Prüfungsfragen Stand Juni 2011

1. *Welche Typen der Wartung werden unterschieden? Erklären sie die unterschiedlichen Typen der Wartung und erläutern sie warum diese Klassifizierung/Unterscheidung erforderlich ist.*

   Wartung ist ein wesentlicher Bestandteil des SW-Lebenszyklus.
   Korrektive Wartung
   - Ausbesserung von Fehlern
   - Abweichungen von Spezifikation
   - Nicht-Erfüllung von SLAs (Service-Level-Agreement)

   Präventive Wartung
   - Verhinderung zukünftiger Fehler

   Adaptive Wartung
   - Anpassung an geänderte Umgebung
   - Änderung verwendeter SW
   - Änderung der Schnittstellen
   - Änderung der Hardware

   Perfektionierende Wartung
   - Verbesserung der Applikation
   - keine funktionale Änderung
   - Benutzerführung
   - Performance
   - Wartbarkeit

   Erforderlich, damit eine optimale Wartung möglich ist. Man muss so nur spezielle Kategorien bearbeiten und nicht alles überprüfen. Weiters können unterschiedliche Personen versch. Wartungstypen übernehmen. So kann dieser Prozess ebenfalls parallelisiert werden. Vermutlich werden sich diese Typen jedoch oftmals überschneiden.

2. *Objektorientierte Kopplung und Kohäsion und Zusammenhang und Auswirkung auf Wartung*

   Gehören zu Konzepten von <u>wartungsfreundlicher</u> Implementierung.

   Kopplung
   - Maß der Abhängigkeit <u>zwischen</u> Klassen
   - hohe Kopplung zeugt von einem verbesserungswürdigen Programmierstil

   Kohäsion
   - Maß der Abhängigkeit <u>innerhalb</u> von Klassen
   - hohe Kohäsion gewährleistet durch Methodengruppierung nach Zweck

   Ziel: Geringe Kopplung erreicht durch hohe Kohäsion.

3. *Was ist der Unterschied zwischen Internationalisierung und Lokalisierung?*

   Internationalisierung
   - Anpassung an andere Kulturen und Sprachen, durch spezielle Gestaltung der SW.
   - Lokalisierbarkeit: Wie gut lässt sich ein Produkt übersetzen ohne in den Source-Code eingreifen zu müssen?
   - Lokalisierung ist ein Teil von Internationalisierung.
   - kein „hard-coden“ von Nachrichten und Beschriftungen

- extrahieren in Ressource Files
- Schriftgrößen nicht einschränken
- Nachricht nicht stückeln
- identische Nachrichten in unterschiedlichem Kontext mehrfach speichern
- gleich bei Implementierungsbeginn vorsehen – nachträgliche Umsetzung sehr teuer

Lokalisierung

- Übersetzungsprozess
- Anpassung von lokalen Datenformatierungen (Zahlen, Datum)
- Hinzufügen von spezifischen Code
- für jede Sprache/Kultur separat notwendig
- vollständige Lokalisierung sehr aufwendig

4. *Erklären Sie das Modell des komponentenbasierten Systementwurfs und seine Bedeutung für die Wartung.*

Modularisierung: Folie: Design_Architektur, 101ff
Aufgabenteilung: Jedes Modul hat eine genau festgelegte Verantwortung, die es zu erfüllen hat. Bei auftretenden Fehlern kann aufgrund dieser bekannten Verantwortungen die Quelle eines Fehlers eng eingegrenzt werden. Auch Änderungen lassen sich dadurch gezielt an der richtigen Stelle durchführen.
Skalierbarkeit: Durch Aufteilung in Module und die definierte Zusammenarbeit können einem System Module hinzugefügt werden, welche die Funktionalität erweitern. Module können ebenso entfernt werden, ohne die Funktionalität von anderen Modulen zu beeinflussen.
Wartbarkeit: Einzelne Module können gegen neuere Versionen ausgetauscht werden, ohne das Verhalten von anderen Modulen zu beeinflussen.

5. *Unterschied Anforderungsanalyse UP und XP*

RUP

- Das Analysemodell soll die Anforderungsanalyse auf Vollständigkeit, Konsistenz und Machbarkeit überprüfen.
- Das System wird nach objektorientierten Gesichtspunkten von innen beschrieben.
- Das System wird in die Klassentypen Schnittstelle, Kontroller und Entitäten unterteilt. Durch die Zusammenarbeit dieser Klassen sollen alle Systemfunktionen für die Benutzer verfügbar gemacht werden.
- Das Analysemodell entspricht einer Aufteilung des Systems nach dem MVC-Paradigma.
- Dynamische Diagramme dienen zur Darstellung von komplexen Abläufen in oder zwischen einzelnen Anforderungen oder innerhalb des Analysemodells.
- Dynamische Diagramme werden nur für besondere Fälle eingesetzt, in denen das Verständnis der dynamischen Funktionsweise des Systems nicht trivial erkennbar ist.
- Die Analyse stellt die Vorbereitung für den Entwurf dar. Während imEntwurf die technische Umsetzung des Systems geplant wird und daher alle technischen Faktoren wie Zielumgebung oder Systemarchitektur eine große Rolle spielen, werden diese in der Analyse außer Acht gelassen.

XP

- Anforderungen werden mit User-Stories beschrieben. Diese werden vom Kunden verfasst
- und dienen auch als Grundlage für die Aufwandsabschätzung.
- Der Kunde muss das ganze Projekt über vor Ort sein.
- Die Metapher ist, dass der Kunde und das Entwicklerteam
  - eine gemeinsame Sprache im Bezug auf das System sprechen
  - eine gemeinsame Vision/Sicht auf das Problem haben

- Generativität: Projektmitarbeiter und Kunde müssen sich gegenseitig umeinander "kümmern".
- Architektur ?

6. *Was sind die wesentlichen Qualitätsanforderungen zur Verifikation von Software-Anforderungen?*
   Vollständig: alle Anforderungen des Kunden müssen explizit beschrieben sein, es darf keine impliziten Annahmen des Kunden über das zu entwickelnde System geben.
   Eindeutig definiert / abgegrenzt: präzise Definitionen helfen, Missverständnisse zwischen Entwickler und Auftraggeber zu vermeiden.
   Verständlich beschrieben: damit sowohl der Auftraggeber als auch der Entwickler mit vertretbarem Aufwand die gesamte Anforderung lesen und verstehen können.
   Atomar: es darf nur eine Anforderung pro Abschnitt oder Satz beschrieben sein.
   Identifizierbar: jede Anforderung muss eindeutig identifizierbar sein (z. B. über eine Kennung oder Nummer).
   Einheitlich dokumentiert: die Anforderungen und ihre Quellen sollten nicht in unterschiedlichen Dokumenten stehen oder unterschiedliche Strukturen haben.
   Notwendig: Sind gesetzliche Vorschriften unabdingbar.
   Nachprüfbar: die Anforderungen sollten mit Abnahmekriterien verknüpft werden, damit bei der Abnahme geprüft werden kann, ob die Anforderungen erfüllt wurden (Ableitung von Testfällen aus den Abnahmekriterien).
   Rück- und vorwärtsverfolgbar: damit einerseits erkennbar ist, ob jede Anforderung vollständig erfüllt wurde und andererseits für jede implementierte Funktionalität erkennbar ist, aus welcher Anforderung sie resultiert, also nicht Überflüssiges entwickelt wird.

7. *Was ist der Unterschied zwischen Verifikation und Validierung von Anforderungen.*
   Validierung: Überprüfen, ob die Anforderungen wirklich abbilden, was der Benutzer / Auftraggeber benötigt.
   Anforderungs-Validierungsfehler sind die teuersten im SE Prozess: Ausbessern eines Validierungsfehlers nach Auslieferung kostet ca. 100x mehr als eine Fehlerausbesserung in der Entwicklung.
   Verifikation: möglichst hohe Abdeckung des Systems mit entsprechenden Testfällen

8. *Was sind funktionale, nicht-funktionale und Domain-Anforderungen? Was gibt es für Typen nicht-funktionaler Anforderungen?*

   funktionale Anforderungen
   - Beschreibung der Funktionen/Services, die das System zur Verfügung stellen soll.
   - Wie reagiert das System auf bestimmte Eingaben und Situationen?

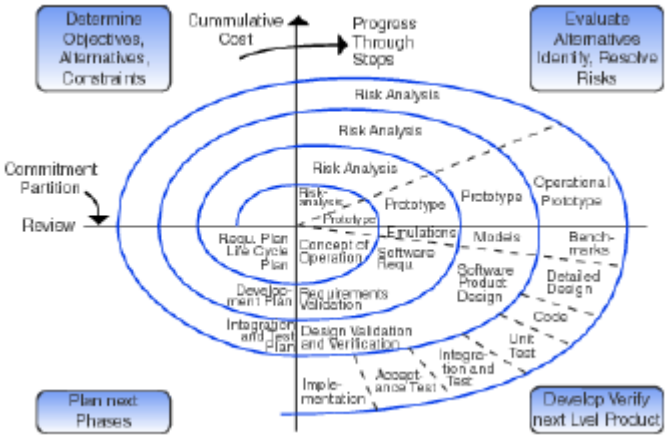
   nicht-funktionale Anforderungen
   - Anforderungen an Funktionen/Services, welche nicht deren Verhalten beschreiben, sondern Eigenschaften des Produktes.
   - Qualitätseigenschaften
   - Entwicklungsprozess
   - Standards
   - Normen
   - gesetzliche Rahmenbedingungen
   - Typen: Produkt, organisatorische und externe Anforderungen

   Domänenanforderungen
   - funktionale und nicht-funktionale Anforderungen, die von der Domäne des Systems definiert werden
   - stellen besondere Eigenschaften/Einschränkungen des Gebietes dar
   - oft inhärent und nicht explizit pro Projekt dokumentiert

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

   Vorgehensmodelle werden benötigt da SW-Produkte immer komplexer und umfangreicher werden. Ein Vorgehensmodell organisiert einen Prozess der gestaltenden Produktion in verschiedene, strukturierte Phasen. Diese Phasen werden durch spezielle Techniken und Methoden organisiert. Die Aufgabe eines Vorgehensmodells ist es, die in einem Gestaltungsprozess auftretenden Aufgabenstellungen und Aktivitäten in einer logischen Ordnung darzustellen.
   Es gibt keine einheitliche Systematik der SW-Entwicklung, daher ist je nach Projektart ein geeignetes Vorgehensmodell zu wählen.

   Spiralmodell nach Boehm
   - iterativ aufgebaut
   - berücksichtigt Risiken
   - Prozess ist in 4 Phasen gegliedert (werden mehrmals durchlaufen)
     - 1. Definition von Zielen
     - 2. Risikoanalyse
     - 3. Durchführen von Arbeitsschritten
     - 4. Planen der nächsten Phase



   - Anzahl der Durchläufe erst bei Durchführung erkennbar, abhängig von Risiken

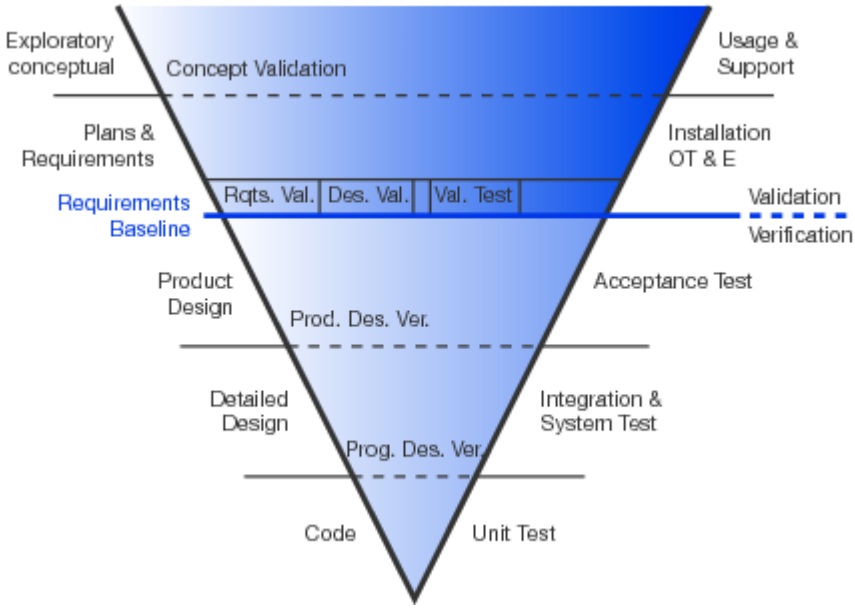
10. *Was sind hybride Vorgehensmodelle, nennen Sie Beispiele und welche Bedeutung haben sie in der Praxis?*
    - wurden 2000 entwickelt
    - Kombination aus theoretischem Wissen des Software Project Engineerings und den praktischen Anforderungen der täglichen Entwicklungsarbeit
    - Vereinigung von V-Modell XT und Extreme Programming → auch als Vorgehensmodell zur strukturiert agilen Softwareentwicklung bezeichnet
    - Flexibilität

11. *Erklären des V-Modells + Skizze - Wofür steht das XT im V-XT Modell und Ziele des XT Modells, Arten des Tailorings*

    Basis für viele Entwicklungsmodelle im öffentlichen Sektor.

Eigenschaften

- korrelierende Phasen
- Validierung und Verifikation



V-Modell XT

XT....extreme Tailoring, extendable (Anpassung an spezielle Bedürfnisse)

- bessere Anpassungs- und Anwendungsmöglichkeit bei unterschiedlichen Projekten und Organisationen
- Skalierbarkeit verschiedene Projektgrößen
- Bessere Änderungs- und Erweiterungsmöglichkeiten des Modells
- Anpassung an aktuelle Normen und Vorschriften
- Betrachtung des gesamten Systemlebenszyklus
- Anpassung an organisationsspezifische Gegebenheiten

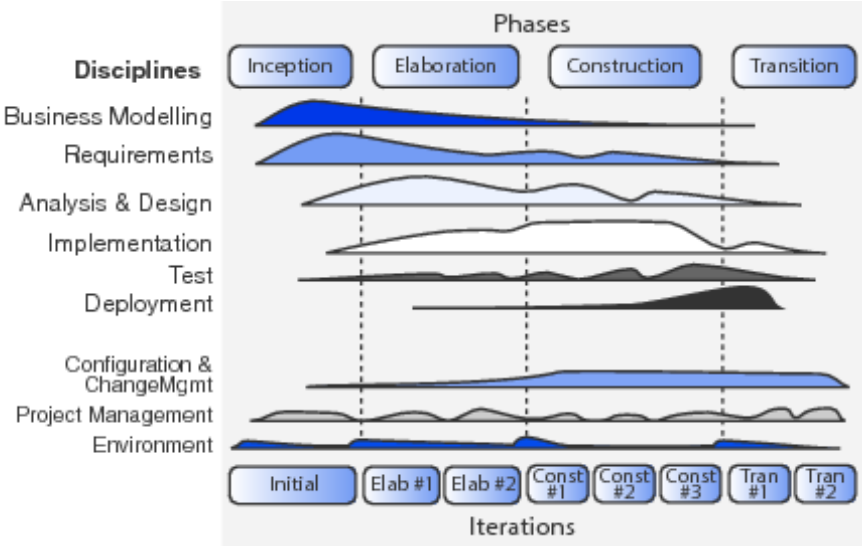
Eigenschaften:

- Ziel –und Ergebnisorientierte Vorgehensweise
- Produkte stehen im Mittelpunkt stellen zentrale Projektergebnisse dar
- Projektdurchführungsstrategien und Entscheidungspunkte geben die Reihenfolge der Produktfertigstellung vor und Projektverlauf
- Auf Basis der Bearbeitung und Fertigstellung der Produkte wird die detaillierte Projektplanung und Steuerung vorgenommen

12. Erklären sie die Grundcharakteristik, die wesentlichen Elemente und Rollen des Unified Process (UP).

Ist ein bekanntes iteratives und inkrementelles SW-Development Process Framework.

- Rational Unified Process (RUP) ist eine spezielle Verwirklichung des Modells von IBM. Dieses wird ständig weiterentwickelt.
- Meist verbreitetes und detailliert beschriebenes Prozessmodell
- Beruht auf folgende Vorgehensweisen:
  - SW iterativ entwickeln
  - Anforderungen managen
  - Verwenden von komponentenbasierten Architekturen
  - SW visuell modellieren
  - SW-Qualität kontinuierlich prüfen
  - Änderungen kontrolliert in SW einbringen

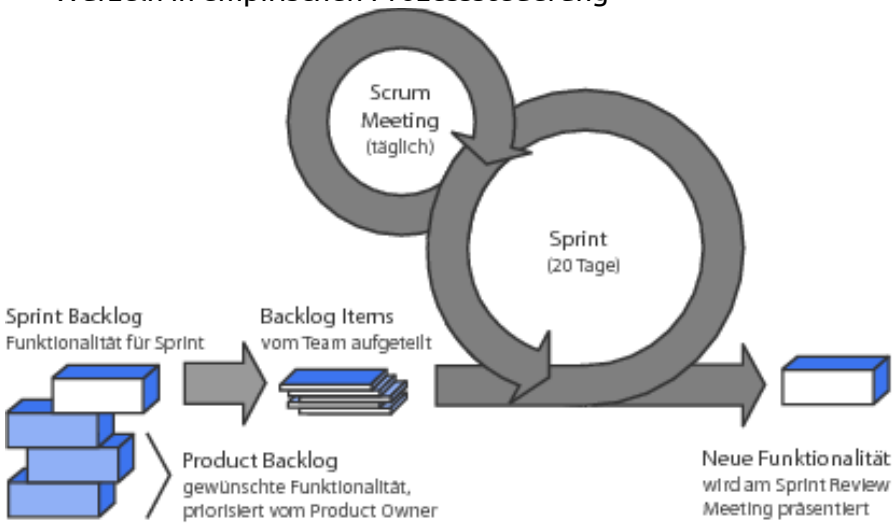


13. *Wartung im RUP: welche Rollen, welche Dokumente, warum wird zwischen Fehler und Erweiterungswunsch unterschieden?*

14. *Erklären sie die Grundcharakteristik (Grundlagen), die wesentlichen Elemente und Rollen des SCRUMs Prozessmodell.*

Grundlagen

- Personen geprägtes Prozessmodell
- hyperproduktive Produktentwicklung nach Sutherland, Schwaber & Beedle
- hohe Produktivität
- wenig Risiko und Ungewissheit
- größerer Komfort für Projektmitglieder
- Wurzeln in empirischen Prozesssteuerung



Elemente

- *Sprint:* so schnell wie möglich zum Sprint-Goal
- *Daily Scrum:* Meeting selber Ort/Zeit
- *Product Backlog:* nach Dringlichkeit geordnete Liste der Anforderungen (von allen einsehbar)
- *Increment:* SW-Produkt entsteht aus Inkrementen (erste und spätestens zweite Sprint muss ein lauffähiges System (Kernfunktionalität) liefern)

- *Burndown Chart:* Scrum-Master trägt in dieses Diagramm Tagesaktuelle Aufwandsabschätzungen für den aktuellen Sprint ein

Rollen

- *Product Owner:* Erstellung und Priorisierung des Product Backlogs, lenkt das Team (frei von Störungen)
- *Scrum Team:* Umsetzung, der im Sprint Backlog geforderten Funktionalität, Erreichung des Sprint-Goals
- *Scrum Master:* Dynamik zur ungestörten Entfaltung bringen, achtet auf Grundsätze und Regeln und schottet das Team von äußeren Einflüssen ab

15. Was für Qualitätsanforderungen gibt es beim extreme Programming?

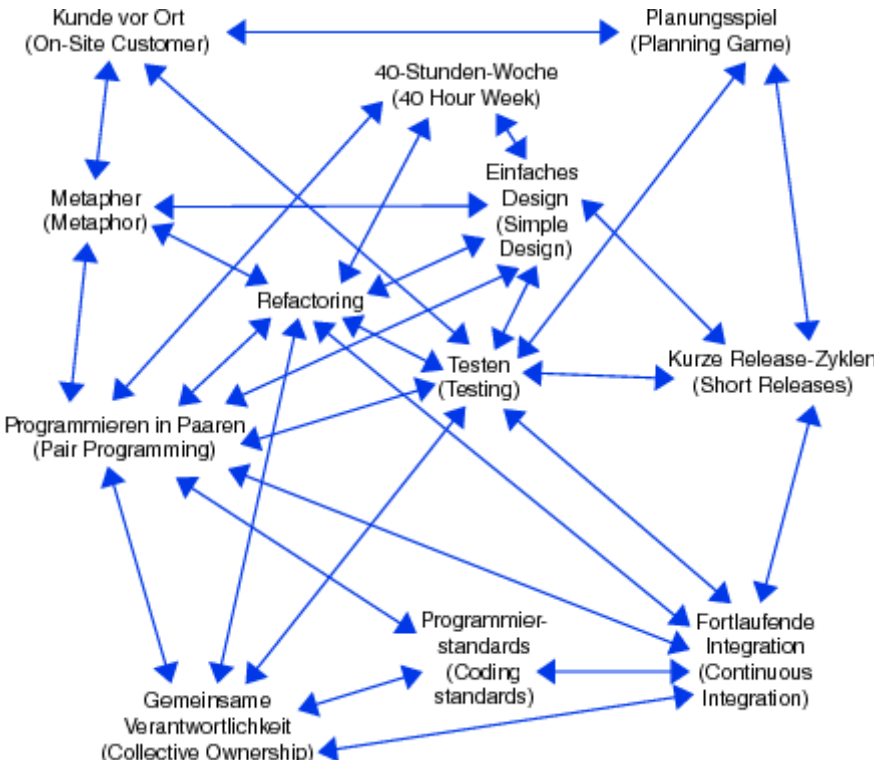
Entwicklung von SW in kleinen bis mittelgroßen Teams. Arbeit unterliegt raschen Änderungen der Anforderungen.
Hauptziel: termingerechtes Abliefern auftragsgemäßer SW

charakteristische Merkmale

- eigenes Wertesystem
- viele Prinzipien und ein System von Techniken, welche einander gegenseitig unterstützen

Qualitätsanforderungen

- Kommunikation (zwischen Beteiligten)
- Einfachheit (einfachstes Design)
- Feedback (Wissen wo man steht empirisches Prozessmodell)
- Mut (zu Offenheit und Transparenz, Änderungen durchführen)
- Respekt (vor sich selbst/Team/Produkt)

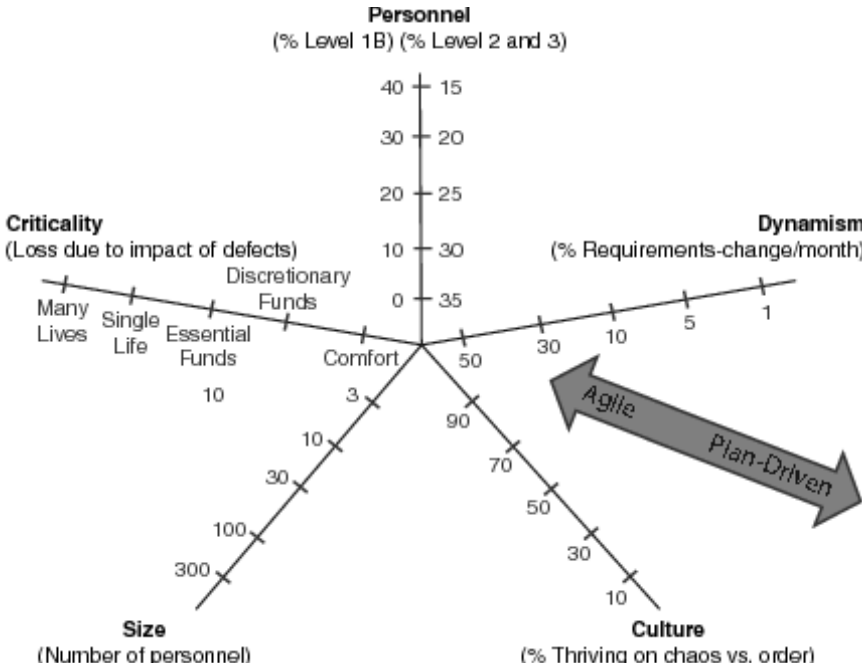


Rollen

- Entwickler/Programmierer
- Kunde
- XP-Trainer

- Tracker
- Tester
- Berater
- Big Boss

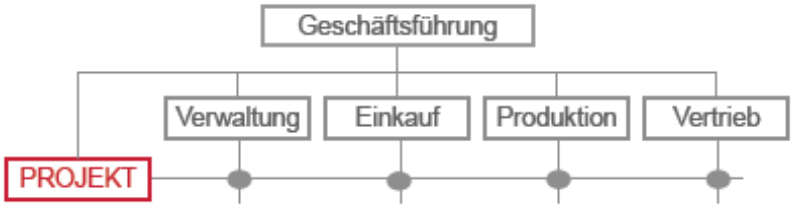
16. *Erklären und Erläutern Sie die Entscheidungskriterien für „Agile vs. Plan-Driven" Methoden von Boehm und Thurner.*



Entscheidungsfaktoren nach Boehm und Thurner

- Teamstruktur (Personell): Qualifikation des Teams muss bei agilen Methoden höher sein als bei plangetriebenen
- Dynamik der Anforderungen (Dynamism):Je häufiger Anforderungen geändert werden, desto einfacher muss dies möglich sein.
- Entwicklungskultur (Culture): Agile Methoden können je Entwicklungskultur eingesetzt werden.
- Teamgröße (Size): Agile Methoden werden oft bei kleinen, oder mittelgroßen Teams effizient eingesetzt.
- Kritikalität (Criticality): Höhere Prüfung der SW, wenn Menschenleben davon abhängen.

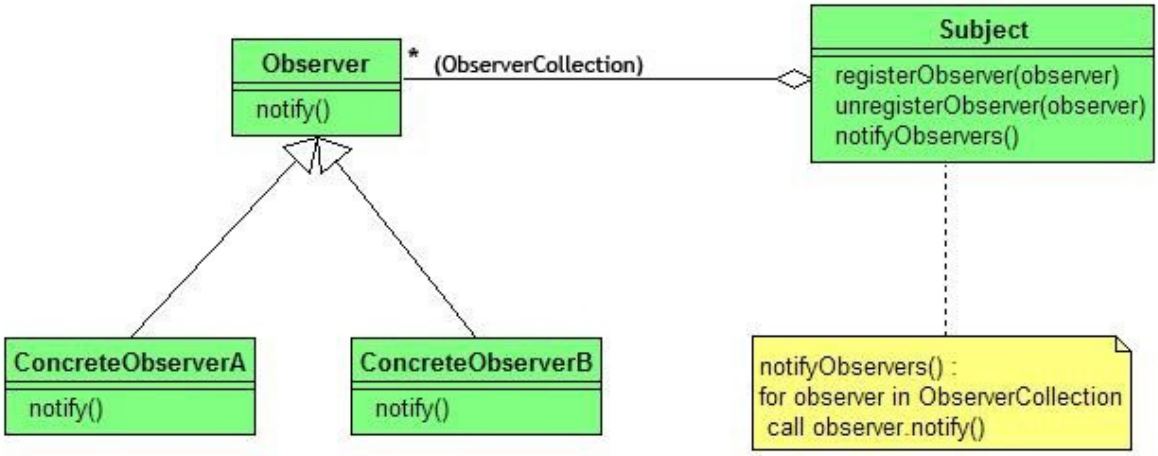
17. *Matrix-Projektorganisation*
Projektleiter trägt Gesamtverantwortung /nicht volle Weisungsbefugnis



Vorteile	Nachteile
+ Projektverantwortung durch PL	- Mitarbeiter Diener 2er Herren
+ Sicherheitsgefühl der	- Konfliktpotential

Mitarbeiter	„Linie/Projekt“
+ flexibler Personaleinsatz	- hoher Koordinationsaufwand

18. Observer-Pattern erklären/aufzeichnen (Klassendiagramm) und Beispiel nennen



- gehört zu den Behavioral Patterns
- Objekte sind voneinander losgelöst, was bewirkt, dass Änderungen eines Objekts zwar viele andere Objekte beeinflussen können, diese aber voneinander keine Details wissen müssen
- wird bei MVC verwendet
- <u>Observer Design Pattern vs. Composite Design Pattern</u>

19. Refactoring + Beispiel + Vorteile/Nachteile

SW leichter verständlich und verstehbar machen, wobei das observalble behavior der SW nicht verändert wird

- Verbesserung des Designs von Quellcode
- Verbesserung der Lesbarkeit von Quellcode
- Verstehen von fremdem oder altem Code
- Restrukturierung Code
- Verbesserung nicht-funktioneller Attribute

Vorgehensweise

- schlecht strukturierten Code identifizieren
- ausreichende Testabdeckung sicherstellen
- Refaktorisierung anwenden
- Test durchführen
- Refaktorisierung abschließen
- Funktionale Anpassung des Quellcodes erst nach Abschluss der Refaktorisierung

Beispiel: Pull-Up Methode in Eclipse, Rename Method

Advantages include improved code readability and reduced complexity to improve the maintainability of the source code, as well as a more expressive internal architecture or object model to improve extensibility.

20. kritischer Pfad bei GANT Diagrammen. 5 Beziehungen eines GANT Diagramms nennen und erklären (Anfang-Ende-Beziehung, etc.)

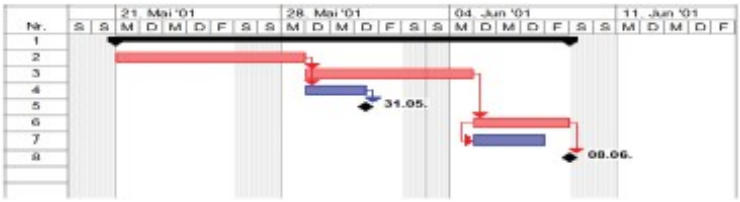
Allgemein:

- älteste, weit verbreitete grafische Methode zur Terminplanung
- Aufgaben oder Personenbezogen

Pufferzeit eines Vorgangs

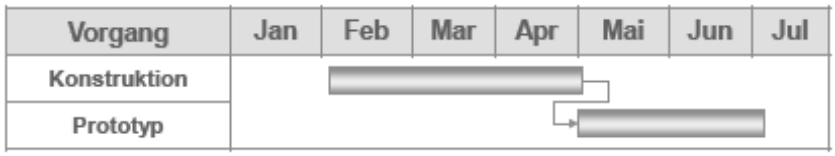
- Gesamtpuffer: Zeitspanne, wie lang sich ein Vorgang verzögern darf, ohne Endergebnis zu verzögern
- Freier Puffer: Zeitspanne, wie lang sich ein Vorgang verzögern darf, ohne andere Vorgänge zu verzögern

Kritischer Vorgang: Vorgang ohne Pufferzeit
Kritischer Pfad: enthält nur kritische Vorgänge

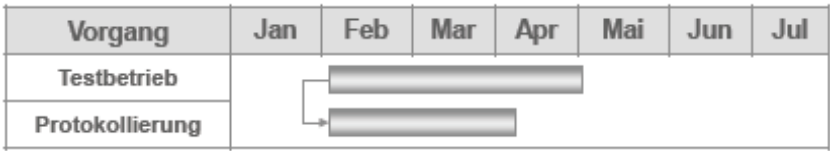


Vorgangsbeziehungen:

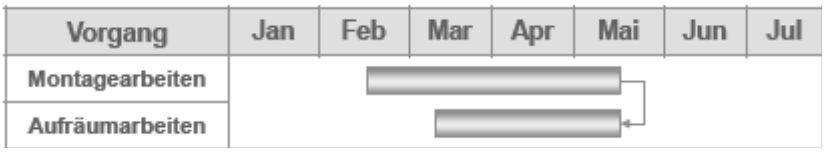
- Ende-Anfangs-Beziehung (EA)
  - Ende Vorgang1 Vorraussetzung für V2
  - Sequentielle Abarbeitung



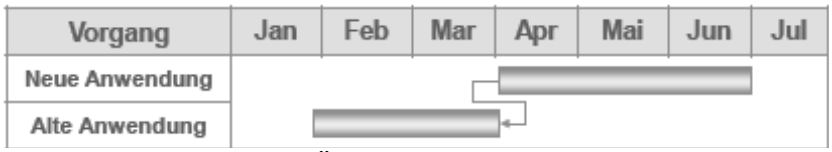
- Anfangs-Anfangs-Beziehung (AA)
  - Anfang V1 ist Vorraussetzung für Anfang V2
  - Parallele Abarbeitung



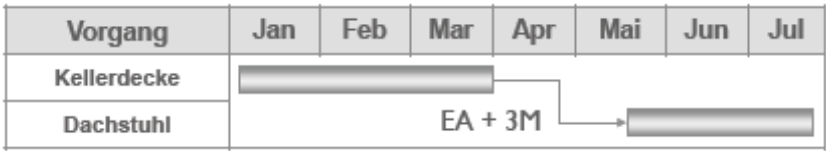
- Ende-Ende Beziehung (EE)
  - Ende V1 Vorraussetzung für Ende V2
  - Parallele Abarbeitung



- Anfangs-Ende Beziehung (AE)
  - Anfang V1 Vorraussetzung für Ende von V2
  - Parallele oder sequentielle Abarbeitung



- Vorgangsbeziehung mit Überlappung oder Verzögerung
  - Zeit/Prozentangaben



21. Wozu dient das "3-Experten-Konzept"/3-Experten-Methode? Erklären Sie dieses.

- Dient zur Aufwandsabschätzung
- 3 optimistische, 3 realistische, 3 pessimistische Schätzungen

- Abklärungen durchführen, bis Ergebnisse in jeder Kategorie übereinstimmen
- Gemeinsamen Schätzwert ableiten

22. Nennen und erklären sie die Delphi-Methode.

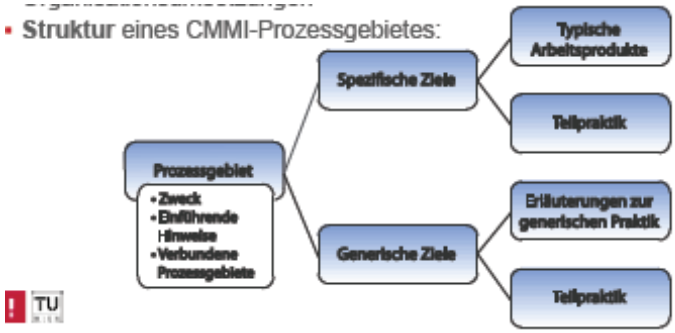
- Dient zur Aufwandsabschätzung
- Moderator und mehrere Experten
- Verfahren:
  - Zu jedem AP gibt jeder Experte einen Schätzwert ab
  - Mittelwert, falls Schätzwerte in gewissen Bandbreite
  - Sonst, Argumente austauschen und neue Schätzung
  - Zuschläge zu Einzelschätzungen: plus 10-15% = Aufwand für Projektm.

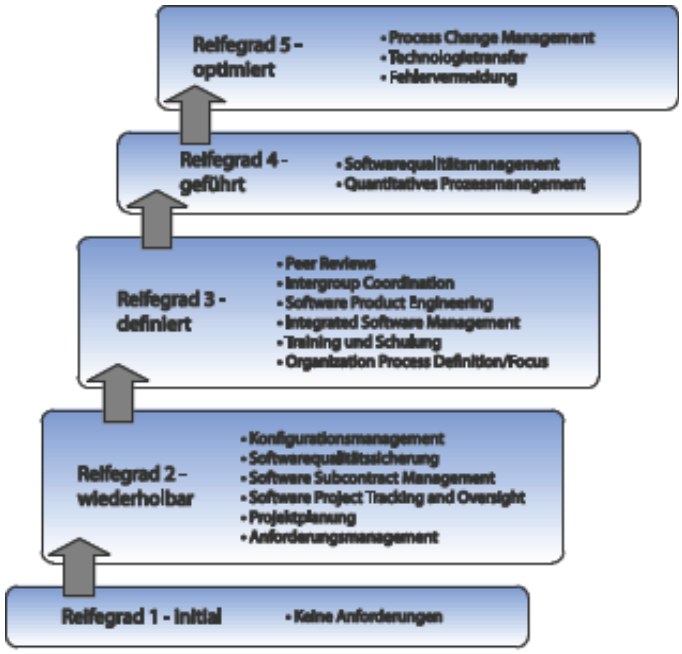
23. Wozu dient CMM? 5 Reifegrade!

- Teil der Qualitätssicherung
- Vergleichbarkeit und Qualität von Softwareprozessen im Rahmen der Vergabe an externe Lieferanten zu verbessern
- CMMI (Capability Maturity Model Integrated)), vereinheitlichter und modularer Nachfolger
  - Liefert Analyse des IST-Zustands eines Unternehemens
  - Kein international anerkanntes Zertifikat
- CMM-Assesment beurteilt Fähigkeit einer Organisation, SW unter bestimmten Rahmenbedingungen erstellen zu können

Prinzipien und Konzepte

- Quantitative Managementmethoden → Qualität kontrollieren, Produktivität steigern
- Fünf-Ebenen-Capability Maturity Model → Bewertung der Leistungen, bestimmt Aufwände, die zum Erreichen der nächsten Stufe notwendig sind
- Generische Prozessgebiete → definieren „was“, nicht „wie“ → breite Palette möglich





24. Erläutern sie das Vorgehen bei einem "SQL Injection" Angriff! Wie kann man sich davor schützen?

Ausnützen fehlerhafter/nicht sicherer Einbindung SQL-Statements (Manipulation)

Listing 1: action.php

```
$pw=$HTTP_GET_VARS["pw"];
...
$result=mysql_query("SELECT * FROM users
                     WHERE password='$pw'");
```

Listing 2: index.html

```
<form method="get" name="reg" action="action.php">
  Login: <input type="text" name="pw">
  <input type="submit" name="do" value="Login">
</form>
```

- Einfaches Hochkomma: “te’st”. Fehlermeldung der Datenbank wird angezeigt. Schwachstelle erkennbar. Zusätzlich Informationen zur Datenbank und zum Aufbau des SQL Statements werden ausgegeben.
- Verwenden von Tautologien wie ’ or ’1’=’1 was zu SELECT * FROM users WHERE user_name=’’ or ’1’=’1’ führt
- Auslesen von Daten aus weiteren Tabellen mittels UNION oder Sub-Queries

“’ UNION SELECT * FROM system”

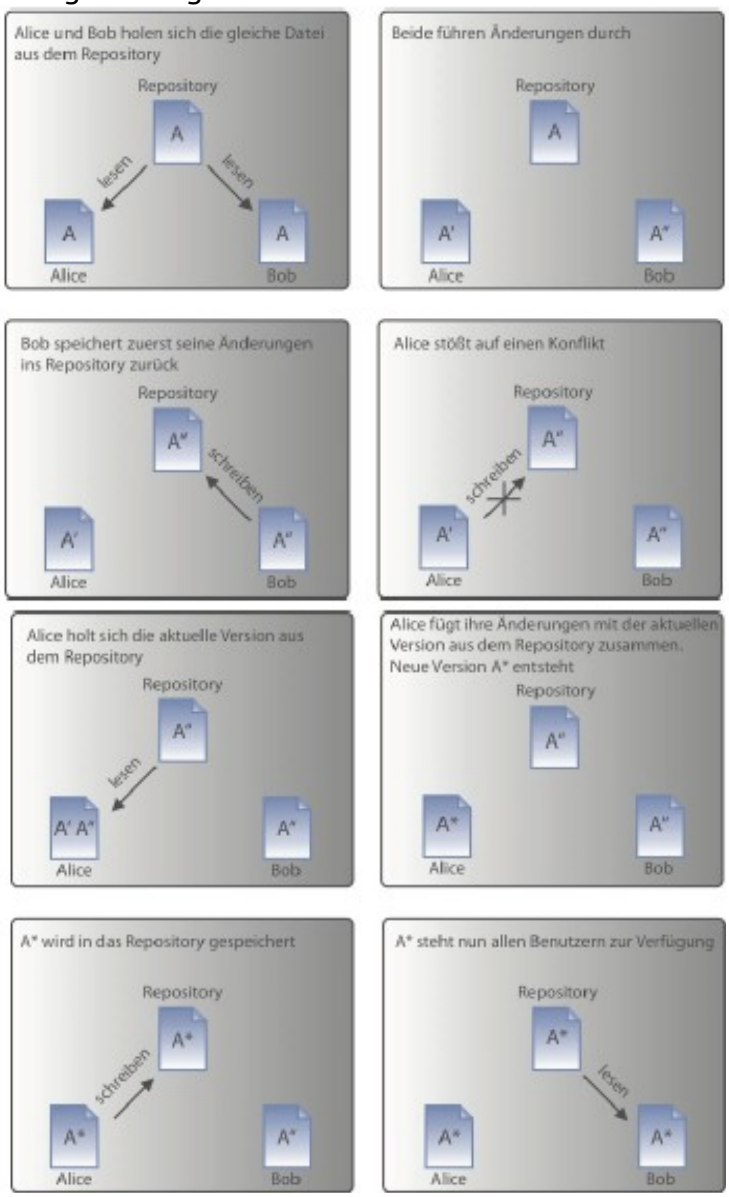
Gegenmaßnahmen:

- Jede Eingabe muss validiert werden. Es ist nicht ausreichend nur Formularfelder zu validieren. Injection z.B. auch durch HTTP Header-Felder
- Filterung spezieller Datenbankzeichen wie z.B. einfache Hochkomma → Blacklist Filtering
- Besser: Prüfung der Eingaben auf g¨ltige Zeichen → Whitelist Filtering
- Precompiled Statements verwenden. Das Statement wird in der Datenbank mit Platzhaltern kompiliert. Spätere Manipulation des Statements durch Eingaben nicht mehr möglich.
- Minimierung des Schadens bei Angriffen: Einschränkung der Rechte in der Datenbank → Principle of Least Privilege

25. *Erklären sie den Unterschied zw. Optimistic Revision Control und Pessimistic Revision Control. Erklären sie diese Strategie anhand einer Skizze. (Unterschied zwischen pesimistischer/optimistischer Versionierung)*

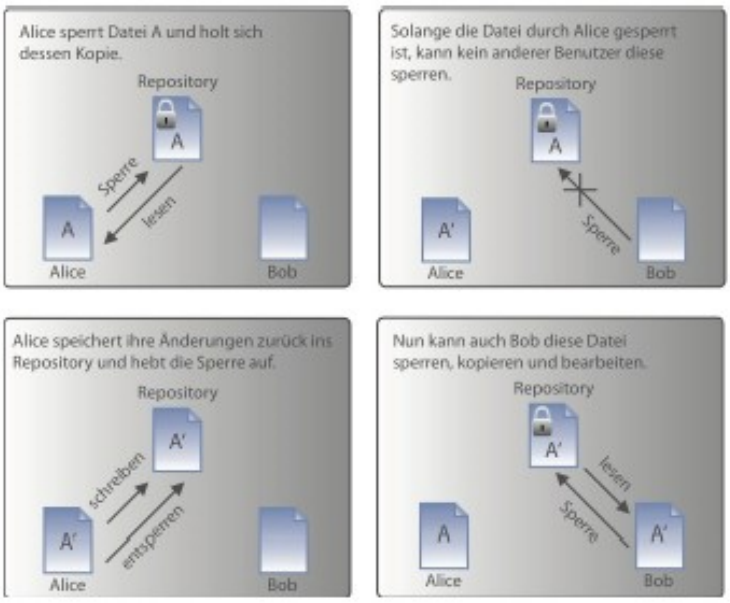
Optimistic:

- Copy-modify-merge Strategie



Pessimistic:

- Lock-Modify-Unlock Strategie



26. *Erklären sie den Unterschied zw. Zentrale, dezentrale, lokale Versionskontolle + je 2 Bsp.*

Zentrales VM:
- Client/Server Ansatz
  - Zentraler Server
  - Beliebig viele Clients
  - CVS, SVN

Dezentrales VM:
- Kein zentrales Repository
- Entwickler haben eigenes Repository
  - Änderungen zwischen Repos ausgetauscht
  - GIT, Bazaar

Lokales VM:
- Älteste Form
- Versionierung einzelner Dateien
- Nicht für Team-Projekte geeignet
- RCS, SCCS

27. *Was ist ein "thin-client" und was ein "fat-client"? Anwendung*

3 Schichten:
- Benutzerschnittstelle
- Business Logic (Ganze logik, die implementiert wird)
- Datenspeicherung (zb datenbank)

thin client:
- Server kümmert sich um business logic, client hat nur benutzerschittstelle (Bankomat).
- Wird verwendet, wenn alte Systeme auf einen Server ausgelagert werden.
- Nachteil: Belastung für Netzwerk und Server

fat client:
- business logic und Benutzerschnittstelle beim client implementiert.
- Server verwaltet nur Daten. Beispiel: C/S Systeme, bei denen Leistungsdaten des Clients vorher schon bekannt sind.
- Komplexer als thin client. neue versionen der applikationen müssen auf allen clients installiert werden

28. *Wie werden Frameworks üblicherweise spezialisiert/angepasst? Wie geschieht dies bei Whitebox Frameworks und Blackbox Frameworks? Unterschied zwischen Blackbox und Whitebox Frameworks. Erklären von "Method Hooks" und "Hot Spots"*

Frameworks werden vom Entwickler angepasst und ist ein wieder verwendbarer Entwurf.

Eigenschaften:

- Method Hooks
  - Abstrakte Methoden, die vom Entwickler überschrieben werden, um gewünschtes Verhalten zu erstellen

```
JPABasicAuthenticator.java
package ac.at.tuwien.inso.sepm.framework;

import com.sun.net.httpserver.BasicAuthenticator;

public class JPABasicAuthenticator extends BasicAuthenticator {

    public JPABasicAuthenticator(String realm) {
        super(realm);
    }

    @Override
    public boolean checkCredentials(String username, String password) {
        User u = loadUser(username);
        return u.checkCredentials(username, password);
    }
```

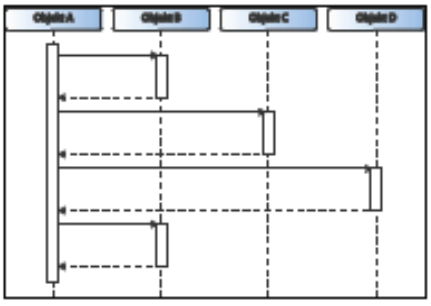
- Hot Spots
  - Punkte, an denen ein Framework spezialisiert wird
- Whitebox Frameworks
  - Innere Struktur muss sichtbar sein, um es spezialisieren zu können
  - <u>Spezialisierung durch Vererbung</u>
- Blackbox Frameworks
  - Enthält stabile Hotspots
  - <u>Spezialisierung Kompositionsbasiert</u> (Komponenten an Framework übergeben)

29. *Was versteht man unter "Stakeholder"? Nennen sie fünf typische Stakeholder in einem Softwareprojekt*

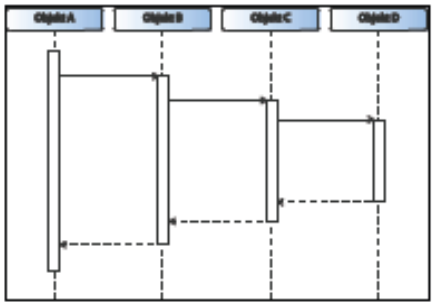
    Stakeholder = "Anteilhalter der ein Interesse im Projekt vertritt"
    Typische Stakeholder: Kunde, Geschäftsführung, Promoter, Legacy Owner, Betreiber, Projektleiter, Projektsteuergruppe,....

30. *Unterschiede Stairs und Fork SW-Architektur erklären +Skizze*

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



Fork



Stair

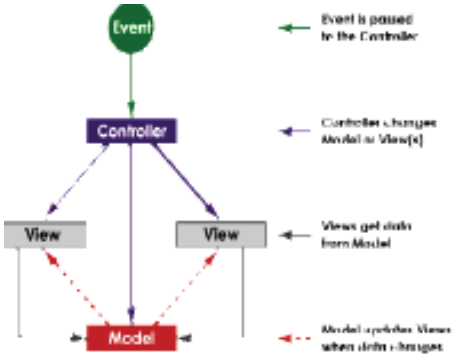
31. *Wozu dienen Architekturmuster? Beschreiben Sie das Model View Controll Architekturmuster, und nennen sie zwei weitere Architekturmuster!*

    - Beschreiben wiederholt auftretende Struktur von miteinander kommunizierenden Komponenten
    - Allgemein definiertes Problem lösen
    - Lösung wiederkehrender Probleme
    - Architekturpatterns, Designpatterns
    - MVC, Pipes and Filters, Layered Architecture

    MVC

    - Model kapselt Daten und Funktionen

- View stellt Informationen dar
- Controller empfängt Events



32. *Was versteht man unter "SOA"? Welche Eigenschaften zeichnen SOA aus? Welche Paradigmen werden dabei verfolgt? Was ist SOAP, WSDL und UDDI? Was ist BPM? Zusammenhang SOA und BPM?*

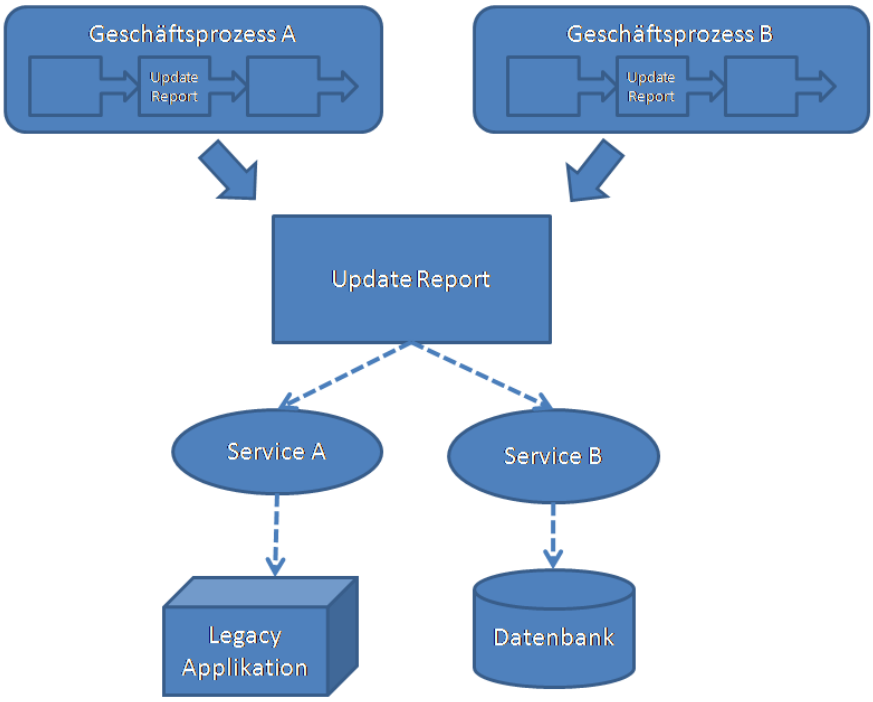
SOA…Service-Orientated Architecture

Paradigmen

- Loose Coupling (□ hohe Reuse und Flexibilität)
- Abstraktion (Reduktion von Neuentwicklungen)
- Standardisierung (erlaubt Zusammenarbeit von Services)
- Composable (können zu Komponenten zusammengefasst werden)

BPM und SOA

- Business Service: im Service-Orientierten Entwurf wird ein Schritt in einem Geschäftsprozess extrahiert und gekapselt (müssen nicht zwingend im Geschäftsprozess sein)
- Business-driven: an Geschäftsprozesse angelehnt
- Business Process Modelling (BPM) und SOA eng miteinander verknüpft □ Anwendungen maßgeblich von Geschäftsprozess-Modellierung beeinflusst



Eigenschaften:

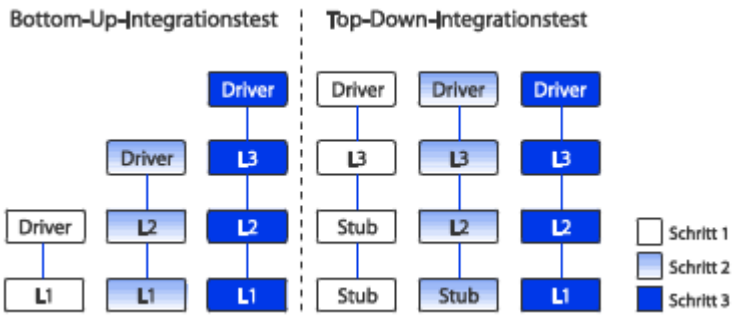
- Geschäftsprozess-orientiert
- Anwendungen durch mehrere Services gelöst
- Services sollen auch von anderen Anwendungen aus aufgerufen werden
  - Reuse
  - Modularisierung
- Orchestrierung: Zusammenstellen von Services in bestimmten Abläufen zur Umsetzung von Geschäftsprozessen
- Services existieren auf mehreren Abstraktionsebenen (Business Service, Component Service, usw.)
- Beschreibt Architektur, keine Technologie (Webservices zum Realisieren verwenden)

33. *Was sind Integrationstests? Erklären sie unterschiedliche Integrationsformen! Erläutern sie in diesem Zusammenhang die Begriffe Stub and Driver!*

Integrationstest

- prüft die Schnittstelle zwischen Komponenten
- mehrere Integrationsstufen möglich (unterschiedliche Größe)
- mit Größe des Integrationsumfanges wächst Schwierigkeit der Insolation von Fehlerwirkungen in Komponenten oder Systemen
- <u>inkrementelle Integrationsstrategien vs. Big-Bang Strategie</u>
- basierend auf Systemarchitektur, funktionale Aufgaben, Transaktionsverarbeitungssequenzen, oder weitere Aspekte des Systems oder der Komponenten
- <u>horizontale vs. Vertikale Integration</u>

Bottom-Up Integrationstest vs. Top-Down Integrationstest



- Stub: Softwaremodule, die untenliegende Schichten simulieren
- Driver: Softwaremodule, die obenliegende Schichten simulieren

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

- Niemals ein echtes Einfrieren der Anforderungen
- stabile Zustände in der Analyse unabdingbar
  - nur change control procedure möglich
- nach setzen Baseline, wird nach Änderungen folgendes durchgeführt:
  - Change Impact Analysis
  - Re-Casting

35. *Erläutern sie die 5 Arten von kontrollflussorientierten Testverfahren (Co,..C4 Test)! Warum ist eine Testabdeckung von 100% nicht immer möglich/sinnvoll?*

Anweisungsüberdeckungstest (C0) (zumindest ein Testfall pro Knoten eines Kontrollflussdiagramms)

Zeigüberdeckungstest (C1) (Jede Kante des Kontrollflussgraphen muss einmal durchlaufen werden)
Einfacher Bedingungsüberdeckungstest (C2) (Atomare Entscheidungen müssen einmal einen wahren und einmal einen falschen test haben)
Mehrfach-Bedingungsüberdeckungstest (C3) (zusätzlich zu den atomaren Teilentscheidungen werden die Gesamtentscheidungen einmal mit wahr und einmal mit falsch gestestet)
Pfad-Überdeckungstest (C4) (Alle unterschiedlichen Pfade des Testpbjekts zumindest einmal durchlaufen

36. Wozu dient SPICE? 6 Reifegrade!

SPICE = Software Process Improvement for Capability Determination

- Teil der Qualitätssicherung
- Gemeinsame Initiative von ISO und IEC
- SPICE und CMMI sehr ähnlich → 80-90% Überdeckung der Ziele und Forderungen
- Konkurrent zu CMMI
- Zweidimensionales Referenzmodell
    - Prozessdimension,
    - Fähigkeitsdimension zur Reifegradbestimmung einzelner Prozesse

Prinzipien und Ziele

- Zusammenführung verschiedenster existierender Assessment-Methoden zu einem zusammenhängenden Systemmodell
- Universelle Einsetzbarkeit, um alle Kategorien von Softwareprojekten sowie deren Kunden und Zulieferer zu unterstützen
- Hohe Professionalität
- Erreichung hoher internationaler Akzeptanz als weltweiter Standard

Strukturierte Vorgehensweise und Zielsetzungen

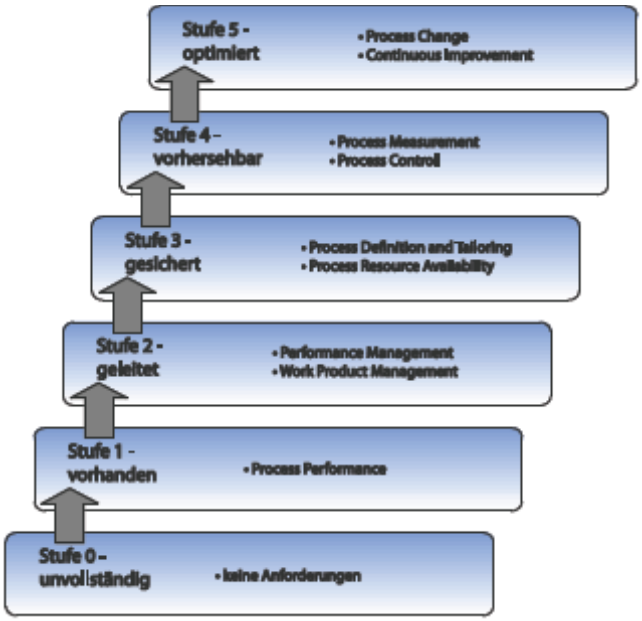
- Das Verstehen des Entwicklungsstandes der eigenen Prozesse, um Verbesserungen vornehmen zu können
- Die Eignung der eigenen Prozesse in Bezug auf Anforderungen oder Klassen von Anforderungen zu bestimmen
- Die Eignung der Prozesse anderer Organisationen und Organisationseinheiten in Bezug auf Verträge oder Klassen von Verträgen zu bestimmen

Erfüllungsgrade

- nicht erreicht (not achieved) 0 – 15 %
- teilweise erreicht (partially achieved) 16 – 50 %
- weitestgehend erreicht (largely achieved) 51 – 85 %
- vollständig erreicht (fully achieved) 86 – 100 %

Prozesskategorien

- Engineering (ENG)
- Customer-Supplier (CUS)
- Management (MAN)
- Support (SUP)
- Organisation (ORG)



37. UML erklären + 5 (!!) verhaltens- und 5 Strukturdiagramme nennen

UML = Unified Modelling Language

- Dient der Anforderungsmodellierung

Verhaltensdiagramme:

- Anwendungsfalldiagramm
- Aktivitätsdiagramm
- Zustandsdiagramm
- Kommunikationsdiagramm
- Interaktion-Übersichts-Diagramm
- Sequenzdiagramm

Strukturdiagramme:

- Klassendiagramm
- Paketdiagramm
- Komponentendiagramm
- Verteilungsdiagramm
- Kompositions-Struktur-Diagramm
- Objektdiagramm

38. Was sind Buildmanagementsysteme? Erklärung und 3 Beispiele.

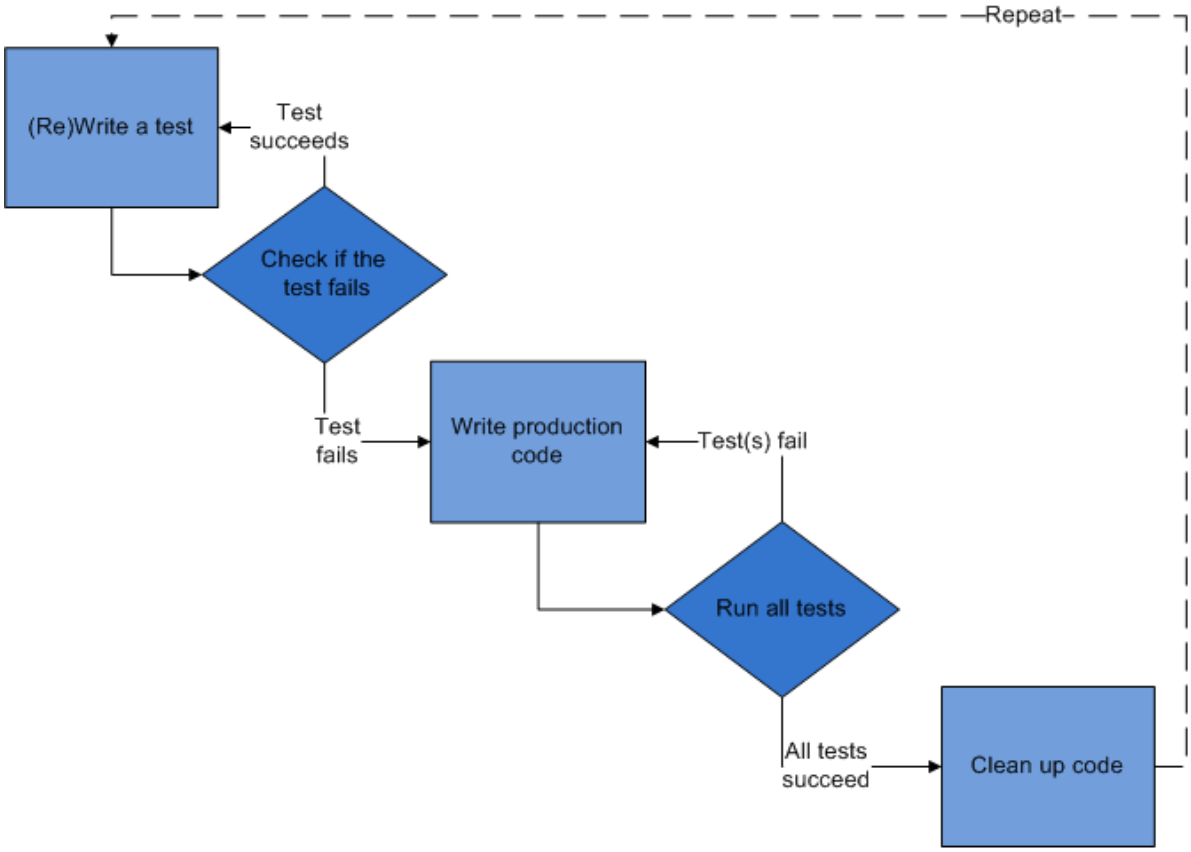
- Werkzeug zum Automatisieren immer wiederkehrender Aufgaben zum Bauen einer Applikation.
- Stellt Abhängigkeiten zwischen Komponenten sicher.
- Gängige Systeme: Ant, Make, Maven, Buildr

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

- Ermöglichen die Qualität von SW zu interpretieren
- Prozessmetriken: quantitative Daten des SW-Entwicklungprozesses
- Produktmetrik: messen von SW, ohne Entstehungshintergrund zu berücksichtigen
- Umfangmetrik: LOC Lines of Code
- McCabe-Metrik: zyklomatische Zahl V(g)
- Durch Einsatz OOP wurden neue Metriken notwendig (zwischen Objekten)
    - CBO (coupling between objects)
    - DIT (depth of inheritance tree)

- NOC (number of children)
- RFC (response for a class)
- WMC (weighted methods per class)
- LCOM (lack of cohesion in methods)

40. *Erklären Sie das Test Driven Development anhand einer Skizze. Welchen Bezug zu automatisierten Tests zu TDD gibt es?*



Man kann Funktionen des automatisierten Testens mit TDD kombinieren. Zum Bespiel kann man Eclipse Unit Tests für das eigene Programm schreiben und mit einfachem Ausführen wird es ausgeführt und auf gewünschte Fehler getestet.

41. *Was ist der Unterschied zwischen Bottom-Up und Top-Town-Integration? Wo liegen die jeweiligen Vor- und Nachteile, was wird eher in der Praxis eingesetzt und weshalb?*

Bottom-Up Integration

Hier arbeitet man sich ausgehend von den hardwarenahen Schichten nach oben vor. Vorteil ist die frühe Integration der risikobehafteten Teile (wodurch man später auf ein stabiles Framework aufbauen kann), Nachteil ist dass man Driver (Softwaremodule, oben liegende Schichten simulieren, z.B. GUI-Interaktionen durch ein Script simulieren) schreiben muss. Der Kunde sieht lange keinen Fortschritt, weil die sichtbaren Teile erst spät integriert werden. Es müssen daher zusätzlich Prototypen erstellt werden.

Top-Down Integration

Die Komponenten werden von höher liegenden Schichten der Software-Architektur ausgehend nach unten integriert.
Beispielsweise wird zuerst das GUI mit der Business Logic zusammengefügt, später kommt unten noch der Data-Access-Layer hinzu.
Vorteil ist, dass die höher liegenden Schichten - das sind die für den Benutzer sichtbareren, früh verfügbar sind, was sich für Demo-Zwecke gut eignet. Allerdings sind Stubs zu schreiben (das sind Softwaremodule, die die noch nicht integrierten unteren Schichten simulieren). Außerdem werden hardwarenahe Teile (das sind üblicherweise die fehleranfälligeren) erst spät integriert werden.

42. Erklären des MDD und Unterschied zwischen PSM und PIM

MDD = Model Driven Development

- Technik, die automatisiert aus formalen Modellen, lauffähige SW erzeugt
- Einsetzen von Domänenspezifischen Sprachen (DSL), zusammen mit Codegeneratoren und Interpretern

PSM = Platform Specific Model

- Modell von SW oder business System, welches mit einer spezifischen Technologieplattform verbunden ist

PIM = Platform Independent Model

- Eigenständig

43. Was versteht man unter „round trip engineering" beim MDD

- Sorgt für Konsistenz zwischen Implementierung und den Diagrammen

Bsp UML:

- Forward Engineering: Änderung Klassendiagramm → autom. Anpassung Quelltext
- Reverse Engineering: Änderung Quelltext → autom. Anpassung im Diagramm

44. Äquivalenzklassenanalyse Grenzwertanalyse erklären und Unterschiede

Gehören zu funktionalen Methoden beim Testen.

Äquivalenzklassenanalyse

- Einteilung möglicher Ein- und Augabewerte in Klassen gleichem Systemverhalten
- Werte von Klassen gewählt, von denen angenommen wird, dass Fehler auftreten, die auch bei allen anderen Werten der Klasse auftreten können

Grenzwertanalyse

- Spezialfall der Äquivalenzklassenanalyse
- Tatsache, dass Fehler oft an den Grenzen der Äquivalenzklassen auftreten (valid,invalid class)

45. Meilensteintrendanalyse Bsp. Achsen erklären. Diagramm interpretieren

- Dient der Projektüberwachung

46. Beschreiben des Review-Prozesses nach IEEE. Erklären sie den Ablauf, und die Rollen. Wieso sind Reviews in der Praxis so wichtig?

- Formell organisierte Zusammenkunft von Personen, die inhaltliche oder formelle Produktteile prüfen (Dokumente, Programmcode) nach vorgegebenen Prüfkriterien/Listen (IEEE).
- Alle Entwicklungsergebnisse können mit Reviews geprüft werden.

- Verschiedene Ausprägungen von Reviews, für alle existieren definierte Regeln und Zuständigkeiten

Vorteile

- Lassen sich früh durchführen, auch wenn noch keine Testprogramme vorliegen
- Latenzeit eines Fehlers kann früh erkannt werden.
- Kosten für Behebung des Fehlers sind geringer und kompensieren den Review-Aufwand.

Rollen

- Manager: Projektleiter, der sich für das Testobjekt gemeldet hat und Freigabe gibt
- Moderator/Review-Leiter: plant, organisiert und leitet Review
- Schreiber: protokolliert währrend der Sitzung
- Autor: Urheber Testobjekt / Repräsentant des Teams
- Gutachter/Reviewer: vorbereitende Gutachtung und berichten in der Sitzung
- Leser: Aufbereitung prüfender SW-Elemente und Durchführung in der Sitzung

Review-Prozess

47. Unterschied zwischen Audit und Review

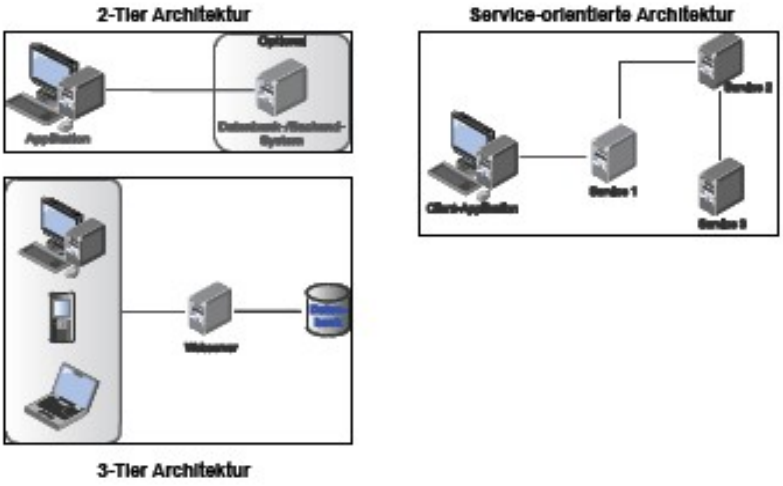
Audit: Aktivität, bei der sowohl Angemessenheit und Einhaltung vorgegebener Vorgehensweisen, Anweisungen und Standards, als auch deren Wirksamkeit und Sinnhaftigkeit geprüft werden (ANSI-Norm)

Review: Formal geplanter und strukturierter Analyse- und Bewertungsprozess, in dem Projektergebnisse einem Team von Gutachtern präsentiert und von diesem kommentiert oder genehmigt werden (IEEE-Norm)

48. Irgendwas mit Testablauf in der inkrementellen Entwicklung...

49. *2-Tier/3-Tier Architektur*

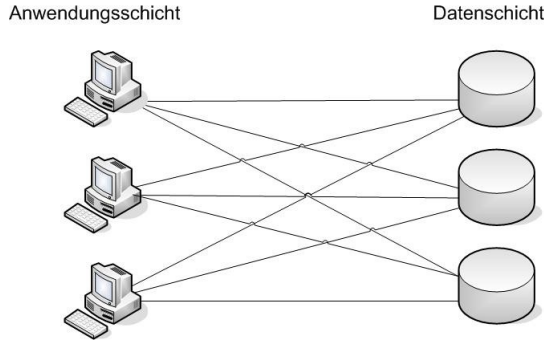
- Historische Entwicklung des Entwurfs von SW-Systemen
  - 2-Tier → 3-Tier → Service orientierte Architektur



2-Schichten-Architektur

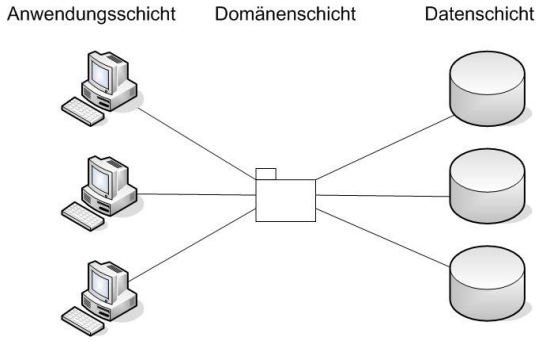
- Höhere darf nur auf niedere zugreifen

- Niedrigere Schicht ist Dienstanbieter (Server)
- Client/Server Architektur



3-Schichten-Architektur

- Eine zusätzliche Schicht (Logikschicht) übernimmt die Datenverarbeitung
- Präsentationsschicht → Logikschicht → Datenhaltungsschicht



50. 3 Vorteile/3 Nachteile Teamwork

Vorteile

+ Mehr Ideen durch Wechselwirkung in der Gruppe (erhöhte Kreativität)
+ Produktiver Wettbewerbszustand durch gegenseitiges Anspornen
+ Direkt vernetzte Kommunikation beschleunigt Informationsverteilung
+ Gesamtproblemsicht aller Beteiligten
+ Identifikation mit Gesamtlösung
+ Gruppenentscheidungen sind im Allgemeinen besser
+ Gruppendynamik (Teamgeist)

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

51. Migration - Definiton und 3 Arten

Ist ein Teil der Inbetriebname/Wartungsprozess, welche eine Applikation in die neue/geänderte Umgebung einbindet (anpassen).
Anpassung an geänderte Rahmenbedingungen, welche den laufenden Betrieb nicht beeinflussen darf. Es muss außerdem einem geordneten Prozess unterliegen.

Softwaremigration

- Umstieg einer eingesetzten SW

Datenmigration

- Transfer von Daten von einem System in ein anderes

Hardwaremigration

- Umstellung der verwendeten HW