

## Prüfung zu Software Engineering vom 27.1.2005

Allgemein: Der Prüfer, Professor Kaindl, stellte drei Kandidaten abwechselnd nach einander jeweils eine Frage, die dieser kurz beantwortete. Jeder Kandidat bekam insgesamt zwei Fragen. Die Atmosphäre war entspannt. Die Prüfung für drei Kandidaten dauerten im Schnitt ca. 15 bis 20 Minuten. Benotung: Der Notenschnitt der ersten 9 Kandidaten ergab: 1,3.

Fragen:

**1. Geben Sie eine Übersicht über die behandelten Kapitel der Vorlesung.**

Antwort des Kandidaten: Requirements Analyse, Entwurf und Architektur (Patterns, ...), Krit. Systeme, Rapide Prototyping, ...

**2. Worum geht es bei Software Requirement Engineering?**

A: Unterschied zw. Software- und User Requirements.

Zusatzfrage: OOAD-Modelle = User-Requirements?

A: Nein

**3. Modelle; Worum geht es dabei?**

A: Erfassen des Gesamt-Systems; Erfassung der Struktur der Daten;

ZF: Entity-Relationship?

A: Alle Beteiligten sollen das Modell verstehen.

ZF: Idealfall eines Requirements-Modells?

A: Komplette Problembeschreibung.

ZF: Reicht es, das Requirements-Modell zu haben?

A: Es schränkt zu sehr ein.

A des Prüfers: Man darf nur die Lösung von außen sehen. Die spezielle Lösung des Programms (innen) darf man noch nicht machen und sie dadurch einschränken.

**4. Wie hängen die Requirements mit der Architektur zusammen?**

A: Vorgaben über die Schichten vom Kunden (Kompatibilität);

**5. Wieso kann es zu Konflikten zwischen der Architektur und den Requirements kommen?**

A: Beispielanforderungen: Möglichst sicher und möglichst einfach; => ev.

Widersprüche und Konflikte; Modelle vorhanden => Welches passt am besten.

**6. Was versteht man unter Re-Engineering?**

A: z.B. Jahr-2000-Problem: Re...(?) => Straight-Forward

**7. Welche Vor- und Nachteile hat die natürliche Sprache beim Requirement Engineering?**

A: In einem Satz sollten nicht mehrere Requirements eingebaut werden. Kein Jargon; Nicht Stilisieren;

A d. Prüfers: Vollständige Genauigkeit ist nicht erreichbar.

**8. Wofür sind die Szenarien beim Requirements Engineering gut?**

A: Der Ablauf ist besser vorstellbar. Use-Cases sind Spezialfälle;

ZF: Unterschied zw. Szenario und Usecase?

A: Usecase ist ein spez. Ansatz für Szenario (?)

ZF: Kann man ein Szenario als Requirement verwenden?

A: Ja

AdP: Somit ist ein Szenario auch für Systemtests geeignet.

### **9. Safety vs. Security bei Requirements Engineering?**

A: Safety: z.B. medizinisch und chemisch => Vermeidung von Unfällen durch die Software.

A: Security: Hacker sollen von außen nicht auf die Software zugreifen können.

### **10.Blackboard?**

A: z.B. Spracherkennung: Hier liegen bislang wenig Forschungsergebnisse vor, auf die man aufbauen könnte. Tafel und jeder schreibt drauf (?).

ZF: Pattern? Das, was auf der Tafel steht steuert. Greifen alle darauf zu?

AdP: Moderator mit im Spiel

A: Moderator sagt: „Jetzt bist du dran“

### **11.Composite Pattern?**

A: Es geht um Objekte. Interface für Kapselung und Gleichbehandlung aller Objekte.

A: (Zeichnung der Klassenstruktur (Action ist Superklasse, A-Action und B-Action sind Subklassen, Composite-Action ist auch Subklasse , aber mit Rückkopplung zu Action) auf der Tafel)

ZF: Kann eine Composite-Action wieder eine Comp. Act. Enthalten?

A: ja, unendlich viele

### **12.Was versteht man unter Debugging?**

A: Testen

P: Nicht Richtig!

ZF: Erklären Sie den Unterschied zw. Testen und Debuggen.

A: Testen: Input und Output gegeben; Man schaut, was passiert. Debuggen: gefundene Fehler eingrenzen.

### **13.Was versteht man unter Pseudocode?**

A: Sinngemäß steht dort, was passieren soll. Schlüsselwörter + natürliche Sprache

### **14. Validieren und Verifizieren beim Testen (Definitionen)?**

### **15.Vererbung?**

A: Wenn Subklasse von zwei Superklassen erbt => Probleme (z.B. beide Superklassen haben eine Variable mit gleichem Namen. Welche Variable wird an die Subklasse vererbt?); => Multiple-...(?) hat sich nicht durchgesetzt.

### **16.Kritische Systeme und Risiken?**

ZF: Was sind Risiken?

A: Zustand den es gibt, der jedoch nicht eintreten darf.

ZF: Was ist in diesem Zusammenhang wichtig?

A: 1. Wahrscheinlichkeit des Eintretens, 2. Schaden der beim Eintreten entstehen kann.

### **17.Was versteht man unter Architektur?**

A: Zerlegen in Schichten, Subsysteme und Komponenten;

ZF: Genügt eine einfache Zerlegung?

ZF: Wodurch erfüllt Software die Funktionalität?

A: Durch das Ablaufen des Programms. Zustands-Veränderungen mit der Zeit.

AdP: Zustände der Datenflüsse bei der Architektur sind notwendig!