UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Software Engineering Wintersemester 2015/16
Prof. Dr. Martin Leucker, Torben Scheffel 21.03.2016

Klausur Software Engineering

Aufgaben und Punkte

Die Bearbeitungszeit der Klausur umfasst 90 Minuten. Es gibt 10 Aufgaben mit insgesamt 100 zu erreichenden Punkten.
Notieren Sie Ihre Lösungen wenn möglich direkt auf dem Aufgabenblatt. Sollte der Platz nicht ausreichen, verwenden Sie zusätzliche Blätter, die Ihnen gestellt werden. Benutzen Sie jede Seite der zusätzlichen Blätter nur für genau eine Aufgabe und notieren Sie Ihren Namen und Ihre Matrikelnummer am oberen Rand des Blattes.
Rechts oben auf jeder Seite der Klausur stehen die Punkte für eine gesamte Aufgabe. Die in Klammern gesetzten Zahlen dahinter geben die Punkte für die einzelnen Teilaufgaben an, von links nach rechts, jeweils beginnend mit Teilaufgabe a).
Wenn Sie in einer Aufgabe Lösungen ankreuzen müssen, so erhalten Sie für jedes richtig gesetzte Kreuz Punkte und für jedes falsch gesetzte Kreuz werden Ihnen Punkte abgezogen. Wenn Sie kein Kreuz setzen bekommen Sie weder Punkte, noch werden Ihnen Punkte abgezogen. Die genauen Punktzahlen stehen dabei an jeder Aufgabe, bei der Sie etwas ankreuzen müssen, dabei. Sie können in jedem Aufgabenteil aber nicht weniger als 0 Punkte bekommen.

Persönliche Daten

Notieren Sie im Folgenden Ihre persönlichen Daten. Notieren Sie Ihren Namen und Ihre Matrikelnummer außerdem auf jedem weiteren Blatt der Klausur am oberen Rand sowie auf jeden zusätzlichen Zettel, den Sie benutzen, wie vorher erläutert.

Vorname: _____

Nachname: _____

Geburtsdatum: _____

Matrikelnummer: _____

Studiengang: _____

Viel Erfolg!

Name: *Matrikelnummer:*

Aufgabe 1: Softwarequalität **6 Punkte (3/3)**

a) Nennen Sie drei Qualitätskriterien.

b) Beschreiben Sie kurz, was konstruktive Maßnahmen sind.

Name: *Matrikelnummer:*

Aufgabe 2: Softwareentwicklungsprozess **8 Punkte (2/4/2)**

a) Um welche Aspekte erweitert das Wasserfallmodell das Lebenszyklusmodell?

b) Nennen Sie zwei Vorteile und zwei Einschränkungen von Extreme Programming.

c) Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind. Es gibt $+0.5$ Punkte für korrekte Kreuze und -0.5 Punkte für falsche.

	Wahr	Falsch
Das Lebenszyklusmodell ist ein inkrementelles Vorgehensmodell.	☐	☐
Bei dem V-Modell steht jeder spezifizierenden eine testende Phase gegenüber.	☐	☐
Bei dem Spiralmodell sind mehrere Risikoanalysen vorgesehen.	☐	☐
Das Spiralmodell beinhaltet die Wartung.	☐	☐

Name: *Matrikelnummer:*

Aufgabe 3: Phasen der Softwareentwicklung **9 Punkte (3/4/2)**

a) Nennen Sie drei Methoden zur Aufwandsschätzung in der Planungsphase.

b) Erläutern Sie, was eine Algebraische Spezifikationen eines Datentyps beschreibt.

c) Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind. Es gibt $+0.5$ Punkte für korrekte Kreuze und -0.5 Punkte für falsche.

	Wahr	Falsch
Entity-Relationship-Modelle werden für den Datenbankentwurf verwendet.	☐	☐
Komponentendiagramme geben eine sehr grobe Sicht auf die Architektur des Systems.	☐	☐
Aktionen und Aktivitäten können in Objektdiagrammen vorkommen.	☐	☐
Verifikation stellt sicher, dass ein System die Anforderungen des Kunden erfüllt.	☐	☐

Name: *Matrikelnummer:*

Aufgabe 4: Management **8 Punkte (3/3/2)**

a) Erläutern Sie kurz, was das Vier-Ohren-Modell beschreibt.

b) Erläutern Sie die Unterschiede von konstruktiver und destruktiver Fehlerkultur.

c) Nennen Sie zwei typische Probleme von Legacy-Systemen.

Aufgabe 5: Gantt-Diagramm **9 Punkte (9)**

Gegeben ist folgendes Software-Projekt zur Entwicklung eines Twitter-Bots. Die Vorgangsdauer wird in Tagen angegeben und die Modellierung soll mit Tag 1 beginnen.

- Zum Projektstart wird zuerst die **Schnittstelle** zu Twitter erkundet (Dauer: 2 Tage).
- Danach können in beliebiger Reihenfolge ein **Entwurf** angefertigt (Dauer: 3 Tage), **Tests** geschrieben (Dauer: 2 Tage) und die **Nachrichten**, die der Bot verschicken soll, ausgedacht werden (Dauer: 2 Tage).
- Der Projektleiter hat danach den ersten Meilenstein **M1** angesetzt, da dies ein guter Zeitpunkt ist, um den Stand des Projektes zu evaluieren. Das heißt, die vorher genannten Aufgaben sollen bis zum Meilenstein **M1** nach Tag 6 fertiggestellt sein.
- Nach Meilenstein **M1** kann in beliebiger Reihenfolge mit der **Implementierung des Bots** (Dauer: 5 Tage) und der **Dokumentation** (Dauer: 3 Tage) begonnen werden. 1 Tag nach Meilenstein **M1** kann auch noch parallel zum Rest mit der Implementierung der äußeren **Schnittstelle des Bots** (Dauer: 6 Tage) begonnen werden, da der Kunde, der diese äußere Schnittstelle benutzen möchte, angekündigt hat, dass er vorher keine genaue Beschreibung liefern kann, was er dort haben möchte.
- Zum Abschluss wird dann noch ein Meilenstein **M2** nach Tag 13 festgelegt, bis zu dem alles fertig sein soll.

Entwickeln Sie ein Gantt-Diagramm, dass die oben genannten Schranken einhält. Ihnen stehen zwei Teams zur Verfügung, die parallel Arbeitspakete abarbeiten können. Markieren Sie, was von welchen Team bearbeitet wurde.

Aus Platzgründen sollten Sie die nächste, komplett freie Seite für die Lösung benutzen!

Name: *Matrikelnummer:*

Aufgabe 6: Petri-Netz **12 Punkte (12)**

Es gibt ein Lager von Holzbalken. Jedes mal, wenn dort ein Balken herausgenommen wird, entsteht sofort ein neuer. Außerdem darf erst ein neuer Balken herausgenommen werden, wenn der vorherige Balken komplett fertig verarbeitet wurde. Nachdem ein Balken herausgenommen wurde, wird er geteilt und eine Hälfte geht in die Fertigung des Körpers und die andere in die Fertigung der Beine. Der Körper wird einfach aus dem Balken ausgesägt und ist dann fertig. Bei der Fertigung der Beine entstehen aus der Hälfte des Balkens fünf Teile. Vier davon werden als Beine genommen und eines als Ersatzbein. Wenn die vier Beine fertig sind, werden diese mit dem Körper zu einem Schaukelpferd zusammengebaut. Danach wird dieses Schaukelpferd mit dem Ersatzbein zusammen als ein Produkt verpackt. Das Produkt bleibt dann im Produktlager liegen und der Balken gilt als komplett fertig verarbeitet. Das heißt, dass ein neuer Balken verarbeitet werden darf. Zu Beginn ist ein Balken im Lager und es liegen bereits zwei Ersatzbeine bereit.
Erstellen Sie ein Petri-Netz, das diese Ablaufsteuerung eines Fertigungssystems für Schaukelpferde modelliert. Dabei sollen Balken, der Körper, jedes Bein, das Schaukelpferd und das Produkt jeweils als Token modelliert werden. Beschriften und markieren Sie dabei die einzelnen Teile des Petri-Netzes so, dass die oben genannten Lager und Fertigungsschritte wiederzuerkennen sind.

Name: *Matrikelnummer:*

Aufgabe 7: Lineare Temporallogik **13 Punkte (2/6/2/3)**

Sei im Folgenden AP $= \{a, b, c\}$ und $\Sigma = 2^{\text{AP}}$.

a) Sei $\{\}\{b, c\}\{a\}\{a\}\{\}^\omega$ ein Lauf. Geben Sie eine LTL-Formel an, die
von diesem Lauf erfüllt wird und nicht semantisch äquivalent zu *true* ist.

von diesem Lauf nicht erfüllt wird und nicht semantisch äquivalent zu *false* ist.

b) Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind. Es gibt $+1$ Punkt für korrekte Kreuze und -1 Punkt für falsche.

	Wahr	Falsch
Der Lauf $\{b, c\}\{b\}\{b\}\{b, c\}^\omega$ erfüllt $c \wedge \mathcal{X}\,\mathcal{G}(b\,\mathcal{U}\,c)$.	☐	☐
$(\mathcal{G}\,a) \wedge (c\,\mathcal{U}(\mathcal{R}\,\mathcal{G}\,b))$ ist eine syntaktisch korrekte LTL-Formel.	☐	☐
Jeder Lauf, der $a\,\mathcal{U}\,b$ erfüllt, erfüllt auch $a \vee b$.	☐	☐
$(\mathcal{G}\,\mathcal{F}\,b) \vee (\mathcal{G}\,\mathcal{F}\,\neg b)$ ist semantisch äquivalent zu *true*.	☐	☐
Der Lauf $\{a, b, c\}\{a, b, c\}\{\}(\{c\}\{a, c\}\{a\})^\omega$ erfüllt $(\mathcal{G}(b \rightarrow (a \wedge c)) \wedge \mathcal{F}\,\mathcal{G}(c \rightarrow \mathcal{X}\,a))$.	☐	☐
Jeder Lauf, der $\mathcal{F}(a\,\mathcal{U}\,b)$ erfüllt, erfüllt auch $a\,\mathcal{U}\,b$.	☐	☐

c) Zeichnen Sie für die Formel $\mathcal{G}\,\mathcal{F}(a \wedge \mathcal{X}\,\neg a)$ ein Transitionssystem, bei dem **alle** unendlichen Läufe die Formel erfüllen.

d) Zeichnen Sie für die Formel $(\mathcal{X}(a\,\mathcal{U}(\mathcal{X}(b\,\mathcal{U}\,c)))) \wedge \mathcal{X}\,\mathcal{X}\,\neg c$ ein Transitionssystem, bei dem **alle** unendlichen Läufe die Formel erfüllen.

Name: *Matrikelnummer:*

Aufgabe 8: UML-Anwendungsfalldiagramm **12 Punkte (12)**

Erstellen Sie ein UML-Anwendungsfalldiagramm für das folgende System einer Kasino-App.

- Ein normaler Nutzer kann Poker, Roulette und Blackjack spielen. Bei allen dreien kann er vorher eingeben, dass er um Echtgeld spielen möchte. Andernfalls startet die App automatisch ein Spiel ohne Echtgeld. Wenn der Nutzer um Echtgeld spielen möchte, muss er seine Kreditkartendaten eingeben.
- Ein normaler Nutzer kann seinen Spielernamen ändern.
- Ein normaler Nutzer kann sich das gewonnene Geld auszahlen lassen. Dazu muss er ein Konto angeben.
- Ein normaler Nutzer kann ein Premiumupgrade erwerben, so dass er zum Premiumnutzer wird und 10% Rabatt auf alle Echtgeldausgaben erhält. Um das Premiumupgrade zu erwerben muss er vorher seine Kreditkartendaten eingeben.
- Ein Premiumnutzer kann alles tun, was ein normaler Nutzer auch kann, außer ein Premiumupgrade erwerben.
- Ein Premiumnutzer kann an Turnierspielen teilnehmen. Dabei kann er Geld einzahlen, um an Turnieren mit mehr Gewinn teilzunehmen. Ansonsten spielt er Turniere, für die er sich umsonst anmelden kann, die aber nur ein kleines Preisgeld haben.
- Ein Administrator kann die selben Aktionen durchführen wie ein normaler Nutzer und ein Premiumnutzer.
- Ein Administrator kann in den Wartungsmodus gehen. Wenn er im Wartungsmodus ist, kann er die Server der App neu starten. Um in den Wartungsmodus zu kommen, muss er sich vorher authentifizieren.
- Ein Testnutzer kann das Selbe tun wie ein normaler Nutzer und zusätzlich noch einen Feedbackbogen ausfüllen.
- Ein Entwickler-Testnutzer kann das Selbe tun wie ein normaler Nutzer, ein Premiumnutzer und ein Testnutzer.

Aus Platzgründen sollten Sie die nächste, komplett freie Seite für die Lösung benutzen!

Aufgabe 9: UML-Klassendiagramm **10 Punkte (10)**

Erstellen Sie für den folgenden Code ein UML-Klassendiagramm. Nutzen Sie dabei wann immer möglich Assoziationen, Aggregationen und Kompositionen. Vermeiden Sie Redundanzen.

```
public interface ExamExercise {

  public void print();
}

public class DesignPatternExercise implements ExamExercise {

  public ClassDiagramExercise classDiagram;
  private Exam exam;

  @Override
  public void print() { }
}

public abstract class DrawingExercise implements ExamExercise {

  private String text = "";
  protected Code c;

  @Override
  public void print() { }

  public abstract String buildFromCode();
}

public class ClassDiagramExercise extends DrawingExercise {

  private ExamExercise subtask;

  @Override
  public String buildFromCode() { return "Class Diagram building finished!"; }
}

public class Code { }

public class Exam {

  protected DesignPatternExercise patternExercise;
  private ExamExercise[] otherExercises;

  public Exam(ExamExercise[] otherExercises) {
    this.otherExercises = otherExercises;
  }
}
```

Name: *Matrikelnummer:*

Name: *Matrikelnummer:*

Aufgabe 10: Design-Pattern **13 Punkte (3/5/5)**

a) Nennen Sie drei Design-Pattern außer dem Model-View-Controller Pattern.

b) Erläutern Sie das Model-View-Controller Pattern. Welche Elemente gibt es und wie sollen Programme nach dem Pattern aufgebaut sein?

c) Erläutern Sie, warum Anti-Pattern bei der Softwareentwicklung vermieden werden sollten. Nennen Sie außerdem zwei negative Effekte zu denen es im Quellcode kommen könnte, wenn Anti-Pattern genutzt werden und beschreiben Sie, welche Auswirkungen diese Effekte auf den Softwareentwicklungsprozess haben.

Korrektur

Diese Seite wird von den Korrektoren ausgefüllt.

Aufgabe	Erreichte Punkte	Mögliche Punkte
1 Softwarequalität		6
2 Softwareentwicklungsprozess		8
3 Phasen der Softwareentwicklung		9
4 Management		8
5 Gantt-Diagramm		9
6 Petri-Netz		12
7 Lineare Temporallogik		13
8 UML-Anwendungsfalldiagramm		12
9 UML-Klassendiagramm		10
10 Design-Pattern		13
Gesamtpunktzahl		100

Note: _____