Lösungshinweise für Übungsblatt 14: **Probeklausur**

Alle Antworten sind ohne Gewähr!

Aufgabe 14-1: Allgemeinwissen, 8 Punkte

	r	f
In UML-Klassendiagrammen sind Mehrfachvererbungen erlaubt.	X	
Das statische Modell in der Analyse unterscheidet sich meistens vom späteren statischen Modell im Entwurf.	X	
UML ist ein Modellierungsprozess, bei dem die Erstellung von Diagrammen im Vordergrund steht.		X
Nicht-funktionale Anforderungen werden in Anwendungsfällen nicht optimal repräsentiert.	X	
Die Signatur einer Methode ist ein Teil ihrer Schnittstellenbeschreibung.	X	
Klient-Dienstgeber-Architekturen (client server architectures) sind immer auch Schichtenarchitekturen.		X
Der Programmierer eines Moduls eines Softwareprodukts ist ein Beteiligter (stakeholder) des Produkts.	X	
Anwendungsfälle lassen sich vollständig mittels UML-Anwendungsfalldiagrammen beschreiben.		X
Extreme Programming ist ein Beispiel für ein inkrementelles Prozessmodell.		X
Sequenzdiagramme in UML zeigen Interaktionen zwischen Klassen.		X
Entwurfsmuster werden in der Regel als Klassenbibliotheken ausgeliefert.		X
Ein Wasserfall-artiger Entwicklungsprozess ist gut geeignet für hardwarenahe Systeme.	X	
Wenn die Kohäsion eines Moduls steigt, sinkt die Kopplung und andersherum.		X
Ein Prozessmodell ist eine Abstraktion von Prozessen.	X	
Bei agilen Softwareprozessen werden Bereichsklassen und Lösungsklassen in der Regel nicht auseinander gehalten.		X
Durchsichten (reviews) können zum Beispiel mittels Black-Box- oder mittels White-Box-Techniken durchgeführt werden.		X

Aufgabe 14-2: Strukturierte Methode, 8 Punkte

*Finden Sie zunächst Akteure, d.h. alles, was mit dem System interagiert, aber selbst nicht Teil des **Systems** ist. Identifizieren Sie danach für jeden **Akteur**, was er mit dem System tun will und bilden Sie daraus einen Anwendungsfall. Jede genannte **funktionale** Anforderung muss damit abgedeckt sein. Schreiben Sie für jeden **Anwendungsfall** ein Szenario bestehend aus einer Vorbedingung, einer nummerierten Abfolge der Tätigkeiten und einer **Nachbedingung**. Ein solches Szenario können Sie auch grafisch in Form eines **Sequenzdiagramms** darstellen. Daraus ergehen die Objekte, die interagieren. Definieren Sie zu den so ermittelten Objekten passende **Klassen**. Fügen Sie dies grafisch zu einem Klassendiagramm zusammen. Modellieren Sie auch **Assoziationen** zwischen den **Klassen**. Ergänzen Sie zusätzlich die **Operationen**, die sich aus den Szenarien ergeben. Wenn möglich, beschreiben Sie danach jeweils grafisch in einem **Zustandsdiagramm** den dazugehörigen Lebenszyklus.*

Aufgabe 14-3: Anforderungsanalyse, 20 Punkte

1. *"Anmeldung vor neues Thema anlegen"* versus *"Jeder muss Möglichkeit erhalten .. ein neues Thema zu erzeugen"*

"Darf neuen Beitrag oder neues Thema hinzufügen, sonst nichts" versus *"Auch Löschen möglich"*.

2. Thema erzeugen, Beitrag hinzufügen, Anmelden, Alte Version Wiederherstellen, Löschen.

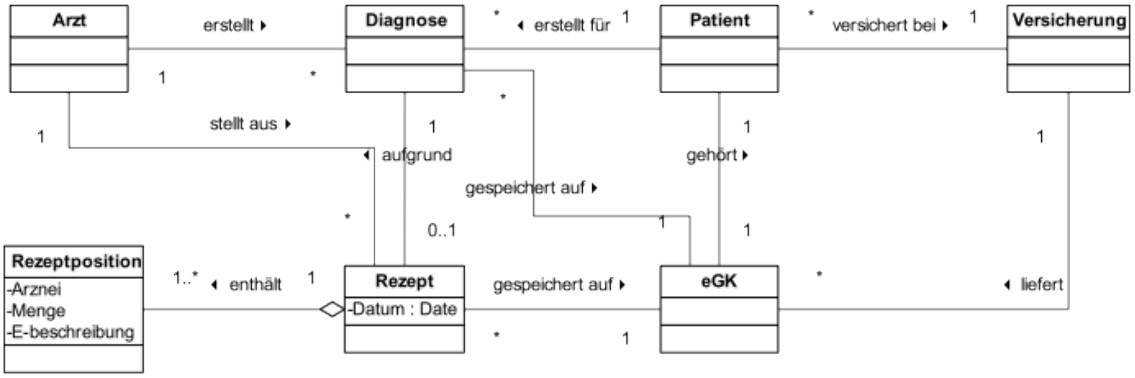
3. "leicht verständlich", "unkomplizierte Textsyntax", "Versionsverwaltung benutzen"

4.

Thema	K
Name	A
Version	K (evtl. auch A)
Leser	S
Versionsverwaltung	E
Textsyntax	S

Aufgabe 14-4: Klassendiagramm, 25 Punkte

1.



2. **Q** nein

S ja (0..1 Rezept aufgrund von 1..2 Diagnosen)

V ja (* Patient versichert bei 1..* Versicherung)

Aufgabe 14-5: Datenmodelle, 20 Punkte

1. Am 26. April um 14:00 Uhr **(1)** spricht Prof. Dr. Günther von Lojewski **(1)**, Leiter des Journalisten-Kollegs der Freien Universität **(1)** und Bruder des bekannten Fernsehjournalisten Wolf von Lojewski **(3)** über „Die Rolle des Journalismus in jungen Demokratien" **(1)**.

Diese Auftaktveranstaltung **(2 o. 3)** findet im Hörsaal 1a **(1)** in der Habelschwerdter Allee 45 (Silberlaube) **(1)** statt und gehört zu **(1)** der Ringvorlesung „Journalismus: Die vierte Gewalt" **(1)**, die während der Vorlesungszeit des Sommersemesters (also bis 15. Juli) **(2 o. 3)** beinahe **(3)** wöchentlich am selben Ort zur selben Zeit **(1)** mit zahlreichen höchst aktuellen Themen **(2 o. 3)** stattfinden wird.

2. 2 Events und 1 Participant

Aufgabe 14-6: Zustandsübergangsdiagramme, 16 Punkte

1. Die Knöpfe in den Ebenen und die Knöpfe im Fahrstuhl sind paarweise identisch, denn es ist egal, ob jemand von innen in eine Ebene möchte oder jemand auf einer Ebene in den Fahrstuhl möchte.

2. Modellierung Z: 16 - Vier Zustandsklassen aufgrund Ebene/Richtung (Ebene 1 als „1 von oben“ und „1 von unten“) mit je vier Knopfzuständen (die Knöpfe der „eigenen“ Ebene sind nicht von Belang). Zustände:

   - Für Ebene 0: E0, E0-K1, E0-K2, E0-K1-K2
   - Für Ebene 1 (von oben): E1o, E1o-K0, E1o-K2, E1o-K0-K2
   - Für Ebene 1 (von unten): E1u, E1u-K0, E1u-K2, E1u-K0-K2
   - Für Ebene 2: E2, E2-K0, E2-K1, E2-K0-K1

   Erläuterung: Der erste Teil des Zustandsbezeichners steht für die aktuelle Ebene und bei Ebene 1, aus welcher Richtung der Fahrstuhl zuvor gekommen ist. Dahinter stehen die gedrückten Knöpfe (K*) oder nichts, wenn kein Knopf gedrückt ist. Damit sind alle Zustände, die der Fahrstuhl einnehmen kann, abgedeckt. Dann braucht man für die Übergänge:

   - Ereignisse: das Drücken der Knöpfe
   - Aktionen: hochfahren, runterfahren

   Alternative Lösung: 8 - Zustände repräsentieren alle möglichen Kombination gedrückter Knöpfe, das sind $2^3 = 8$. Dann braucht man für die Übergänge zusätzlich Guards/Bedingungen, die die auszuführende Aktion von der aktuellen Ebene (und ggf. aus welcher Richtung der Fahrstuhl kommt) abhängig macht.

3. Modellierung E: Naiver Ansatz (mit Einschränkungen): 4 – Aus Z fallen die Zustandsklassen aufgrund der gedrückten Knöpfe weg, da sie nun in den Eingaben kodiert sind. Es gibt nur noch die Zustände: E0, E1o, E1u, E2. Ein Übergang von E1o zu E0 könnte folgendermaßen aussehen:

   - Ereignis: Knopf 0 und 2 gedrückt
   - Aktion: runter; K0 löschen; hoch; hoch; K2 löschen

   Das würde bedeuten, dass der Fahrstuhl solange keine Eingaben beachtet, bis er sowohl das EG als auch OG 2 angefahren hat. Doof, aber würde funktionieren.

4. Zustandsdiagramme „explodieren“ hier mit steigendem N, daher eher ungeeignet. Eine Spezifikation müsste das Regelmäßige und Wiederkehrende der einzelnen Stockwerke nutzen, denn jedes neue Stockwerk verbindet sich auf immer die gleiche Weise mit den schon vorhandenen. (Modularität, Parametrisierung)

Aufgabe 14-7: Entwurfsmuster, 16 Punkte

1. Kommando-Muster. Vorteile: Beeinflussung der Kommandos ist zentral möglich (enable/disable, `new()`). Entkopplung von Befehlsgeber (Menü / Knopf) und Funktionalität.

2. Kompositum-Muster. Vorteile: Jedes Teil lässt sich über dieselbe Schnittstelle bedienen (`paint()`). Man braucht nicht zu wissen, dass eigentlich viele Objekte betroffen sind. Komposita können beliebig tief geschachtelt werden.

3. ```
   class GUIContainer implements GUIElement {
       public Vector<GUIElement> elements = new Vector<GUIElement>();
       public void paint(){
           for (GUIElement e : elements)
               e.paint();
       }
       // möglich, aber nicht nötig: add und remove für Elemente
   }
   ```
   ```

Aufgabe 14-8: OCL, 18 Punkte

1. **a)** `context Konto::einzahlen(betrag:double) post: kontostand = kontostand@pre + betrag`

   b) `context Konto::auszahlen(betrag:double) pre: kontostand + dispolimit >= betrag`

   c) `context Person inv: beschäftigungsstelle->notEmpty() implies bankverbindung->exists(b | b.dispolimit >= 500.0)`

   d) `context Filiale inv: kunden->forAll(k | k.bankverbindung->size > 0)`

   e) `context Filiale inv: mitarbeiter->forAll(m | self.kunden->exists(k | k = m))`

   f) `context Filiale inv: konten->forAll(k | self.konten->forAll(k2 | (k.blz = k2.blz) and (k.kontoNr = k2.kontoNr implies k = k2)))`

2. Eine Person hat Konto bei genau einer Filiale, aber beliebig viele (also 0 möglich als auch bei verschiedenen anderen Filialen) Bankverbindungen. Vermutlich soll `Person::filiale` die Multiplizität 0..* haben.

Aufgabe 14-9: Testfallauswahl, 10 Punkte

Betrachten Sie den Test einer Anmeldefunktionalität für ein Prüfungsverwaltungssystem wie in der Übung besprochen: Ein Student gibt Benutzername und Passwort ein und wird dann für eine Sitzung zugelassen oder abgelehnt.

1. Testfall-Notation: (Benutzername, Passwort, Ergebnis) mit Benutzername aus {Be, Bn}
   Be = existierender Benutzername
   Bn = nicht existierender Benutzername

   Passwort aus {Pr, Pf, Pb}
   Pr = zu Be passendes Passwort
   Pf = zu Be nicht passendes Passwort
   Pb = beliebiges, existierendes Passwort ungleich Pr

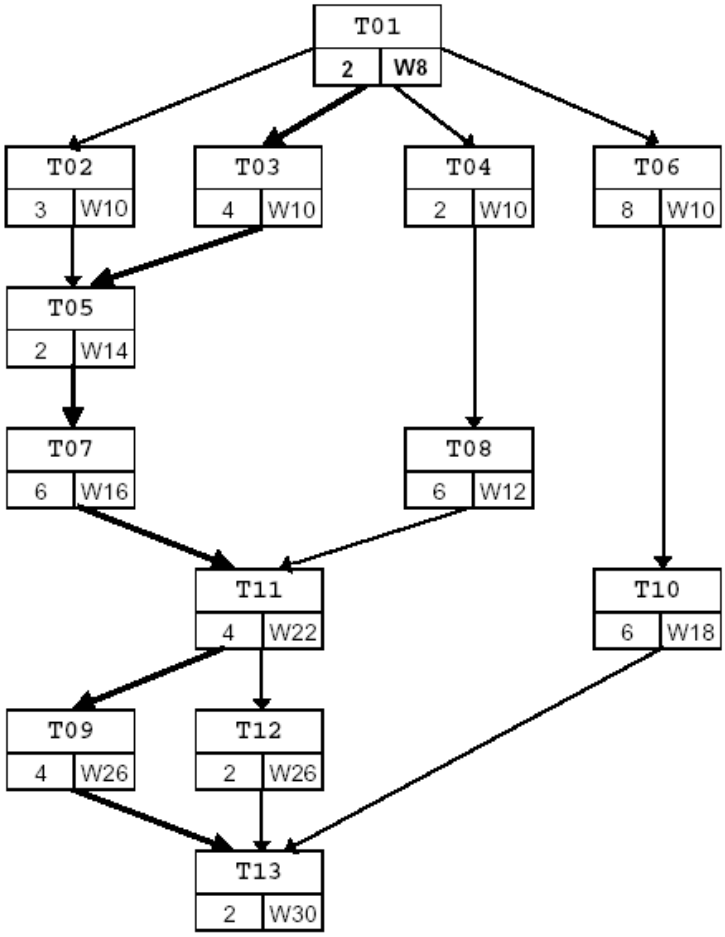
   Ergebnis aus {Zulassung, Ablehnung}

2. A (Be, Pr, Zulassung)
   B (Be, Pf, Ablehnung)
   C (Bn, Pb, Ablehnung)
   D (Be, Pb, Ablehnung)
   Man könnte auch auf leere Benutzernamen und Passwörter testen.

3. A und B

Aufgabe 14-10: Projektplanung, 20 Punkte

1.



2. 6 Wochen Pufferzeit haben die Aufgaben T06 und T10

3. T07

Aufgabe 14-11: Überschlagsrechnung, 15 Punkte

1. Wenn man etwas in Auftrag gibt, wird man niemals unter einem Arbeitstag abgerechnet bekommen, also 8 Stunden, jede kostet etwa 50 Euro, ergibt 400 Euro. Der eigentliche Programmieraufwand (inkl. Qualitätskontrolle!) wird vielleicht bei 4 oder 5 Stunden liegen. Es bedarf aber einiges an Vor- und Nachaufwand.

Manuell: Die Datei hat 400 Datensätze mit je 20 Ziffern, also 8000 Zeichen. Mit etwas Übung schafft man 100 Zeichen pro Minute, also dauert es 80 Minuten = 1,3 Stunden. Eine Kontrolle wird doppelt so viel Zeit kosten, also 3 Stunden. Die Qualifikation für eine solche Arbeit ist deutlich geringer und auch im Haus erledigbar, sagen wir 40 Euro, ergibt 120 Euro, also weniger als 20% der Kosten.

2. Wenn eine solche Konvertierung mehrmals nötig ist, rentiert sich eine programmierte Lösung schnell. Zudem ist anzunehmen, dass eine solche weniger unentdeckte Fehler verursacht.

Aufgabe 14-12: Rundumschlag, 5 Punkte

Hier nur ungeordnete Stichworte und Ideen: Agiler Prozess, Testautomatisierung, gute Dokumentation und guter Stil des Codes, funktionierendes Konfigurationsmanagement, dokumentierter Entwurf. Es dürften noch einige mehr vorhanden sein.