SQS Theorie

Dies ist eine Zusammenfassung des Stoffes basierend auf den Folien von WS13/14 und aus den Fragen welche im Vowi zu finden waren.

Ersteller: Marco Handl

Keine Garantie auf Vollständigkeit ;)

Block 1 (Biffl)

Was ist Qualität?

Hierzu gibt es keine eindeutige Definition. Jedoch sind

- Kundenzufriedenheit
- Erfüllung der Anforderungen
- Erfüllung von Vorgaben und Richtlinien
- Erfüllung von gesetzlichen Regelungen

Kennzeichnend für Qualität.

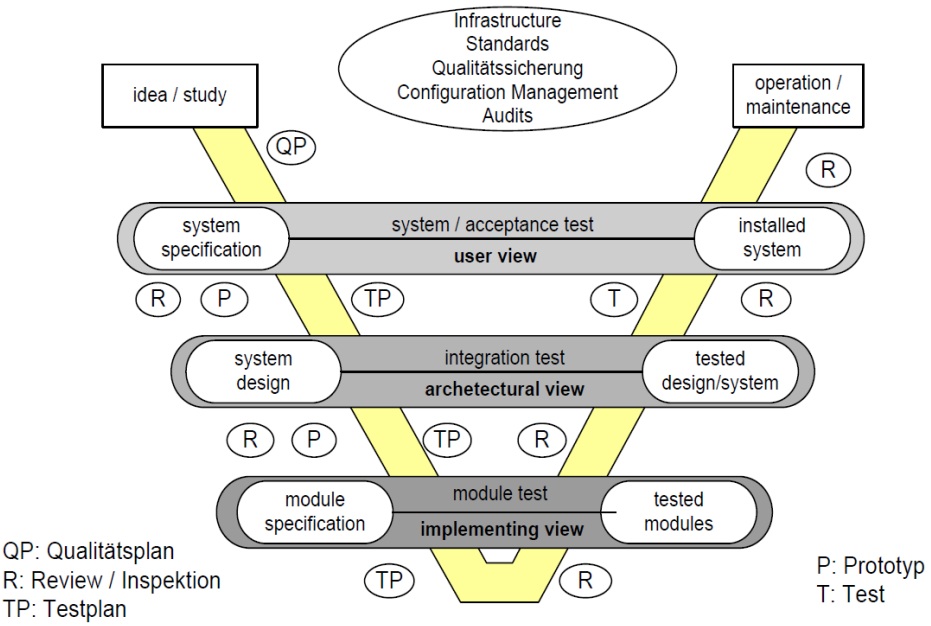
Welche Eigenschaften hat eine qualitativ hochwertige Software?

1. Termingerecht und im Rahmen des Budgets erstellt
2. Für den Anwender verwendbar
3. Für den Professionisten verständlich und änderbar
4. Für den Betreiber effizient und administrierbar

Was ist der Unterschied von Verifikation und Validierung?

- Verifikation: Unter Verifikation versteht man die Prüfung eines Produkts im Hinblick auf die Spezifikation(Pflichtenheft, etc.)
    - Sollte nach jeder Phase durchgeführt werden
- Validierung: Im Rahmen der Validierung wird die Frage beantwortet ob ein Produkt richtig entwickelt wurde. (negative Validierung wäre somit eine Themenverfehlung). Sollte zu Beginn und am Ende durchgeführt werden
    - Am Beginn: Validierung des Pflichtenheftes/der Spezifikation
    - Am Ende : Validierung der erstellten Software

Erkläre das V-Model unter besonderer Berücksichtigung der Qualitätsicherungsmaßnahmen.



Modul – Test, Integrationstests, Systemtest sind hierbei die Testdurchführung auf Synthetischer Seiten der einzelnen Ebenen die auf der Analytischen Seite im Testplan erstellt worden sind.

Nenne 5 Qualitätsfaktoren

1. Korrektheit
2. Zuverlässigkeit
3. Wartbarkeit
4. Testbarkeit
5. Effizient
6. Integrität
7. Verwendbar
8. Wiederverwendbarkeit
9. Portabilität
10. Verknüpfbarkeit

Wie kann man die Wahrscheinlichkeit steigern ein hoch qualitatives Produkt zu erstellen?

- Angemessener **Softwareprozess** (z.B.: Livecyclemodel)
- Angemessene **Vorgehensmodel** (V-Model, RUP, Spiral-Model, agile Ansätze)
- Wirkungsvolle Methode zur Produkt**erstellung**
- Wirkungsvolle Methode zur Produkt-, Prozess**verbesserung** (Review, Inspektion, Testen)

Was bedeutet PDCA, welche Ebenen gibt es in PDCA?

Dieses Prinzip sagt aus, dass die Erkenntnisse aus der Qualitätssicherung nicht nur im aktuellen Projekt genutzt werden sollten, sondern auch in der gesamten Arbeitsgruppe und im gesamten Unternehmen.

Nach dem Planen (Plan) und der Implementierung (Do) werden analytische Qualitätssicherungs-Maßnahmen (siehe unten) angewandt (Check). Die Erkenntnisse werden rückgekoppelt (Act).

Man unterscheidet 3 Ebenen: Unternehmens-, Projekt- und Arbeitspaketebene.

Auf was sollte Qualitäts-Management besonders achten?

1. Hygienefaktoren: Sind solche die unbedingt umgesetzt werden müssen da ansonsten das gesamte System keinen Sinn mehr ergibt.
2. Faktoren die für die Entwickler keine direkten Auswirkungen haben
3. Faktoren die für die Entwickler nicht besonders wichtig sind

Welche Qualitäts-Maßnahmen sind besonders zu unterstützen?

- Hygienefaktoren
- Maßnahmen die vorbeugend wirken sollen

Welche Qualitäts-Maßnahmen sind besonders wirksam?

- Maßnahmen die das richtige machen möglichst einfach und das falsche möglichst schwer machen.
- Rasches Feedback an die Entwickler

Beschreibe Qualitätsmodel nach McCall.

Dieses Modell sieht eine stufenweise Verfeinerung von Qualitätsfaktoren vor.

- **Qualitätsfaktoren**: beschreibt das Verhalten des Systems; Qualitätsfaktoren sind bestimmte die Qualität betreffende Forderungen
- **Qualitätskriterien**: Eigenschaften von Qualitätsfaktoren in Bezug auf Softwareentwicklung
- **Qualitätsmetriken**: sind Messungen die Definierte Aspekte der Qualitätsfaktoren beschreiben

Welche 4 wesentlichen Einflussfaktoren sind in der Softwareentwicklung zu finden?

1. Qualität (Anzahl der Fehler in einem Projekt)
2. Quantität (Anzahl der Programmierten Funktionen)
3. Entwicklungsdauer/Aufwand (kleine/große Projekte)
4. Kosten

Diese 4 Einflussfaktoren wirken wechselseitig auf die Produktivität. (Teufelsquadrat nach Sneech)

Welche Analytischen Qualitätsmaßnahmen können ergriffen werden?

- **S**tatische Prüfung (Review, Inspektion, Audits)
- **D**ynamische Prüfung (Softwaretests)

Welche Konstruktiven Qualitätsmaßnahmen können ergriffen werden?

- **T**echnische Maßnahmen (Verwendung von Prinzipien in der Softwareentwicklung)
- **O**rganisatorische Maßnahmen (Verwendung von Vorgehensmodellen)
- **M**enschliche Maßnahmen (Schulung der Mitarbeiter)

Wie können QS in den Einzelnen Vorgehensmodellen integriert werden?

- V-Modell
  - Analytische Seite: Testplanerstellung in jeder Phase und nach jeder Phase ein Review
  - Synthetische Seite: nach jeder Ebene die Tests des Testplans durchführen
- V-Model-XT
  - QS ist hier ein eigener nicht optionaler Vorgehensbaustein
- RUP
  - Review bei allen Meilensteinen
  - QS ist ständiger Begleiter
- SCRUM
  - Review und Test sind fixer Bestandteil eines Sprints

- Testdriven Development

Erkläre Qualitätssicherung

Qualitätssicherung besteht in der Durchführung von Verifikation und Validierung in jeder Phase der Softwareherstellung.

Erkläre Qualitätsmanagement

Merke: Menge von Aktive HiTech Vorgehensweise

Qualitätsmanagement ist die Menge aller **Aktiv**itäten, **Vorgehensweise**n, **Tech**niken und **Hi**lfsmittel, die sicherstellen, dass ein Software-Produkt vordefinierte Standards erreicht oder übertrifft.

Nennen Sie 3 Prinzipen des Qualitätsmanagements und erklären Sie diese.

- Mehraugenkontrolle: Andere Personen andere Herangehensweise
- Unabhängigkeit bei Qualitätsprüfung
- Produkt und Projektabhängige Qualitätsprüfung
- Frühzeitiges Entdecken und Beheben von Fehlern
- Bewertung der eingesetzten Qualitätsmaßnahmen
- Pair-Programmierung: 2 sehen und wissen mehr.

Paar wichtige Sätze

- Es gibt keine einheitliche Definition für "Qualität" aber verschiedene Ansätze, um "Qualität" begrifflich zu fassen.
- Überprüfung, Beurteilung und Verbesserung der Produkt- und Prozessqualität erfordert messbare Attribute.
- Qualität, Quantität, Kosten und Entwicklungsdauer bilden ein Spannungsfeld in der Softwareentwicklung.
- QS ist integraler Bestandteil von Vorgehensmodellen (z.B. V-Modell (XT), Rational Unified Process, Agile Ansätze usw.)
- Das Prinzip der ständigen Verbesserung von Produkten und Prozessen ist ein integraler Bestandteil von Qualitätsmanagementsystemen und ist auf allen Ebenen (Arbeitspaket, Projekt, Unternehmen) anwendbar.
- Produkte und Prozesse können immer verbessert werden!

Block 2 – Review und Inspektion

Wozu verwendet man Reviews?

Reviews dienen zur qualitativen Beurteilung von Produkten und Prozessen, die Quantitativ nur sehr schwer bis gar nicht beurteilt werden können.

Welche Arten von Review kann man unterscheiden?

- Mit Kunde
  - SRR: Software Requirement Review. eines der Wichtigsten Reviews, hier wird besprochen ob die Anforderungen klar und richtig verstanden wurden. Wird in der Systemspezifikationsphase verwendet
  - PDR: Primary Design Review: Vergleich momentanes Design mit Software Designbeschreibung
  - CDR: Critical Design Review: der Kunde soll bei kritischen Entscheidungen informiert werden und soll auch die Entscheidung mitunterschreiben.
  - IPR: In Prozess Review
- Ohne Kunde
  - **MR: Management Review**
  - **Code Walkthrough:** Der Autor stellt seine Module ablauforientiert vor. Das eignet sich um gemeinsam mit den Reviewern die Qualität der Implementierung zu begutachten und Verbesserungsvorschläge einzuholen. Weiters können neue Mitarbeiter dabei zusehen und sich so mit dem Projekt vertraut machen. Letztendlich entscheidet hier der Autor selbst ob bzw. was überarbeitet wird.
  - **Technical Review:** Dies ist bereits ein formaleres Review, bei dem auch noch ein Moderator sowie ein Protokollführer anwesend sind. Das Produkt wird mit dem Ziel begutachtet, die Stärken und Schwächen zu finden. Das Review-Team gibt eine Empfehlung an das Management, das letztendlich entscheidet.
  - **Inspektion:** Eine Inspektion ist ein formales Review das speziell in frühen Phasen der Entwicklung durchgeführt wird. Dabei wird das zu reviewende Produkt von einem Leser Schritt für Schritt vorgetragen. Das Review-Team versucht schwere Fehler zu finden und ein Protokollführer macht Notizen dazu. Schließlich entscheidet der Moderator wie weiter verfahren wird.

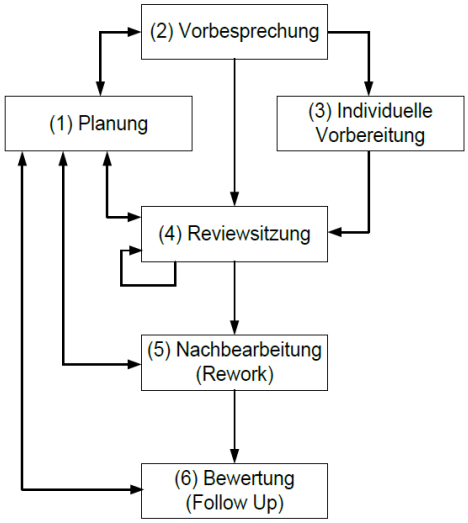
Welche Rollen gibt es bei Review Sitzungen?

Moderator, Leser, Schreiber, Gutachter, Autor

Welche Richtlinien sind für technische Review zu beachten?

- Produkt nicht der Autor soll Reviewed werden
- Verwendung eines Arbeitsplan
- Diskussion sachlich und Kurz
- Keine Lösungen suchen nur Fehler
- Schriftliche Aufzeichnung führen
- Anzahl der Teilnehmer begrenzen
- Teilnehmer sollen gut vorbereitet sein
- Auch Review beurteilen

Wie ist der Ablauf/Aufbau eines Reviews?



Was sind Inspektionen?

Eine Inspektion ist eine formale und wirtschaftliche Methode um Fehler zu finden. Sie sind somit eine spezielle Form der Reviews. Sie können sehr gut in frühen Phasen der Fehlerfindung eingesetzt werden.

Welche Dimensionen hat eine Inspektion?

- Prozess
  - Planung => Übersicht => Fehlersuche => Fehlersammlung => Fehlerbehebung => Follow up
- Produkt
  - Anforderung, Design, Code,
- Rollen
  - Moderator, Leser, Schreiber, Autor, Gutachter
- Lesetechniken
  - Ad-hoc, Checklisten, Szenario-Basiert, Perspektiven-Basiert, Fehlerbasiert

Nenne 5 Lesetechniken und beschreibe sie Kurz.

- Ad-hoc
  - Unstrukturiertes Lesen des Dokuments
  - Erfordert hohe Kenntnisse des Gutachters
  - Eher schlecht da kein strukturiertes Vorgehen
- Checklistenbasiert
  - Durcharbeitung des Dokuments mithilfe einer Checkliste
  - Checkliste ist meist eine generische und eine Projektspezifische
  - Vorteil: Strukturiertes Vorgehen das heißt es ist gut wiederholbar
- Perspektivenbasiert
  - Hierbei nehmen die einzelnen Gutachter unterschiedliche Rollen ein
  - Idee ist das durch die Unterschiedlichen Sichten unterschiedliche Fehler gefunden werden
- Szenario basiert – Usage-Based-Reading => Best Practice
  - Einteilung des Projektes in Use Cases (sollte sowieso vorliegen)

    - Jeder Use-Case wird nach Wichtigkeit Beurteilt
    - Starte mit dem Wichtigsten => Überprüfung der relevanten Teile => Reporting von Fehlern => Auswahl des nächsten Use-Case
    - Idee ist das Fehler in wichtigen Use-Case größere Priorität haben
- Fehlerbasiert
    - Spezielle Form der Checklistenbasierten Lesetechnik
    - Fehler werden in Fehlerkategorien eingeteilt damit sie besser gefunden werden können

Wie können Anforderungen Klassifiziert werden?

Stichwort: FURPS deutsch: FBZPW

- **F**unctional (Funktional)
    - Features, Fähigkeiten
- **U**sability (Benutzerfreundlichkeit)
    - Menschliche Faktoren, Hilfe
- **R**eliability (Zuverlässigkeit)
    - Häufigkeit eines Ausfalls, Wiederherstellbarkeit
- **P**erformance
    - Durchsatz, Antwortzeiten
- **S**upportability (Wartbarkeit)
    - Änderbarkeit, Modularität, Wartung

Welche Typen von Anforderungen gibt es?

- Funktionale Anforderungen
    - Beschreibt **was** das System tut
    - Typisch hierfür ist: Das System kann Funktion x ausführen
- Nicht Funktionale Anforderungen
    - Erwünschte Charakteristika
    - Qualitätsanforderungen, Frage nach **wie gut**
- Constrains
    - Einschränkungen des Systems
- Klassifizierungen
    - Stichwort FURPS

Wie können Anforderungen Kategorisiert werden?

- Testbare Aussagen
- Funktionen oder
- Qualitative Anforderungen
    - Definition von Qualität
    - Wie ist Qualität messbar?
    - Wieviel Qualität ist genug

Was ist ein Anwendung-Szenario?

Funktionale Anforderungen bilden ein Anwendungs-Szenario.

Nennen Sie Vorgehensweisen zu Sicherung/Verbesserung der Qualität von Projekt-Anforderungen?

Welche Kriterien sollten brauchbare Anforderungen erfüllen?

- Notwendige Kriterien
  - Typ(Funktional, nicht Funktional)
  - Rückverfolgbarkeit: Ist die Anforderung in der Projektbeschreibung auffindbar
  - Überprüfbar: Ist die Anforderung testbar
- Erstrebenswerte Kriterien
  - Knapp verständliche Beschreibung
  - Umsetzbar
  - Konsistent

Wie ist ein Anwendungsfall aufgebaut?

- Allgemein
  1. Titel
  2. Kurzbeschreibung
  3. Welche Aktoren sind beteiligt
  4. Vorbedingung
  5. Nachbedingung
- Flow of events
  1. Hauptszenario
  2. Alternativszenario
  3. Fehlerszenario
- Nichtfunktionale Anforderungen

Block 3 – Testen

Was ist das Ziel der Softwareentwicklung und durch was kann dies erreicht werden?

Ziel der SE ist es hoch **qualitative** und **testbare** Software zu erstellen. Dies kann erreicht werden durch:

- Anwendung von Software-Vorgehensmodel(V-Model, SCRUM,…)
- Konstruktive Methoden (Model-Driven, Test-Driven)
- Analytische Methoden (Review, Inspektion, Testen)

Welche Phasen hat man im Testprozess?

1. Planung & Steuerung
2. Analyse/Design
3. Durchführung
4. Auswertung und Berichterstellung
5. Abschluss

Was ist der Unterschied zwischen Reviews und Testen?

Review werden verwendet um Fehler in frühen Phasen der Entwicklung zu erkennen.

Testen dient der Verifikation(Testen gegen die Spezifikation) bzw. der Validierung(Testen gegen Kundenanforderungen) von bereits implementierter Software.

Welche Schritte gibt es bei Test –Driven-Development?

1. Think
    - Auswahl der zu implementierenden Anforderung
    - Spezifikation der Testfälle
2. Red
    - Implementierung und Ausführung der Testfälle
    - Alle Tests müssen fehlschlagen
3. Green
    - Implementierung der Komponenten und Klassen
    - Wenn alle Tests grün sind weiter zu 4 sonst auf 3 bleiben
4. Refactor
    - Änderung und Optimierung der Implementierung ohne Änderung der Funktionalität
    - Testfälle dürfen nicht fehlschlagen => weiter bei Schritt 1.

Was versteht man unter Refactoring? Was wird dadurch erreicht?

Refactoring bezeichnet in der Software-Entwicklung die manuelle oder automatisierte Strukturverbesserung von Quelltexten unter Beibehaltung des beobachtbaren Programmverhaltens. Dabei sollen die

- Lesbarkeit
- Verständlichkeit
- Wartbarkeit
- Erweiterbarkeit

verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken.

Erkläre der Begriff Testsuit und wann ist ein Test erfolgreich?

Tests bestehen aus einer Menge von Testfällen dieses Set nennt man Testsuit.

Ein Test ist erfolgreich wenn man damit einen Fehler findet

Aus was besteht ein Testfall?

- **V:** Vorbedingung
- **E:** Eingabewert
- **A:** Aktion zur Eingabe
- **R:** erwartetes Ergebnis (Rückgabewert/Resultate)

Welche Fälle von Testfällen gibt es?

- Normalfall: sollte bei korrekter Implementierung funktionieren
- Fehlerfall: sollte bei korrekter Implementierung zu einer Fehlermeldung/Exception führen
- Sonderfall: an der Grenze zwischen Normalfall und Fehlerfall

Welche Abdeckungsarten gibt es bei der Testabdeckung?

- Abdeckung aller Funktionen
    - jede Funktion wird mindesten einmal ausgeführt
- Abdeckung aller Eingabeklassen
    - Für jede Funktion wird jede Eingabeklasse mindestens einmal verwendet
- Abdeckung aller Ausgabeklassen
    - Für jede Funktion wird jede Ausgabeklasse mindestens einmal erzeugt.

Was ist eine Äquivalenzklasse und wie findet man diese?

Zerlegung einer Menge von Daten (Ein-,Ausgabewerte) in Untermengen(Klasse) welche äquivalente Ergebnis oder Auswirkungen liefern.

Vorgehensweise:

1. Finden eines Eingabevektors (A,B,C,…)
2. Finde für jede Komponente des Vektors seine ungültige und gültige Äquivalenzklasse und wähle einen Wert davon aus. (A1,A2,… ; B1,B2,… ; ….)
3. Kombination dieser Werte bestimmt die Intensität. [(A1,B1), (A1,B2), …..]

Was versteht man unter Grenzwertanalyse?

Bei der Grenzwertanalyse werden die Testdaten ebenfalls in Äquivalenzklassen unterteilt, jedoch wird hier nicht ein beliebiger Wert aus der Klasse verwendet sondern der beziehungsweise die Grenzwerte der Klasse.

Was ist Value-Based-Testing wie kann man es Unterteilen?

Value-Based-Testing beruht darauf dass nicht alle Testfälle bzw. alle Fehler gleich viel Wert sind.

Man unterteilt dies in

- Requirements-Based-Testing
    - Welche Anforderungen bringen den Kunden am meisten
- Risk-Based-Testing
    - Welche Risiken gefährden diesen Nutzen. (Auf einer Anwendung bauen 10 weitere auf)
- Test Case Selection
    - Testfälle welche all diese Risiken , Nutzen abdecken

Was sind Test-Stufen was Test-Typen und wie hängen diese zusammen?

Teststufen stellen unterschiedliche Tests auf unterschiedlichen Ebenen des Softwareentwicklungsprozesses dar. Test-Typen sind spezielle Ausprägungen von Tests. Eine Teststufe besteht aus einem oder mehreren Test-Typen.

Beispiele:

Test-Stufen:

- Privater-Test, Unit-Test, Integrationstest, System-Test, Regressionstest, Akzeptanztest

Test-Typen:

- Review, Funktionaler-Test, Massentest, Stresstest, Performancetest

Erkläre kurz die einzelnen Test-Stufen.

- Privater Test
  - Vom Entwickler vor der Freigabe durchgeführter meist undokumentierter Test.
- Unit-Test
  - Test des Moduls bzw. Klasse gegen die Spezifikation.
  - Einzelne externe Komponenten können mittels Stubs oder Mocks simuliert werden.
- Integrationstest
  - Test der Interaktion zwischen den einzelnen Modulen.
  - Es können mehrere Integrationsstufen eingeführt werden.
  - Bottom-up, Top-Down, Big-Bang Strategien werden verwendet.
- Systemtest
  - Test des Systems gegen die Kundenanforderung (Validierung)
  - Basiert auf: Anforderungsspezifikation, Use-Cases, Geschäftsprozesse
- Akzeptanztest
  - Wie Systemtest jedoch mit bzw. beim Kunden

Erkläre Integrationstest und die Vorgehensweisen Bottom-Up und Top-Down?

Bottom-Up:

Hier wird zuerst die Basis ausprogrammiert. Um diese zu Testen werden sogenannte Treiber Objekte erstellt die diese Objekte erzeugen und Testen. Nach jeder Integrationsstufe wird der Treiber verändert und eine Ebene höher angesiedelt.

Top-Down:

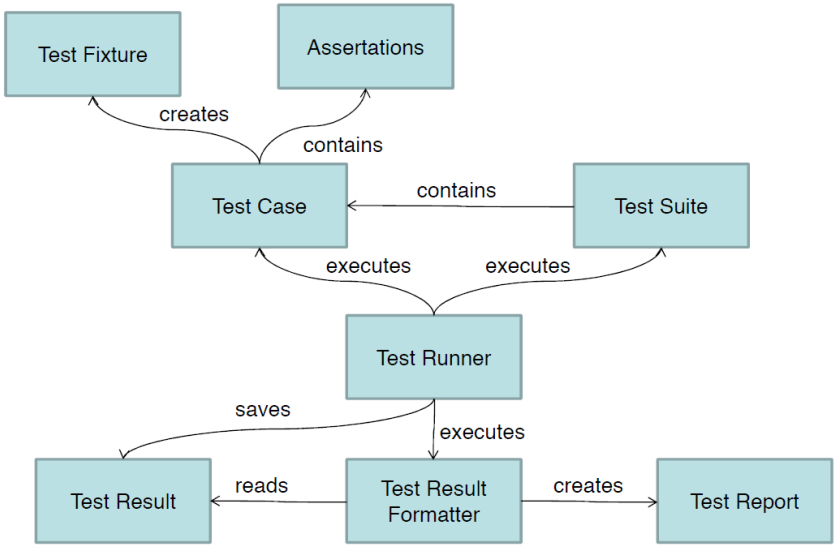
Hier werden zuerst die obersten Module entwickelt auf welchen der Testtreiber aufsetzt. Alle darunterliegenden Module werden mithilfe von Stubs simuliert. Diese Stubs werden je nach Integrationsstufe im echten Code befüllt. Vorteil das es nur einen Testtreiber gibt.

Block 4 - Softwaretesten

Was ist die Grobstruktur eines Unit-Tests?

1. Set-Up (Vorbereitung)
2. Exercise (Ausführung)
3. Verify (Auswertung)
4. Tear-Down (Nachbearbeitung / zerstören)

Aus welchen Bestandteilen besteht das xUnit Framework?



Test Fixture: Definierter Status der Software/des Systems

Was ist der wesentliche Unterschied zwischen JUnit 3 und 4?

In Version 3 mussten die Klassen noch expliziert von der Testklasse abgeleitet werden, weiters musste auch eine spezielle Namenskonvention eingehalten werden. In Version 4 geschieht dies alles über Annotationen.

Welche Livecycle Methoden gibt es in JUnit 4, erkläre diese kurz?

- @BeforeClass: wird einmal bei der Initialisierung der Klasse ausgeführt.
- @AfterClass: wird einmal vor verlassen der Klasse ausgeführt
- @Before: wird vor jeder Test-Methode ausgeführt
- @After: wird nach jeder Test-Methode ausgeführt

Was sind Assertions?

Assertions sind Zusicherungen. Sie bestehen aus einem überprüfenden Teil und einem Fehlertext welcher ausgegeben wird falls der überprüfende Teil „false“ liefert.

Wieso teilt man Tests in unterschiedliche Kategorien ein?

Wenn man Test Kategorien zuweist kann man den Test auf eine spezielle Kategorie beschränken.

Was sollte bei Unit-Test beachtet werden?

- Tests sollten prinzipiell unabhängig voneinander laufen, Reihenfolge egal.
- Pro Test sollte nur eine Funktion überprüft werden
- Klassen und Tests liegen in verschiedenen Verzeichnisstrukturen aber im selben Packet
- Definition und Einhaltung der Namenskonvention

Welche Test-Doubles gibt es, erkläre diese kurz?

- Dummy Objekt: keinerlei Funktionalität besitzt nur Methodenköpfe
- Fake Objekt: Ausführbare Implementierung liefert jedoch keine Echtdaten
- Stub: Liefert vordefinierte Werte an den Aufrufer
- Mock: Liefert vordefinierte Werte an den Aufrufer überprüft jedoch die Übergabeparameter und wird somit Teil des Tests
- Spy: Proxy-Objekt. Einzelne Methoden werden durch eigene Implementierungen ersetzt.

Block 4 – Code Coverage

Welche Arten von Überdeckungstests kann man unterscheiden?

Merke: die **Anweisung** an einen **Zweig**, **Bedingt** einen **Pfad**

- C_0 – Anweisungsüberdeckung
  - Jeder Knoten des Kontrollflussgraphen wird durchlaufen
  - Spürt nicht ausführbaren Code auf
- C_1 – Zweigüberdeckung
  - Jede Kante wird mindestens durchlaufen
  - Spürt nicht ausführbare Programmzweige auf
  - Jede Entscheidung muss zumindest einmal true und einmal false liefern
- C_2 – Bedingungsüberdeckung
  - Behandelt die Überprüfung von Entscheidungen. Zwei Varianten:
    - einfache Bedingungsüberdeckung: Alle atomaren Entscheidungen (True oder False) müssen einmal durchlaufen werden.
      Bsp.: (x||y) → 1.Test x=true, y=true ; 2.Test x=false, y=false
    - mehrfache Bedingungsüberdeckung: Alle möglichen Kombinationen von atomaren Entscheidungen müssen einmal durchlaufen werden
      Bsp.: (x||y) → alle 2^n möglichen Kombinationen (hier 4).
- C_3 – Pfadüberdeckung
  - Alle möglichen Pfade (Knotenfolgen) vom Anfang bis zum Endknoten.
  - Meist nicht durchführbar da die Anzahl der Kombinationen zu hoch.

Block 5 – Automatische Qualitätssicherung

Welchen Fokus kann ein Test haben?

- Funktionale Anforderungen
  - Features
- Integration
  - In ein spezifisches Umfeld
- User Interface
  - Bedienbarkeit, WAI
- Nicht Funktionale Anforderungen
  - Performance, Wiederhochlaufzeit
- User Akzeptanz
  - UI-Design

Was sind die Probleme von Unittest mit Datenbanken wie kann man das lösen?

Unittest auf Datenbanken haben das Problem das man immer von einem bestimmten Datenbankzustand ausgehen muss dieser aber oft nicht einfach hergestellt werden kann z.B.: weil das System schon in Betrieb ist.

Lösung hierbei ist die Verwendung von DBUnit welches genau dieses Problem löst.

JUnit

Ermöglicht das Schreiben von Unit-Testfällen in Java.

HTTP/HTML-Unit

Simulieren eines HTTP- bzw. HTML-Client. Man kann dadurch von der Testumgebung auf Websites zugreifen daraus folgt es können Befehle an den Webserver gesendet werden und die Rückgabe überprüft werden. Vor allem dienen diese Bibliotheken dem Test von Webapplikationen.

JMeter

Diese Bibliothek kann verwendet werden um Performancetests durchzuführen. Es kann damit beispielsweise ein Webserver mit vielen Anfragen konfrontiert und das Verhalten beobachtet und gemessen werden.

Diese Tools muss man aber verstehen bevor man sie einsetzt! Bei falscher Anwendung haben die Ergebnisse keine Aussagekraft.

Automatisierte Code-Quality-Checks

Auch die Qualität des Quelltextes selbst kann überprüft werden. Quellcode-Qualität zeichnet sich durch Aspekte wie ausreichende Dokumentation, Einhaltung von Namens und Formatierungskonventionen, Vermeidung von dupliziertem Code und ähnlichem aus. Natürlich kann man solche Messungen stören indem man einfach ”Dummy-Kommentare“ einfügt. Es ist daher nicht sinnvoll von den Mitarbeitern einen bestimmten Prozentsatz und Dokumentation zu fordern, da es stark auf den Kontext ankommt ob das sinnvoll ist oder nicht.

Was kann man Code-Quality-Checks überprüft werden?

- Kommentare
- Imports (*)
- Duplizierter Code
- Namenskonvention
- Modifier

Profiling

Bei lauffähigen Anwendungen lässt sich ein Profil erstellen. Das sind Berichte darüber wie oft eine Funktion aufgerufen wird, wie lange ein Durchlauf dauert, etc. . Entsprechende Tools hängen sich dazu in die Java-VM ein. Man kann diese Informationen nutzen um etwa die Performance zu verbessern (um gute Ergebnisse erzielen zu können muss man wissen wo die meiste Zeit verbraucht wird).

Continuous Integration

Darunter versteht man die Zusammenschaltung vieler Entwicklungstools zu einem ganzen Entwicklungssystem, das Routineaufgaben periodisch und automatisiert durchführen kann. Damit werden tägliche Builds erstellt, automatische Testläufe durchgeführt (Unit-Tests) und Benachrichtigung an die Teammitglieder versendet.

Wichtig dabei ist, dass auch das Repository, in das der Sourcecode eingecheckt wird, gekoppelt ist. Nach einem Check-In sollte automatisch ein Build erzeugt werden. Wenn man Maven manuell anstoßen muss, dann ist das noch nicht ”Continuous Integration“. ”Cruise Control“ und ”Apache Continuum“ sind entsprechende Tools, die den ganzen Vorgang unterstützen. Auch Bug-Tracking-Tools (mit denen alle Stakeholder gefundene Bugs bekanntgeben können) sollten berücksichtigt werden.

Was sind Kennzeichen für Probleme oder schlechte Designs?

- Lange Klassen
- Doppelter Code
- Lange Parameterlisten
- Langsame Klassen
- Breite Schnittstelle der Klasse
- Öffentliche Objektvariablen