Fragensammlung SEPM VO 2022W

Nennen und erklären Sie drei der sieben Grundsätze des Softwaretests nach ISTQB!

1. Nur weil keine Tests fehlschlagen, System nicht fehlerfrei
2. Möglichst früh mit Testen beginnen
3. Tests zeigen Fehler auf
4. Tests hängen von Umfeld ab
5. Vollständiges Testen unmöglich

Was versteht man unter dem in der LVA diskutierten Vorgehen “Entwicklung auf Zuruf”? Ist dieses Vorgehen einem Softwareprojekt zuträglich? Begründen Sie Ihre Antwort

Jede Anforderung und Change Request wird ohne formalen Prozess einngeführt. Es wird schnell sehr unübersichtlich was die Änderungen, also schlecht für Softwareprojekte.

Was versteht man unter Anforderungsnachverfolgbarkeit (Traceability)? Warum ist Traceability bei der Durchführung eines Softwareprojekts wichtig?

Traceability beschreibt die Übereinstimmung zwischen Anforderung und Software.

- Nachverfolgbarkeit der Quelle: Anforderung, Wann/Wie/Von wem?
- Nachverfolgbarkeit zu anderen Anforderungen: Interdependenzen (gegenseitige Abhängigkeit)
- Nachverfolgbarkeit zu abhängigen Artefakten: Systemdesign, Testfälle

Wichtig um Überblick zur Verbindung von Anforderung zu Implementierung und Testing zu haben

Was versteht man unter den Begriffen Kopplung und Kohäsion? Welcher Zusammenhang besteht?

- Kopplung: wie stark einzelne Komponenten von einander abhängig sind
- Kohäsion: wie stark einzelne Komponenten Funktionalitäten zusammenfassen

starke Kohäsion führt zu schwacher Kopplung

Nennen und erklären Sie die vier in der LVA genannten Paradigmen der serviceorientierten Architektur. Was versteht man unter “business driven” im Kontext von SOA?

- niedrige Kopplung: Services untereinander unabhängig, Funktionalitäten gekapselt, ermöglicht reuseability
- Abstraction: reduziert Neuentwicklung
- Standadisierung der Schnittstellen: Zusammenarbeit zwischen Services
- Composable: Standadisierte Services werden zu Komponenten zusammengefasst

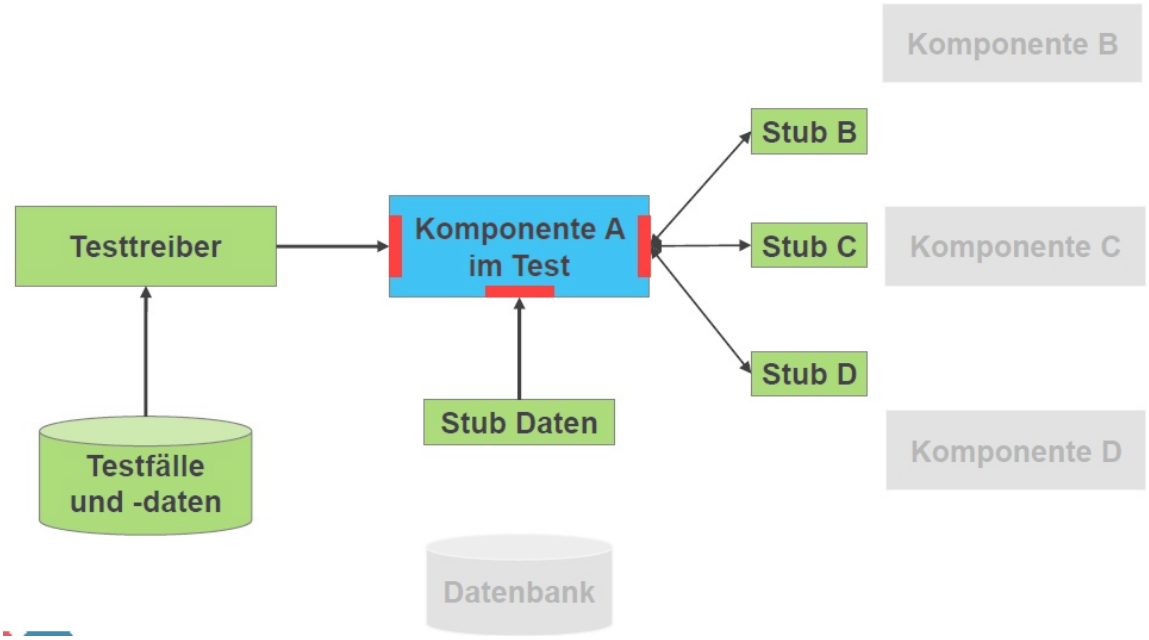
business driven: Archtitektur an Geschäftsprozesse angelehnt, Wirklichkeit soll abgebildet werden

Was versteht man unter Microservices? Welche Vorteile und Herausforderungen bringen Microservices mit sich?

Bei Microservices werden Services in kleine, einzeln entwickelte, betriebene und eingesetzte Einheiten geteilt

- Vorteile: versch. Techonologien für einzelne Services, Teams für Service voll verantwortlich, kurze Entwicklungszeit, Modulare Struktur
- Nachteile: komplexe Operationen müssen pro Service immer neu gemacht werden (Build, Testen, ...), Kommunikation zwischen Services, Dezentrale Datenhaltung, Teams cross-funktional

Was versteht man unter Komponententest? Skizzieren und beschreiben Sie das in der LVA vorgestellte Test-Setup für den Komponententest!



- Komponententest: eine einzelne Komponente wird getestet, Beziehungen zu anderen Komponenten werden dabei simuliert (Stubs)

Wozu dienen Design Patterns? Nennen Sie drei Design Patterns der Kategorie “Creational Patterns”/“Structural Design Patterns”, erklären Sie eines davon anhand eines Beispiels.

Design Patterns helfen und bieten Lösungen für Probleme die auftreten können

- Creational Patterns: Singleton (es darf nur ein Element existieren, wird ein neues erstellt und es

existiert schon eines, wird das alte zurückgegeben, wenn nicht wird es erstellt), Factory, Builder
- Structural Design Patterns: Proxy (über die Proxy Klasse greift der Client auf eine andere Klasse zu und geht quasi über ihr einen Umweg), Facade, Bridge, Decorator

Erklären Sie das Vorgehen bei Test Driven Development (TDD)! Erläutern Sie weiters die "zwei goldenen Regeln" von Kent Beck!

TTD:

1. Think: zu implementierende Anforderungen wählen, Testfälle spezifizieren
2. Red: Testcases schreiben, es sollen alle failen
3. Green: Implementation verfassen, es sollen alle Testcases erfolgreich abschließen
4. Refactor: Code refaktorisieren und optimieren ohne das Funktionalität verloren geht, danach Schritt 1 für nächstes Feature

zwei Regeln: erst Implementation schreiben, wenn Tests failen, redundanten/meehrachen Code vermeiden

Was versteht man unter Inversion of Control? Erläutern Sie ein Beispiel dafür!

IOC: statt das source code library aufruft, ruft Framework den source code auf (deswegen Inversion), kümmert sich zB.: unter anderem um die Dependencies (dependency injection)
Beispiel: Spring-Framework ruft bei Rest Aufruf Endpoint Methode auf

Was versteht man unter den Begriffen Verifikation und Validierung im Kontext des Softwaretestens?

- Verifikation: Richtig gebaut? entspricht Software den Spezifikationen
- Validierung: das Richtige gebaut? entspricht Software den Anforderungen

6 Typische Eigenschaften eines Softwareprojekts

- Zeit
- Risiko
- Ressourcen
- Einmaligkeit
- Neuwertigkeit
- Zielvorgabe
- Größe
- Komplexität

Statische/Dynamische/Organisatorische Qualitätssicherung erklären mit jeweils 2 Elementen

- Statische Qualitätssicherung: Programm wird statisch, also außerhalb der Laufzeit analysiert, 3 Elemente: Statische Analyse, Review, Walthrough

- Dynamische Qualitätssicherung: Programm wird dynamsich, also während der Laufzeit analysiert, 3 Elemente: Dynamische Analyse, Testing, Debugging
- Organisatorische Analyse: Programm wird organisatorisch, also aus abstrakter Projekt Sicht analysiert, 3 Elemente: Checklisten, Templates, Wissensmanagement

SOLID Prinzipien aufzählen und 3 Punkte näher beschreiben

- Single Responsibility Principle: eine Komponente hat nur eine responsibility
- Open-Close Principle
- Liskov Substitution Principle
- Interface Seggregation Principle
- Dependency Inversion Principle

Unterschiede Framework und Library

Framework ruft den source oder Programm code auf (IOC, Hollywood-Prinzip), während der source code libraries aufruft

Unterschiede zentrales und verteiltes Versionsmanagementsystem. Erklären und je ein konkretes Beispiel. Was sind Versionkontrollsysteme?

- Zentrales Versionsmanagementsystem: alle Operationen laufen über einen Server (bottleneck/Flaschenhals) und es werden lokal bei den Clients keine Daten gespeichert, Beispiel: SVN
- Verteiltes Versionsmanagementsystem: Operationen können auf allen Clients laufen, Daten müssen lokal gespeichert werden, Änderungen zwischen repos getauscht, Beispiel: git

Versionskontrollsystem: mit diesen Tools können Softwareteams auf vorherige Versionen zugreifen und vorherige Changes einsehen, Changes werden alle vom Tool gespeichert

Erklären Sie Begriffe Äquivalenzklasse und Grenzwertanalyse. Wozu werden diese verwendet? Geben Sie ein konkretes Beispiel für den Nutzen einer Äquivalenzklasse.

- Äquivalenzklasse: beim Testen einer Funktionalität mit einem Input weisen bestimmte Inputs das gleiche Verhalten auf, werden diese Inputs jeweils in Klassen mit gleichem/äquivalenten Verhalten geteilt
- Grenzwertanalyse: Wichtig sind dabei die Randfälle zwischen den Klassen, da dort oft Fehler entstehen
  Beispiel: f(x) wird getestet, Äquivalenzklassen: x < 0, 0 <= x, getestet wird x = -1 und x = 1, mit Grenzwertanalyse: x = -1, x = 0, x = 1

1 agiles und ein traditionelles Modell beschreiben

agiles
SCRUM: Backlog mit Anforderungen, Sprints dauern 20 Tage, Scrums 1 Tag, jeden Tag Scrum Meeting,

danach Sprint Review und nächster Sprint mit neuen Anforderungen
Rollen:

- Product Owner (erstellt Backlog, keine Einflüsse von außen)
- Scrum Master (schaut das Scrum richtig funktioniert, keine Einflüsse von außen)
- Scrum Team (Developer, Tester, UI/UX)

traditionelles
Wasserfall Modell: Fehler am Anfang sehr schlecht, weil ein Schritt nach dem anderen geschieht, bei Fehlern geht man Schritt für Schritt zurück (zB.: falsche Anforderung: von Testing zu Analyse und dann Design ...)

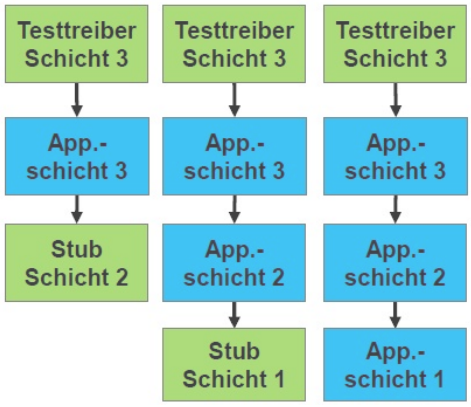
- Analyse <-> Design & Entwurf <-> Implementation & Integration <-> Testing <-> Deployment

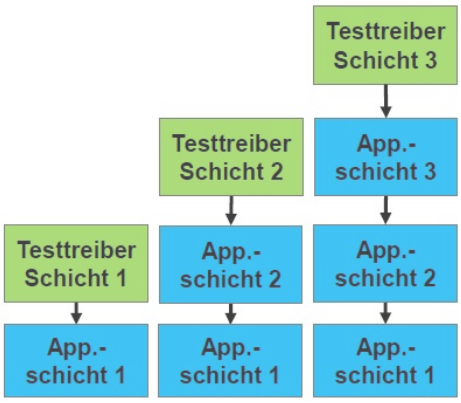
Einen Führungsstil beschreiben und alle nennen

- autokratisch: eine Person führt das Team alleine an
- demokratisch
- kooperativ
- authentisch
- situativ

Eine Skizze zu horizontalen Integrationstests vervollständigen und beide Arten erklären

- Top-Down: es wird von oben nach unten gearbeitet, man hat früh externe Schnittstellen, da man die unteren Schichten simuliert, allerdings kann man muss sehr viel simulieren
- Bottom-Up: von unten nach oben, wenig Simulation notwendig, spät erst externe Schnittstellen, viel Testen notwendig





3 Arten von Vorgangsbeziehungen erklären

- Normalfolge: Das Ende von Vorgang 1 ist Vorraussetzung für Anfang von Folge 2: sequentiell, Ende-Anfangs Beziehung
- Anfangsfolge: Der Anfang von Vorgang 1 ist Vorraussetzung für Anfang von Folge 2: parallel, Anfangs-Anfangs Beziehung
- Endfolge: Das Ende von Vorgang 1 ist Vorraussetzung für Ende von Folge 2: parallel, Ende-Ende Beziehung

3 Gegenüberstellungen strategischer/taktischer Entwurf

Strategischer Entwurf = Architektur:

- Architektur im Großen
- Entwurf auf Systemebene
- Nichtfunktionale Anforderungen sind Faktoren

Taktischer Entwurf = Design:

- Architektur im Kleinen
- Entwurf auf Bausteinebene
- Funktionale Anforderungen sind Faktoren

User Stories (3 Vor- und Nachteile)

- Vorteile: kurz und prägnant, fördern Diskurs zwischen Kunden und Entwickler, reduziert Risiko für Fehlentwicklung
- Nachteile: starke Involvierung von Kunden (unrealistisch), personenzentriert, keine Vertragsgrundlage

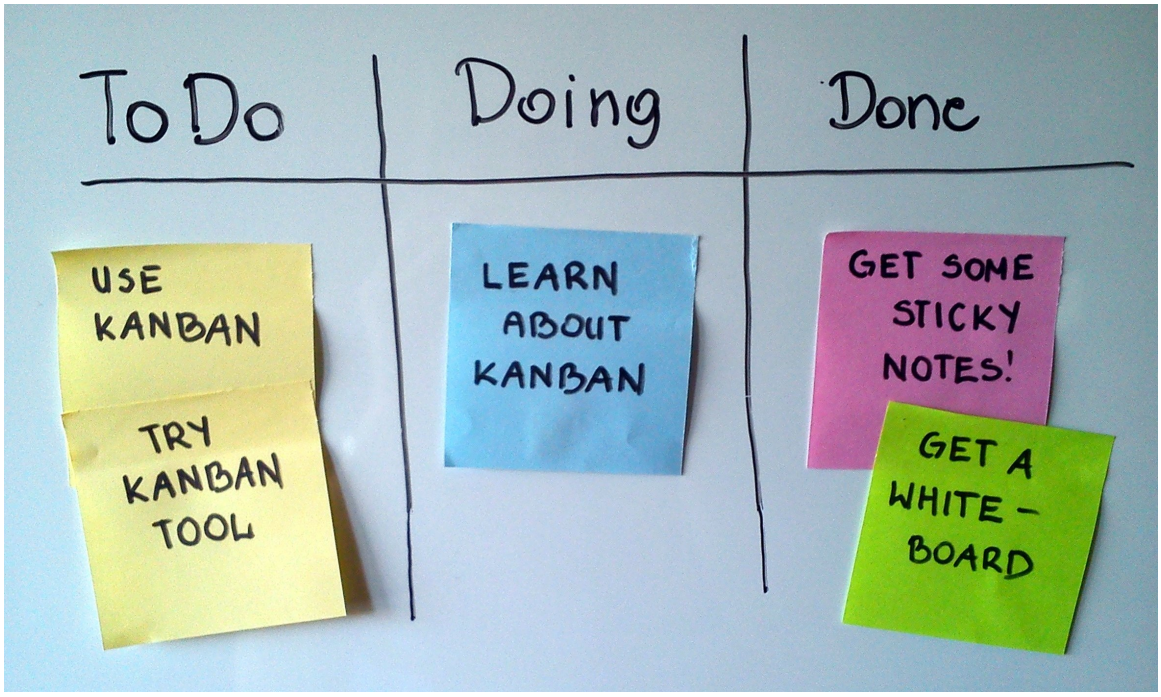
UML (Die 2 Typen und jeweils 1 Beispiel)

- Verhaltensmodelle: Zustandsdiagramm
- Strukturmodelle: Klassendiagramm

Top-Down und Bottom-Up erklären

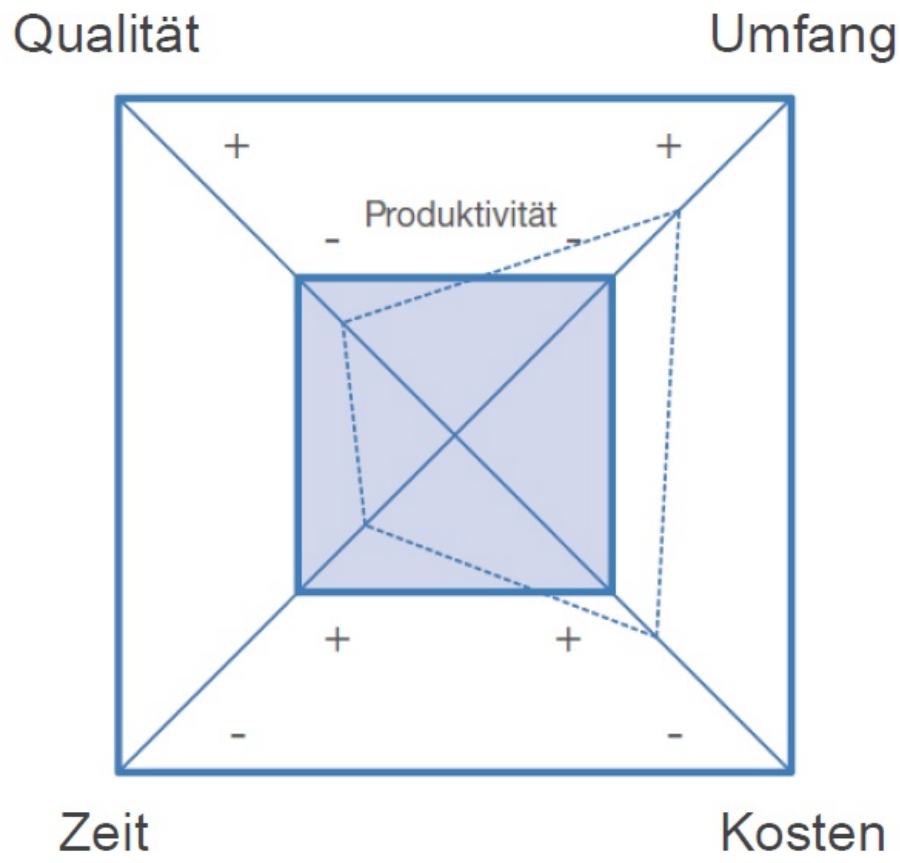
- Top-Down: es wird von oben nach unten gearbeitet, man hat früh externe Schnittstellen, da man die unteren Schichten simuliert, allerdings kann man muss sehr viel simulieren
- Bottom-Up: von unten nach oben, wenig Simulation notwendig, spät erst externe Schnittstellen, viel Testen notwendig

Kanban (Erklären + Skizze)



Board (Backlog, Doing, Done) mit Notizen, Pull System: es werden die Anforderungen von links nach rechts gezogen

Teufelsquadrat beschreiben und skizzieren



Quadrat, Flächeninhalt bleibt immer gleich, Ecken aus: Kosten, Zeit, Umfang, Qualität, die Fläche ist die Produktivität, zeigt das Verhältnis der genannten Punkte beim Projekt

4 Risiko dinger aus der vorlesung nennen und beschreiben. Was ist risiko?

Risiko = Eintrittswahrscheinlichkeit x Schadenhöhe, Risiken sind Dinge die beim Projekt eintreten können die zu höheren Kosten, Zeitverzögerungen oder ähnlichem führen können

- Risikoidentifikation: welche Risiken können eintreten?
- Risikoanalyse: Was passiert bei diesen Risiken?
- Risikobehandlung: was kann man machen wenn das Risiko eintritt?
- Risikokontrolle: prüfen ob das Risiko eintritt

3 Wartungstypen nennen und beschreiben

- Präventive Wartung: Fehler beheben bevor sie auftreten können
- Korrektive Wartung: Fehler beheben nach dem sie auftreten
- Perfektionierende Wartung: System optimieren (Speicherverbrauch, Performance, ...)

Nenne 3 arten von anforderungen, beschreibe diese und nenne je 2 beispiele im context einer webplattform die studenten das anmelden zu LVAs ermöglicht.

- Funktionale Anforderungen: Funktionen die die Software können muss (Bsp.: Studenten müssen sich anmelden können, Studenten müssen sich zu LVAs anmelden können)
- Nichtfunktionale Anforderungen: Anforderungen die nicht Verhalten sondern Vorgaben an Funktionalität stellen, zB.: Qualität, Standards, usw. prüfen (Bsp.: die Seite muss 24/7 erreichbar sein, man muss sich innerhalb 5 Clicks anmelden können)
- Domänenanforderungen: Anforderungen die automatisch durch die Domäne gestellt sind (Bsp.: nicht Studenten dürfen keinen Zugriff haben, jeder Student hat eine Matrikelnummer)

Was ist Serviceorientierte Architektur? Welche Modellierungssprachen werden eingesetzt?

Anwendungen durch mehrere Services gelöst, Services können andere Services aufrufen
4 Paradigmen:

- Schwache Kopplung: Services voneinander unabhängig, Services wiederverwendbar
- Abstraction: reduziert Neuentwicklungen
- Standadisierung von Schnittstellen: Zusammenarbeit zwischen Services
- Composeable: Standadisierte Services können zu Komponenten zusammengefasst werden

Modellierungssprachen: UML, BPM

Was ist BPM? Zusammenhang mit SOA?

- Service-orientierte Architektur: Fokus auf Services, Anwendungen durch Services realisiert, Services können andere Services aufrufen
- Business Process Modelling eng mit SOA verknüpft, Anwendungen von realen Geschäftsprozessen beeinflusst

Strukturelle Abdeckung: Anweisungs-, Bedingungs- und

Pfadabdeckung erklären:

Beim White-Box Testen werden Tests aus innerer Struktur herausgeleitet:

- Wie weit jede Anweisung einmal durchlaufen werden: zeigt toten Code
- Wie weit jede Bedingung einmal geprüft werden: einfache Bedingungsabdeckung: jede Bedingung muss mal true und mal false sein, mehrfache Bedingungsabdeckung: alle Kombinationen von true und false bei Bedingungen
- Durchführung aller Pfade der Software, normalerweise unmöglich (unendlich Zweige) und wenn nicht unendlich trotzdem extrem groß

Was ist der Unterschied zwischen Software Reengineering und Reverse Engineering?

Reengineering: Teile der Software werden neu geschrieben ohne die Funktionalität zu beeinflussen
Reverse Engineering: Source Code wird von Grund aus untersucht um einen abstrakten Überblick über das ganze System zu erstellen

Nennen Sie die vier in der VO besprochenen Methoden zur Erhebung von Anforderungen!

- Interviews
- Workshops
- Beobachtungen
- Mitarbeit

5 Risiken nennen

- zu wenig Zeit
- zu wenig Ressourcen
- mangelnde Anforderungen
- Team nicht qualifiziert
- Konflikte mit Projektleitung
- Konflikte innerhalb Teams
- sich ändernde Anforderungen

Anweisungs-/Bedingungsüberdeckung mit Codebeispiel (Wie viele Testfälle sind nötig? Nennen sie 2 Testfälle)

Beispiel:

if (x == 10) // do something

- Anweisungsüberdeckung = 1 Testfall:
  - int x = 10
- Bedingungsüberdeckung = 2 Testfälle:
  - int x = 10

    - int x = 5

Eigenentwicklung vs Framework, wann sollte ich mich für was entscheiden.

Deckt das Framework alle gewollten Anforderungen ab, ist Wartung und Weiterentwicklung garantiert, und sind Kosten in Ordnung -> Framework
Sonst Eigenentwicklung (aber teuer, dauert, ...)

5 Entscheidungsfaktoren für agil vs. planungsgetrieben

- Personal muss bei agil qualifizierter sein
- bei wechselnden Änderungen agil
- nach Entwicklungskultur kann agil eingesetzt werden
- Teamgröße klein (1-2) bis mittelgroße (<= 12) Teams für agile Methoden
- Kritikalität: falls Menschenleben von Software abhängig sind, mehr und stärkere Prüfung auf allen Ebenen