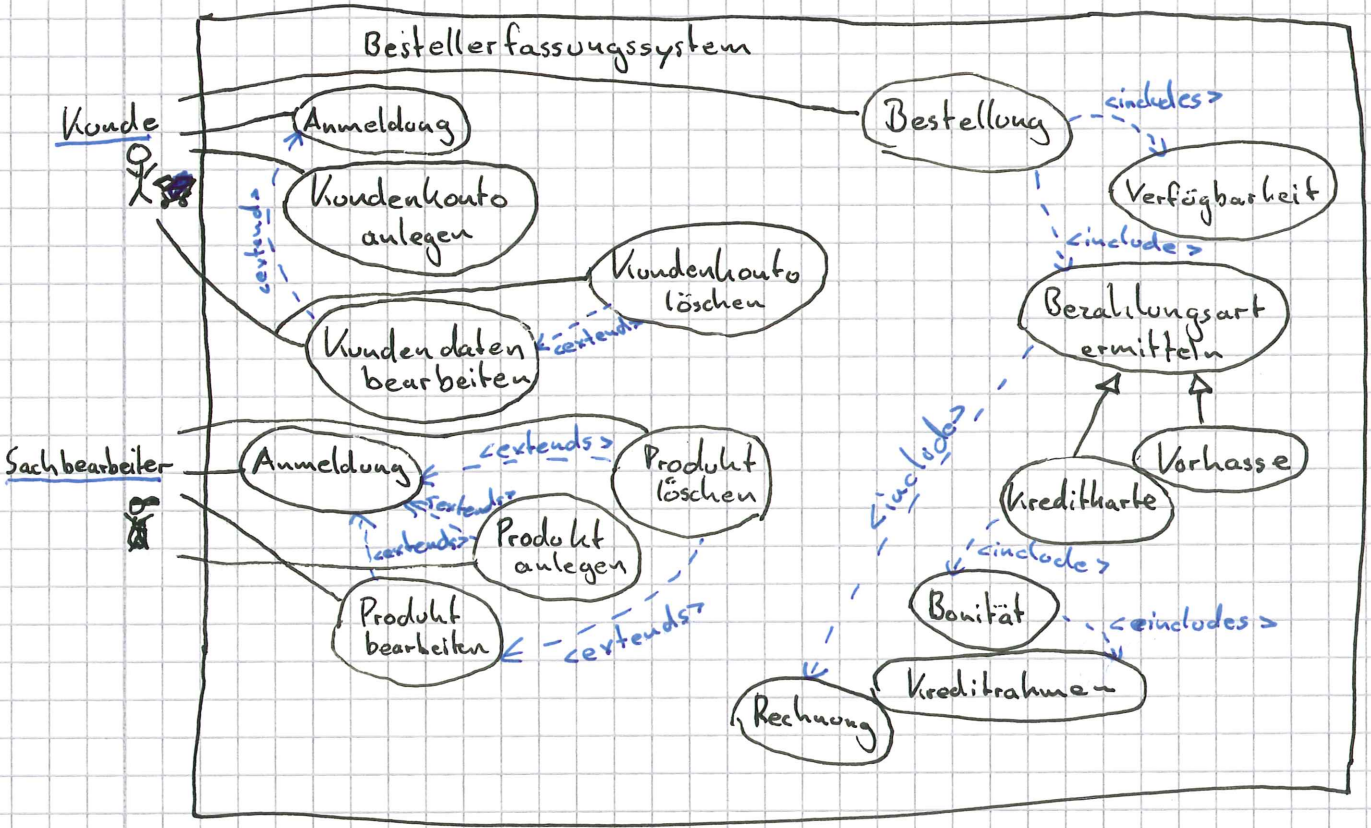
b) A2: p=0

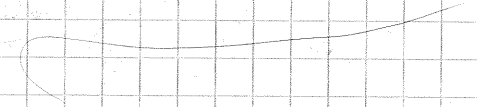
c) 1. Ist falsch
Bsp: Spezifikation: „Programm gibt 4 zurück"
Auftraggeber wollte aber ein Zufallszahl

2. Ist falsch
Bsp: - Zeit und Kosten sind nicht realisierbar
- Unlösbares mathematisches Problem

3. Ist falsch
Bsp: - Nicht realisierbare Anforderungen

Aufgabe 5: Use-Case-Modellierung



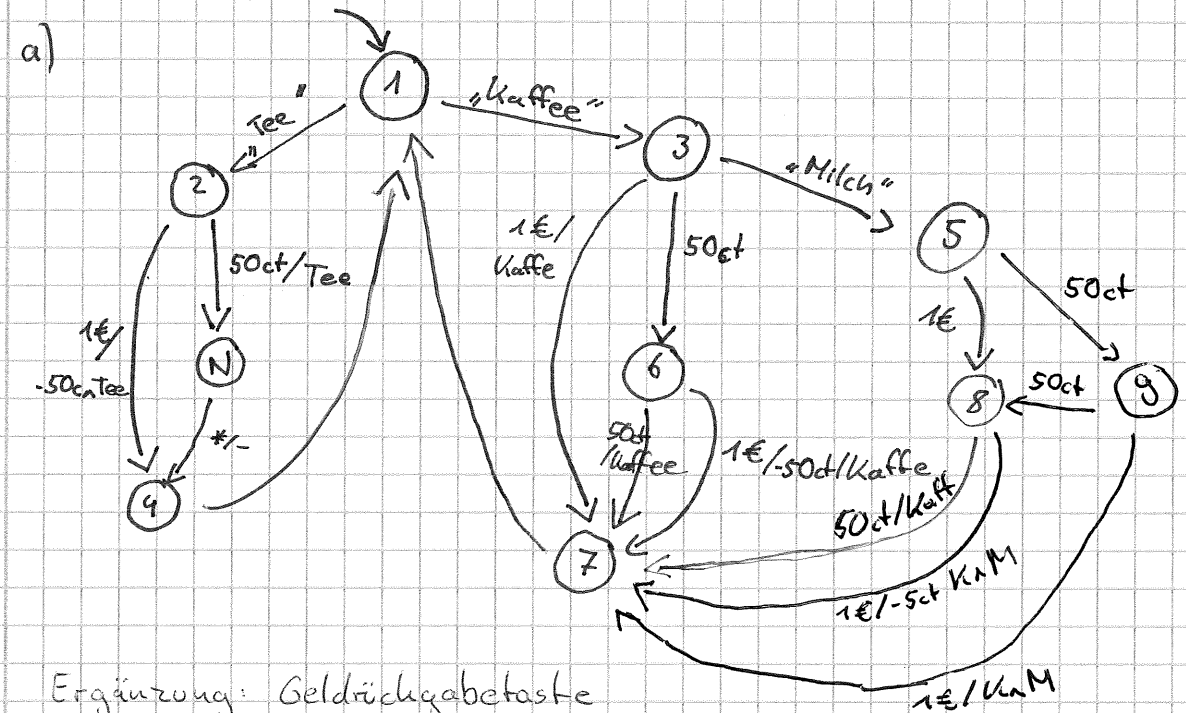


12.11.2012
SoSy-Ü

4. SoSy-Übung

Aufgabe 6: Endlicher Zustandsautomat

a)



Ergänzung: Geldrückgabetaste

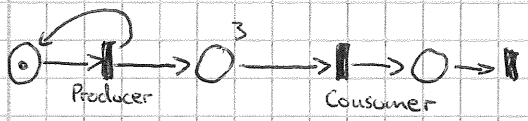
Extended final state machine: Hat Variablen

a

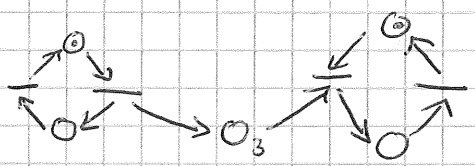
a

Aufgabe 7: Petri Netze I

Meins:

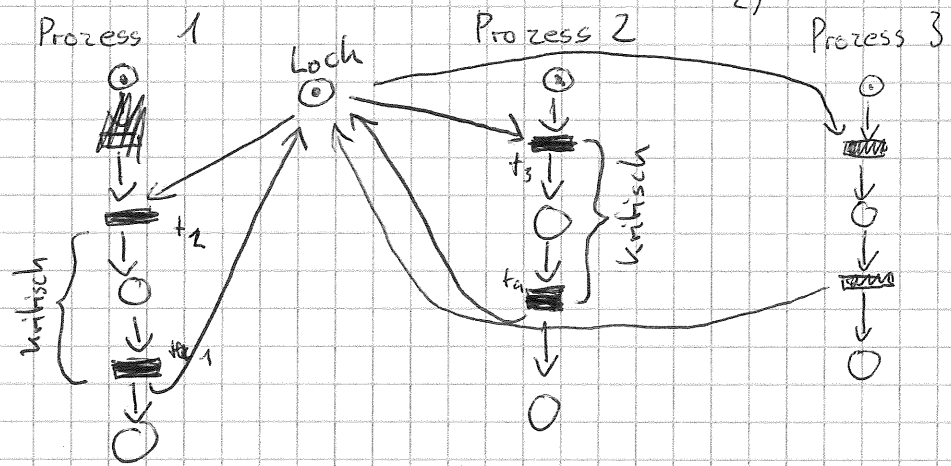


Laut Übung:



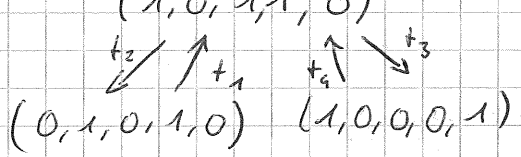
Aufgabe 8: Petri Netze II

a)



(An der Tafel: Prozess = unendliche Schleife

b)



19.11.2012
SoSy-Ü
(1)

5. SoSy Übung

~~Aufgabe 10: OCL I~~

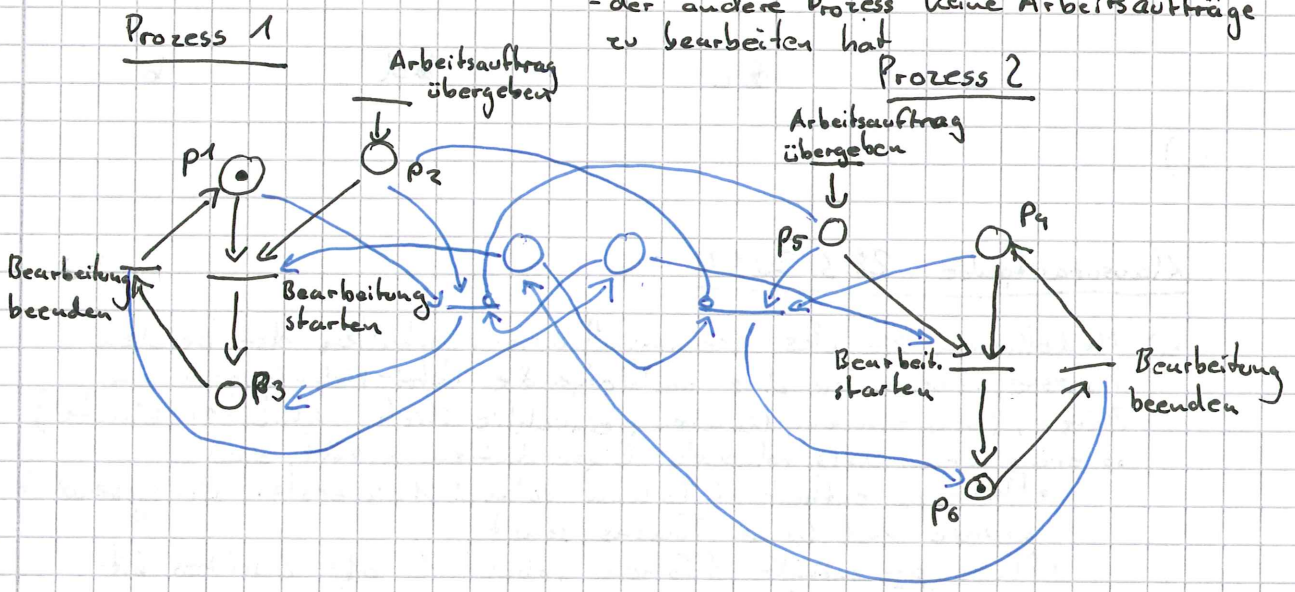
~~a)~~

1 Klausuraufgabe 28.9.2012

Teil 2

Zwei Prozesse greifen zur Bearbeitung ihnen übergebener Arbeitsaufträge auf eine gemeinsame Ressource zu.
Dabei gelten folgende Bedingungen:

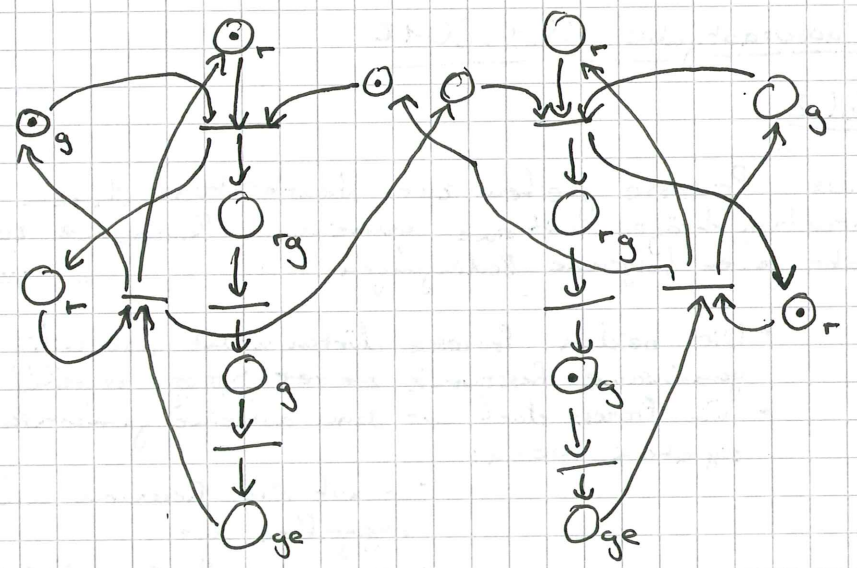
- Die beiden Prozesse dürfen nicht gleichzeitig auf eine gemeinsame Ressource ~~zur Verfügung~~ zugreifen
- Ein Prozess darf nur dann auf eine gemeinsame Ressource zugreifen wenn:
  - er auf die Ressource nicht zuletzt zugegriffen hat
  - der andere Prozess keine Arbeitsaufträge zu bearbeiten hat



Aufgabe 9

a) a: F_1, F_2, F_5, F_6 b: F_3, F_4, F_7, F_8 c: A_1, A_4 d: A_2, A_3

b)



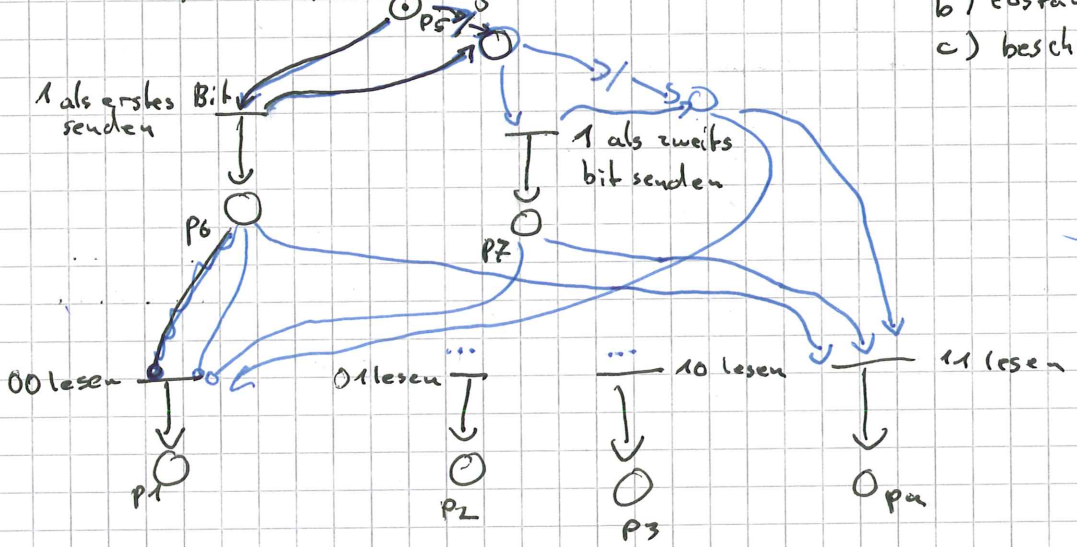
a c d b

c)

Klausuraufgabe 28.3.2012

a) Modellieren Sie das Kommunikationsverhalten der beiden Prozesse, indem Sie das nachfolgende Petri-Netz-Gerüst um Plätze, Transitionen, Kanten (einschließlich Inhibitor-Kante) so erweitern, dass folgende Eigenschaften erfüllt sind:
- Platz p_0 enthält 1 Token sobald 1 als erstes Bit gesendet wurde und keine Tokens sonst
- Platz p_2 enthält 1 Token sobald 1 als zweites bit gesendet wurde und keine Tokens sonst
- Platz p_i $(1 \leq i \leq 4)$ enthält 1 Token, sobald die entspr. Nachricht vollst. gelesen wurde

b) Zustandsgraph
c) beschränkt?



3.12.2012
SoSy - Ü
(1)

Aufgabe 12

Kundenverwaltung - Lagerverwaltung: —
Kundenverwaltung - Bestellungserfassung: Datenkopplung | Datenstruktur
Kundenverwaltung - Auslieferungsplanung: —
Kundenverwaltung - Datensicherung: Kontrollkopplung
Lagerverwaltung - Bestellungserfassung: globale Kopplung
Lagerverwaltung - Auslieferungsplanung: —
Lagerverwaltung - Datensicherung: —
Bestellungserfassung - Auslieferungsplanung: Datenkopplung
Bestellungserfassung - Datensicherung: —
Auslieferungsplanung - Datensicherung: —

Aufgabe 13

a) Funktionale Kohäsion
b) logische Kohäsion
c) prozedurale Kohäsion / sequentielle Kohäsion
d) sequentielle Kohäsion
e) sequentielle Kohäsion / prozedurale Kohäsion / temporale Kohäsion / komm. Koh.
f) kommunikative Kohäsion

Klausur 15.4.2011

Seite 3

e) Zwei Module p und q haben Inhaltskopplung, wenn Modul p eine Datenstruktur als Parameter an Modul q übergibt, Modul q aber nur Teile der Datenstruktur verwendet.
Falsch. (Begründung)

f) Ein Modul hat prozedurale Kohäsion, wenn es eine Reihe von Aktionen auf gemeinsamen Daten ausführt, wobei die Reihenfolge irrelevant ist

c) Eine Transition t in einem Petri Netz ist schaltbar, falls jede ihrer Eingabestellen (Plätze) mindestens einen Token enthält

d) Die Markierung eines Petri Netzes steht für dessen Zustand

b) Eine Anforderung ist korrekt, wenn es möglich ist, die Anforderung unter den ihm ~~zu~~ voraus bekannten

Klausur 10.9.2012

Seite-3

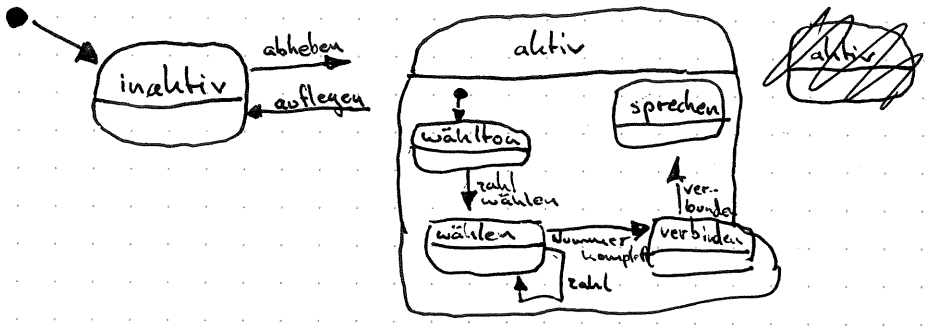
c) Nennen Sie neben dem „Datenspeichermodell" zwei weitere Beispiele für Architekturmodelle.

d) Benennen sie die Kopplungsart zu folgender Beschreibung: Zwei Module greifen auf gemeinsame globale Variablen schreibend und lesend zu

e)

Aufgabe 14: Statechart I

a)



6. SoSy Übung

26.11.2012
SoSy-Ü
(1)

Aufgabe 10: OCL I

a) context Automat
inv: self.anzahlZustaende > 0

b) context Konto :: abheben (betrag: real): void
pre: guthaben >= Betrag
post: ~~betrag~~ guthaben = guthaben@pre - betrag

Ruft man Methoden mit . oder -> auf?

c) context Kegel :: volumen (): real
pre: radius > 0 and hoehe > 0
post: result = $r^{radius} * r_{radius} * 3,141592 * hoehe / 3$

d) context Autovermietung
inv: mietwagen->size() > 0

e) context Autovermietung
inv: Autovermietung.allInstances()->size() > 0

f) context Autovermietung
inv: mietwagen->exists (a: Auto | a.mietpreis <= 20)

Aufgabe 11: OCL II

a) context Besprechung
inv: teilnehmer->includes (verantwortlicher)

b) context Raum
inv: Raum.allInstances()->forAll (r: Raum | r <> self implies r.nummer <> self.nummer)

c) context Besprechung :: teilnehmerHinzufuegene (p: Person): void
pre: not teilnehmer->includes (p)
post: teilnehmer->includes (p) and
teilnehmer->size() = teilnehmer@pre->size() + 1

d) context Besprechung
~~inv: Besprechung.allInstances()->forAll (b: Besprechung |~~
~~not exists (a,b. b.ausweichtermine = self))~~
~~->(self.ausweichtermine->size()>0))~~

meins: inv: Besprechung.allInstances()->forAll (b: Besprechung |
(not b.ausweichtermine->includes (self))
-> (self.ausweichtermine->size()>0)

implies

Übung: inv: self.ausweichtermine->size()>0 or
Besprechung.allInstances()->exists (b. Besprechung | b.ausweichtermin->includes(self)

e) context Besprechung
inv: teilnehmer -> size () <= raum.sitzplaetze

f) context Besprechung

inv: Besprechun.allInstances () -> forAll (b. Besprechung |

Von der Logik her self<>b gehts glaub ich ?
(b.Raum = self.Raum) implies
(b.start ≥ self.ende for b.start ≤ self.start))

inv: Besprechung.allInstances() -> forAll (b: Besprechung |
(self <> b and b.raum.nummer = self.raum.nummer)
implies (self.start.sekunden > b.ende.sekunden or
b.start.sekunden > self.ende.sekunden))

g) context Besprechung

inv: Besprechung.allInstances -> forAll (b: Besprechung |

(b <> self and (b.start.sekunden ≤ self.ende.sekunden
or b.start.sekunden ≤ self.ende.sekunden))
implies

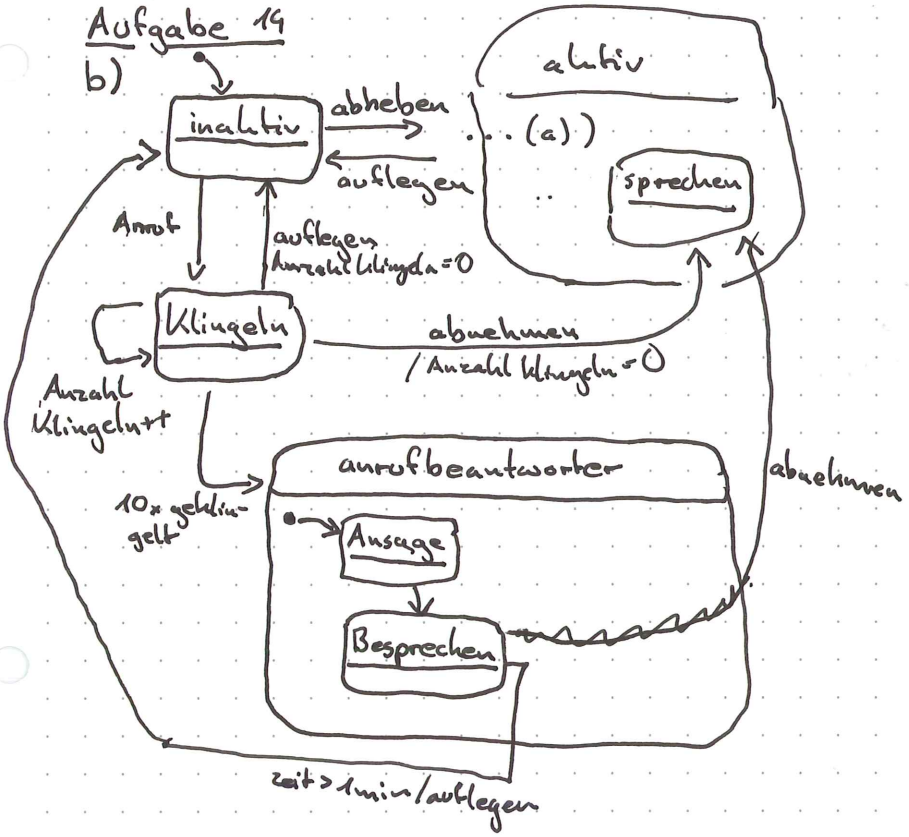
(b <> self and (b.start.sekunden >= self.start.sekunden
and b.start.sekunden <= self.ende.sekunden))
implies
b.teilnehmer -> forAll (t | not self.teilnehmer -> includes (t))

Aufgabenblatt 7

10.12.2012
SoSy-B
(1)

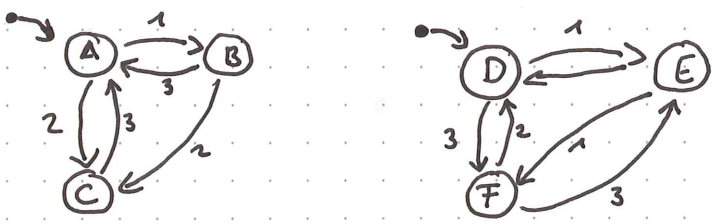
Aufgabe 14

b)

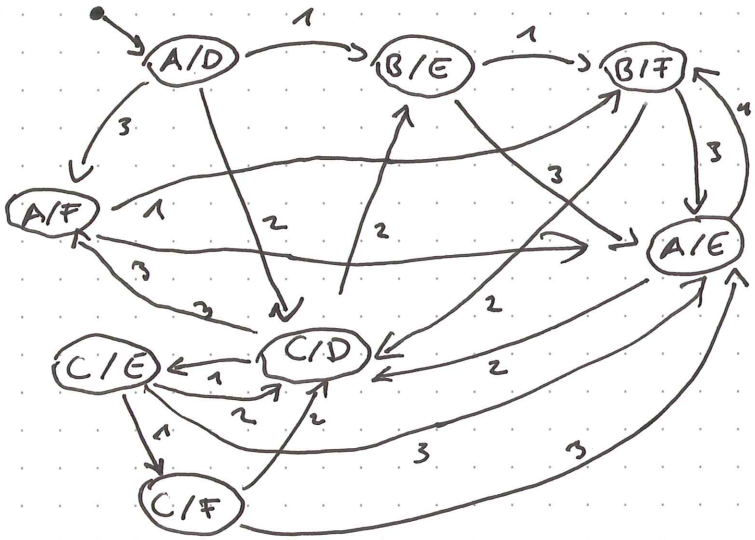


Aufgabe 15

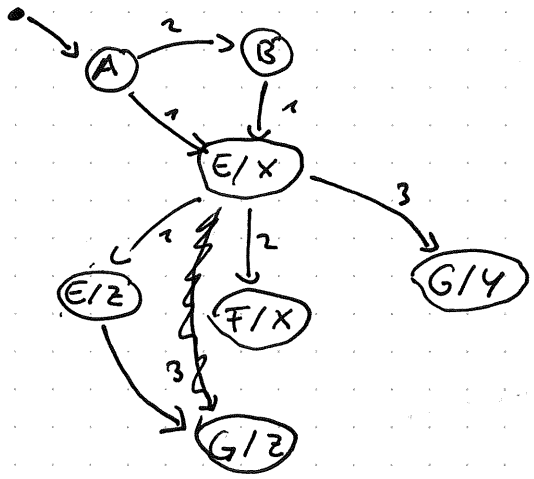
a)



b)



Aufgabe 16



17.12.2012
SoSy-Ü
(1)

Klausur 29.9.2011

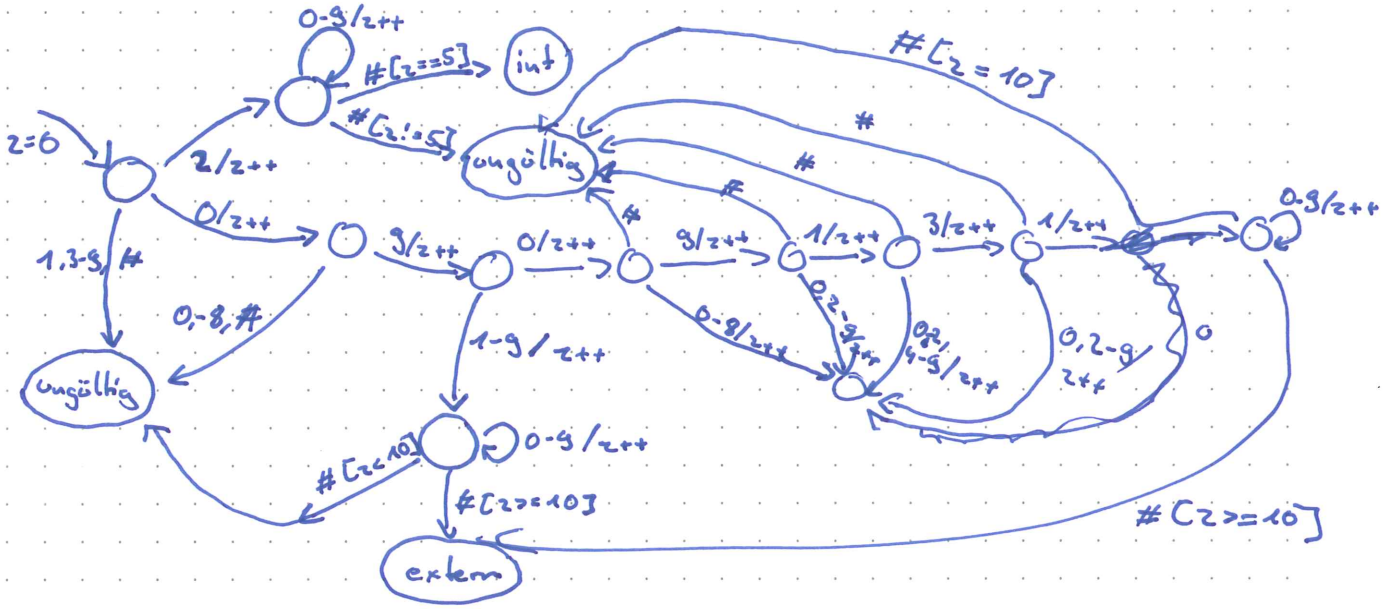
Aufgabe 6

Ein Erlanger Telekommunikationsunternehmen klassifiziert Telefonnummern nach folgenden Kriterien:

1. Eine interne Rufnummer beginnt immer mit der Ziffer »2« und enthält genau 5 Ziffern

3. Alle anderen Rufnummern, die keine der oben beschriebenen Bedingungen erfüllen, sind als ungültig zu klassifizieren

Geben Sie eine Erweiterte Zustandsmaschine an, die eine Telefonnummer ziffernweise einliest und entsprechend obiger Klassifikation einen der folgenden Endzustände erreicht: »Interne Rufnummer«, »Externes Ortsgespräch«, »Externes Ferngespräch« oder »Ungültig«. Eine Telefonnummer gilt dabei als vollst. eingelesen, wenn die Zustandsmaschine das Zeichen »#« liest.



Klausur 13.4.2011

20.12.2012
~~GLoLoP~~
SoSy
(1)

$P_1 \rightarrow E_1$

$P_1, P_2 \in \mathcal{P}$

$P_2 \rightarrow E_2$

$\mathcal{P} \rightarrow \mathcal{E}$

Menge von Problemen — Entwurfsmuster

Wir lernen $\mathcal{E}_1 \dots \mathcal{E}_{12}$ kennen $\mathcal{P}$ $\mathcal{T}$ $\mathcal{J}$ J $\mathcal{R}$

Klassifikation von Mustern

~~• Ereignismuster~~

• Klassenbasiert oder Objektbasiert ← spannender

E_1: Singleton

Stellt sicher, dass es von einer Klasse immer nur eine Instanz gibt

Lösung: private Variable Instance, die mit getInstance() geholt werden kann und privater Konstruktor

E_2: Abstrakte Fabrik

Blatt 8

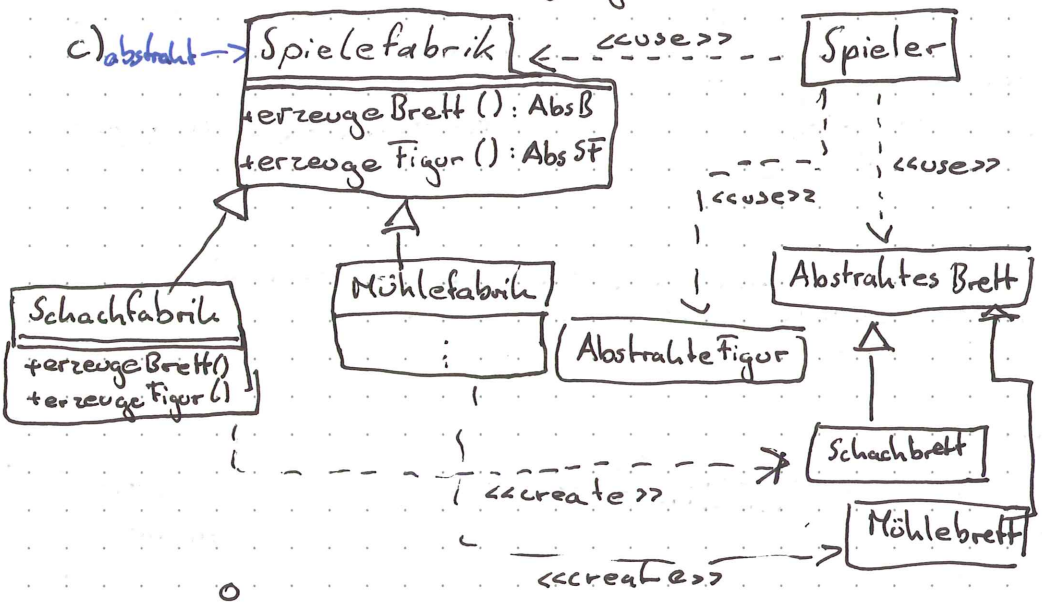
7.1.2012
SoSy-Ü
(1)

Aufgabe 17: Entwurfsmuster I

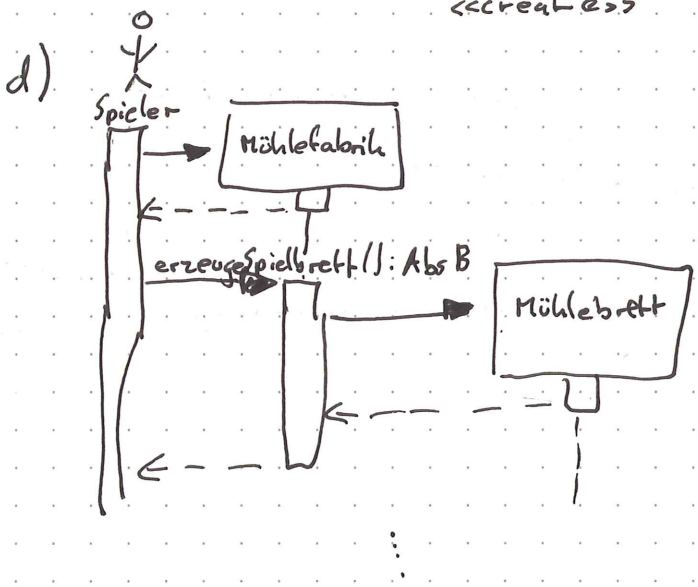
a) Abstrakte Fabrik

b) Gültigkeitsbereich: Objektbasiert
Klasse: Erzeugungsmuster

c)



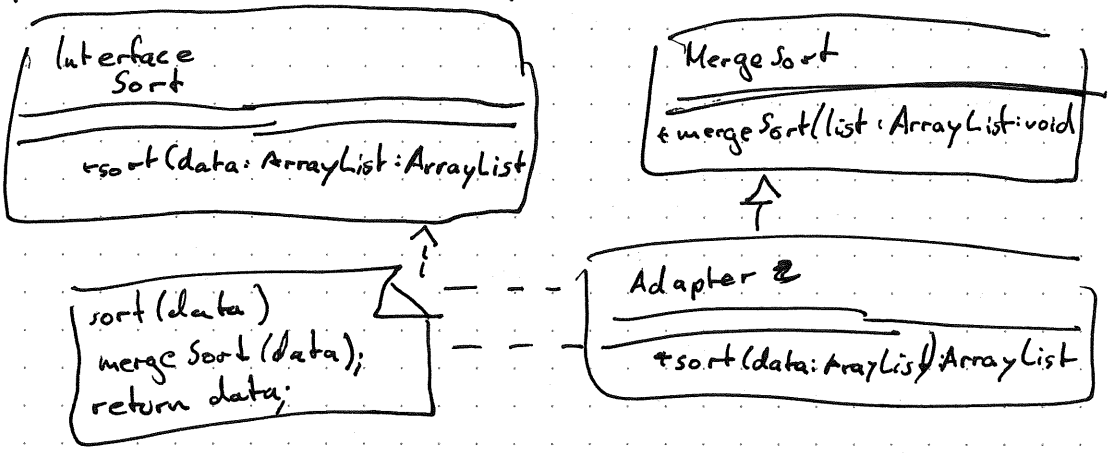
d)



Aufgabe 18: Entwurfsmuster II

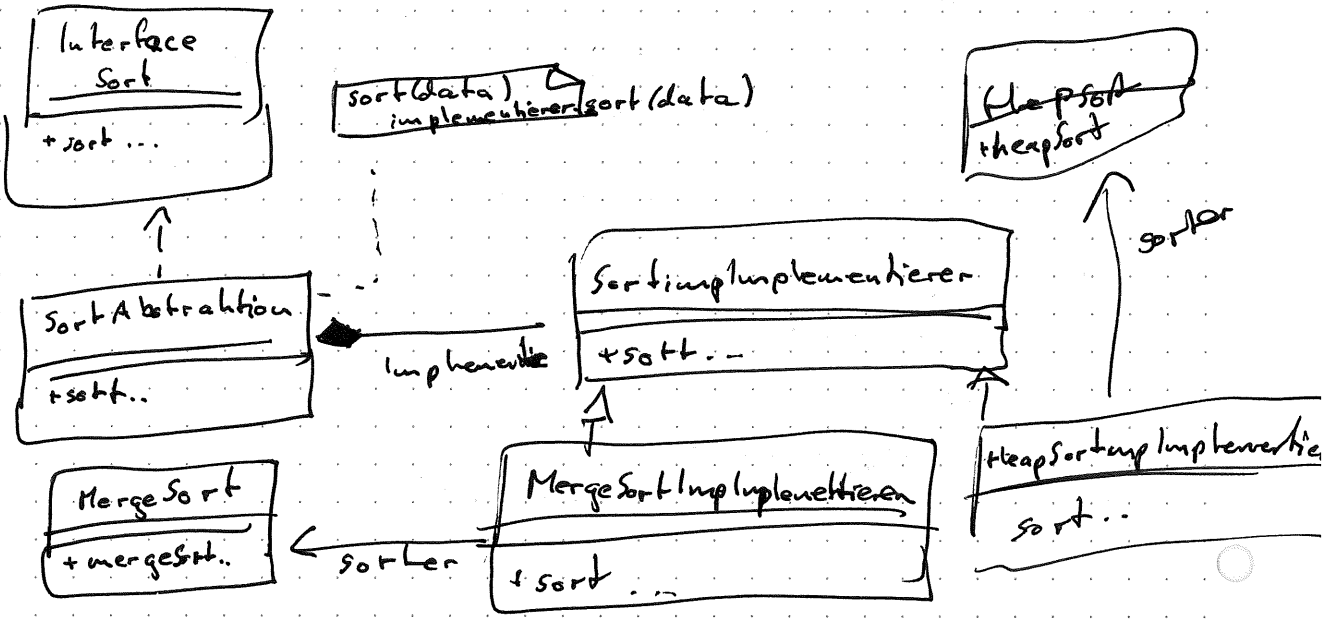
a) Adapter (klassenbasiert oder objektbasiert)

b)



Nachteil: Wenn Adapter auch noch von was anderem erben muss
Mehrfach vererbung nicht immer möglich

c) Strategie oder Brücke



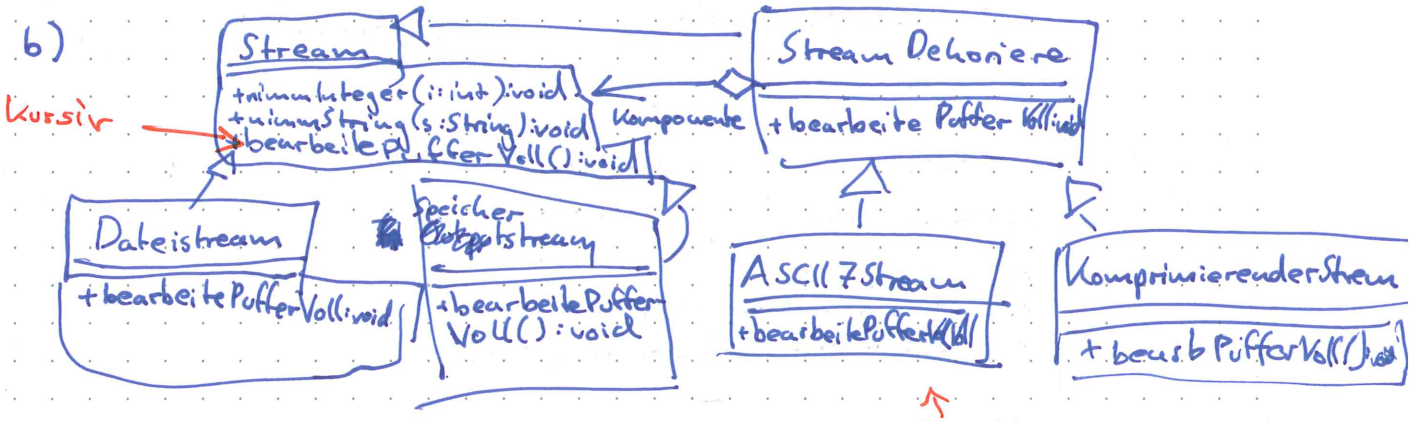
Blatt 10

14.1.2013
-Ü
(1)

Aufgabe 21: Entwurfsmuster III

a) Dekorierer

b)

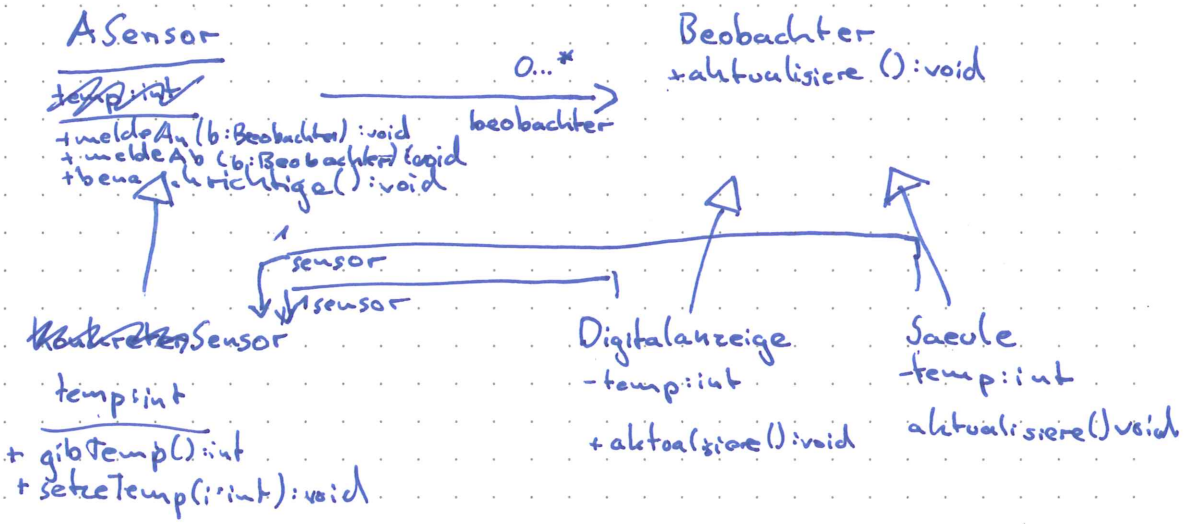


c) SpeicherStream um Komp.Stream ergänzen:

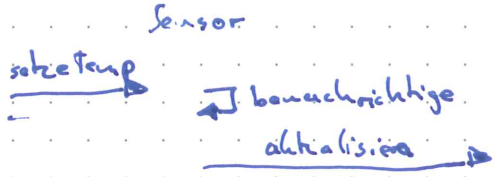
[Bild]

Aufgabe 22: Entwurfsmuster IV

a) Observer, 1 Klasse stellt Daten zur Verfügung, verschiedene Anzeigeklassen wollen informiert werden



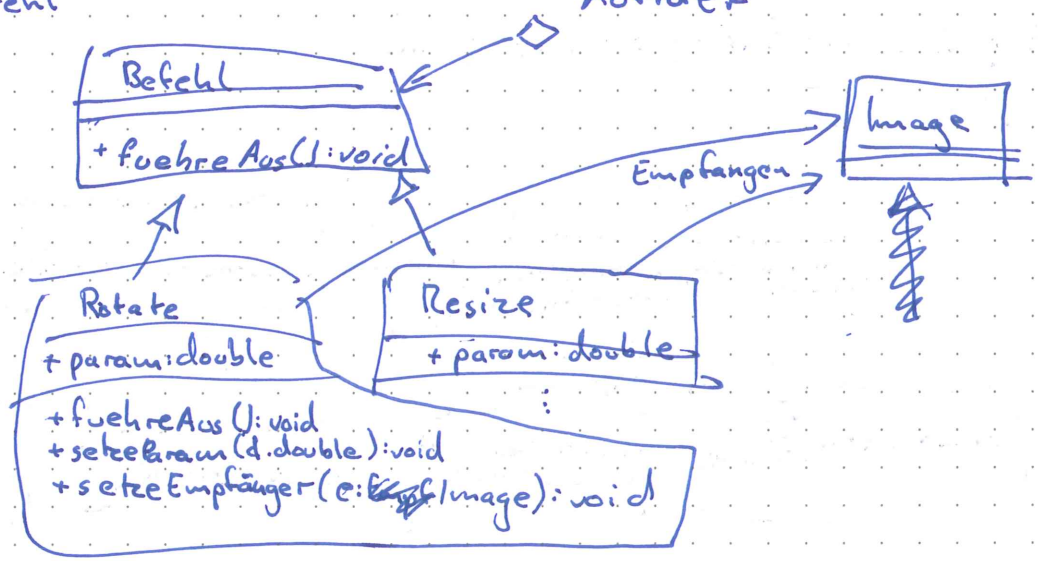
d)



Klausur 28.9.2012 (GWE)

a) Befehl

b)



21.1.2013
SoSy-Ü

Aufgabe 23

a) Fehler finden
Zuverlässigkeit nachweisen

b) Modultest
Integrationstest
~~X~~ Systemtest
Annahmetest

c)
strukturelle test: anhand des Programmcodes ~~Test~~ struktur wird getestet

funktionaler Test: Testfälle anhand der spezifikation auswählen

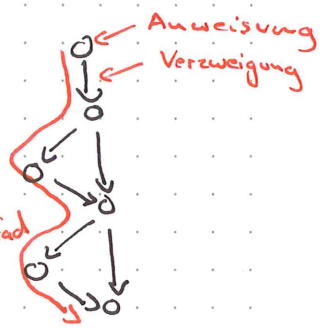
Aufgabe 24

a) 1 (zB t=3 u. s=3 | t=4 s=0)

b) 2 (zB t=3, s=3 | t=4 s=0)

c) 4 (t=3, s=3 | t=4, s=0 | t=4, s=1 | t=0, s=0)

Skizze:
(Kontrollfluss)



Aufgabe 25

a) 2, 2

b) 4, 6

c) Schleife und nicht alle Pfade möglich

Verzweigungsüberdeckung
⟹ Anweisungsüberdeckung

Klausuraufgabe 13.4.2011 S.14 GSWE

b)
a
abaa

Blatt 12

28.1.2013
SoSy-Ü
(1)

Aufgabe 26: Testen IV

a) Äquivalenzklassentest ist Blackboxtest

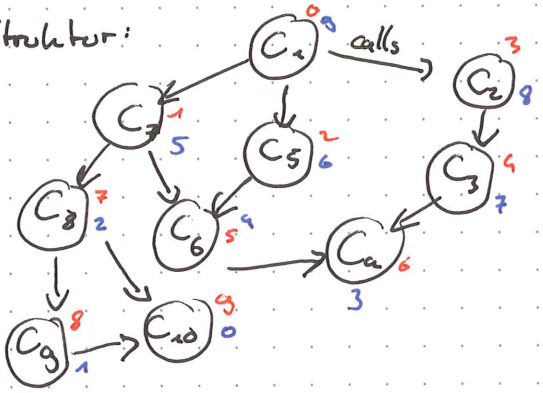
$x = 2$, $x = 101$, $x = 0$

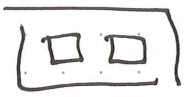
b) {0, 1000, 1, 99, 999, 100}, INT_MIN, INT_MAX}

c) Äquivalenzklassen: ~~Vereinigungsüberdeckung~~ alles ???

Aufgabe 28: Integrationstest

a) Struktur:



Komponente: 

b) top-down: Aufrufende Komponente zuerst testen

c) bottom-up: Aufgerufene Komponente zuerst testen

Aufgabe 27: Testen V

4. Unklar, weil nicht klar ist was der Konstruktor tut
20. Auch unklar was der Konstruktor macht
29. c)
25. d)
16. a)

Probeklausur

5.2.2013
SoSy-Ü

Aufgabe 1

a) Irrtum, Produktfehler, fehlerhafter Zustand, Versagen
b) (Mentale → Fehler)

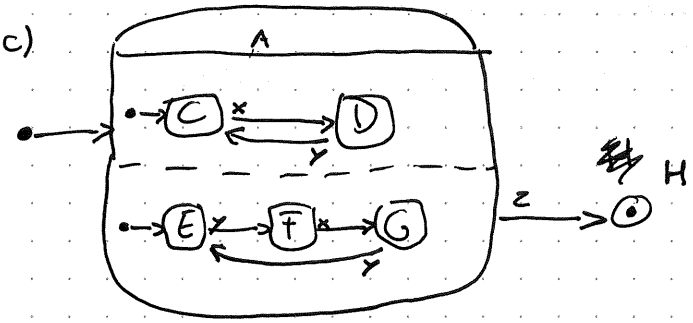
c) OCL, Petri-Netze, Zustandsautomaten
d) Wohldefinierte Semantik
e) Vorteile: eindeutig
Verifikation
Widerspruchsfreiheit
Nachteile: schwer zu erlernen
aufwendig

f) Klassenkandidaten
Beziehungen
Variablen

Aufgabe 2

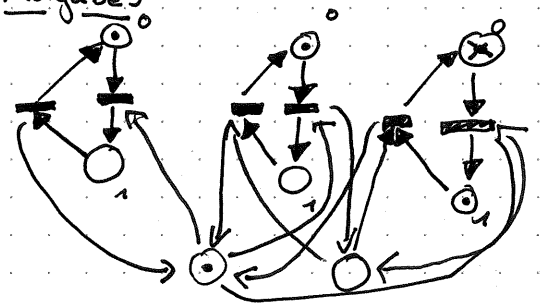
a) (C, G), (D, F)
b) (H),

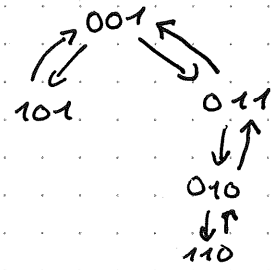
c)



Reflexive Kanten kann man weglassen

Aufgabe 3





Aufgabe 4

a) context Abteilung
inv: mitarbeiter -> includes (abteilungsleiter)

b) context Abteilung
inv: nachfolger -> size() = 1

c) context Firma
inv: nachfolger -> size() = 0

d) context Firma
inv: abteilungen -> forAll (a | a.mitarbeiter -> includes (geschäftsführer))

e) context Antrag
inv: Person.allInstances -> exists (p | p.antraege -> includes (self))

f) context Abteilung
inv: Abteilung.allInstances -> exists (a | a.abteilungsleiter
= self.abteilungsleiter
and a <> self)

g) context Person :: entscheide (a: Antrag)
pre: Abteilung.allInstances -> exists (ab | ab.abteilungsleite = self
and a.kosten <= ab.genehmigungsLimit)

Aufgabe 5

a) Kompositum
b) Objekt basiertes Strukturmuster
c)