Information
Als Basis für diesen Fragenkatalog wurden, alle im VoWi bzw. Informatikforum verfügbaren Prüfungsangaben von 2010-2014 benutzt. Die Theoriefragen wurde mit bestem Gewissen anhand der Vorlesungsfolien des WS 13/14 sowie andere Ausarbeitungen beantwortet. Es besteht keine Garantie auf Richtigkeit bzw. Vollständigkeit der Antworten. Für die Vorlesungsprüfung vom 29.04.14 hat das Lernen dieses Fragenkatalogs sowie eine kurze Vorbereitung für den praktischen Teil ausgereicht um ein Sehr Gut zu erhalten.
P.S: Das Dokument enthält mit ziemlich sicherer Wahrscheinlichkeit Tipp- bzw.- Rechtschreibfehler

Fragen

1) Was versteht man unter "Stakeholder"? Nennen sie fünf typische Stakeholder in einem Softwareprojekt
2) Was versteht man unter stabilen und volatilen Anforderungen? Welche Auswirkungen haben diese auf das Softwaredesign
3) Was versteht man unter einer "Baseline" in der Analyse? Was passiert üblicherweise bei Änderungen (Changes) nachdem eine Baseline gesetzt wurde?
4) Wass versteht man unter funktionalen und nicht funktionalem Anforderungen. Nennen sie 3 Beispiele für nicht funktionale Anforderungen
5) Nennen und erklären Sie 6 Qualitätskriterien für Anforderungen
6) Beschreiben Sie Qualitätskriterien und Probleme beim Spezifizieren/Schreiben von Anforderungen. Nennen Sie Methoden um Vollständigkeit und Qualität der Anforderungen zu testen.
7) Was ist UML. Nennen sie in diesen Zusammenhang je 5 Verhaltens- und Strukturmodelle
8) Was ist Nachvollziehbarkeit (Traceability) im Anforderungsmanagement
9) Was versteht man unter "SOA"? Welche Eigenschaften zeichnen SOA aus? Welche Paradigmen werden dabei verfolgt?
10) Service-orientierte Architektur erklären. Was ist BPM? Zusammenhang SOA und BPM?
11) Wofür werden Softwarepatterns verwendet? Was ist der Unterschied zwischen Design- und Architekturpatterns? Beschreiben sie ein Architekur Pattern
12) Stairs und Fork SW-Architektur erklären
13) Wozu dienen Architekturmuster? Beschreiben Sie das Model View Controll Architekturmuster, und nennen sie zwei weitere Architekturmuster!
14) Erklären Sie die wesentlichen Unterschiede zwischen einem Framework und einer Bibliothek!
15) Wie werden Frameworks üblicherweise spezialisiert/angepasst? Wie geschieht die bei Whitebox Frameworks und Blackbox Frameworks?
16) Zusammenhang zwischen Kopplung und Kohäsion und Auswirkung auf die Wartung
17) Was versteht man unter kollektivem Codebesitz.Vorteile, Nachteile?
18) Was ist das Observerpattern+ Skizze
19) Erklären Sie das Test Driven Development anhand einer Skizze. Welchen Bezug zwischen automatisierten Tests und TDD gibt es?
20) Was versteht man unter Refactoring? Welche Methoden des Refactoring gibt es? Welche Rolle spielt Refactoring im TDD
21) Was ist der Unterschied zwischen Internationalisierung und Lokalisierung?
22) Wozu dient ein Loggin-Framework und warum sollte man nicht System.out.println() verwenden, um zu loggen?
23) Was sind Buildmanagementsysteme? Erklärung und 3 Beispiele.
24) Erklären sie den Unterschied zwischen lokaler, zentraler und dezentraler Versionskontrolle
25) Erklären sie den Unterschied zw. Optimistic Revision Control und Pessimistic Revision Control. Erklären sie diese Strategie anhand einer Skizze.
26) Was versteht man unter struktureller Testabdeckung und welche Bedingungsarten gibt es?
27) Erläutern Sie die Teststufen in der iterativen Softwareentwicklung. Erläutern sie in diesem Zusammenhang de Begriffe Stub und Treiber

28) Was versteht man unter Äquivalenzklassenanalyse bzw. Grenzwertanalyse. Was sind die Unterschiede
29) Warum prüft man Komponenten isoliert? Was für Arten von Isolation kennen Sie?
30) Was versteht man unter Migration. Nennen sie 3 Arten von Migration
31) Was versteht man unter Continous Delivery.?Prinzipien und Vorteile
32) Welche Typen der Wartung werden unterschieden? Erklären sie die unterschiedlichen Typen der Wartung
33) Beschreiben Sie die Wartungsmaßnahmen Software Reengineering und Reverse Engineering
34) Nennen und erklären sie die Delphi-Methode
35) Was ist ein GANTT-Diagramm:
36) Wozu dient das "3-Experten-Konzept"? Erklären Sie dieses.
37) Nennen Sie 3 Vorteile/Nachteile im Zusammenhang mit Teamwork
38) Erklären Sie die Meilenstein-Trendanalyse anhand des Diagramms.
45) Erklären Sie die Matrix zu Projektorganisation
46) Fragen aus Foliensatz 8
47) Erklären Sie den Begriff eXtreme Programming
48) Was sind Vorgehensmodelle und wozu sind sie gut?Beschreiben Sie das Spiralmodell von Boehm und zeichnen sie die Phasen ein. Welche Phasen gibt es? Beschreiben sie das Modell. Unterschied zwischen traditionellen und agilen Methoden erklären
49) Erklären sie die Grundcharakteristik, die wesentlichen Elemente und Rollen des Unified Process.
50) Was ist Scrum. Welche Elemente und Rollen spielen in diesem Zusammenhang eine Rolle?
51) V-Modell Beschreiben + Skizze wofür steht das XT im V-XT Modell und Ziele des XT Modells
52) Was sind Softwaremetriken in der Qualitätssicherung, warum werden sie verwendet? Geben Sie 3 Beispiele objektorientierter Metriken an.
53) Erklären Sie den Ablauf und die Rollen des Review-prozesses. Warum sind Reviews wichtig?
54) Wozu dient CMM? Nennen Sie die 5 Reifegrade!
55) Wozu dient SPICE? Nennen sie die 6 Reifegrade!
56) Was erklären Sie das Prinzip des Misuse-Case anhand eines Beispiels
57) Wie läuft eine SQL-Injection ab? Wie kann man sich davor schützen?

Fragen aus dem Foliensatz 2

1) Was versteht man unter "Stakeholder"? Nennen sie fünf typische Stakeholder in einem Softwareprojekt

Stakeholder= „Anteilhalter“ der ein Interesse im Projekt vertritt und somit eine Anforderungsquelle darstellt
Jede Stakeholdergruppe hat eine eigene Sicht auf das Projekt → es können Konflikte entstehen → Anforderungspriorisierung
Typsiche Gruppen sind z.B: Kunde, Geschäftsführung, Benutzer, Softwareentwickler, Projektleiter, Betrieb, Wartung

2) Was versteht man unter stabilen und volatilen Anforderungen? Welche Auswirkungen haben diese auf das Softwaredesign

- Stabil: Anforderungen die sich (voraussichtlich) nicht mehr oder nur mehr geringfügig ändern werden
- volatil: Anforderungen die sich (voraussichtlich) während der Anforderungsanalyse oder während des Betriebs des Systems ändern werden
- Auswirkungen
  - stabile Anforderungen können mit simplerem Designs (bzw. hard-wired) implementiert werden
  - für die Umsetzung volatiler Anforderungen benötigt man ein adaptives Systemdesign (soft-wired)
  - stabile Anforderungen können früher und detaillierter dokumentiert und modelliert werden
  - stabile Anforderungen können früh im Entwicklungsprozess implementiert werden

3) Was versteht man unter einer "Baseline" in der Analyse? Was passiert üblicherweise bei Änderungen (Changes) nachdem eine Baseline gesetzt wurde?

Ein echtes Einfrieren (Freeze) der Anforderungen ist nie möglich, trotzdem sind stabile Zustände in der Analyse unabdingbar. Weshalb man sogenannte Baseline einsetzt. Diese stellen ein System oder eine Spezifikation dar, welche bereits reviewed und überarbeitet wurde und nun als Basis für die weitere Entwicklung dient.

Nach dem Setzen einer Baseline werden bei Änderungen üblicherweise eine Change-Impact-Analyse und ein Re-Costing durchgeführt.

4) Wass versteht man unter funktionalen und nicht funktionalem Anforderungen. Nennen sie 3 Beispiele für nicht funktionale Anforderungen

- Funktional: legen fest was ein System können soll, also welche Funktionen/Services zur Verfügung stehen sollen. Liefern typischerweise einen Mehrwert für den Kunden bzw. Auftragsgeber.
- Nicht funktional: beschreiben Eigenschaften welche das System haben sollte. Sie sind in erster Linie Qualitätsanforderungen (wie „gut“ das System etwas können sollte)
- Beispiele:
  - Die Speicherung von Passwörtern soll nur verschlüsselt passieren!
  - Bei Kauf von Produkten sollen verschlüsselte Kanäle verwendet werden.
  - Eine Produkt-Abfrage soll nicht länger als 2 Sekunden in Anspruch nehmen.

5) Nennen und erklären Sie 6 Qualitätskriterien für Anforderungen

- Vollständig: Anforderungen müssen explizit beschrieben sein, es darf keine impliziten Annahmen eines Stakeholder über das zu System geben

- Atomar: Es darf nur eine Anforderung pro Anforderungs-ID beschrieben werden
- Verständlich: alle Stakeholder müssen die Anforderung unter vertretbarem Aufwand lesen und verstehen können.
- Identifizierbar: Jede Anforderung muss eindeutig identifizierbar sein (z. B. über eine ID).
- Notwendig: Klären ob die Anforderung unabdingbar ist.
- Nachprüfbar: Anforderungen sollten mit Abnahmetestfällen verknüpft werden, damit bei der Abnahme die Erfüllung überprüft werden kann

6) Beschreiben Sie Qualitätskriterien und Probleme beim Spezifizieren/Schreiben von Anforderungen. Nennen Sie Methoden um Vollständigkeit und Qualität der Anforderungen zu testen.

- Kriterien: siehe Frage davor.
- Probleme:
  - Verständlichkeit
  - Selbstverständlichkeit: Domain-Spezialisten erachten gewisse Anforderungen als impliziert
  - zu wenig Präzision
  - Vermischung von funktionalen und nicht funktionalen Anforderungen
  - Mehrdeutigkeit: Spezifikationen in natürlicher Sprache sind oft mehrdeutig
  - Überflexibilität: Beschreibungsart der Anforderung ist oft mehrdeutig interpretierbar
- Methoden: ?

7) Was ist UML. Nennen sie in diesen Zusammenhang je 5 Verhaltens- und Strukturmodelle

- Unified Modeling Language: eine grafische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software-Teilen.
- Verhaltensmodelle: Anwendungsfalldiagramm, Aktivitätsdiagramm, Zustandsdiagramm,Sequenzdiagramm, Kommunikationsdiagramm
- Strukturmodelle: Klassendiagramm, Paketdiagramm, Komponentendiagramm, Verteilunsdiagramm, Objektdiagramm

8) Was ist Nachvollziehbarkeit (Traceability) im Anforderungsmanagement

3 wesentliche Faktoren:

- Nachvollziehbarkeit der Quelle: Zusammenhang mit dem Stakeholder die Anforderung eingebracht hat
- Nachvollziehbarkeit zu anderen Anforderungen: Zusammenhänge zwischen interpendenten Anforderungen
- Nachvollziehbarkeit zu abhängigen Artefakten: Zusammenhänge der Anforderungen z. B. zu Elementen des Systemdesigns

Fragen aus dem Foliensatz 3

9) Was versteht man unter "SOA"? Welche Eigenschaften zeichnen SOA aus? Welche Paradigmen werden dabei verfolgt?

SOA= Service orientierte Architekturen
Eigenschaften:

- Business-driven (Geschäftsprozess-orienterte) Architektur
- Anwendungen werden durch mehrere Services gelöst
- Services in einer Anwendung sollen auch aus einer anderen Anwendung heraus aufgerufen werden
- Zusammenstellung von Services in bestimmten Abläufen zur Umsetzung von Geschäftsprozessen
- Services existieren in mehreren Abstraktionsebenen

- Beschreibt eine Architektur, keine Technologie

Paradigmen:
- Lose Kopplung: Niedrige Kopplung der Services ermöglicht hohen Reuse
- Abstraktion: Reduktion von Neuentwicklungen
- Standardisierung: erlaubt die Zusammenarbeit von Services
- Composable: Services könne zu Komponenten zusammengefasst werden.

10) Service-orientierte Architektur erklären. Was ist BPM? Zusammenhang SOA und BPM?

Im Service-orientiertem Entwurf wird ein Schritt in einem Geschäftsprozess extrahiert und gekapselt → dabei handelt es sich um einen sog. Buisiness Service. SOA sind buisiness-driven → sie sind an Geschäftsprozesse angelehnt.
Buisines Process Modelling (BPM) und SOA sind eng miteinander verknüpft. So werden Anwendungen maßgeblich von der Prozess-Modellierung beeinflusst.

11) Wofür werden Softwarepatterns verwendet? Was ist der Unterschied zwischen Design- und Architekturpatterns? Beschreiben sie ein Architekur Pattern

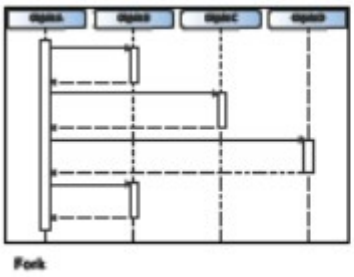
Patterns stellen prinzipiell allgemeine Lösungsstrategien für wiederkehrende Probleme in der Softwareentwicklung da. Sie sollten daher möglichst abstrakt und weitgehend technologie- unabhängig sein. Sie bieten wesentlichen Input in Entwurfsentscheidungen.

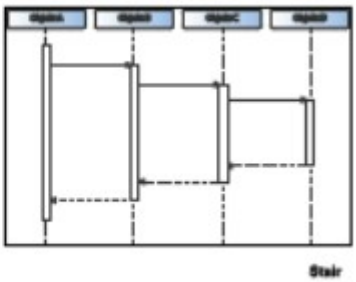
- Architektur-Patterns: dienen als Schablonen für konkrete Softwarearchitekturen. Sind also Muster auf architektonischer Ebene. Des Weiteren spezifizieren sie Systemweite, strukturelle Eigenschaften einer Anwendung und stellen so fundamentale Designentscheidungen dar.
- Design-Patterns: beschreiben wiederholt auftretende Strukturen von kommunizierenden Komponenten. Lösen ein allgemein definiertes Problem innerhalb eines bestimmten Kontext. Sind Patterns auf Abstraktionsebene unter Architekturpatterns.

mvc

12) Stairs und Fork SW-Architektur erklären

- Fork: zentrales Objekt übernimmt Kontrolle.
- Stair: Kapselung in Schritten.





13) Wozu dienen Architekturmuster? Beschreiben Sie das Model View Controll Architekturmuster, und nennen sie zwei weitere Architekturmuster!

Patterns stellen prinzipiell allgemeine Lösungsstrategien für wiederkehrende Probleme in der Softwareentwicklung da. Architektur-Patterns: dienen als Schablonen für konkrete Softwarearchitekturen. Sind also Muster auf architektonischer Ebene. Des weiteren spezifizieren sie Systemweite, strukturelle Eigenschaften einer Anwendung und stellen so fundamentale Designentscheidungen dar.
Beispiele: MVC, Pipe and Filters, Layered Architecture

MVC-Pattern:
Es existieren die Komponenten Model, View und Controller. Der Controller empfängt die Events und stellt darauffolgend die Views zusammen. Er verändert die Views und die Models gemäß dem eingetretenen Event. Die Views bekommen ihre Daten vom Model (sind somit abstrakt mit Platzhaltern für konkrete Werte). Somit verändert auch ein Model eine oder mehrere Views wenn sich die Daten ändern.
Dies ist sehr sinnvoll, wenn sich GUI's oft ändern. Die Daten und Funktionen werden in den Models gekapselt, und die Views sind nur für die konkrete graphische Abbildung der Daten verantwortlich (findet vor allem in der Web-Entwicklung seine Anwendung).

Fragen aus Foliensatz 4

14) Erklären Sie die wesentlichen Unterschiede zwischen einem Framework und einer Bibliothek!

- Framework ist ein Applikationsskelett welches vom Entwickler angepasst werden kann um Anforderungen optional umzusetzen.
- Eine Bibliothek bietet wiederverwendbare Funktionalität an, während ein Framework wiederverwendbares Verhalten bereitstellt.
- Methoden von Bibliotheken werden durch den eigenen Code aufgerufen, während bei der Verwendung von Frameworks der eigene Code durch das Framework aufgerufen wird
- Frameworks bestehen aus abstrakten Klassen und bieten Funktionalität in (teilweise) implementieren Methoden an. Diese Methoden können Funktionalität aus Bibliotheken beziehen.

15) Wie werden Frameworks üblicherweise spezialisiert/angepasst? Wie geschieht die bei Whitebox Frameworks und Blackbox Frameworks?

Zwei wesentliche Konzepte:
- Method Hooks: Abstrakte Methoden die vom Entwickler überschrieben werden um das gewünschte Verhalten zu erstellen
- Hot Spots: „Punkte“ an denen ein Framework spezialisiert wird.

Whiteboxframeworks:
- innere Struktur muss sichtbar sein um es spezialisieren zu können
- Spezialisierung erfolgt meist durch Vererbung

Blackbox Frameworks:
- enthält stabile Hot Spots
- Spezialisierung komponentenbasiert
  - Komponenten werden an Framework übergeben

16) Zusammenhang zwischen Kopplung und Kohäsion und Auswirkung auf die Wartung

Kopplung
- Maß der Abhängigkeit zwischen Klassendiagramm
- Hohe Kopplung zeugt von einem verbesserungswürdigen Programmierstil

Kohäsion
- Maß der Abhängigkeit innerhalb von Klassen
- Hohe Kohäsion wird meistens durch Methodengruppierungen nach Zweck gewährleistet

<u>Ziel:</u> Geringe Kopplung durch hohe Kohäsion erreichen und somit den Wartungsaufwand zu reduzieren

17) Was versteht man unter kollektivem Codebesitz.Vorteile, Nachteile?

Jedes Teammitglied hat volle Kontrolle über den kompletten Quellcode, aber nicht jeder ist für den gesamten

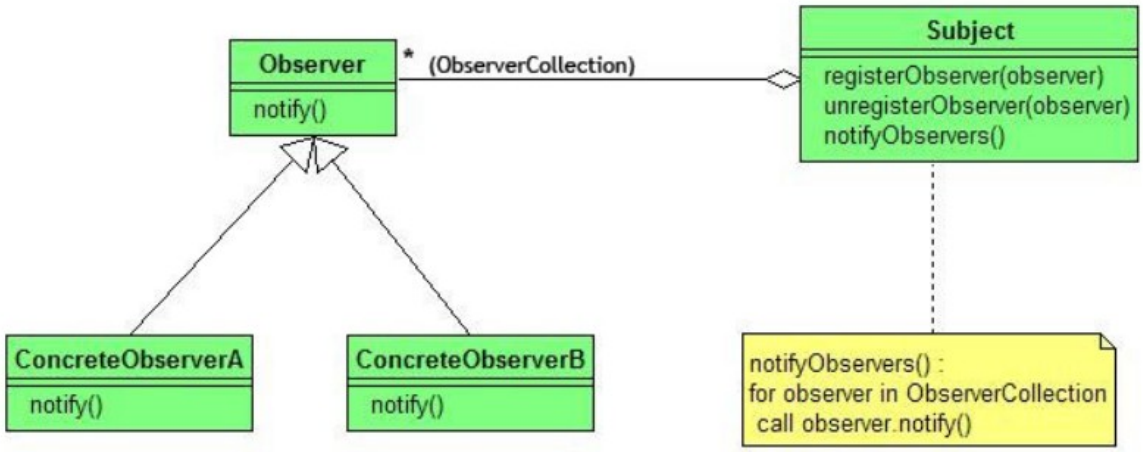
Quellcode verantwortlich. Standardvorgehen vor allem in agilen Methoden.

Vorteil:
- jeder Entwickler kann Änderungen und Verbesserungen am gesamten Code vornehmen
- Wissen über Implementierungsdetails ist besser verteilt
- Auswirkungen bei Ausfall eines Mitglieds sind geringer

Nachteil:
- Verantwortlichkeit für „schlechten“ Code ist schwerer nachvollziehbar

18) Was ist das Observerpattern+ Skizze

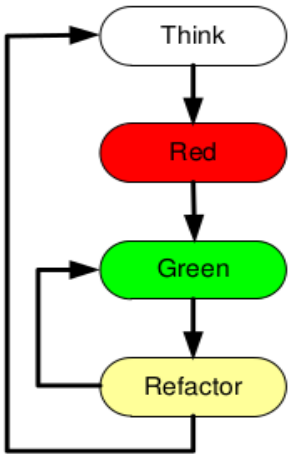
Grundgedanke dieses Pattern ist, ein einzelnes Objekt möchte eine Nachricht an eine Vielzahl vom Empfängern (Obervern) wahrgenommen wird.



19) Erklären Sie das Test Driven Development anhand einer Skizze. Welchen Bezug zwischen automatisierten Tests und TDD gibt es?

Grundsätzlich eine Methode wo die Tests für Komponenten vor der Implementierung dieser erstellt werden, und die Komponenten dann aufbauend auf den Testfällen entwickelt werden.

TDD Schritte:



1. Think:
   - Auswahl der zu implementierenden Anforderungen
   - Spezifikation der Testfälle
2. Red: Implementierung und Ausführung der Testfälle
   - Alle Testfälle müssen fehlschlagen
3. Green: Implementierung der Komponenten und Klassen und Ausführung der Testfälle
   - Test erfolgreich → weiter mit 5
   - Test schlägt fehl → weiter mit 3
4. Refactor: Änderung und Optimierung der Implementierung ohne Änderung der Funktionalität, Ausführen der Testfälle

Automatisierte, fein granulierte Funktionstests (Unittests) sind ein wesentlicher Bestandteil des TDD.

20) Was versteht man unter Refactoring? Welche Methoden des Refactoring gibt es? Welche Rolle spielt Refactoring im TDD

Refactoring = "Aufräumen des Codes"
nach Refactoring darf sich die Funktion des Programms nicht ändern, Testfälle müssen danach immer noch alle positiv sein!

Refactoring dient
- der Verbesserung des Quellcodes
- der Lesbarkeit des Quellcodes
- des Verstehen des Quellcodes

Es gibt ähnlich wie Design Pattern diverse Refaktorisierungsmuster z.B:
- Rename Method
- Extract Method/Class
- Pull Up Methoden
- Split Temporary Variable

Essentieller Bestandteil im TDD zum Ändern und Optimieren der Implementierung ohne dass sich die Funktionalität ändert.

21) Was ist der Unterschied zwischen Internationalisierung und Lokalisierung?

Internationalisierung bezeichnet sämtliche Maßnahmen, eine Software so zu gestalten, dass diese möglichst einfach an andere Sprachen und Kulturen angepasst werden kann.

Lokalisierung hingegen bezeichnet den eigentlichen Übersetzungsprozess
- Anpassung von lokalen Datenformatierungen (Zahlen, Datum, Währung)
- Hinzufügen von spezifischen Code
- ist für jede Sprache separat notwendig.
- Vollständige Lokalisierung ist sehr aufwändig

22) Wozu dient ein Loggin-Framework und warum sollte man nicht System.out.println() verwenden, um zu loggen?

Logging-Frameworks dienen zur Laufzeitprotokollierung. Jede Logging-API stellt üblicherweise folgende Log Levels zur Verfügung
- DEBUG: granulierte Informationen zur Programmausführung
- INFO: wichtige Programmereignisse die im regulären Betrieb vorkommen
- WARN: unerwartete Situation
- ERROR: Fehler is aufgetreten und die entsprechende Fehlerbehandlung wurde eingeleitet
- FATAL: ein nicht behandelbarer Fehler ist aufgetreten. Programm kann nicht stabil fortgesetzt werden

System.out.println() für das Logging zu verwenden gilt als Bad Practice. Zum einen ist das Loggen direkt in den Standardausgabekanal alles andere als ideal, zum anderen ist die Logging-Funktionalität sehr beschränkt und lässt sich auch nicht nach Belieben an oder ausschalten.

23) Was sind Buildmanagementsysteme? Erklärung und 3 Beispiele.

Ein BMS ist ein Werkzeug um immer wiederkehrende Aufgaben zum Bauen einer Applikation automatisiert ablaufen zu lassen. Stellt die Abhängigkeiten zwischen den Komponenten sicher.
<u>Gängige Buildmanagmentsysteme:</u>

- Make
- Ant, nAnt
- Maven

24) Erklären sie den Unterschied zwischen lokaler, zentraler und dezentraler Versionskontrolle

VMS sind Computersysteme die Änderungen an Dateien und Verzeichnissen über die Zeit hinweg archivieren und in einem oder mehreren Entwicklungszweigen für die gemeinsame Bearbeitung bereitstellen.

Lokales VM:
- älteste Form
- beschränkt sich auf die Versionierung einzelner Dateien
- nicht geeignet für Teamprojekte
- RCS (Revision Control System)

Zentrales VM
- Client/Server Ansatz
  - zentraler Server
  - beliebig viele Clients
- SVN (Suversion)

Dezentrales VM:
- keine zentrales Repository
- jeder Entwickler besitzt eigenes Repository
  - Änderungen werden zwischen den Repositories ausgetauscht
- Git

25) Erklären sie den Unterschied zw. Optimistic Revision Control und Pessimistic Revision Control. Erklären sie diese Strategie anhand einer Skizze.

Optimistic Control (Copy-Modify-Merge Strategie):
- Alle Entwickler arbeiten mit lokalen Kopien der Dateien vom Repository, können diese verändern und wieder zurück speichern. Dabei kann es zu Konflikten beim zurück speichern kommen. Ist dies der Fall, dann müssen die beiden Versionen der Datei zu einer neuen Version gemergt werden

Pesimistc Control:
- Wenn ein Nutzer eine Datei anfordert wird diese am Server gesperrt und der Nutzer arbeitet mit der Kopie. Solange die Datei gesperrt is kann kein anderer Benutzer sie sperren oder bearbeiten
- Erst beim zurück speichern der überarbeiteten Datei wird die Sperre aufgehoben

26) Was versteht man unter struktureller Testabdeckung und welche Bedingungsarten gibt es?

Die strukturelle Testabdeckung erfolgt mittels White-Box Tests. Das heißt es wird die konkrete Implementierung überprüft. Und je nach Abdeckungsart müssen gewissen Teile des Kontrollflussgraphen durchlaufen werden um diese zu gewährleisten.
- Anweisungs-Abdeckung: Alle Knoten des Graphen (sprich jede Anweisung muss einmal durchlaufen werden)
- Zweigabdeckung (C1): Alle Kanten des Graphen müssen mindestens einmal druchlaufen werden
- Bedingungsabdeckung (C2)
  - einfache Bedinungsabdeckung: jede atomare Teilentscheidung muss einmal true und false annehmen

  - mehrfache Abdeckung: Alle Wahrheitskombinationen der atomaren Teilentscheidungen werden getestet
- Pfadabdeckung (C3): Alle Pfade im Graphen werden abgedeckt. (Geht bei wachsender Softwaregröße gegen unendlich

Fragen aus Foliensatz 5

27) Erläutern Sie die Teststufen in der iterativen Softwareentwicklung. Erläutern sie in diesem Zusammenhang de Begriffe Stub und Treiber

- Komponetentests:
  - Prüfung von separat testbaren Komponenten (Modulen, Programmen, Objekten, Klassen..) auf vorhandene Fehler.
  - Isolation der einzelnen Komponenten vom Rest des Systems mit Hilfe von Testdoubles (Stubs, Mocks..) bzw. Simulatoren und Testtreiber
- Integrationstests
  - Prüft Schnittstellen zwischen Komponenten
  - können mehrere Integrationsstufen zum Einsatz kommen
  - mit Größe des Umfangs wächst auch die Schwierigkeit der Isolation von Fehlerauswirkungen
  - Bottom-up vs. Top-Down Integrationstests
- Systemtest
  - Im Fokus liegt spezifiziertes Verhalten eines Gesamtsystems → Testen des Systems gegen die Anforderungen
  - sowohl funktionale als auch nicht-funktionale Anforderungen sollten abgedeckt werden
  - Basiseren auf Anforderungsspezifikation, Use-Case, Geschäftsprozessen, Risikoanalysen etc.
- Akzeptanztest
  - auch Abnahmetest, hier wird versucht die Erbringung der vereinbarten Leistung nachzuweisen.
  - Wird meist von Kunden oder Benutzern des Systems ausgeführt

Stub: Code-Sequment welches Aktivitäten von Fehlenden Komponenten simuliert
Driver: Code-Sequment welches Test-Parameter zu einem anderen Code-Sequment weiterleitet.

28) Was versteht man unter Äquivalenzklassenanalyse bzw. Grenzwertanalyse. Was sind die Unterschiede

Äquivalenzklassenzerlegung: Einteilung von Ein- und Ausgabewerten in Klassen mit dem gleichen Systemverhalten. Also Klassen die,die selbe Ausgabe produzieren. Danach werden Werte aus diesen Klassen genommen, bei denen angenommen wird, dass Fehler auftreten können und diese dienen als Repräsentant für die gesamte Klasse. Dadurch wird die Anzahl an benötigten Testfällen reduziert.

Grenzwertanalyse ist ein Spezialfall der Äquivalenzklassenanalyse bei der als Repräsentanten einer Klasse die obere und untere Grenze gewählt werden. Basiert auf der Annahme, dass Fehler besonders oft an den Grenzen auftreten.

29) Warum prüft man Komponenten isoliert? Was für Arten von Isolation kennen Sie?

Die Isolation von einzelnen Komponeten zu Testzwecken, dient in erster Linie dazu Testaufwand für diese Komponente sowie mögliche Fehlerquellen zu reduzieren. Es kann somit effizenter getestet werden. Dazu gibt es mehre möglichkeiten zum einen werden oft Testdoubles (Stub, Mock,Fake,Dummy etc.) eingesetzt um Abhänigkeiten gewissen Komponeten zu simulieren, zum anderen greift man oft auf die Hilfe von Treibern und Simulatoren zurück

Fragen aus Foliensatz 6

30) Was versteht man unter Migration. Nennen sie 3 Arten von Migration

Migration bezeichnet die Ablöse eines bestehenden Systems durch ein neues. Dabei gilt zu beachten, dass der laufende Betrieb nicht beeinflusst werden darf. Und Migration muss einem geordnetem Prozess unterliegen.

- Softwaremigration
  - Austausch einer bestehenden durch eine neue Software
- Datenmigration
  - Transfer von Daten von einem System in ein anders
  - Verwendung von Extract-Transform-Load-Tools
  - Änderung des Datenschemas
- Hardwaremigration
  - Umstellung der verwendeten Hardware

31) Was versteht man unter Continous Delivery.?Prinzipien und Vorteile

Bezeichnet die kontinuierliche Auslieferung von Software durch automatisierte Prozesse
z.B: tägliche Releases ohne Service-Unterbrechung bei Facebook

Prinzipien
- Zuverlässiger & Wiederholbarer Release-Prozess
- Alles is versioniert
- „Done means releast“

Vorteile
- Höhere Zuverlässigkeit aufgrund weniger Fehler
- Weniger Stress
- kürzere Cycle-Time
-

32) Welche Typen der Wartung werden unterschieden? Erklären sie die unterschiedlichen Typen der Wartung

- <u>Korrektive Wartung</u>
  - zur Behebung von Fehlern
  - Abweichungen von der Spezifikationen
  - Nicht-Erfüllung von SLAs
- <u>Präventive Wartung</u>
  - Zur Verhinderung vermutlicher zukünftiger Fehler
- <u>Adaptive Wartung:</u>
  - Zur Anpassung an eine geänderte Umgebung
  - Änderung von verwendeter Softwar
  - Änderung von Schnittstellen
  - Änderung von Hardware
- <u>Perfektionierende Wartung</u>
  - zur Verbesserung der Applikation
  - Verbesserung der Performance

- Verbesserung der Wartbarkeit
- Verbesserung in Usability

33) Beschreiben Sie die Wartungsmaßnahmen Software Reengineering und Reverse Engineering

- Software Reeingineering
  - Neuentwicklung bei gleichbleibender Funktionalität
  - Motivation meistens Qualitätssteigerung
  - Hohes Abstraktionsniveau
- Reverse Engineering
  - Gewinnung von Code & Modellen aus vorhandenen Artefakten
  - Notwendig wenn Source Code nicht verfügbar (Urheberrecht)
  - Re-Dokumentation
  - Trägt zum Code Verständnis bei
  - Oft Vorstufe zum Reengineering

Fragen aus Foliensatz 7

34) Nennen und erklären sie die Delphi-Methode

Ist eine Methode zur Aufwandsabschätzung in der Projektplanung

- Moderater & mehrere Experten
- Verfahren
  - Zu jedem Arbeitspaket gibt ein Moderator einen Schätzwert ab
  - Mittelwert falls alle Schätzwerte in eine bestimmte Bandbreite fallen, ansonsten Argumente austauschen und neue Schätzung
- Zuschläge zu den Einzeleinschätzungen:
  - plus 10-15% Aufwand für Projektmanangment

35) Was ist ein GANTT-Diagramm:

- älteste, am weitesten verbreitete grafische Methode
- Aufgaben zur Terminplanung dargestellt
- aufgaben- oder personenbezogen
- Pufferzeit eines Vorgangs:
- Gesamtpuffer: Zeitspanne, um die sich ein Vorgang verzögern darf, ohne dass das Projektende verzögert wird
- Freie Puffer: Zeitspanne, um die sich ein Vorgang verzögern darf, ohne dass andere Vorgänge verzögert werden
- Kritischer Vorgang: Vorgang ohne Pufferzeit
- Kritischer Pfad: enthält nur kritische Vorgänge

36) Wozu dient das "3-Experten-Konzept"? Erklären Sie dieses.

Ist eine Methode zur Aufwandsabschätzung in der Projektplanung

- 3 optimistische, 3 realistische, 3 pessimistische Schätzungen
- Abklärungen durchführen, bis die Ergebnisse in jeder Kategorie übereinstimmen
- Daraus Schätzwert gemeinsam ableiten

37) Nennen Sie 3 Vorteile/Nachteile im Zusammenhang mit Teamwork

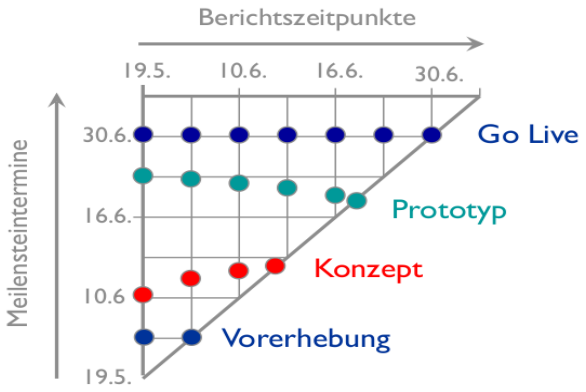
Vorteile

- Mehr Ideen durch Wechselwirkung in der Gruppe
- Direkt vernetzte Kommunikation beschleunigt Informationsverteilung
- Gruppendynamik (Teamgeist)
- Gruppenentscheidungen sind im Allgemeinen besser

Nachteile:

- Gruppenkommunikation ist besonders zeitaufwendig, wenn man jeden einzelnen ernst nimmt
- Gruppendiskussionen laufen manchmal in eine unproduktive Richtung
- zu heterogene Gruppen finden keine gemeinsame Plattform für Gruppenregeln und Gruppennormen

38) Erklären Sie die Meilenstein-Trendanalyse anhand des Diagramms.



39)

Steigende Kurve: Terminüberschreitung **40)**

Waagrechte Kurve: Termin wird gehalten **41)**

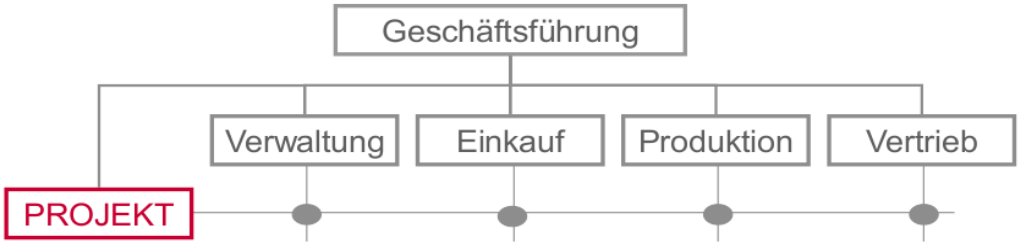
Fallende Kurve: Terminunterschreitung **42)**

43)

44)

45) Erklären Sie die Matrix zu Projektorganisation

Projektleiter trägt Gesamtverantwortung, keine volle Weisungsbefugnis



Vorteile:

- Projektverantwortung durch Projektleiter
- Sicherheitsgefühl der Mitarbeiter
- flexibler Personaleinsatz

Nachteile:

- Mitarbeiter Diener zweier Herren
- Konfliktpotential Linie/Projekt
- hoher Koordinationsaufwand

46) Fragen aus Foliensatz 8

47) Erklären Sie den Begriff eXtreme Programming

Dabei handelt es sich um eine kompakte Methode zur Entwicklung von Software in kleinen bis mittelgroßen Team, deren Arbeit vagen oder sich rasch ändernden Anforderungen unterliegt.
Hauptziel dabei ist die termingerechte Ablieferung von Software. Charakteristische Merkmale von XP sind das eigene Wertesystem, viele Prinzipien und ein System von Techniken, die einander gegenseitig in höchstem Maße unterstützen.

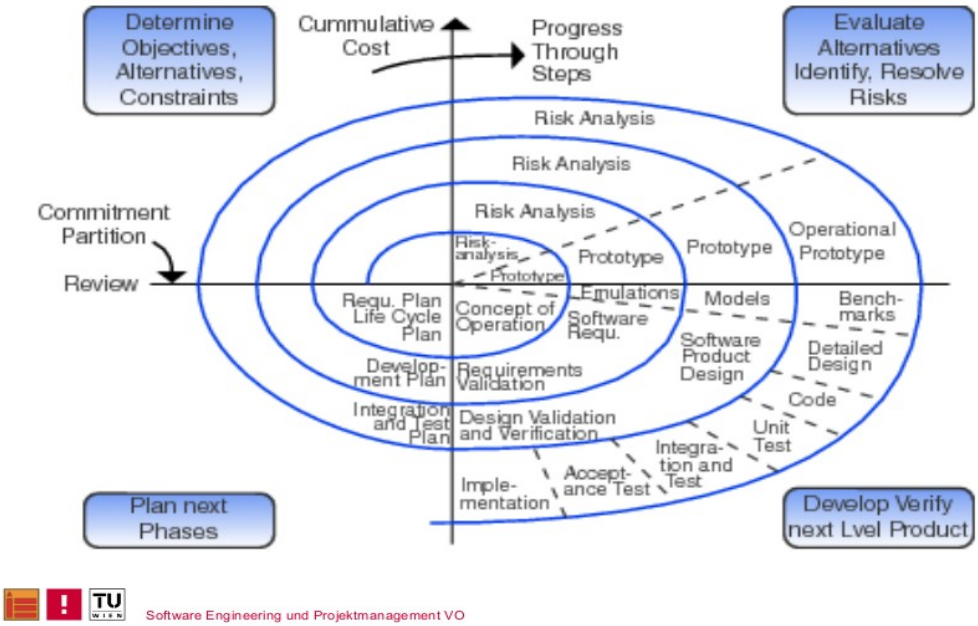
Prinzipien: Kommunikation. Einfachheit, Feedback, Mut, Respekt
Techniken: Kunde vor Ort, Planungsspiel. Metapher. Kurze Release-Zyklen, Testen, 40-Studen Woche...

48) Was sind Vorgehensmodelle und wozu sind sie gut?Beschreiben Sie das Spiralmodell von Boehm und zeichnen sie die Phasen ein. Welche Phasen gibt es? Beschreiben sie das Modell. Unterschied zwischen traditionellen und agilen Methoden erklären

Es gibt bisher keine einheitliche Systematik der Softwareentwicklung, je nach Projektart ist eine geeignetes Vorgehen auszuwählen, das sogenannte Prozessmodell. Das am weiten verbreitete Vorgehensmodel ist das Wasserfallmodel in unterschiedlichsten Ausprägungen. In seiner Grundform ist es vollständig sequentiell.

Spiralmodell nach Böhm:
Berücksichtigt die möglichen Projektrisiken der sequentiellen Entwicklung und ist deshalb iterativ aufgebaut. Der gesamte Prozess ist in 4 Phasen geteilt. Alle Phasen werden im Rahmen der Entwicklung mehrmals durchlaufen.
Phasen: Definition von Zielen, Risikoanalyse, Durchführung der Arbeitsschritte, Planen der nächsten Phase
Die Anzahl der Durchläufe ergibt sich im Projektverlauf

Der Kerngedanke der agilen Softwareentwicklung ist ein zyklischen Vorgehen auf allen Ebene mit häufigen Rückkopplungsprozessen. Anders als bei klassischen Methoden wird das neue System nicht Voraus in allen Einzelheiten genau geplant und dann in einem langen Durchlauf entwickelt. Stattdessen wechseln sich kurze Planungs- und Entwicklungsprozesse ab. Dadurch kann besser auf noch nicht vollständige oder sich ändernde Anforderungen reagiert werden.



Software Engineering und Projektmanagement VO

49) Erklären sie die Grundcharakteristik, die wesentlichen Elemente und Rollen des Unified Process.

Beim RUP handelt es sich um ein industriell geprägtes Prozess-Framework, es gehört zu den detailliert beschriebenen und meisten verbreiteten Modellen. Beruht auf:
- Software iterativ entwickeln
- Anforderungen managen
- Komponenten basierte Architektur verwenden
- Software visuell modellieren
- Softwarequalität kontinuierlich prüfen
- Änderungen kontrolliert in die Software einbringen

Der RUP muss für jedes Projekt spezifisch angepasst werden, um ihn optimal zu nutzen. Dafür gibt es viele Tools, die vordefinierte Prozesse enthalten und eine Spezialisierung erlauben.

Die Phasen sind gegliedert in Inception, Elaboration, Construction und Transition. Es existieren verschiedenste Disziplin (Business Modelling, Analyse, Design, Implementierung, Testen,...) die auf die Phasen aufgeteilt werden. Die 4 Phasen werden dann iteriert.

50) Was ist Scrum. Welche Elemente und Rollen spielen in diesem Zusammenhang eine Rolle?

Ist ein relativ neues Prozessmodell welches auf agiler Entwicklung beruht. Grundgedanke dabei ist die Entwicklung in kleinen Teams, auf Basis eines kleinen Anforderungpools in einem relativ kurzen Zeitraum.
<u>Elemente</u>

- Sprint: Innerhalb eines Sprints wird versucht, so schnell wie möglich zum zuvor definierten Sprint Goal zu kommen.
- Daily Scrum: Dieses Meeting findet immer zur selben Zeit am selben Ort statt.
- Product Backlog: Diese nach Dringlichkeit geordnete Liste aller Anforderungen kann von allen Projektbeteiligten jederzeit eingesehen werden
- ncrement: Bei Scrum entsteht ein Softwareprodukt in Inkrementen, wobei schon der erste, aber spätestens der zweite Sprint ein lauffähiges Softwaresystem (mit Kernfunktionalität) liefert.
- Burndown Chart: In dieses Diagramm werden vom Scrum Master tagesaktuelle

Aufwandsschätzungen für den aktuellen Sprint eingetragen

Rollen:

- Product Owner: Neben der Hauptaufgabe, der Erstellung und Priorisierung des Product & Release Backlogs, ist diese Person wesentlich dafür verantwortlich, dass das Team frei von Störungen und Unterbrechungen bleibt.
- Scrum Team: Die Mission des Teams ist die Umsetzung der über den Sprint Backlog geforderten Funktionalität bzw. die Erreichung des über das Sprint Goal bestimmten Ziels.
- Scrum Master: Die wesentliche Aufgabe dieser Rolle besteht darin, die Dynamik von Scrum zur ungestörten Entfaltung zu bringen.

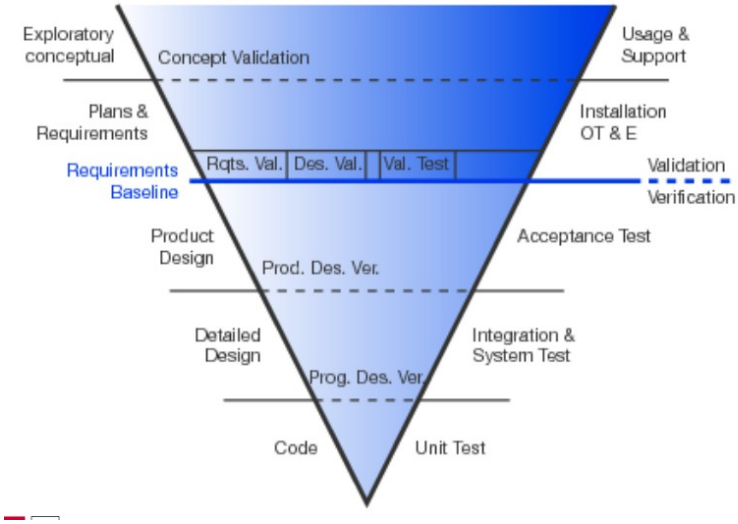
51) V-Modell Beschreiben + Skizze wofür steht das XT im V-XT Modell und Ziele des XT Modells

Basis für viele Entwicklungsmodelle. Basiert auf korrelierenden Phasen und Verifikation und Validierung.

Entwicklungsphilosophie:

- inkrementelle Entwicklung
- zu Beginn Gesamtfunktionalität abgegrenzt, einzelne Funktionen priorisiert
- Funktionen gemäß Priorität Ausbaustufen zugeordnet, im Rahmen dieser umgesetzt
- Ergebnis einer Stufe Endanwendern zur Verfügung gestellt
- Umfang wächst ausgehend von Basisfunktionalität ständig an

XT steh für extreme tailoring. Dabei handelt es sich um ein adaptierte Version des V-Modells welche bessere Anpassung an unterschiedliche Projekte und Organisationen bieten und generell besser skaliert.



Fragen aus Foliensatz 9

52) Was sind Softwaremetriken in der Qualitätssicherung, warum werden sie verwendet? Geben Sie 3 Beispiele objektorientierter Metriken an.

Softwaremetriken werden in der Qualitätssicherung benutzt und ermöglichen es die Qualität von Software zu interpretieren. Bei Prozessmetriken handelt es sich um quantitative Daten des Entwicklungsprozesses. Bei Produktmetriken werden die Software oder Teile davon gemessen, ohne dabei zu berücksichtigen wie das Produkt entstanden ist oder in welchem Zustand es sich gerade befindet. Objekt-orientiere Metriken wurden notwendig um die Beziehung von Objekten untereinander und deren Struktur zu berücksichtigen.
Bsp: CBO (coupling between objects), NOC (number of children), RFC(Response for class)

53) Erklären Sie den Ablauf und die Rollen des Review-prozesses. Warum sind Reviews wichtig?

Der Review-Prozess beginnt mit der Planungsphase. Das Review selbst wird erst durch einen Initialisierungsvorgang gestartet. Den Abschluss des Reviews bildet die Freigabe. Ein Review besteht aus den Schritten Vorbereitung, Sitzung und Nacharbeit. Im Anschluss an das durchgeführte Review wird versucht anhand von Metriken und Analysen Aussagen über die Effektivität des Reviews zu treffen.

Rollen

- Manager: ist der Projektleiter, welcher den Auftrag zur Erstellung des Testobjekts gegeben hat
- Moderater: Er plant, organisiert und leitet das Review.
- Schreiber: notiert werden der Sitzung anfallende Erkenntnisse des Review Teams
- Autor: Ist Urheber des Testobjekts oder Repräsentant des Teams welches das Objekt erstellt hat
- Reviewer: haben die Aufgabe, das zu prüfende Material in der Vorbereitung zu begutachten und in der Sitzung über ihre Erfahrungen zu berichten
- Leser: Ist für die Aufbereitung der zu prüfenden Softwareelemente und für die zeitlich angemessene Durchführung der Sitzung verantwortlich.

Reviews sind essentiell für die Qualitätssicherung unter anderem weil sie sich schon sehr für im Entwicklungsprozess durchführen lassen und Fehler oder potentielle Fehler somit früher gefunden und effizienter behoben werden können.

54) Wozu dient CMM? Nennen Sie die 5 Reifegrade!

Das CCM (Capability Maturiy Modul) Modell hat als Ziel, die Vergleichbarkeit und Qualität von Softwareprozessen im Rahmen der Vergabe an externe Lieferanten zu verbessern. CCM liefert eine Analyse des Ist-Zustands eines Unternehmens und ist kein international anerkanntes Zertifikat.

- Reifegrad 1-initial
  - keine Anforderungen
- Reifegrad 2: Wiederholbarkeit
  - Konfigurationsmangament, SQS, Projektplanung, Anforderungsmanagament etc.
- Reifegrad 3: definiert
  - Peer Reviews, Software Product Engineering, Training und Schulung etc.
- Reifegrad 4: geführt
  - Softwarequalitätsmanangment
  - Quanitatives Prozessmanagment
- Reifegrad 5: optimiert
  - Process Change Managment
  - Technologietransfer
  - Fehlervermeldung

55) Wozu dient SPICE? Nennen Sie die 6 Reifegrade!

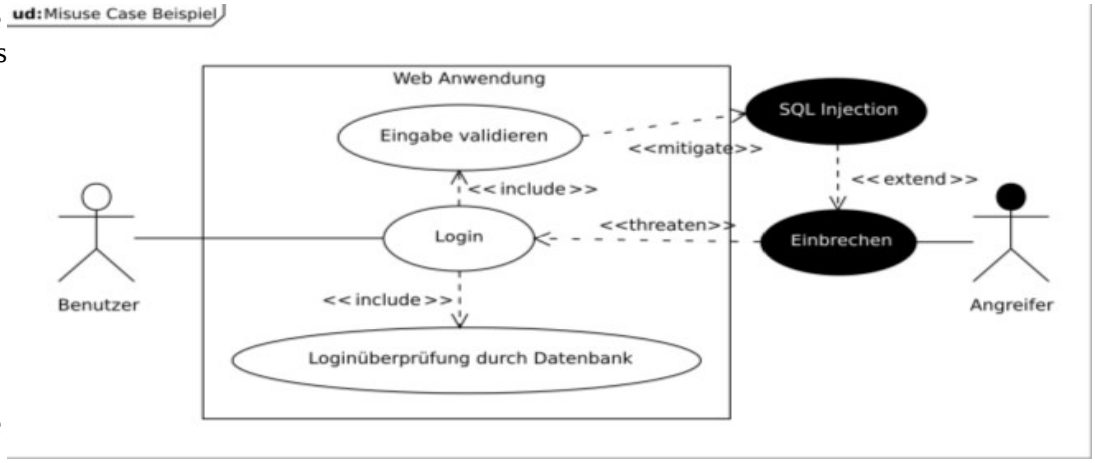
Ist Qualitätsmanagementstandard ähnlich wie CCM. (ca. 80% Überdeckung bei Zielsetzung und Forderung beider Modelle). Somit ist Spice ein Konkurrent zum US-dominanten CCM. Besteht aus einem zweidimensionalen Referenzmotel (Prozessdimension und Fähigkeitsdimesion)

- Stufe 0: unvollständig
  - keine Anforderungen
- Stufe 1: vorhanden
  - Process Perfomance
- Stufe 2: geleitet
  - Performance Managment, Work Product Managment
- Stufe 3: gesichert
  - Process Defintion & Tailoring..
- Stufe 4: vorhersehbar
  - Process Messaurement
  - Process Control

- Stufe 5: optimiert
  - Process Change, Continous Improvement

Fragen aus dem 10 Foliensatz

56) Was erklären Sie das Prinzip des Misuse-Case anhand eines Beispiels



Während Use-Case die erwartete Funktion, das erwartete Verhalten eines System modelliern, ist der Grundgeanke des Misuse-Cases der Missbrauch eben solcher UseCases (z.B. durch Angriffe von außen via SQL-Injection) Während Use Cases mittlerweile weitgehend akzeptiert werden, haben sich Misuse Cases noch nicht durchgesetzt und es existiert auch noch keine Toolunterstützung.

57) Wie läuft eine SQL-Injection ab? Wie kann man sich davor schützen?

Eine SQL-Injection ist ein Angriff auf ein System, bei dem versucht wird mittels Eingabe von Zeichen und Werten, den Code so zu hintergehen, dass er Informationen aus der Datenbank ausspuckt, die durch die aufgerufene Funktion aber nicht erscheinen sollten. Dies ist z.B. mit einstelligen Hochkommas möglich ('), da diese von der Datenbank als Werte interpretiert werden. Somit ist es z.B. möglich bei einem Eingabefeld eine WHERE-Klausel so zu manipulieren, so dass sie Werte aus ganz anderen Tabellen zurückgibt (mit Hilfe von Unions und Tautologien '1 == 1'). Somit können sensible Daten aus der Datenbank erlangt werden.
Um dies zu vermeiden, sollte man datebankspezifische Zeichen in Eingabefeldern von GUI's immer Escapen bzw. die richtigen Berechtigungen in der Datenbank setzen (typischer Ansatz: die minimalsten Rechte vergeben, die zum Erfüllen einer Aufgabe notwendig sind).
Zum Escapen verfolgt man 2 Ansätze:

- Black-List: Darin stehen nicht erlaubte Zeichen
- White-List: Darin stehen alle erlaubten Zeichen

Sicherer ist es natürlich White-Lists zu verwenden!