DR. MICHAEL EICHBERG

Einführung in Software Eingineering

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Wintersemester 2014/2015

Vorwort (dt.)

Dieses Dokument enthält – die Folien ergänzende – Informationen zur Vorlesung Software Engineering.

Chapter 1

Organisation (dt.)

Im Folgenden werden organisatorische Details bzgl. der Veranstaltung Einführung in Software Engineering dargestellt.

Bonusreglung

ÜBERBLICK

1. **Durch die regelmäßige Teilnahme an den Übungen kann ein Bonus für die Klausur erarbeitet werden**
2. **Mind. 30% der Übungspunkte müssen erreicht werden.**
3. **Der Bonus wird in Klausurpunkte umgerechnet und entspricht max. 50% einer kompletten Notenstufe**
4. **Dient lediglich der Verbesserung der Klausurnote/der Bonus kann nicht zum Bestehen verwendet werden**

Überblick

Der maximal zu erreichende Bonus entspricht 50% der Anzahl Klausurpunkte, die notwendig wären, um in der Klausur ein Ergebnis zu erreichen, dass eine komplette Notenstufe besser wäre. D.h.wenn durch die Übungen der volle Bonus erreicht wurde, und aus der Klausur heraus 60 Punkte geholt wurden und der Abstand zwischen zwei Noten (z.B. von 3,0 → 2,0) 10 Klausurpunkte beträgt, dann wäre der Bonus 5 Klausurpunkte und die Gesamtzahl der in der Klausur erreichten Punkte 65. Der Bonus kann nicht zum Bestehen der Klausur führen.

Erlangen des Bonuses

Durch die Bearbeitung der Übungsblätter kann immer eine bestimmte Anzahl an Übungspunkten erreicht werden. Die maximale Anzahl der Punkte, die pro Übungsblatt erreicht werden kann, ist immer auf dem Blatt vermerkt und liegt bei ca. 10 Punkten (ausgenommen Übung 0).

Die maximal zu erreichende Anzahl Übungspunkte ergibt sich somit aus der Summe der Übungspunkte über alle bewerteten Übungsblätter. Um ein Bonus für die Klausur zu erreichen, müssen mind. 30% der Übungspunkte erreicht werden.

Verrechnung mit der Klausur

Der Anteil der erreichten Übungspunkte wird in Klausurpunkte umgerechnet und auf die erzielten Klausurpunkte aufaddiert. Die Gesamtzahl der Klausurpunkte

bestimmt dann die Endnote. D.h. wenn in der Klausur zum Beispiel zehn Punkte mehr erreicht werden müssen, um sich eine komplette Notenstufe zu verbessern (2,0 → 1,0), und im Rahmen der Übungen 90% der maximal möglichen Übungspunkte erreicht wurden, dann ist der effektive Bonus 0,9 * 5 = 4, 5 Klausurpunkte.

Es ergibt sich somit folgende Berechnungsvorschrift: Anteil der erreichten Übungspunkte * (0,5 * Anzahl der Klausurpunkte, die notwendig sind, um eine Note, die eine volle Notenstufe besser ist, zu erreichen) = zusätzliche Klausurpunkte

Klausurpunkte	Note
> 80	1,0
77-79,5	1,3
73-76,5	1,7
70-72,5	2,0
...	...

Sie müssen mind.30% der Übungspunkte erreichen.

Gültigkeit des erreichten Bonus

Der Bonus ist gültig für das aktuelle Wintersemester und das darauf folgende Sommersemester. Der Bonus verfällt insbesondere auch dann nicht, wenn die Klausur am Ende des Wintersemesters nicht bestanden wird.

Material/Quellen

- Folien zur Vorlesung Software Engineering
  http://stg-tud.github.io/eise/
- Weiterführende Quellen
  - SE-Radio
    http://www.se-radio.net
  - ACM Queue
    http://queue.acm.org
  -

Chapter 2

Object-Oriented Thinking

Next...

Chapter 3

Klausur (dt.)

Im Folgenden finden Sie Informationen über die wesentlichen Bestandteile der Klausur.

Section 1

Testen und Testabdeckung

Eine Beispielaufgabe zum Thema Testen und Testabdeckung wäre:

Die folgende Java-Implementierung einer Methode, zur Partitionierung einer Liste (hier ein Array) von double Werten, soll getestet werden.

```
int partition(double[] a, int left , int right) {
    int i=left-1;
    int j = right;
    while ( true ) {
         while (a[++i] < a[right]) ;
         while (a[right] < a[--j]) if (j == left) break ;
         if (i >= j) break ;
         swap(a, i, j);
         swap(a, i, right);
    return i ;
}
```

Bewerten Sie die Testabdeckung des Testfalls a = {12.0,8.3}, left=0, right = 1 in Hinblick auf Anweisungsabdeckung (statement coverage). Geben Sie für jede Anweisung an, ob diese durch den Testfall abgedeckt wird.

Section 2

Design Patterns

Fragen zum Thema Design Patterns fallen in die folgenden Kategorien:

- Ein Stück Code oder ein UML Diagramm ist gegeben und Sie müssen erkennen welche Variante eines Design Patterns umgesetzt wurde und welche Klassen welche Rollen inne haben.
- Geben ist ein bestimmtes Designproblem und Sie müssen entscheiden welches Design Pattern Sie zur Lösung einsetzten wollen. Darüber hinaus ist es oft erforderlich die Lösung – unter Verwendung des vorgeschlagenen Patterns – zu skizzieren.
- Es gibt ein partielles Design und Sie müssen dieses erweitern.

Section 3

Use Cases

Es gibt typischerweise eine Aufgabe, bei der ein einfaches Anwendungsfalldiagramm zu erstellen ist. Darüber hinaus gibt es meist eine Aufgabe, die das Erstellen eines “fully-dressed” Use Cases verlangt.

Section 4

UML

Es gibt meist eine Aufgabe, im Rahmen derer ein Klassendiagramm zu erstellen oder zu übersetzen ist.

Section 5

Software Design

Aufgaben zum Thema Software Design erfordern meist die Beurteilung eines gegebenen Designs (unabhängig von konkreten Design Patterns oder Idiomen).

Eine Aufgabe wäre:

Geben ist folgendes Design:

```
class Student
class BachelorStudent extends Student
class MasterStudent extends Student
```

Ist dieses Design sinnvoll? Begründen Sie Ihre Antwort! Wenn Sie nicht genügend Informationen haben sollten, um eine Entscheidung treffen zu können, dann geben Sie an wovon Ihre Entscheidung ggf. abhängt.

Section 6

Wissensfragen

Es wird eine Aufgabe mit Wissensfragen geben.

Beispiele:

- Benennen Sie drei allgemeine Entwurfsprinzipien. Begründen Sie für jedes Prinzip, warum es sich um ein allgemeines Entwurfsprinzip handelt und warum es nicht ausschließlich für die Entwicklung objektorientierter Programme gilt.

- ...