Prüfung 13 Oktober 2011, Fragen und Antworten - WK, summer

Einführung in die Softwaretechnik (IN0006) (Technische Universität München)

Wiederholungsklausur zur Veranstaltung „Einführung in die Softwaretechnik“

Lehrstuhl für Informatik 19 (sebis), 13. Oktober 2011, Wintersemester 2011/2012

Nachname:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Wichtige Hinweise

- Füllen Sie auf diesem Blatt gut lesbar die obigen Felder (Nachname, Vorname etc.) aus.
- Notieren Sie auf jeder Seite des Arbeitspapiers gut lesbar Ihren Namen und die laufende Seitennummer!
- Es stehen 120 Minuten Bearbeitungszeit zur Verfügung. Maximal zu erringen sind 74 Punkte.
- Für korrekte Lösungsansätze werden auch dann Punkte vergeben, wenn das Endergebnis fehlerhaft ist oder fehlt. Erläutern Sie daher Ihre Lösungswege möglichst genau!
- Bei einem Täuschungsversuch wird die Klausur mit 5,0 bewertet.

Erlaubte Hilfsmittel

- Ein beidseitig handbeschriebenes DIN A4 Blatt.

Aufgaben

Teil 1: Wissensfragen

1. **Erläutern Sie den Unterschied zwischen einer *strengen* und einer *offenen Schichtenarchitektur*. Nennen Sie für jede dieser beiden Alternativen einen Vorteil. (3 Punkte)**
2. **Erläutern Sie die Idee des *Composite-Patterns* und geben Sie ein UML 2.0 Klassendiagramm an, das alle wesentlichen Element enthält und benennen Sie diese entsprechend. (6 Punkte)**
3. **Erläutern Sie, was man in der Softwaretechnik unter *Refactoring* versteht und nennen Sie zwei Ziele von *Refactoring*. Erläutern Sie außerdem, wie das Refactoring „Pull up field" funktioniert (4 Punkte)**
4. **Erläutern Sie die Idee von *CRC-Cards* und wie sie eingesetzt werden. Nennen Sie je eine *Stärke* und eine *Schwäche* dieser Methode. (4 Punkte)**
5. **Erklären Sie die beiden Softwarequalitätsmetriken „depth of inheritance tree" und „number of children". Erläutern Sie für beide was ein hoher Wert der jeweiligen Metrik bedeutet und ob ein hoher Wert problematisch ist. Begründen Sie Ihre Aussagen. (5 Punkte)**
6. **Erläutern Sie den Unterschied zwischen *funktionalen* und *nicht-funktionalen* Anforderungen und geben Sie *jeweils eine* Beispiel-anforderung für eine Überwachungssoftware, die heimlich Daten von privaten PCs an einen Regierungsserver sendet, an. (3 Punkte)**
7. **Nennen Sie vier Elemente des Konfigurationsmanagementprozesses, wie er in der Vorlesung vorgestellt wurde. Geben Sie drei Arten von Artefakten an, die dem Konfigurationsmanagementprozess unterliegen und erläutern Sie, welche Rolle eine Konfigurationsdatenbank in diesem Zusammenhang spielt. (5 Punkte)**
8. **Grenzen Sie Standardsoftware und Individualsoftware voneinander ab. Nennen Sie für beide Arten von Software jeweils Nachteile für das einsetzende Unternehmen. (4 Punkte)**

Teil 2: Verständnis- und Modellierungsfragen

1. Testing (22 Punkte)

Folgender Code soll getestet werden:

```
1   public int div(int dividend, int divisor) {
2      int result = 0;
3      if (divisor > 0 & dividend >= 0) {
4         if (divisor == 1) {
5            result = dividend;
6         } else if (dividend > 0) {
7            while (dividend >= divisor) {
8               dividend = dividend - divisor;
9               result++;
10           }
11        }
12        System.out.println("ok");
13     } else {
14        System.out.println("fehler");
15     }
16     return result;
17  }
```

Folgende Testfälle sind vorgegeben:

Parameter ‚dividend'	Parameter ‚divisor'	Ergebnis	Ausgabe
23	4	5	ok
-5	-5	0	fehler
1234	1	1234	ok

A) Erstellen Sie zunächst einen *Kontrollflussgraphen* für die Methode „div". (14 Punkte)

B) Stellen Sie fest, ob mit den gegebenen Testfällen vollständige *Anweisungsüberdeckung* erreicht wird. (1 Punkt)

C) Fügen Sie einen neuen Testfall hinzu, der entweder die *Anweisungsüberdeckung* oder die *Zweigüberdeckung* verbessert. Geben Sie an, welche Überdeckung verbessert wurde und ob noch weitere Verbesserungen der Überdeckungen möglich sind. (4 Punkte)

D) Begründen Sie, warum mit den bestehenden Testfällen keine vollständige *Pfadüberdeckung* erreicht ist. Lässt sich durch das Hinzufügen zusätzlicher Tests vollständige *Pfadüberdeckung* erreichen? Falls ja, geben Sie die entsprechenden Testfälle an, falls nein, begründen Sie Ihre Antwort. (3 Punkte)

2. Konzeptuelle Modellierung (18 Punkte)

Erstellen Sie ein konzeptuelles Klassendiagramm, das die für die Entwicklung eines Softwaresystems für die Fluggesellschaft EIST-Air relevanten Konzepte beinhaltet. Die Anforderungen an das System sind in folgender Beschreibung festgehalten:

Das Softwaresystem der Fluggesellschaft EIST-Air soll alle Mitarbeiter und Kunden sowie alle angeflogenen Flughäfen und die Flüge erfassen. Desweiteren können Kunden über das System Flüge buchen.

Zu jedem Mitarbeiter werden Vorname, Nachname und das Einstellungsdatum gespeichert. Mitarbeiter sind entweder Piloten, Flugbegleiter oder Bodenpersonal. Für Piloten wird außerdem die gesamte Zahl ihrer Flugstunden gespeichert, wobei auch angebrochene Stunden berücksichtigt werden. Für Bodenpersonal der Flughafen, an dem der jeweilige Mitarbeiter arbeitet.

Ein Flughafen hat einen Namen und ein Kürzel (z.B. „MUC“ für den Münchner Flughafen).

Ein Flug gehört immer zu einer Flugserie, das sind alle Flüge mit derselben Flugnummer (z.B. fliegen die Flüge der Serie mit der Flugnummer „EA 2011“ jeden Tag um 8:00 Uhr von München nach Augsburg). In der Flugserie wird neben der Flugnummer auch die Abfluguhrzeit und die Flugdauer in ganzen Minuten erfasst. Es ist außerdem der Startflughafen und der Zielflughafen festgelegt.

Ein Flug findet an einem bestimmten Datum statt. Falls ein Flug ausfällt, wird dies im System erfasst. Wenn er verspätet ist, wird zum Flug die Verspätung in Minuten angegeben. Einem Flug ist immer genau ein Pilot und zusätzlich ein Copilot zugeordnet. Außerdem gibt es auf jedem Flug mindestens einen Flugbegleiter.

Kunden der Fluggesellschaft werden mit ihrem Vornamen und ihrem Nachnamen erfasst. Ein Kunde kann beliebig viele Flüge buchen. Zu jeder Buchung ist die Bezeichnung des Sitzplatzes und die Zahl der erworbenen Bonusmeilen angegeben.

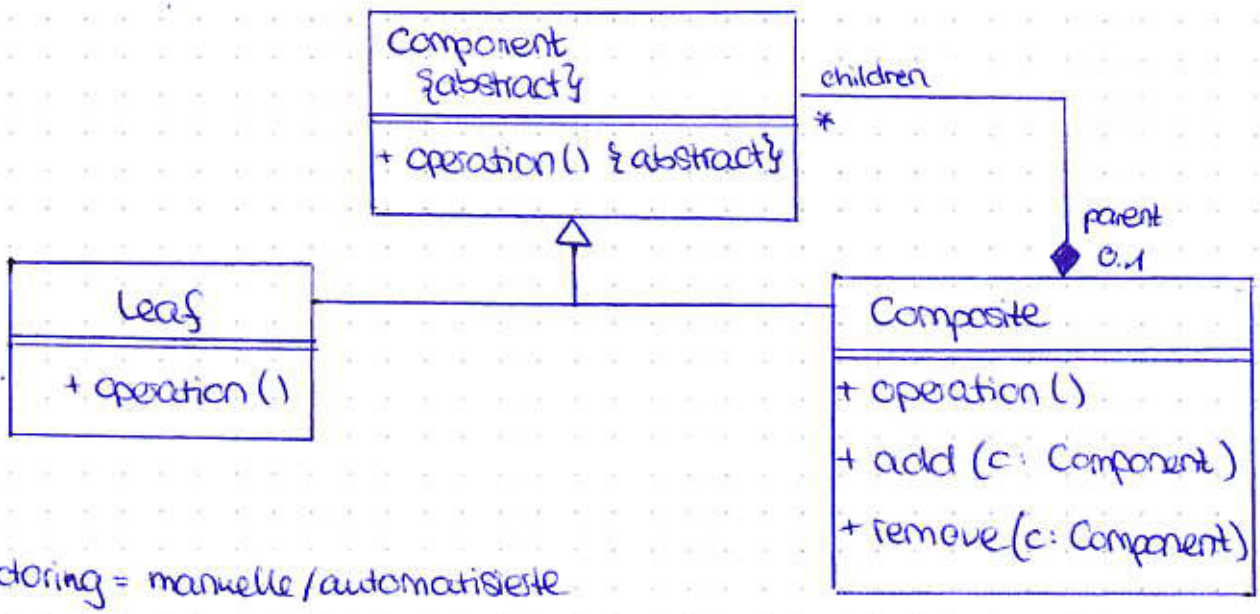
Beachten Sie, dass es sich um ein konzeptuelles Klassendiagramm handelt und dass möglicherweise nicht alle Inhalte der obigen Beschreibung sinnvoll in einem Klassendiagramm dargestellt werden können.

Klausur 2011 (Wiederholung)

Teil 1:

① Bei einer strengen Schichtenarchitektur kann eine Komponente nur auf die Komponenten aus der Schicht, die direkt unter ihrer Schicht liegt, zugreifen. Das ist im Sinne geringer Kopplung & erleichtert Änderungen. Bei einer offenen Schichtenarchitektur können die Komponenten auf alle darunter liegenden Schichten zugreifen. Ein Vorteil ist die Performanz, da dabei kein "Umweg" über andere Komponenten gegangen werden muss

② Das Composite-Pattern ermöglicht es Teil-Ganzes-Hierarchien zu veranschaulichen. Alle Elemente, einzelne Objekte sowie Kompositionen, können einheitlich behandelt werden. Dadurch können außerdem neue Elemente leichter eingefügt werden (als Unterklasse)



③ Refactoring = manuelle/automatisierte Strukturverbesserung unter Beibehaltung des Programmverhaltens
Ziele: verbessertes Codedesign → bessere Lesbarkeit, Fehleridentifikation
„Pull up field" → 2 Subklassen haben dasselbe Attribut. Hier kann das Attribut in die Oberklasse verschoben werden.

④ CRC Cards dienen zur Darstellung, welche Verantwortlichkeiten, Namen und Schnittstellen eine einzelne Klasse hat. So kann herausgefunden werden, ob eine Klasse zu viele Verantwortlichkeiten hat, und es hilft die genauen Schnittstellen einer Klasse zu finden. (= Stärke)
Diese Technik wird allerdings nicht von UML unterstützt und erfolgt meistens manuell → unübersichtlich, unstrukturiert → Ergebnisse sind also nicht digital. (= Schwäche)

⑤
DIT → misst die Länge des Weges von der untersten Klasse bis zu ihrer Wurzel
→ hoher DIT bedeutet eine hohe Komplexität des Codes und damit ein großer Aufwand für Entwicklung & Wartung
NOC → wie viele direkte Unterklassen hat eine Klasse.
→ hoher NOC deutet auf hohen Einfluss der Klasse. Häufige Wiederverwendung, hoher Testaufwand.

⑥
funktionale Anforderung
→ beschreiben Interaktion zwischen Kunde & System → können als Aktivitäten beschrieben werden
„wenn ein Nutzer eines privaten PCs eine neue Datei auf seinen Arbeitsplatz speichert, soll diese automatisch an die Überwachungssystem-datenbank gesendet werden"

nicht-funktionale Anforderung
→ beschreiben Eigenschaften des Systems → können als negative Zusicherungen formuliert werden
„Die Upload-dauer sollte unter 30s liegen" (Performanz)

⑦
- Definition, welche Artefakte erfasst werden
- Festlegung, wer für das Konfigurationsmanagement verantwortlich ist
- Beschreibung der Werkzeuge, die für das ↑ verwendet werden
- Beschreibung der Konfigurationsdatenbank

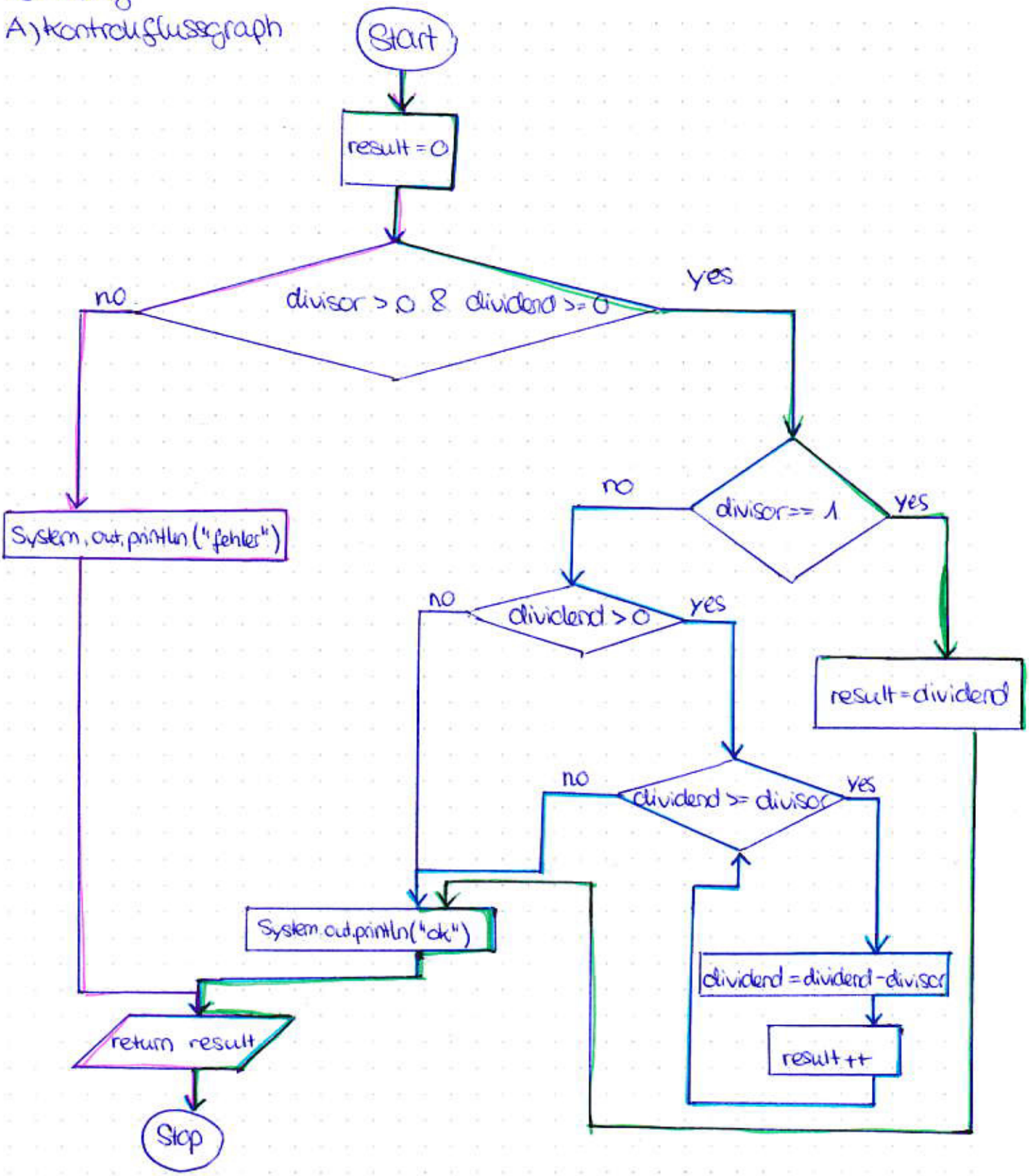
Artefakte: Projektpläne, Testsuiten, Programme → Auf der Konfigurationsdatenbank werden alle relevanten Daten für das Konfigurationsmanagement gespeichert. Sie hilft die Auswirkungen einer Änderung am System abzuschätzen und bietet dem Management Informationen zum Konfigurationsmanagementprozess.

⑧ Individualsoftware ist speziell für ein Unternehmen individuell entwickelt.
↳ maßgeschneidert. Nachteil: Wartung ist teuer
Standardsoftware ist für den gesamten Markt entwickelt und bildet im Kern alle Standardprozesse ab. Nachteil: nicht individuell auf alle Bedürfnisse angepasst, besitzt auch Funktionen, die nicht gebraucht.

Teil 2

① Testing

A) Kontrollflussgraph



B) Anweisungsüberdeckung:

Testfall 1 Testfall 2 Testfall 3 → Jeder Knoten wurde mit den 3 Testfällen abgedeckt → es liegt also 100% Anweisungsüberdeckung vor

C)

Parameter „dividend"	Parameter „divisor"	Ergebnis	Ausgabe
-2	2	0	ok

→ Zweigüberdeckung! → Nein, nicht möglich zu verbessern, da Kantenüberdeckung wegen Schleife unmöglich

D) Pfadüberdeckung ist wegen der Schleife nicht möglich.
Durch das Hinzufügen von Tests können noch mehr Pfade getestet werden, eine vollständige Überdeckung übersteigt aber jeglichen umsetzbaren Aufwand und Nutzen-Kosten-Verhältnis → dauert ewig.

② Konzeptuelles Klassendiagramm