Übungen zur Vorlesung

Softwaretechnologie

-Wintersemester 2013/2014-

Achtung: Privat (von den Tutoren) erstelltes Dokument!

Fragen zur Klausurvorbereitung

Die Fragen auf diesem Zettel sollen Ihnen helfen, den eigenen Wissenstand einzuschätzen. Vor der Klausur sollten Sie in der Lage sein den Großteil dieser Fragen beantworten zu können. Sie bilden einen groben Querschnitt über wichtige Themen der Vorlesung, sind aber KEINE komplette Übersicht über die relevanten Themen! Es handelt sich lediglich um eine mehr oder wenige willkürliche Auswahl, die Sie beantworten können sollten. Die Fragen und die Art der Fragen stehen in keiner Beziehung zu den Aufgaben der Klausur. Die Quellen für die Klausur sind die Vorlesungsfolien und die Übungsblätter. Die Testate bieten einen Vorgeschmack, wie die Prüfungsklausur in etwa aussehen könnte. Nutzen Sie auch die Musterlösungen der Übungsblätter zur Vorbereitung.

1. Wofür benötigt man SCM?
2. Was sind CVS, SVN, GIT und ClearCase? Was sind die konzeptionellen Ideen, welche Möglichkeiten bieten sie, und wie unterscheiden sie sich?
3. Nenne die wichtigsten Funktionen, die SVN und GIT anbieten (Check-Out, Commit…)
4. Was sind Tags und Branches?
5. Was sind Konflikte bei SVN/GIT? Wann treten sie auf? Wie löst man sie auf?
6. Was ist das Konzept hinter verteilter Versionskontrolle? Wo liegen Vor- und Nachteile?
7. Was versteht man in der Softwareentwicklung unter „Issues“?
8. Wozu dienen Jira und Mylyn? Wie spielen SVN, Jira und Mylyn zusammen?
9. Wofür benötigt man UML?
10. Ist UML sprachspezifisch?
11. Wie sehen Klassendiagramme aus, und was modellieren sie?
12. Was ist der Unterschied zwischen Assoziation, Aggregation und Komposition?
13. Was sind abstrakte Klassen und Interfaces? Wie werden sie in Klassendiagrammen dargestellt?
14. Wie werden Assoziationen der Multiplizität 0..1 und 0..* in Code realisiert?
15. Was ist der Unterschied zwischen Klassen- und Objektdiagrammen in Hinsicht auf Konzept und UML-Syntax?
16. Wie wird in UML die Vererbung (Java: *extends*) von Klassen, und Realisierung (Java: *implements*) von Interfaces dargestellt?
17. Wie sehen Sequenzdiagramme aus, und was modellieren sie?
18. Sequenzdiagramm: Was unterscheidet synchrone von asynchronen Nachrichten? Was kommt in normalen Programmen hauptsächlich vor?
19. Sequenzdiagramm: Wie werden Schleifen und Verzweigungen (if-else) dargestellt?
20. Wie sehen Kommunikationsdiagramme aus, und was modellieren sie?
21. Wie sehen Aktivitätsdiagramme aus, und was modellieren sie?
22. Aktivitätsdiagramm: Wie werden Entscheidung/Vereinigung und Parallelisierung/Synchronisierung dargestellt?
23. Aktivitätsdiagramm: Wo ist der Unterschied zwischen den Knoten Ablaufende und Endzustand?

24. Wie sehen Zustandsdiagramme aus, und was modellieren sie?
25. Zustandsdiagramm: Beschreibe die Syntax von Transitionen/Zustandsübergängen.
26. Zustandsdiagramm: Wie wird Parallelisierung/Synchronisierung dargestellt?
27. Zustandsdiagramm: Wie funktionieren Unterautomaten? (Ein-/Ausstiegspunkte?)
28. Zustandsdiagramm: Was sind Entry- und Exit-Aktivitäten eines Zustandes, und wie werden sie modelliert? Wann werden diese Aktivitäten ausgelöst?
29. Zustandsdiagramm: Wo ist der Unterschied zwischen Endzustand und Terminierungsknoten?
30. Welchem Zweck dient die Anforderungserhebung?
31. Was sind funktionale/nichtfunktionale Anforderungen und Nebenbedingungen?
32. Was unterscheidet Szenarien und Use Cases?
33. Wie sehen Use Case Beschreibungen aus?
34. Wie sehen Anwendungsfalldiagramme (Use Case Diagramm) aus, und was modellieren sie?
35. Use Case Diagramm: Was ist der Unterschied zwischen Include-, Extends- und Generalisierungsbeziehungen?
36. Use Case Diagramm: Was sind Extension Points und wie verwendet man sie?
37. Was ist Abbotts Textanalyse? Wofür wird sie verwendet?
38. Was ist das Domain Object Model (DOM)?
39. Welche Schritte werden in der Anforderungserhebung behandelt? Wie geht es danach in der Anforderungsanalyse weiter?
40. Was unterscheidet das Use Case Modell vom Analyse Modell? Welche Personen sind jeweils bei der Erstellung involviert?
41. Beschreiben sie die drei Analysetypen Boundary, Entity, Controller. Welche Aufgaben haben sie, und wie können sie im BEC-Modell miteinander in Verbindung stehen?
42. Was sind Design Patterns und wofür benötigt man sie?
43. Nennen Sie mindestens vier Design Patterns, und erklären sie ihren Aufbau, Funktion und Anwendungskontext. Überlegen sie sich auch jeweils ein Anwendungsbeispiel.
44. Welche zwei Varianten des Observer-Patterns kennen sie?
45. Was ist der Unterschied zwischen Factory und Abstract Factory?
46. Was ist der Zweck des Visitor Patterns?
47. Was wird im Schritt Systementwurf festgelegt? Welche Schritte werden dafür unternommen?
48. Was sind Dienste und Subsysteme?
49. Was versteht man unter „Kohärenz“ und „Kopplung“? Welche Ziele verfolgen Sie diesbezüglich beim Design der Subsystem-Struktur?
50. Welchen Zweck hat das Facade-Pattern?
51. Was sind die Ideen und Vorteile von Komponenten-basierter Softwareentwicklung?
52. Wie sehen Komponentendiagramme aus, und was modellieren sie?
53. Nennen Sie mindestens 3 Software-Architekturen und beschreiben Sie ihren Aufbau, sowie geeignete Anwendungskontexte.
54. Beschreiben Sie die Model-View-Controller (MVC) Architektur. Welche Analysetypen dürfen innerhalb der einzelnen Ebenen vorkommen?
55. Welche Ziele werden im Schritt Objektentwurf verfolgt?
56. Was sind CRC-Cards? Welchen Zweck erfüllen sie?
57. Was sind die drei Kernkonzepte von Design by Contract?
58. Was steht hinter dem Konzept „Split Objects“? Welche Patterns bezeichnet man als „Split Object Pattern“ und wieso?

59. Beschreiben Sie das Decorator-Pattern, erklären Sie seinen Aufbau, Funktion und Anwendungskontext. Überlegen Sie sich ein Anwendungsbeispiel.
60. Wo ist der Unterschied zwischen Strategy und State Pattern?
61. Wie kann man UML Klassendiagramme, die multiple Vererbung verwenden, so umstrukturieren, dass das Modell in Sprachen wie Java (ohne multiple Vererbung) realisierbar ist?
62. Was sind Assoziationsklassen und qualifizierte Assoziationen in UML?
63. Welche Aufgaben werden in dem Schritt „Anforderungserhebung“ bearbeitet?
64. Welche Aufgaben werden in dem Schritt „Anforderungsanalyse“ bearbeitet?
65. Welche Aufgaben werden in dem Schritt „Objektentwurf“ bearbeitet?
66. Welche Aufgaben werden in dem Schritt „Systementwurf“ bearbeitet?
67. In welcher Reihenfolge werden die obigen Schritte durchgeführt? Wieso?
68. Wie lauten die 7 Schritte des Test-Zyklus? Welche Schritte werden vom Tester, welche vom Entwickler bearbeitet? Welche sind automatisierbar?
69. Was sind Modultests (Unit Tests)?
70. Welche Aspekte werden von Black-Box und White-Box Tests überprüft? Wie funktionieren diese Tests?
71. Black-Box Tests: Wie werden Äquivalenzklassen zum Testen verwendet?
72. White-Box Tests: Wie sind Anweisungs-, Verzweigungs-, Schleifen-, und Pfadabdeckung definiert?
73. Was ist JUnit? Welche Vorteile hat Testing mit JUnit im Gegensatz zu Testing mit *System.out.print*-Ausgaben?
74. Wie sieht eine TestCase Klasse in JUnit 4 aus? Was bedeutet es, wenn eine Methode mit *@Test* oder *@Before* annotiert ist?
75. Was versteht man unter “Test-Driven Development”?
76. Was versteht man unter “Refactoring”? Welche Ziele verfolgt man damit?
77. Nennen Sie die drei in der Vorlesung vorgestellten Software-Entwicklung Prozessmodelle, und erkläre Sie sie. Wo liegen die Unterschiede, sowie Vor- und Nachteile?
78. Nennen Sie die Kernkonzepte von Agilen Methodologien.
79. Was ist Extreme Programming?
80. Wann und wo findet die SWT-Klausur statt?

Viel Erfolg bei der Klausur!
Wir sehen uns am 25.02 im HS X!