SQS – TEST (THEORIE) SS14

Testfragenausarbeitung bis SS14

Anmerkungen

Diese Ausarbeitung beruht auf der Sammlung der bisherigen Testfragen vom VoWi stand Mai. 2014. Herzlichen Dank an meine Vorarbeiter. Ich hoffe ich konnte mit meiner Arbeit dazu beitragen, den Stoff der Vorlesung etwas angenehmer zum Lernen zu machen.

Jeder ist herzlich eingeladen an dieser Ausarbeitung mit zu wirken. Ich kann nur empfehlen sich die Fragen herzunehmen und diese selber versuchen Auszuarbeiten oder zu verbessern. Es steht jedem frei diese Ausarbeitung nach seinem Ermessen zu korrigieren oder zu verwenden.

Quellenangaben:
https://vowi.fsinf.at/wiki/Spezial:Materialien/TU_Wien:Software-Qualitätssicherung_VU_(Biffl)

Schwarz Gerald 1126385

Inhaltsverzeichnis

1. *Qualitätsmanagement VS Qualitätssicherung, Prinzipien des Qualitätsmanagements nennen.*

- Qualitätsmanagement
  Ist die Menge aller Aktivitäten, Vorgehensweisen, Techniken und Hilfsmittel, die sicherstellen, dass ein Softwareprodukt vordefinierte Standards erreicht oder übertrifft.
- Qualitätssicherung
  (QS) besteht in der Durchführung von Verifikation und Validierung in jeder Phase der Softwareherstellung
- Prinzipien
  - Konkrete **operationalisierbare** Qualitätsmerkmale
  - Produkt- und projekt**abhängige** Qualitätsplanung
  - **Unabhängigkeit** bei Qualitätsprüfung
  - **Mehraugenkontrolle** bei Qualitätsprüfung
  - **Frühzeitige Entdeckung und Behebung** von Fehlern und Mängel
  - **Bewertung** der eingesetzten Qualitätsmaßnahmen
  - Organisation in Form von **Qualitätsmanagementsystemen** (ISO 9001, CMM(I)
  - **Rückkopplung** der Ergebnisse der Qualitätsanforderungen
  - Prinzip der **ständigen Verbesserung von Produkten und Prozessen**

2. *Integration der Qualitätssicherung (Verifikation und Validierung) im V-Modell (Skizze war dabei) einzuschreiben.*

Verifikation vs. Validierung

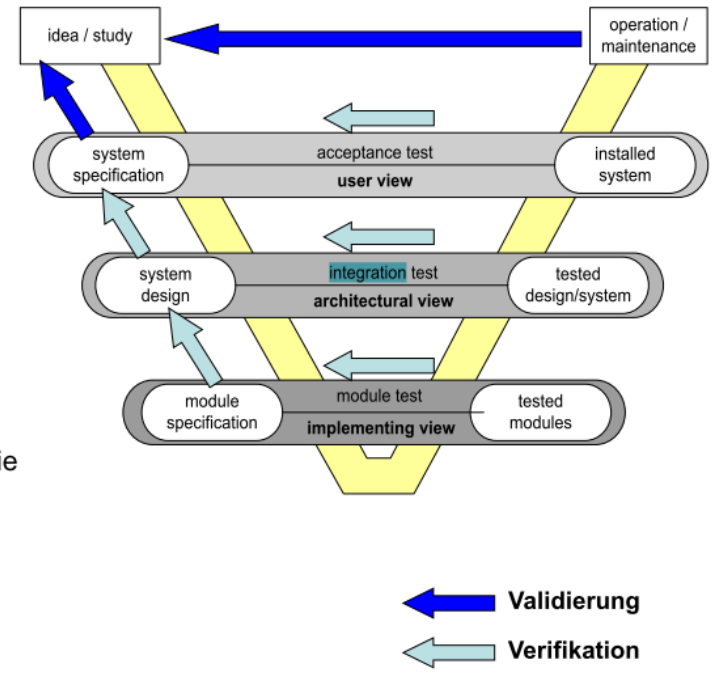
TU WIEN

Verifikation:

- „Bauen wir das Produkt richtig?“
- Umsetzung im Vergleich zur Spezifikation in vorangegangenen Phasen.
- Test der Umsetzung gegen die Spezifikation

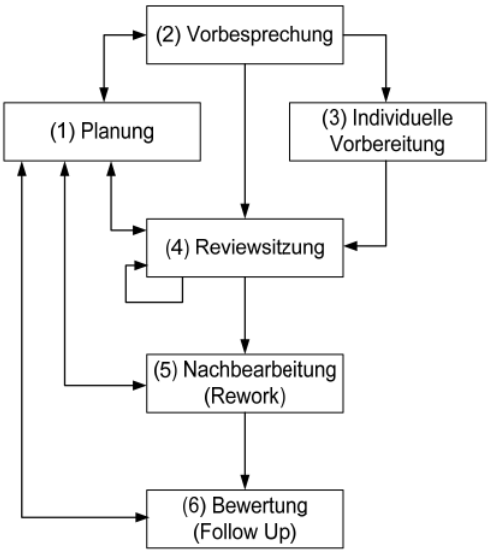
Validierung:

- „Bauen wir das richtige Produkt?“
- Entspricht die Lösung und damit die Spezifikation dem, was der Kunde erwartet?
- Test der Umsetzung gegen die Nutzeranforderungen.



3. Ablauf eines Review beschreiben (Skizze+Stufen kurz erklären) Vorteile eine Reviews gegen den manuellen Test nennen.

Ablauf einer Review



- Planung: Objekt, Prüfziele, Auslösekriterien (Einstiegskriterien), Teilnehmer, Ort, Zeit.
- Vorbesprechung: Vorstellung des Prüfobjekts bei komplexen und neuen Produkten.
- Intensive Einzeldurcharbeitung
- Durchführung: Gemeinsames Lesen, Aufzeichnung von Mängeln; während des Reviews sollen Mängel entdeckt, nicht korrigiert werden.
- In der Nachbearbeitung werden dokumentierte Mängel korrigiert und in der Bewertung überprüft.
- Berichterstattung.
- Wiederholungen von Reviews sind möglich.

- Checklisten unterstützen Reviews.
- Typische Dauer: 2h

Vorteile:

- Software-Entwicklung erfordert die **Herstellung konsistenter Sichten** auf Anforderungen und Entwurf eines Systems
- Modelle helfen den **Überblick** zu bekommen und zu behalten
  - Grundlage für effektives und effizientes **Arbeiten im Team**
  - **Gemeinsame Notation** mit konsistenter Bedeutung
- Herausforderung: Konsistente Verwendung unterschiedlicher Modelle
  - **Systemstruktur**: Subsysteme, Komponenten, Schnitstellen
  - **Verhalten** von Komponenten; **Interaktion** zwischen Komponenten
- Anforderungen -> Modelle – **Daten/Komponenten/Testspezifikationen**
- Tests überprüfen das Laufzeitverhalten von Systemteilen
  - Funktionen, Modelle und Qualitätsmerkmale
  - Ausgangsbasis; Anforderungen, Ein-/Ausgabeparameter des Systems, innere Systemstruktur

4. Black Box Test VS White Box Test. Ist es möglich den White Box Test mit JUnit Framework durchzuführen. Wie? Begründung!

Black-Box-Test

bezeichnet eine Methode des Softwaretests, bei der die Tests ohne Kenntnisse über die innere Funktionsweise des zu testenden Systems entwickelt werden. Er beschränkt sich auf **funktionsorientiertes** Testen, d. h. für die Ermittlung der Testfälle werden nur die Anforderungen, aber nicht die Implementierung des Testobjekts herangezogen. Die genaue Beschaffenheit des Programms wird nicht betrachtet, sondern vielmehr als **Black Box** behandelt. Nur nach außen sichtbares Verhalten fließt in den Test ein.

White-Box-Test

Black-Box-Tests werden eingesetzt um Fehler gegenüber der Spezifikation aufzudecken, sind aber kaum geeignet, Fehler in bestimmten Komponenten oder gar die fehlerauslösende Komponente selbst zu identifizieren. Für letzteres benötigt man White-Box-Tests. Zu bedenken ist auch, dass zwei Fehler in zwei Komponenten sich zu einem vorübergehend scheinbar korrekten Gesamtsystem aufheben könnten. Dies kann durch White-Box-Tests leichter aufgedeckt werden, bei Black-Box-Tests nach der nicht auszuschließenden Korrektur nur einer der beiden Fehler jedoch als vermeintliche Regression zu Tage treten.

Zur Frage mit JUnit:

Ja einen ganz normalen JUnit test aufsetzen und für die Klassen Testdoubles verwenden (Mock, Stubs usw.)

5. Kontrollflussgraph war gegeben. Testfälle bezüglich 100% Anweisungsüberdeckung und 100% Bedingungsüberdeckung waren gefragt. (Multiplechoicefrage)

Zur Kontrolle hier die Anforderungen für die Überdeckungsmöglichkeiten:

Anweisungsüberdeckung c0:

Zielt darauf ab alle Knoten von Anweisungen min. 1 zu durchlaufen, dann ist eine 100%ige Anweisungsüberdeckung erreicht. Diese Testart bietet die Möglichkeit nicht ausführbare Anweisungen (toten Code) aufzuspüren, jedoch ist es ein zu schwaches Kriterium für sinnvolle Testdurchführung, da das Auftreten von Fehlerwirkung an die Ausführung bestimmter Testdaten gekoppelt sein kann.

Zweigüberdeckung c1:

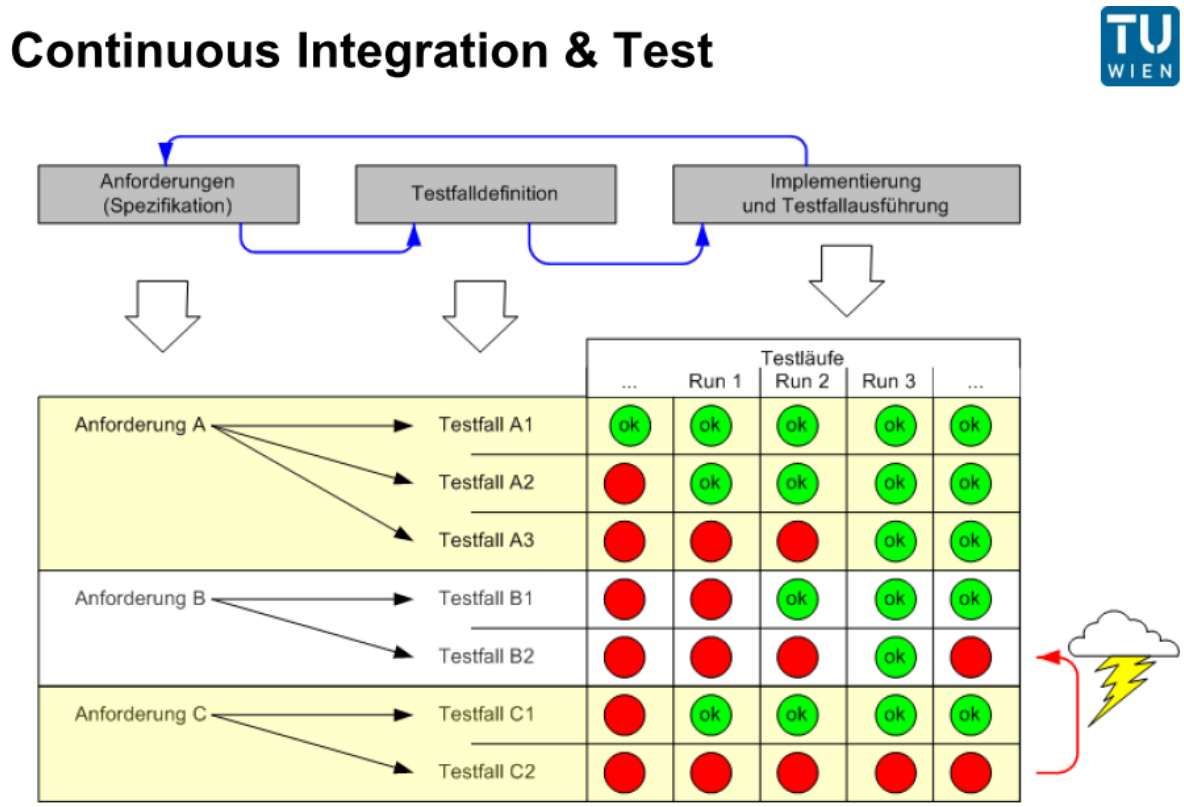
Zielt darauf ab alle Kanten von Anweisungen zu durchlaufen. Im Gegensatz zu c0 muss hier jede Entscheidung mindestens einmal TRUE und FALSE annehmen. Diese Testabdeckung spürt nicht ausführbare Programmzweige auf und besonders oft durchlaufene Programmzweige können gezielt optimiert werden. Er gilt als Minimalkriterium im Kontrollflussorientierten Testen. Problem hierbei ist, dass Schleifen unzureichend getestet werden, da ein Durchlauf genügt und kompliziert aufgebauten Entscheidungen werden nicht berücksichtig.

Bedingungsüberdeckung c2:

Die Grundidee hierbei ist die Überprüfung zusammengesetzter Entscheidungen, Teilentscheidungen. Die Ausprägung „Einfacher Bedingungstest“ subsumiert hierbei nicht c0 od. c1. Er fordert den Test aller atomaren Teilentscheidungen gegen TRUE u. FALSE. Die Ausprägung „Mehrfacher Bedingungstest subsumiert hingegen c1. Dieser fordert den Test aller Wahrheitswertekomtinationen der atomaren Teilentscheidungen,

dies ist demnach exponentiell 2^n. N ist die Anzahl der Variablen die in Anweisungen vorkommen. Dieser Test ist sehr aufwendig da er exponentiell wächst. Einsatzgebiet ist eher dort, wo Fehler eine nicht absehbare Auswirkung haben und es sich demnach Auszahlt diesen durchzuführen oder bei Verlangen des Auftraggebers.

6. Continous Integration Workflow erklären, Einfluss des CI-Systems an die Qualitätssicherung beschreiben.



Der Einfluss besteht darin, dass sofort sichtbar ist, wo ein möglicher Fehler auftritt anhand der kontinuierlichen Testläufe (siehe Bild). Das ist das Wesen, was die Qualitätssicherung ausmacht, zu wissen Wo ein Fehler auftreten kann oder einen Vergleich herstellen, hier war alles in Ordnung und auf einmal nicht mehr. Dann, im Falle von CI, zurück zu diesem Zeitpunkt gehen und dem Verantwortlichen eine Zeit geben um diesen Fehler zu lokalisieren und reparieren.

7. Phasen im Testprozess - Teststufen (Unit, Integration, Regression, System, Akzeptance) mit der richtigen Fällen (Erklärungen) verbinden (so ähnlich wie Multiplechoicefragen)

- *Unit (Modul)*
  - Ziel: Aufspüren von Fehlern in der **Implementierung**
  - **Module** werden jeweils einzeln gegen ihre **Spezifikation** getestet: Übergabe von Daten, Prüfung des Outputs.
  - Modultests erlauben eine **genaue Lokalisierung** und **frühe Erkennung von Implementierungsfehlern**.
  - Modultests können bei größeren Systemen **sehr aufwendig** werden; daher ist bei größeren Tests besonderer Wert auf **strukturiertes Vorgehen** und **Dokumentation des tatsächlichen Vorgehens** zu legen.
- Integrationstests
  - Ziel: Test der **Interaktion zwischen Modulen**
  - Zusammenführung von Modulen zu größeren Strukturen (Häufig durch Entwickler)

  - Integrationsstrategien: „Big Bang“, Top Down, Bottom Up, Build Integration
  - Build-Integration und Integrationstests sind – sofern möglich – zu bevorzugen
- Systemtests
  - Ziel: Test des Gesamtsystems gegen die Anforderungsanalyse bzw. den Systementwurf.
  - Validierung u. Verfikation
- Abnahme- Akzeptanztests
  - Ziel: Validierung des Systems gegen die **Kundenanforderungen**
  - Vom Kunden durchgeführter Systemtest, meist mit einer Submenge an Testfällen aus dem Systemtest.

8. Testdoubles nennen und kurz beschreiben. Wie führt man die Isolation beim Testen?

- *Stub:*
  liefert vordefinierte Werte an den Aufrufer
- *Mock:*
  liefert vordefinierte Werte an den Aufrufer und überprüft im Gegensatz zum Stub die bekommenen Übergangsparameter und wird somit Teil des Tests.

Auf die Frage „Wie führt man die Isolation beim Testen“:
durch den Einsatz von Testdoubles, führt man die Isolation herbei/durch.

9. Welche Aufgaben hat Qualitätsmanagement?

- Qualitätsplanung als Teil der Projektplanung
- Herstellen lokaler Standards auf Projekt- und Organisationsebene
- Review zentraler Projektdokumente
- Organisieren von Reviews: Ausbildung, Planung, Durchführung, Verbesserungsvorschläge
- Unterstützung bei Personalauswahl und Software-Zukauf
- Vorbereitung und Auswertung von Podukttests

10. Was sind die 3 Schwerpunkte/Aufgaben des Softwarequalitätsmanagement?

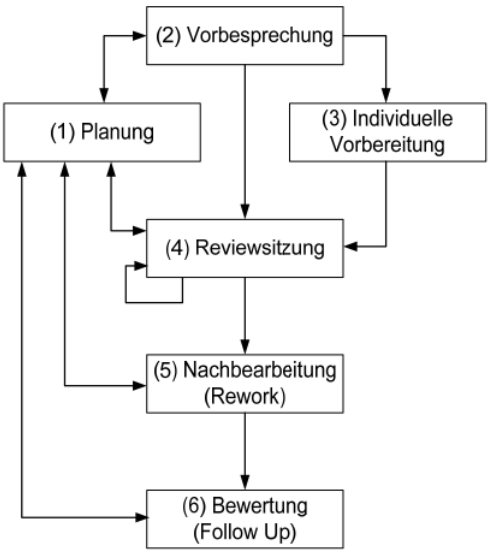
- Qualitätsplanung als Teil der Projektplanung
- Herstellen lokaler Standards auf Projekt- und Organisationsebene
- Review zentraler Projektdokumente

11. Was ist ein Review? unterschied zwischen Review und Test? Vor- und Nachteile?

- Review
  Ein Review ist ein geplanter und strukturierte Analyse- und Bewertungsprozess, in dem Projektergebnisse einem Team von Gutachtern präsentiert und von diesen kommentiert oder genehmigt werden.
- Unterschied Review und Test
  Der zentrale Unterschied liegt darin, dass Tests je nach Ausführung einzeln/automatisiert durchgeführt werden. Bei aufgedeckten Fehlern, werden diese sofort korrigiert. Bei Reviews ist dies ein Analyse- und Bewertungsprozess, bei dem die Ergebnisse einem Team von Gutachtern präsentiert und von diesen kommentiert oder genehmigt werden.
- Vorteil
  - Tragen erheblich zur Verbesserung der Qualität des Produktes schon in der Entwicklungsphase dessen bei und sind in ständiger Begleitung des Produktes.
- Nachteil
  - Meist sehr aufwendig und nicht immer von einzelnen Personen ausführbar. Dies erfordert eine gewisse Planung und kann nicht von heute auf morgen durchgeführt werden.

12. Wie ist ein Review aufgebaut? + Skizze

Ablauf einer Review



- Planung: Objekt, Prüfziele, Auslösekriterien (Einstiegskriterien), Teilnehmer, Ort, Zeit.
- Vorbesprechung: Vorstellung des Prüfobjekts bei komplexen und neuen Produkten.
- Intensive Einzeldurcharbeitung
- Durchführung: Gemeinsames Lesen, Aufzeichnung von Mängeln; während des Reviews sollen Mängel entdeckt, nicht korrigiert werden.
- In der Nachbearbeitung werden dokumentierte Mängel korrigiert und in der Bewertung überprüft.
- Berichterstattung.
- Wiederholungen von Reviews sind möglich.

- Checklisten unterstützen Reviews.
- Typische Dauer: 2h

13. *4 Arten von Test-Doubles nennen und kurz beschreiben. Welche Vorteile bringen Test-Doubles beim Testen in Bezug auf Qualitätssicherung?*

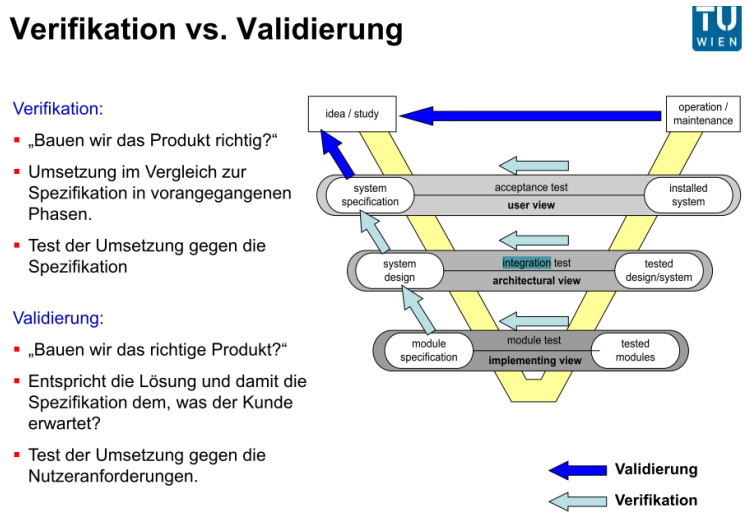
- Dummy Object
  Ohne Funktionalität, dienen als „leere“ Methodenparameter
- Fake Object
  Ausführbare Implementierung, liefern keine Echtdaten, z.B. Emulation einer Datenquelle zur raschen Testausführung
- Stub
  Liefert vordefinierte Werte an den Aufrufer
- Mock
  Liefert vordefiniert Werte an den Aufrufer und überprüft im Gegensatz zum Stub die bekommenen Übergabeparameter und wird somit Teil des Tests.

Vorteil der Testdoubles:

Durch Testdoubles kann schon frühzeitig in der Entwicklung die Funktionalität des Projektes bzw. Modul getestet werden ohne das reale Daten oder Daten die erst später zur Verfügung stehen würden.

14. *2 Prozessmodelle in der Softwareentwicklung nennen. Anhand eines Modells erklären, wie darin Qualitätssicherung gewährleistet wird.*

- V-Modell



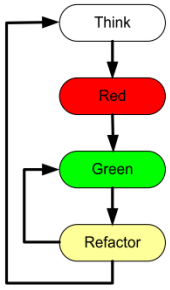
Beim V-Modell wird die Qualitätssicherung dahingehend gewährleistet, dass einerseits die Validierung am Anfang also zu Beginn und am Schluss, zur Abnahme hin durchgeführt wird und somit eine kontrollierte Qualität des Produktes vorhanden ist. Andererseits wird bei jedem Schritt im V-Modell eine Verifikation durchgeführt ob das Entwickelte auch, dass ist was der Kunde verlangte.

- TDD

Test-Driven Development Process

Test Driven Development Schritte:

1. Think
   - Auswahl der zu implementierenden Anforderungen.
   - Spezifikation der Testfälle.
2. Red: Implementierung und Ausführung der Testfälle
   - Alle Tests müssen fehlschlagen.
3. Green: Implementierung der Komponenten und Klassen und Ausführung der Testfälle
   - Testfall erfolgreich → weiter bei Schritt 4.
   - Testfall schlägt fehlt → weiter bei Schritt 3.
4. Refactor: Änderung und Optimierung der Implementierung ohne Änderung der Funktionalität; Ausführung der Testfälle.
   - Testfälle dürfen nicht fehlschlagen.
   - Auswahl der nächsten Anforderung (Schritt 1)



Bei diesem Softwareentwicklungsprozessmodell wird anhand von Testfällen die eigentliche Klasse (Implementierung) heraus entwickelt. Dies gewährleistet eine immer währende fehlerfreie, nach den definierten Testfällen, Software. Was auch Qualitätssicherung ausmacht.

15. *Was bedeutet Refactoring und wann wird es eingesetzt? Wie wird Refactoring durchgeführt? Was wird dadurch erreicht?*

- Refactoring ..
  bedeutet eine Änderung und Optimierung der Implementierung ohne Änderung der Funktionalität. Wird bspw. In TDD eingesetzt.
- Einsatz….
  Ein explizierter Anwendungsfall ist beim Softwareentwicklungsprozess TDD, wo im Letzten schritt ein Refactoring durchgeführt wird. Dies bedeutet unter Umständen alles in eine Zeile (unübersichtlicher, kompakter, im engsten Sinne) zu gestalten. Z.B. Funktionenwerte/Berechnungen die in Variablen welche nur zur Zwischenspeicherung dienten und danach als Parameter übergeben wurden werden gleich als Parameter übergeben.
- Nutzen:
  kürzeren Quellcode, übersichtlicher gestaltet die durch IDE durchgeführtes Refactoring ist Typen geprüft, bessere Wiederverwendbarkeit.

16. *Was ist Software-Qualität? Welche Möglichkeiten gibt es, Software-Qualität sicherzustellen?*

- „Unter Softwarequalität versteht man die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen“ (Ist/Soll).
- Möglichkeiten:
  - Software Prozesse (Life-Cycle Modell, V-Modell)
  - Konstruktive Methoden zur Herstellung der Produkte z.B. Model-Diven-Development, Test-Driven-Development
  - Analytische Methoden zur Verifikation und Validierung der Projektergebnisse z.B. Reviews und Inspektion, Testen.

17. Nennen Sie Vorgehensweisen zu Sicherung/Verbesserung der Qualität von Projekt-Anforderungen? (4 Punkte)

- Review
- Inspektion
- Code Walk-Through

18. Was ist der Unterschied zwischen Testing und Debugging? (2 Punkte)

- **Testing** ist das durchführen vordefinierter Test z.B. UnitTest, Komponententests (Szenarios) welche fehlschlagen oder korrekt ablaufen können.
- **Debugging** hingegen ist das konkrete ausbessern eines durch Testing oder anderer Mittel ermittelten Fehlern.

19. Was ist der Unterschied zwischen Äquivalenzklassenzerlegung und Grenzwertanalyse? (4 Punkte)

- Ziel der **Äquivalenzklassenzerlegung** ist es, Äquivalenzklassen zu bilden und so eine hohe Fehlerentdeckungsrate mit einer möglichst geringen Anzahl von Testfällen zu erreichen. Die Äquivalenzklassen sind also bezüglich Ein- und Ausgabedaten ähnliche Klassen bzw. Objekte, bei denen erwartet wird, dass sie sich gleichartig verhalten.
- Die **Grenzwertanalyse** erweitert die Äquivalenzklassenzerlegung für bessere Überdeckung. Die Grenzwerte werden als Klassenrepräsentanten gewählt -> je Klassengrenze ein Testfall.

20. Eine Methode string10(String s) retourniert die ersten 10 Zeichen des Strings s. Ist der String kürzer als 10, so wird der String unverändert zurückgegeben. Wieviele Äquivalenzklassen werden mindestens benötigt, um die Methode vollständig zu testen. (4 Punkte)

- Antwort 3:
  1) kleiner zehn
  2) größer zehn
  3) gleich zehn (Grenzfall)

21. Nennen Sie 3 Prinzipen des Qualitätsmanagements und erklären Sie diese. Was ist der Unterschied zwischen Qualitätsmanagement und Qualitätssicherung? (6 Punkte)

- Qualitätsmanagement
  Ist die Menge aller Aktivitäten, Vorgehensweisen, Techniken und Hilfsmittel, die sicherstellen, dass ein Softwareprodukt vordefinierte Standards erreicht oder übertrifft.
- Qualitätssicherung
  (QS) besteht in der Durchführung von Verifikation und Validierung in jeder Phase der Softwareherstellung
- Prinzipien
  - Konkrete **operationalisierbare** Qualitätsmerkmale
    z.B. Korrektheit, Zuverlässigkeit
  - Produkt- und projekt**abhängige** Qualitätsplanung
    Software Process Improvement (SPI) mit dem PDCA-Zyklus (Plan,Do,Check,Act)
  - **Unabhängigkeit** bei Qualitätsprüfung
    Unabhängigkeit kann auch durch externe QS-Prüfer erfolgen
  - **Mehraugenkontrolle** bei Qualitätsprüfung
    mit Durchführung von Reviews odä.

- **Frühzeitige Entdeckung und Behebung** von Fehlern und Mängel
  durch geplantes ständiges Test Modellen und deren Testphasen (Unit-, Integrations-,System-,Akzeptanztests)
- **Bewertung** der eingesetzten Qualitätsmaßnahmen
  z.B. CodeCoverage mit vordefiniertem Wert der zu erreichen ist.
- Organisation in Form von **Qualitätsmanagementsystemen** (ISO 9001, CMM(I)
  Qualitätsmanagementsysteme stellen sicher, dass die Systemqualität, Prozessqualität und die Produktqualität in einer Organisation geprüft und verbessert wird. Ziel eines Qualitätsmanagementsystems ist eine dauerhafte Verbesserung der Unternehmensleistung.
- **Rückkopplung** der Ergebnisse der Qualitätsanforderungen
  Vergleich von Ist und Soll Zustand an div. vordefinierten Projektmeilensteinen und dementsprechend handeln
- Prinzip der **ständigen Verbesserung von Produkten und Prozessen**
  Durch die erlangten Erkenntnisse von den div. Projekten und der Grundlage dessen Dokumenten besteht immer eine Verbesserung und Wiederverwendung der angewandten Modelle und Prozessen.

22. Erklären Sie den Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen und geben Sie je ein Beispiel.

Eine funktionale Anforderung legt fest, was das Produkt tun soll.

- Ein Beispiel:
  „Das Produkt soll den Saldo eines Kontos zu einem Stichtag berechnen.“

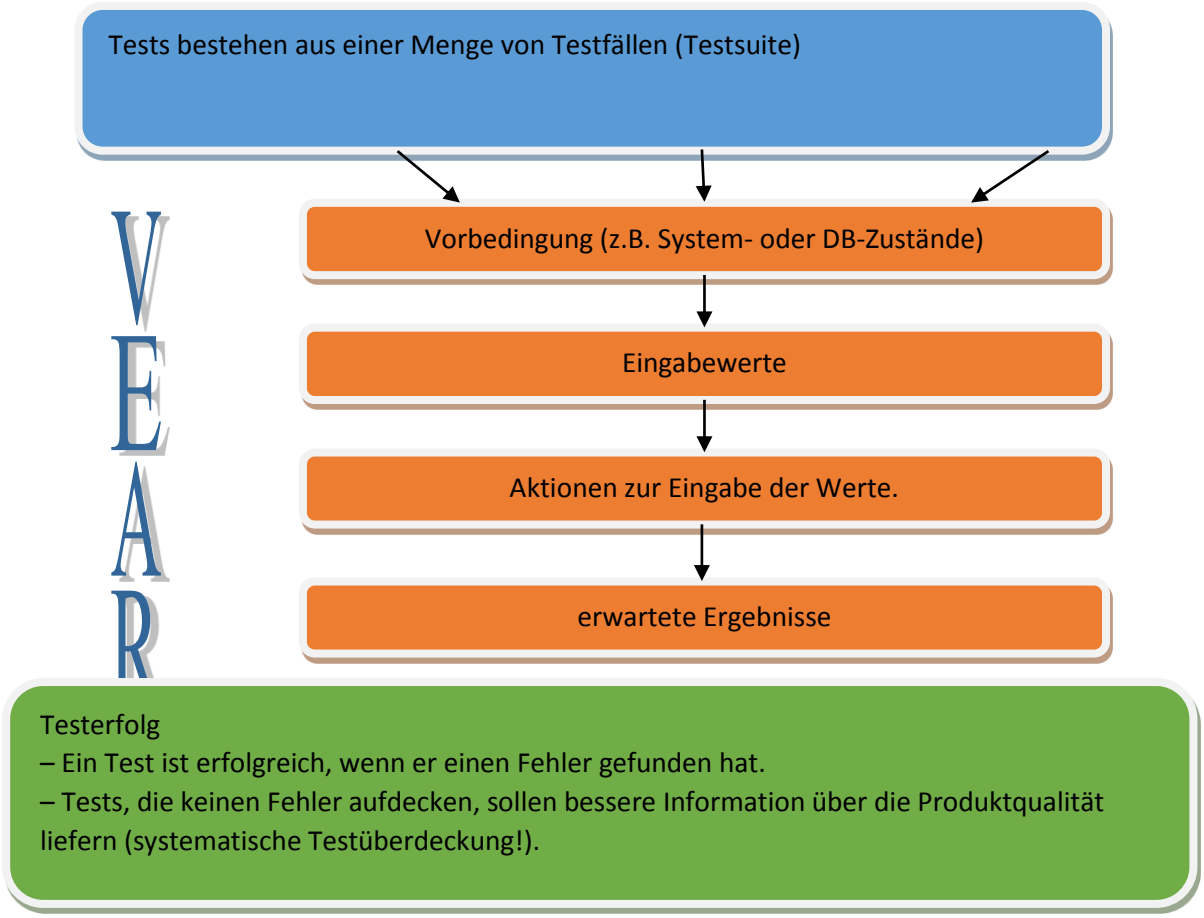
Eine nichtfunktionale Anforderung legt fest, welche Eigenschaften das Produkt haben soll.

- Ein Beispiel:
  „Das Produkt soll dem Anwender innerhalb von einer Sekunde antworten.“

23. Erklären Sie den Unterschied zwischen Unit Tests und Data Driven Tests.

Der große Unterschied zwischen Unit-Test und Data Driven Tests ist jener, das bei Data Driven Tests eine externe Tabelle mit div. zu testenden Daten herangezogen wird. Im Groben gesagt, Test gegen eine Tabelle / Datenbank mit Werten. Bei Unit-Tests werden speziell gewisse Abschnitte (Module) gezielt während (automatisch) der Entwicklung getestet. Dies kann nun hard-coded sein oder mit parametrisierten Werten indem man sie aus einer Datei liest geschehen. Bei Data Driven Tests ist dies eher der Fall wenn ein Großteil des Produktes fertiggestellt ist und somit mit den vorgegebenen Daten getestet werden kann.

24. Erklären Sie den grundlegenden Test-Prozess anhand einer Skizze.



25. Worauf ist beim Erstellen von Äquivalenzklassen zu achten?

Ziel der Äquivalenzklassenbildung ist es, Äquivalenzklassen zu bilden und so eine hohe Fehlerentdeckungsrate mit einer möglichst geringen Anzahl von Testfällen zu erreichen. Die Äquivalenzklassen sind also bezüglich Ein- und Ausgabedaten ähnliche Klassen bzw. Objekte, bei denen erwartet wird, dass sie sich gleichartig verhalten. Bspw. im Programm zur Verwaltung eines Fuhrparks sind äquivalente Klassen Fahrzeuge (Ferrari - BMW nicht jedoch Ferrari - Mitarbeiter). Das Wesen der Äquivalenzklassenbildung besteht darin, die gesamten Eingabedaten und Ausgabedaten eines Programms in Gruppen von Äquivalenzklassen zu unterteilen, so dass man annehmen kann, dass mit jedem beliebigen Objekt einer Klasse die gleichen Fehler wie mit jedem anderen Objekt dieser (Äquivalenz-)Klasse gefunden werden (Bspw. Ferrari ENZO - BMW M3).

26. Was ist der Unterschied zwischen Review und Inspektion?

Bei der Inspektion werden u.a. auch Metriken ermittelt. Im Gegensatz zu einem Review, wo das Prüfteam eine Empfehlung an den Manager weitergibt, ist es bei einer Inspektion so, dass der Moderator eine Freigabe erteilt und der Vorleser das Prüfobjekt Absatz für Absatz vorträgt.

27. Was ist der Ablauf eines Reviews?

Es gibt eine Planungsphase, wo das Review im Detail geplant wird. Daraufhin gibt es eine Vorbesprechung wo das Prüfobjekt dem Prüfteam vorgestellt wird. Nun hat jeder die Möglichkeit sich mit dem Prüfobjekt intensiv einzeln zu beschäftigen. Nach einer kurzen Zeit kommt dann die eigentliche Review Sitzung mit einem Gemeinsamen Lesen und aufzeigen von gefundenen Fehlern/Unstimmigkeiten. Diese Review Sitzung kann mehrmals wiederholt werden und sogar wieder eine Planung einberufen. Darauffolgend kommt die Nachbearbeitungsphase, wo die dokumentierten Mängel korrigiert und in der Bewertung überprüft werden. Dies kann auch wieder in die Planungsphase übergehen.

28. Welche Lesearten gibt es bei Reviews? Beschreiben Sie jede dieser Arten kurz.

Perspektivenbasiertes Lesen

- Perspektivenbasierte Lesetechniken beruhen darauf, dass die Gutachter eine konkrete Anleitung – ein Szenario – erhalten, um sich in die Rolle eines bestimmten Nutzers des Review Objektes hinein zu versetzen und damit die Suche nach Befunden auf bestimmte Aspekte zu konzentrieren. Dabei wird davon ausgegangen, dass jeder Inspektor ein anderes Szenario bekommt und so das gesamte Inspektionsteam effektiver wird.

Fehlerklassenbasierte Lesetechnik

- Das Fehlerklassenbasierte Lesen beruht darauf, dass die Gutachter unterschiedliche Fehlerklassen zugewiesen bekommen. Die historischen Daten werden dann gemäß ihrer Auftretens Häufigkeit und ihrer Schwere priorisiert und die demzufolge wichtigsten Fehlerklassen werden auf die Gutachter verteilt.

Checklistenbasiertes Lesen

- Die Checklistenbasierte Lesetechnik als Hilfsdokument eine Checkliste. Dieses enthält eine Reihe von Fragen enthält, die der Inspektor während der Individuellen Vorbereitung beantwortet. Checklisten passen immer nur zu genau einem Dokumenttyp. Zur Erarbeitung der Checkliste bietet es sich an, die Erstellungsrichtlinien dieses Dokumenttyps zu analysieren und typische Qualitätsmerkmale aufzulisten, die dieser Dokumenttyp erfüllen soll.

Ad-hoc Lesetechnik

- Ad-hoc werden Dokumente ohne spezifische Fragestellung gelesen und um einen ersten Überblick zu erhalten über den Inhalt zu erhalten. Der Gutachter hat dabei den Blick des internen Kunden und konzentriert sich in der Regel auf die Erfüllung eigener Erwartungen bzgl. ob das Dokument als Grundlage der eigenen Arbeit geeignet ist.

29. Was ist die Aufgabe der Qualitätssicherung? Wo ist sie in einem Unternehmen einzuordnen?

Was ist die Aufgabe der Qualitätssicherung

- Planen & Sicherung von Qualität, daraus folgt die Erstellung qualitativ hochwertiger Software (Die Erzeugung eines qualitativ hochwertigen Produktes wird dadurch NICHT garantiert!)

Wo ist sie in einem Unternehmen einzuordnen

- Als eigene Abteilung die bei der Projektabwicklung maßgeblich von Anfang an bis zum Schluss beteiligt ist und deren Qualitätskriterien kontrolliert und für die Umsetzung sorgt, um ein qualitativ hochwertiges Produkt zu erstellen.

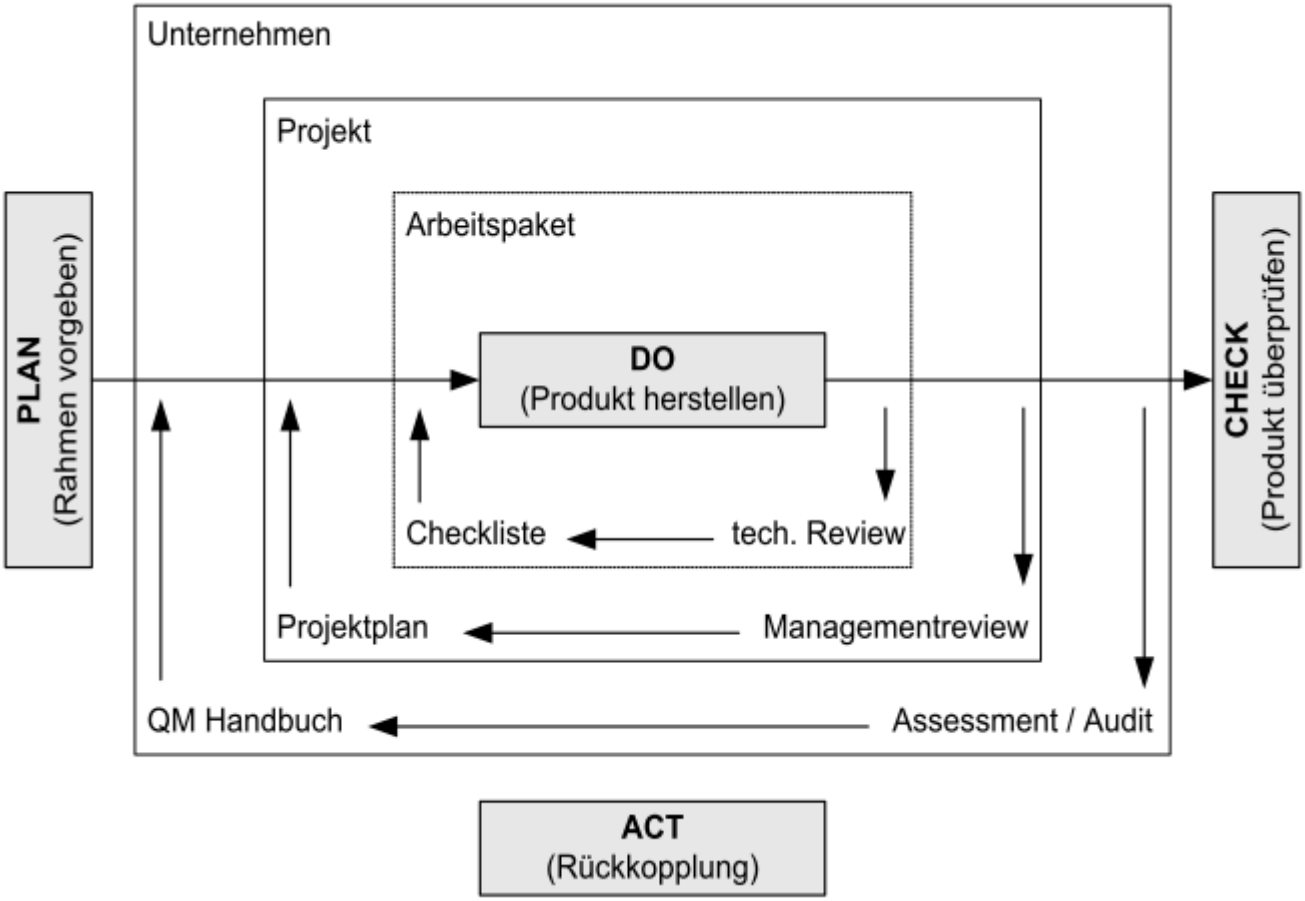
30. Wann wird Qualitätssicherung eingesetzt? Wann/wo/wie sollte Qualitätssicherung eingreifen?

Wann wird Qualitätssicherung eingesetzt

- Zu jedem Zeitpunkt!

Wann/wo/wie sollte Qualitätssicherung eingreifen

- Nach dem PDCA-Zyklus



31. Erklären Sie den Unterschied der Coverage Kategorien c0 - c3. Wann ist der Einsatz welcher Coverage sinnvoll?.

Anweisungsüberdeckung c0:
Zielt darauf ab alle Knoten von Anweisungen min. 1 zu durchlaufen, dann ist eine 100%ige Anweisungsüberdeckung erreicht. Diese Testart bietet die Möglichkeit nicht ausführbare Anweisungen (toten Code) aufzuspüren, jedoch ist es ein zu schwaches Kriterium für sinnvolle Testdurchführung, da das Auftreten von Fehlerwirkung an die Ausführung bestimmter Testdaten gekoppelt sein kann.

Zweigüberdeckung c1:
Zielt darauf ab alle Kanten von Anweisungen zu durchlaufen. Im Gegensatz zu c0 muss hier jede Entscheidung mindestens einmal TRUE und FALSE annehmen. Diese Testabdeckung spürt nicht ausführbare Programmzweige auf und besonders oft durchlaufene Programmzweige können gezielt optimiert werden. Er gilt als Minimalkriterium im Kontrollflussorientierten Testen. Problem hierbei ist, dass Schleifen unzureichend getestet werden, da ein Durchlauf genügt und kompliziert aufgebauten Entscheidungen werden nicht berücksichtig.

Bedingungsüberdeckung c2:
Die Grundidee hierbei ist die Überprüfung zusammengesetzter Entscheidungen, Teilentscheidungen. Die Ausprägung „Einfacher Bedingungstest“ subsumiert hierbei nicht c0 od. c1. Er fordert den Test aller atomaren Teilentscheidungen gegen TRUE u. FALSE. Die Ausprägung „Mehrfacher Bedingungstest subsumiert hingegen c1. Dieser fordert den Test aller Wahrheitswertekomtinationen der atomaren Teilentscheidungen, dies ist demnach exponentiell 2^n. N ist die Anzahl der Variablen die in Anweisungen vorkommen. Dieser Test ist sehr aufwendig da er exponentiell wächst. Einsatzgebiet ist eher dort, wo Fehler eine nicht absehbare Auswirkung haben und es sich demnach Auszahlt diesen durchzuführen oder bei Verlangen des Auftraggebers.

32. Er Nennen Sie mindestens zwei Gründe für den Einsatz von Testdoubles (Mock, Stubs, etc.).

Testdoubles sind dann gut einsetzbar wenn, das reale Objekt:

1. Zum Testzeitpunkt nicht verfügbar ist
2. Nicht deterministische Ergebnisse liefert (Datum / Zeit)
3. Langsam od. nicht vorhanden ist
4. Schwer ein bestimmten Status zu setzen ist

33. Erklären Sie den Unterschied zwischen Mock Objects und Stubs.

Stub:
liefert vordefinierte Werte an den Aufrufer

Mock:
liefert vordefinierte Werte an den Aufrufer und überprüft im Gegensatz zum Stub die bekommenen Übergangsparameter und wird somit Teil des Tests.