Analyse

Vision, Scope und Terminologie müssen klar definiert und abgegrenzt sein.
Alle relevanten Stakeholder müssen identifiziert und qualifiziert eingebunden werden.
Die Machbarkeit muss vor dem formellen Projektbeginn sichergestellt sein.
Alle Komponenten müssen auf „Make or Buy“ evaluiert werden.
Eine Architektur sollte während der Analyse grob feststehen und Stakeholdern dargestellt werden.
Alle Schnittstellen müssen bekannt und geprüft sein.
Alle funktionalen, nichtfunkt. und Domänenanforderungen identifiziert und qualitativ beschrieben sein.
Anforderungsbeschreibungen müssen für alle Stakeholderverständlich gemacht werden.
Ein Prozess zum Ändern von Anforderungen (inklusive Re-Costing) muss etabliert sein.

Wie ist eine Anforderung definiert?

„A condition or capability needed by a user to solve a problem or achieve an objective.“
„A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.“
stabil (hard-wired implementiert) VS volatil (soft-wired)

Was ist eine Baseline, warum braucht man sie?

kein echtes Einfrieren der Anforderungen, aber stabiler Zustand um Analyse zu ermöglichen
Bei Änderung: Change Impact Analysis + Re-Costing.

Welche Typen von Anforderungen gibt es, woher kommen sie?

- funktional: bereitgestellte Funktionen und Services, Reaktion auf Verhalten
- nicht-funktional: Verfügbarkeit, (Code-)Qualität, Geschwindigkeit, einzuhaltende Standards, Normen und Gesetze
- Domänen-Anforderung: inherent in der Domäne, nicht Projekt spezifisch, oft implizit

Quellen für Anforderungen: Stakeholder, Standards und Normen, Datenschutz, Gesetze, Anforderungen und Funktion von alternativen Systemen (z.B. Konkurrenz), Verdeckte Regeln: Glaubensgrundsätze/Tabus/Macht/Werte/Einstellungen/Status

Was sind typische Probleme von schlechten Anforderungen?

Verständlichkeit, Selbstverständlichkeit, Wenig Präzision, Vermischung, Zusammenführung, Mehrdeutigkeit, Überflexibilität

Welche Qualitätsattribute haben gute Anforderungen?

Vollständig, Konsistent, Eindeutig definiert, Verständlich beschrieben, Atomar, Identifizierbar, Einheitlich dokumentiert, Notwendig, Nachprüfbar, Rück- und vorwärtsverfolgbar, Priorisiert

Was ist ein Stakeholder, welche gibt es?

= „Anteilhalter“ der ein Interesse im Projekt vertritt und damit eine Anforderungsquelle darstellt
Jede Stakeholdergruppe hat eine eigene Sicht auf das Projekt, einen eigenen Lingo
Konflikte machen Anforderungspriorisierung (!) notwendig
Beispiele: Kunde/Auftraggeber/Sponsor; Geschäftsführung/Abteilungsleiter (Auftraggeber- und Auftragnehmerseite); Promotor; Legacy Owner (Besitzer des Altsystems oder der Altdaten); Betreiber/Entwickler von Schnittstellensystemen (Systemen, mit denen integriert wird); Benutzer/Benutzervertreter (pro Benutzergruppe); Projektleiter; Projektsteuergruppe; Systemarchitekten/IT-Stratege; Softwareentwickler; Qualitätssicherung und Test; Betrieb; Wartung

Wie können Anforderungen modelliert und beschrieben werden?

- Unified Modeling Language (**UML**)
    - Verhalten: Anwendungsfall-, Aktivitäts-, Sequenzdiagramm
    - Struktur: Klassendiagramm
- Domänenspezifische Sprachen (**DSL**s): Modellierungssprache für bestimmten Bereich, oft Erweiterung für XML oder UML
- (Logische) **Datenmodellierung**: Informations-, Persistenzmodell. Oft relationales Modell oder XML Dokumente

Wie wird eine Make or Buy Entscheidung getroffen?

Entscheidungsfaktoren: Entwicklungs- vs Lizenzkosten, Schulungs-, Anpassungsaufwand, Vendor Lock-In, Support & Wartung, Hardwareanforderungen, Abdeckung des Projekt-Scopes

Welche Arten der Anforderungsanalyse gibt es?

- nicht-strukturiert: natürliche Sprache
- semi-strukturiert: natürliche Sprache + Vorlagen, eingeschränktes Vokabular, ergänzende Eigenschaften
- strukturiert: definierte Syntax, Notation

Anwendungsfälle / **Use Cases**: Was aber nicht wie; **User Stories**; Business Processing Modeling Notation (BPMN), End User Programming

Wie sind Use Cases nach Cockburn aufgebaut?

Name + Identifier + Beschreibung; Beteiligte Akteure, Status
Verwendete andere Anwendungsfälle
Auslöser + Vorbedingung + Invarianten + Nachbedingungen
Standardablauf + Alternative Abläufe; Hinweise, Änderungsgeschichte

Was sind Vor- und Nachteile von User Stories?

+ kurz, fördern Diskussion, reduzieren Risiko von Fehlentwicklungen, gut in Kombination mit Metapher
- stark interpretierbar, starke Involvierung des Nutzers, personenzentriert, schwer Vertragsgrundlage

Wie können Anforderungen verwaltet und die Nachverfolgbarkeit sichergestellt werden?

- Spezifikationsdokument
    - RFC2119: MUST - MUST NOT - SHOULD - SHOULD NOT - MAY
    - IEEE830-1998: Einleitung - Beschreibung - Spezifische Anforderungen - Anhang
- Projektglossar: gemeinsame Sprache, wichtige Begriffe definiert
- Anforderungsverwaltung:
    - oft Anforderungsbaum - typischer Dekompositionsstil in SE
    - Verwaltung in Datenbank, Versionierung, Tagging, integriert Systemmodelle
- Traceability
    - zwischen Anforderungen (bei Abhängigkeiten)
    - zu erstellten Artefakten

Wie kann die Machbarkeit überprüft werden?

Machbarkeitsstudie: Prototypen, technische Modellierung, rechtliche Prüfung, Kostenanalyse, Identifikation von ähnlichen Systemen, von wesentlichen Kostentreibern

Entwurf

Historische Entw. des Entwurfs von Softwaresys.: 2-Tier -> 3-Tier -> Service-orientierte Architektur
Im Entwurf werden Entscheidungen zur Mikro- und Makroarchitektur getroffen (taktisch vs. strategischer Entwurf)
Mehrere Sichten und Abstraktionsebenen einer Architektur, je nach betrachtender Rolle
Anforderungen aus der Analysephase müssen in einen Entwurf transformiert werden
Architekturpatterns sollen den Entwurf vereinfachen und bekannte Probleme effizient lösen

Wie unterscheidet sich der Strategische vom Taktischen Entwurf?

Strategisch: Architektur im Großen, Systemebene, Entscheidungen für Technologie, Gesamtaufbau
Taktisch: im Kleinen, Entsch. über Aufbau einzelner Bausteine (zb Klassen), Entwurf im klassischen Sinn (auf Baustein-, nicht Systemebene)

Welche Sichtweisen auf Architektur gibt es?

Business-, Informationssytem- (Anwendungs-, Daten-), Technologiearchitektur

Was enthält ein Pflichtenheft?

funktionale Anforderungen: Use Cases, User Stories
nicht-funktionale: in Prosa, als SLAs: Wartbarkeit, Usability, Performance, Skalierbarkeit

Wie sieht Architektur im agilen Kontext aus?

Emergent Design:

- Basiert auf Deliverables, durch Refactoring entsteht Design
- Code Conventions + Reviews notwendig für Best Practices
- High Quality Design durch: Test everything, eliminate duplication, express all ideas, minimize entities (YAGNI)

braucht trotzdem Rahmenbedingungen: Benachbarte Systeme, etc -> strategischer Rahmen

Wie läuft der Entwurf im Rational Unified Process (RUP) ab?

liefert Bauplan für Software: Analysis -, Architectural -, Design model
iteratives Vorgehensmodell, Phasen werden wiederholt durchlaufen (auch Entwurf und Analyse)

Was ist SOA, was BPM? Wie hängen sie zusammen?

SOA = Service-oriented Architecture
business driven
Anwendungen durch mehrere Services gelöst

- Lose Kopplung -> Reuse, Flexibilität
- Abstraktion: reduziert Neuentwicklungen
- Standardisierung: Zusammenarbeit von Services
- Composable: Services zu Komponenten zusammenfassbar

BPM = Business Process Modelling
ein Schritt im Geschäftsprozess => gekappselt als ein Business Service
Architektur ist business driven, dh Geschäftsprozesse geben Architektur vor
Anwendungen werden durch Services gelöst, diese in mehren Anwendungen wiederverwendet
Services auf mehreren Ebenen - Business -, Component -, technisches Service, etc

Wie funktioniert Objektorientiertes Design, weche Diagramme verwendet es?

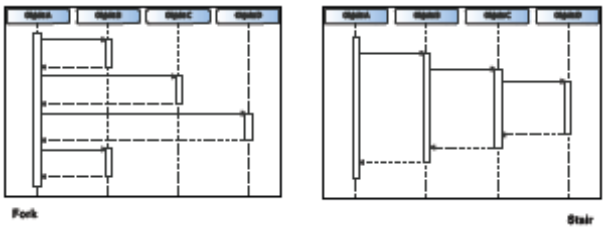
erweitertert Diagramme aus der OO-Analyse, versucht System möglichst genau Abzubilden
nicht vollständig, Klassendiagramm für Struktur, Zustands- u. Sequenzdia. für Verhalten, Komponentendia. für Zusammenhang der Komponenten. Diagramme in UML

Wie wird bei einem objektorientierten Entwurf vorgegangen?

Klassen und Objekte identifizieren, Semantik zwischen ihnen festlegen, Beziehungen zwischen ihnen, Schnittstellen und Implementation der Klassen spezifizieren
UML zur Unterstützung, gleichzeitig Basis für Dokumentation.
Verhaltensmodelle: dynamisches Verhalten - Aktivitätsdia, Zustandsdia., Sequenzdiagramm
Strukturmodelle: statische Teile des Systems: Klassendia, Komponentendia., Verteilungsdia.

Erkläre die Unterschiede zwischen Stair- und Fork-Kontrollstil

Stair: Objekt ruft jeweils das nächst tiefere auf, etc
Fork: zentrales Kontrolobjekt ruft tiefere Objekte direkt auf



Welche Design/Architectural Patterns gibt es, wie funktionieren sie?

Patterns = allgemeine Lösungsstrategien
4 Typen von Design Patterns, auf Klassenebene, wichtigste Beispiele:

- **Creational** (Abstract) Factory, Prototype, Resource acquisition is initialization (RAII), Singleton
- **Structural**: Adapter, Decorator, Proxy,
- **Behavioral**: Observer, Visitor, Iterator
- **Concurrency**: Lock, Thread-Specific Storage

Architectural:

- **Structure**: Component-based, Monolithic, **Layered**, **Pipes and filters**
- **Shared Memory**: Data-centric, Blackboard, Rule-based
- **Messaging** Event-driven aka Implicit invocation, Publish-subscribe, Asynchronous messaging
- **Adaptable systems**: Plug-ins, Microkernel, Reflection, Domain specific languages
- **Distributed Systems**: Client-server (2-tier, 3-tier, n-tier exhibit this style), Shared nothing architecture, Space based architecture, Broker, Peer-to-peer, Representational State Transfer, Service-oriented

Layered Architecture: funktionell getrennte Ebenen: Datenbankzugriff, Geschäftslogik, UI; Kommunikation nur mit darunter liegender Ebene; keine Abhängigkeiten auf darüber liegenden Ebene; Kommunikation mittels Interfaces, Exceptions nach Ebene gekapselt
Model-View-Controller:
Modell kapselt Daten und Funktion, View stellt Information dar, Controller empfängt Events

Implementierung

Umsetzung der Architektur in eine Software, Umsetzung der Anforderungen des Kunden
Zeitgerechte Entwicklung einer validierbaren (abnehmbaren), wartbaren Software
„The process of translating a design into hardware components, software components, or both“

Was ist ein Framework, eine Bibliothek? Unterschiede? Black- vs Whitebox Spezialiserung

Framework = Applikationsskelett, wiederverwendbarer Entwurf / Verhalten. Durch Menge von abstrakten Klassen und Zusammenspiel ihrer Instanzen, ruft eigenen Code auf
+ Reduktion von Entwicklungskosten, kürzere Entwicklungszeit, Wiederverwendung von Quellcode und architektonischen Strukture, Einhaltung von Standards
- Einarbeitungsaufwand, Vererben von Fehlern
Bibliothek = Sammlung von Software und Dokumentation designed to aid in Software development, wiederverwendbare Funktionalität, von eigenem Code aufgerufen
Framework Typen:

- Persistence: Hibernate, JPA, Anorm
- Web Application: Jboss Seam, Struts, Ruby on Rails, Django, Play!, Yesod, Lift
- Rich Client: Eclipse RCP
- Grafik: Qt, Core Graphics, Gtk+, Direct X

Framework Eigenschaften

- Method Hooks: abstrakte Methoden, Entwickler überschreibt für gewünschtes Verhalten
- Hot Spots: Punkte zur Spezialisierung
- Whitebox Framework: Innere Struktur sichtbar, Spezialisierung meist durch Vererbung
- Blackbox: stabile Hotspots, Spezialisierung durch übergabe von Komponenten an Framework

Wie findet die Entscheidung Eigenbau vs vorhandenes Framework statt?

nachträglicher Tausch kommt Neuentwicklung gleich
Auswahl: Deckt Framework Anforderungen? Wartung u Weiterentwicklung? kommerziel <-> frei ? Verfügbarkeit von Entwicklern mit Erfahrung?
Eigenentwicklung nur: kein geeignetes Framework, Anpassung min. gleich aufwändig, spezielle Anforderung erzwingen es

Was sind Integrated Development Environments (IDE), welche Features haben sie?

zB Eclipse, NetBeans, XCode, Visual Studio, KDevelop, QT Creator, IntelliJ IDEA
Source File Editor, Tooling (Compiler, Debugger, Profiler), Refaktorisierung, Unit Testing, Volltext-, Regex-, Semanische Suche, Support für Versionskontrolle, Plugins

Was versteht man unter objektorientierter Kopplung und -Kohäsion?

Kopplung - Zusammenhang verschiedener Klassen, sollte möglichst niedrig sein. Zu hoch -> Klassen können nicht unabhängig verwendet werden
Kohäsion - Zusammenhalt von Methoden innerhalb einer Klasse, sollte möglichtst hoch sein. Zu niedrig -> Trennung in zwei Klassen möglich
Information Hiding: Kommunikation zw Komponenten nur über Interfaces

Was ist kollektiver Codebesitz, wann kommt er vor?

jeder hat volle Kontrolle über den Code, nicht jeder für alles verantwortlich. va bei agilen Methoden
+ jeder kann ändern, wissen über Implementation gut verteilt, Ausfall einer Person nicht so schlimm
- Verantwortlichkeit für “schlechten” Code schwer nachvollziehbar

Erkläre Test Driven Development (TDD). Was ist der Zusammenhang zu automatisierten Tests?

“You maintain an exhaustive suite of programmer tests, no code goes into production unless it has associated tests, you write the test first, the tests determine what code you need to write”
2 **goldene Regeln**

- You should write new business code only when an automated test has failed
- You should eliminate any duplication that you find

automatisierte Tests: jeder Entwickler ist Tester, einfach ausführbar, feingranuliert, vermeiden ungewollte Nebeneffekte

Erklären Sie das Konzept des Refactorings inklusive dessen Vor- und Nachteile.

„First, the purpose of refactoring is to make the software easier to understand and modify. [...]
Only changes made to make the software easier to understand are refactorings. [...]
Refactoring does not change the observable behavior of the software.“ -- (Fowler 1999)
Verbessert Design und Lesbarkeit, dient zum Verstehen von alten und fremden Code
NT: Bei nicht ausreichenden Tests: Bugs, Fehler, Funktionsveränderung
Vorgehen: schlecht strukturisierten Code finden -> ausreichende Testabdeckung sicherstellen -> Refaktorisieren -> Tests überprüfen -> Refaktorisierung abschließen

Nennen und Erläutern Sie ein Beispiel für ein Refactoring Muster.

Beispiele: Rename Method, Extract Method/Class, Pull Up Method, Split Temporary Variable

Wozu wird Code dokumentiert, wie wird er dokumentiert?

Erhöht Lesbarkeit und Übersicht, Code soll auch Monate später von anderen verstanden werden
Kommentare: Bestandteil des Codes, nur Intentionen eines Blocks, Erklärung der Umstände für unkonventionelle Wahl; Achtung: Kommentare können veraltet und falsch sein!
Tagging (TODO, FIXME), automatische Dokumentation (Doxygen, Javadoc)

Erkläre den Unterschied zwischen Internationalisierung und Lokalisierung

Internationalisierung: „sämtliche Maßnahmen, eine Software so zu gestalten, dass diese möglichst einfach an andere Sprachen und Kulturen angepasst werden kann“
Lokalisierbarkeit: Wie gut lässt sich ein Produkt übersetzen, ohne in den Source Code zu müssen?
Lokalisierung (l10n) = Übersetzungsprozess, für jede Sprache/Kultur, Anpassung von lokalen Formatierungen (Zahlen, Datum, Währung, Leserichtung), sehr aufwändig
Richtlinien: kein hard-coden von Text -> Ressource Files; Schriftgröße nicht beschränken, identische Nachrichten in unterschiedlichem Kontext extra speichern, gleich von Anfang an vorsehen

Was ist Fehlerbehandlung (Exception Management)?

einheitliches Konzept von Anfang an wichtig, Exceptional Handling in Programmierspr. vorhanden

Wie sollte man (nicht) loggen?

mit API, nicht system.out.println -> mehrere Log Levels, Sinks, bessere Granulierung:
DEBUG - INFO (info aus normalem Betrieb) - WARN (unerwartet, kein Fehler) - ERROR (fehler behandelbar) - FATAL (nicht stabil fortsetzbar)

Erkläre Build Mangagement Systeme, gib Beispiele

immer wiederkehrende Aufgaben automatisiert, stellt Abhängigkeiten sicher
C/C++: cmake, make, qmake, boost build, scons, autotools,
Java: ant, maven, sbt (Scala), ivy, leiningen (clojure), gradle (Groovy)
Ruby: rake
Haskell: cabal

Welche Arten von Versionskontrolle gibt es? Beispiele dazu? Begriffe?

Repository: Datenbestand mit Versionshistory, aus dem Arbeitskopie generiert wird
Checkout: Abruf der Daten aus dem Repository
Check-in/Commit: Übertragung und Einpflegen der Daten in das Repository
Branching: Anlegen einer Kopie von Objekten im Versionsmanagementsystem
Mergen: Zusammenführen mehrere gleichzeitiger Änderungen aus verschiedenen Commits
Markieren/Tagging: Tag kennzeichnet konkreten Entwicklungsstand (zB zur Auslieferung bestimmt)
Protokollierung/Historisierung:

- Welche Änderungen von welchem Benutzer zu welcher Zeit?
- Wiederherstellung eines beliebigen Versionsstandes aus der Vergangenheit

Arten

- Zentrales Versionsmanagement:
    - Server + beliebig viele Clients
    - Subversion (SVN), Perforce, CVS
- Dezentrales Versionsmanagement:
    - kein zentrales Repository, jeder Entwickler besitzt sein eigenes
    - Änderungen zwischen Repositories ausgetauscht
    - Git, Bazaar
- Lokales Versionsmanagement:
    - älteste Form, beschränkt auf Versionierung einzelner Dateien
    - nicht geeignet für Teamprojekte
    - Revision Control System (RCS), Source Code Control System (SCCS)

Copy-Modify-Merge-Strategie (Optimistic Revision Control) -> Konflikte und Zusammenfügen
Lock-Modify-Unlock-Strategie (Pessimistic Rev. Cont.) -> Datei während Bearbeitung gesperrt

Integration-Testing

Keine Software ist frei von Fehlern -> Systematisches Testen notwendig
Testen von Software gehört zu den wichtigsten Aktivitäten des Software-Entwicklungsprozesses
Eine gut geplante Testphase ist die Grundlage für ein erfolgreiches Projekt
Ziel ist es, Fehler so früh wie möglich zu finden
Softwaretests sind eine dynamische und produktorientierte Qualitätssicherungstechnik
Aufgrund wachsender Komplexität und hohe Abhängigkeiten von Softwaresystemen wird der Softwaretest strategisch immer bedeutender

Wie ist Integration-Testing definiert?

„Software testing is a **formal process** carried out by a **specialized testing team** in which a software unit, several integrated software units or an entire software package are examined by running the programs on a computer. All the associated tests are performed according to **approved test procedures on approved test cases**."

Welche Ziele hat Integration-Testing?

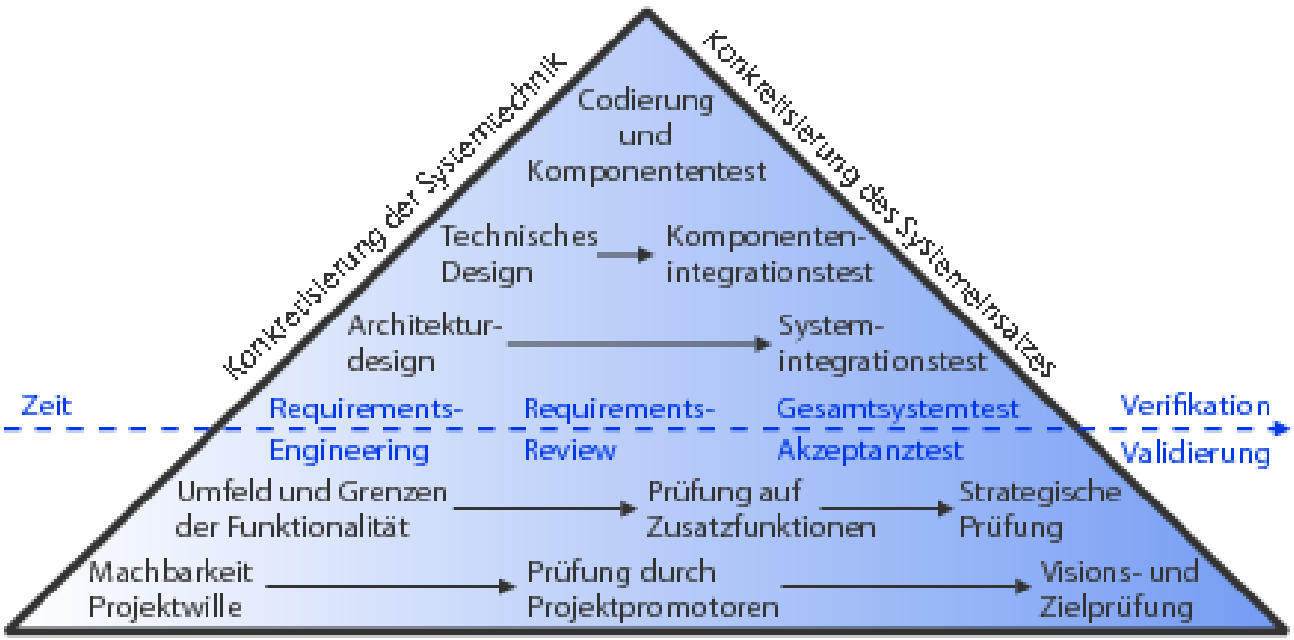
Testfälle mit höchster Wahrscheinl. festzustellen ob das System funktioniert; guter Test hat möglichst große Abdeckung; soll Fehler von Programmen identifizieren; Fehler gefunden -> Erfolg

Welche Grundsätze hat Integration-Testing?

Testen zeigt Anwesenheit von Fehlern, Vollständiges Testen nicht möglich, frühzeitig beginnen, Häufung von Fehlern, Wiederholungen haben keine Wirkung, Testen abhängig vom Umfeld, Keine Fehler != brauchbares System

Was sind Test- und Integrationsstufen, welche gibt es?

Zerlegung in beherrschbare Teile -> frühe Prüfung möglich
Aspekte von Teststufen: allgemeine Ziele, Testbasis = Arbeitsergebnis als Grundlage für Tests, Testobjekt = was getestet wird, auftretende Fehlerwirkungen, Anforderungen an die Tests, Werkzeugunterstützung, Verantwortlichkeiten

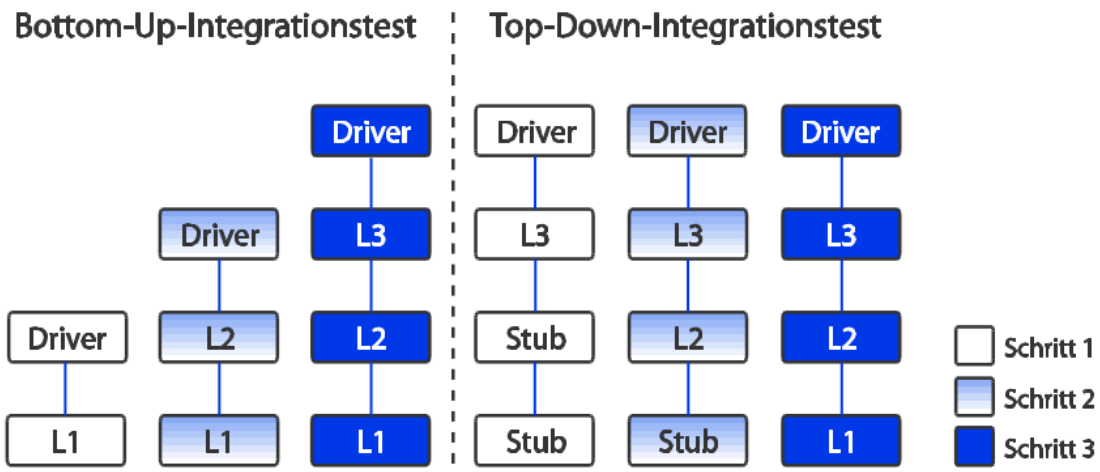


Was ist ein Komponententest?

prüft seperat testbare Komponenten (Module, Programme, Klassen) auf Fehler
Isolation der Komponenten durch Stubs (Platzhalter) / Simulatoren (Fakes) und Testtreiber

Was sind Integrationstests? Bottom-Up VS Top-Down Integrationstests?

prüft **Schnittstelle zwischen Komponenten**, in mehreren Stufen möglich
je größer der Integrationsumfang desto schwieriger Fehler zu isoliern
basierend auf Systemarchitektur, funktionalen Aufgaben, Transaktionsverarb.sequenz, etc.
inkrementell vs Big-Bang, horizontale vs vertikale Integration



Bottom-Up: + keine Stubs, - immer andere Driver, + findet leichter bugs, + auch am Anfang der Entwicklung möglich wenn obere Module noch nicht fertig
Top-Down: + gleiche Driver, - braucht Stubs, + findet leichter fehlende Links

Was ist Big-Bang Integration Testing?

viele oder alle Komponenten werden zu einem System integriert das dann als ganzes getestet wird
+ Zeitsparend, - wenn Test nicht richtig geloggt wird das testen komplizierter / kann das Ziel verfehlen

Was sind Systemtests?

Fokus auf spezifiziertem Verhalten des Gesamtsystems, funktionale + nichtfunktionale Anforderungen
basieren auf: Anforderungsspezifikationen, Anwendungsfällen, Geschäftsprozessen, Risikoanalysen, Erfahrungen im Produktionsumfeld, Systemressourcen, sonstige Beschreibungen des Systems

Was ist ein Akzeptanztest?

Nachweis der Erbringung der vereinbarten Leistung, meist vom Kunden durchgeführt
Arten: Anwender-, Betrieblicher-, Regulatorischer-, vertraglicher-, Alpha- und Beta-Test (oder Feldtest)

Welche Arten von Testabdeckung gibt es?

Ziel: möglichst hohe Abdeckung (Coverage) durch Tests -> Auswahl nötig
strukturelle Abdeckung (white box tests): Anweisungs-, Bedingungns-, Pfadabdeckung
funktionale Abdeckung (black box tests): Äquivalenzklassen-, Grenzwertanalyse

Was ist Äquivalenzklassenanalyse, was Grenzwertanalyse? Unterschiede?

Äquivalenzklassenanalyse = Einteilung von Ein-/Ausgabewerten in Klassen gleichen Verhaltens, wählt Werte bei denen Fehler auftreten die auch bei allen anderen Werten einer Klasse auftreten
Grenzwertanalyse = Spezialfall, Fehler besonders oft an den Grenzen einer Klasse (zb off-by-one)

Was sind zustandsbasierte Testmethoden?

Basieren auf Zustandsautomaten, als UML-Zustandsdiagramme dargestellt
Coverage: State-, Transition-, Event-, Path-Coverage

Was sind funktionale Methoden des Testens?

Klassifikationsbaum: systematische Blackbox-Tests, Eintelung anhand möglicher Eingabebereiche und Systemzustände in Klassen, diese in disjunkte Teilmengen zerlegt
Kombination: Minimal (jede klasse min 1), Maximal (jede mit jeder klasse), Paarweise (jeweils zwei Klassen vollständig), Tripelweise (jeweils drei)

Was sind Informelle Testmethoden?

keine systematische Abbildung, nach Intuiton des Testers, beruhen auf Erfahrung, gleichzeitiges Lernen und Testen eines Objekts, Information fließt in neue Tests; oft mit formellen Tests kombiniert

Was sind nichtfunktionale Tests, welche Arten gibt es?

Aufbau abhänging vom Qualitätsattribut, erfordert mehr Expertenwissen, zb Usability, Security,
Efficiency - **Performance**: Latenz, Durchsatz, Ressourcenverbrauch als Metriken
Load Testing: fähigkeit wachsende Last zu bearbeiten, Multi-User-, Volume-Testing
Stress Testing: ability to handle peak loads >= max capacity, Sonderfälle: Spike-, Bounce-Testing
Scalability, Resource Utilization

Was sind Vor- und Nachteile von Testautomatisierung?

vollautomatische Durchführung, Verifikation -> Wirtschaftlichkeit
Initialaufwand und Wartung höher als für manuellen Test, für Ausführung später viel geringer
-> Spezielle Unit-Test Frameworks mit Code-Coverage, auch für GUI (Capture - Replay)

Welche Rolle spielen Testdaten?

zur Konfiguration des Testaufbaus, Datengetriebener Test -> Trennung von Testfall und -daten.
-> wiederverwendbarkeit von beidem getrennt, wart- und lesbarkeit erhöht, testfälle können ohne daten definiert werden, leichtere Automatisierung
sowohl deterministische als auch random daten

Wie findet Fehlermanagement statt?

melden -> bewerten -> zuteilen -> beheben -> Korrektur ausliefern -> nachtesten -> abschließen

Inbetriebnahme, Rollout und Wartung

Software Lebenszyklus: Beschaffung -> Lieferung -> Entwicklung -> Betrieb -> Wartung

Wie kann die Anwenderakzeptanz bei der Einführung von Software gesteigert werden?

Integration der Systemanwender, Graduelle Einführung, Testbetrieb inkl Anwenderfeedback

Anwenderdokumentation

- Einführungsmaterial, Nachschlagewerk
- für Anwender entsprechend formuliert
- Aufbau anhand der Anwender, nicht dem technischen Aufbau
- technische Doku nicht Teil der Anwenderdokumentation

Schulungsmaßnahmen

- Umfang hängt von Applikation ab
- meist mehrer Gruppen mit unterschiedlicher Nutzung u Berechtigung
- Praxisnahe Vermittlung: Übungen, max 10 Personen, Professionelle Trainer

Was ist bei der Inbetriebsetzung von Software zu beachten?

- Aufwand hängt ab von benötigter Soft/Hardware (eigene Maschinen, Datenbank?)
- Klärung von Verantwortlichkeit: Installation, Einrichtung, Bereitstellung von Soft/Hardware
- Umfangreiche Planung notwendig, Iterativ - kann Planung ändern

Was ist bei der Abnahme durch den Kunden zu beachten?

- Aufwand steigt mit Komplexität - Überprüfung auf Vollständigkeit/Korrektheit schwer
- Basis: WAS wird WANN von WEM nach WELCHEN KRITERIEN abgenommen?
- bei größeren Projekten durch mehrere Stakeholder
- Dokumentation des Abnahmeprozesses inkl Ergebnisse
- Ebenen
    - Technisch - Lauffähigkeit, zb Entwickler- und Funktionstests
    - Inhaltlich - semantische Korrektheit; zb Black Box - u User Acceptance Tests
    - Integrationstests - Zusammenspiel getrennt abgenommener Module

Welche Anti Patterns gibt es bei der Einführung von Software?

Manuelle Deployments, Deployments erst nach Abschluss der Entwicklung, manuelle Konfiguration der Betriebsumgebung (Puppet, Chef), Änderungen direkt in der Betriebsumgebung

Was ist Continuous Delivery, welche Werkzeuge gibt es?

ständige automatisierte Auslieferung nach jeder Änderung / Commit

build tools - make, ant, maven, gradle; **version control** - subversion, git; **contiuous integration** - hudson, jenkins; **deployment**: chef, puppet

-> Automatisierung, zuverlässig und wiederholbar, alles versioniert

-> höhere Zuverlässigkeit, weniger Stress, Nachvollziehbar, kürzere Zyklen

Welche Rollen gibt es im Betrieb?

Operator: Laufende Überwachung (24/7), Systeme als Black Boxes

Technischer Admin: administriert einzelne Systeme (DB, Server), Low-Level Zugriff (Logs, Console)

Fachlicher Admin: admin. Applikation über UI, kein Zugriff auf Infrastruktur

Support Mitarbeiter: verschiedene Level, Ansprechpartner für User => nehmen Fehlermeldung auf

Welche Anforderungen gibt es im Betrieb?

Stabilität & Fehlertoleranz, Sicherheit, gute Dokumentation (Referenz), Wiederherstellbarkeit (Restart/Recovery), Analysierbarkeit, Anpassbarkeit

Erkläre Überwachung und Optimierung im laufenden Betrieb

Überwachung: prüft Erreichbarkeit, erlaubt schnelle Reaktion auf Ausfälle, internes Qualitätsbenchmarkung (Verfügbarkeit, Antwortzeiten), Notwendig zur Erfüllung von SLAs
Optimierung: für Optimale Auslastung der Infrastruktur, Balance zwischen Einsparung und Performanceeinbußen, Trend zur Virtualisierung; Achtung: Single Point of Failure!

Was ist Migration, welche Arten gibt es?

Anpassung an geänderte Rahmenbedingungen - geänd. Anforderungen, Weiterentwicklungen darf laufenden Betrieb nicht beeinflussen. Ziel: Produkt in einem geordneten Prozess überholen
Arten: Software-, Daten-, Hardwaremigration

Welche Arten von Datenintegration gibt es?

Ziel: Heterogene Daten aus unterschiedlichen Quellen in einer Applikation zu verwenden.
Physische: in eine gemeinsame Datenbank kopiert; **Logische**: nicht kopiert, bei Abruf konvertiert

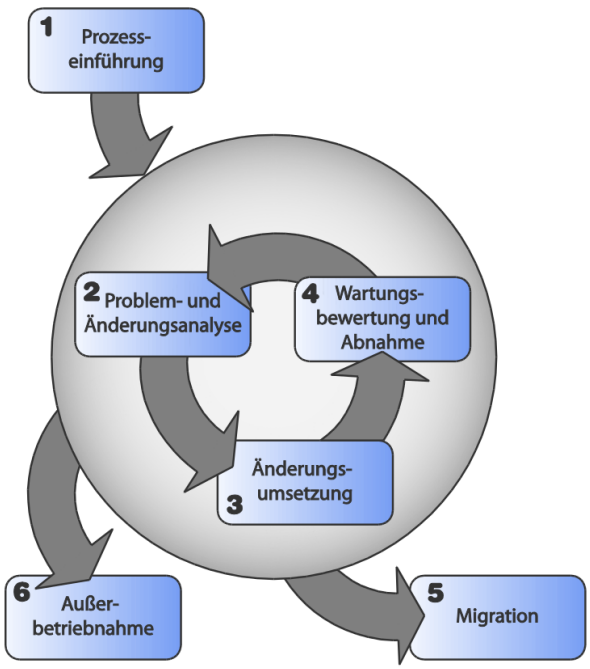
Welche Typen der Wartung werden unterschieden? Erklären Sie die unterschiedlichen Typen der Wartung. Wieso ist die Unterscheidung so wichtig?

Wartung ist keine Weiterentwicklung - keine Änderung der Funktionen

- **korrektive** Wartung - ausbessern von Fehlern; Abweichungen von SLA, Spezifikationen
- **präventive** Wartung - verhindern von vermuteten Fehlern die in Zukunft auftreten könnten
- **adaptive** Wartung - anpassen der Software an veränderte Bedingunen und Umgebungen, Änderung von Schnittstellen, Hardware, verwendeter Software
- **perfektionierende** Wartung - verbessern der Software ohne Funktionalität zu ändern, zb Benutzerfreundlichkeit oder Geschwindigkeit

Welche Wartungsmaßnahmen gibt es?

Reverse Engineering - falls Quellcode nicht vorhanden Rückgewinnung von Code und Modellen aus Artifakten. Trägt zum Code-Verständnis bei und hilft bei der Re-Dokumentation, oft Vorstufe zum Reengineering.
Software Reengineering - Neuentwicklung bei gleicher Funktionalität, soll Qualität verbessern
Refaktorisierung - sowohl während Entwicklung als auch Wartung, soll Qualität hoch halten
Anti-Patterns - schlechte Lösungsansätze die vermieden werden sollen, bieten Verbesserungsansätze



Vorgehensmodelle

Was sind Vorgehensmodelle, wozu sind sie gut?

Vorgehensmodelle sind dazu da, um die Softwareentwicklung auf eine etablierte Ingenieurmäßige Basis zu stellen und Verbesserung in Produktivität, Qualität und Vorhersagbarkeit zu erzielen.
Die Softwareprojekte werden immer umfangreicher-> Vorgehensmodelle werden wesentlich für den Geschäftserfolg.

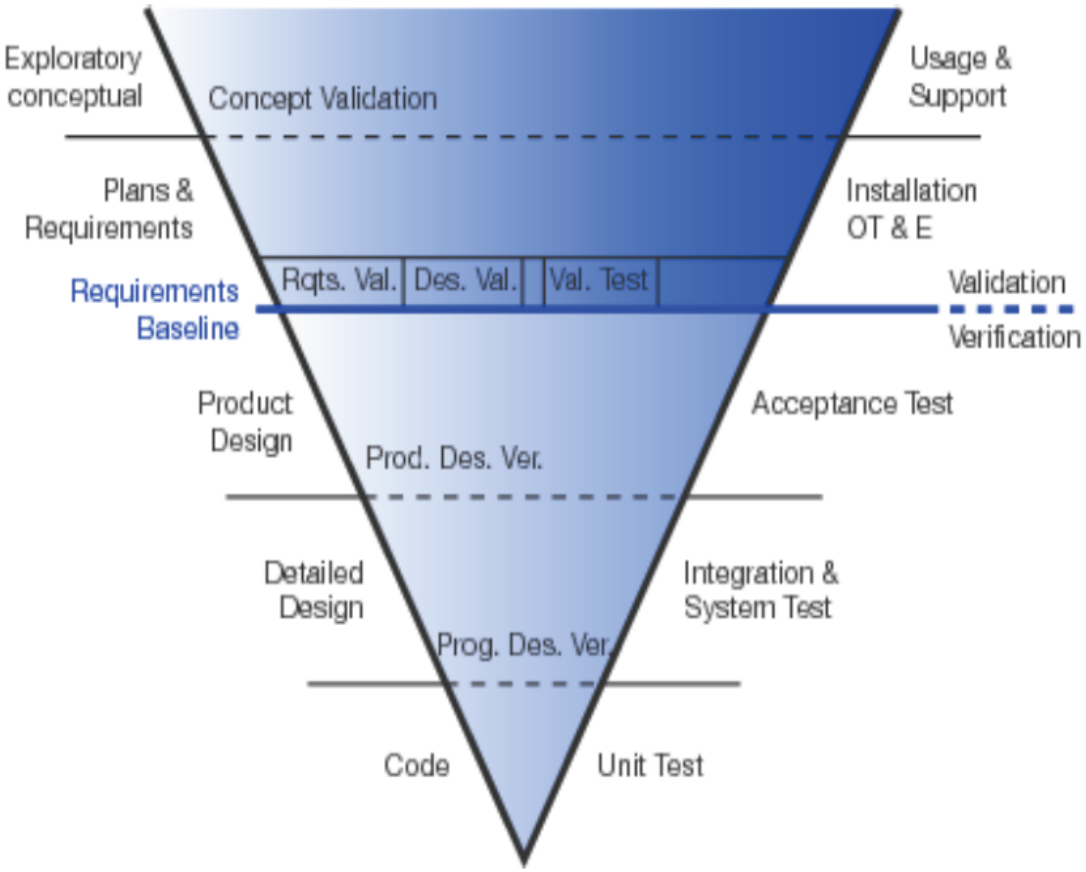
Welche Steuergrößen hat ein Projekt?

Kosten, Qualität, Zeit, Umfang

Beschreibe das Wasserfallmodell nach Royce

- In Grundform vollständig sequentiell.
- lässt in einer Erweiterungsstufe einen Rückwärtsschritt zu
- wird in der modernen Softwareentwicklung als inhärentes Risiko betrachtet und hat zur Entwicklung der iterativen Modelle geführt

Erkläre das V-Modell



Beschreibe das Spiralmodell von Boehm

Berücksichtigt die Risiken der sequentiellen Entwicklung -> iterativ
Prozess in 4 Phasen: 1 Def. von Zielen, 2 Risikoanalyse, 3 Durchf. von Arbeitsschritten, 4 Planen der nächsten Phase
Anzahl der Durchläufe ergibt sich erst im Projektverlauf und hängt von auftretenden Risiken ab.

Erkläre den RUP

Rational Unified Process von IBM
am meist verbreitetes und detailliert beschriebens Prozessmodell
beruht auf:

- Software iterativ entwickeln
- Anforderungen managen
- Komponentenbasierte Architekturen verwenden
- Software visuell modellieren
- Softwarequalität kont. prüfen
- Änderungen kontrolliert in die Software einbringen

Tailoring (Anpassung) mittels kostenpflichtigem Tool RMC

Was ist das Microsoft Solutions Framework (MSF)?

Alternative zum RUP
Projekt setzt sich aus Iterationen und diese wiederum aus kleineren Einheiten (Builds und Check Ins)

Was ist das agile Manifesto?

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge.
- Funktionierende Programme sind wichtiger als ausführliche Dokumentation.
- Die stetige Abstimmung mit dem Kunden ist wichtiger als die ursprüngliche Leistungsbeschreibung in Verträgen
- Der Mut und die Offenheit für Änderungen stehen über dem Befolgen eines festgelegten Plans.

Erkläre SCRUM. Was sind die wichtigsten Element und Rollen?

hyper-produktive Produktentwicklung, beim Rugby angewandte Strategie (adaptive, schnelle, sich selbst organisierende Vorgehensweise mit wenigen Ruhepausen)
Werte

- hohe Produktivität und Anpassungsfähigkeit
- wenig Risiko und Ungewissheit
- größerer Komfort für die Projektmitglieder

Alle Anforderungen im **Product Backlog**, **Product Owner** ist verantwortlich dafür.
Entwicklung in kleinen Teams in **Sprints** - Sprint muss mit neuer Funktionalität enden, dazu **Daily Scrum Meetings** und am Schluss **Sprint Review Meeting**
Softwareprodukt entsteht in **Inkrementen**, spätestens im zweiten Sprint lauffähiges Softwaresystem
Burndown Chart: Scrum Master trägt tagesaktuelle Aufwandsschätzungen ein
Rollen:
Product Owner: Erstellung, Priorisierung des Product & Release Backlogs, Team von Störungen/Unterbrechungen frei halten
Scrum Team: Umsetzung der vom Sprint Backlog geforderten Funktionalität, Erreichung des Sprint Goal
Scrum Master: achtet auf Umsetzung der Grundsätze, Regeln von Scrum, schottet das Team von äußeren Einflüssen ab

Erkläre eXtreme Programming. Welche Qualitätsanforderungen gibt es?

kompakte Methode zur Softwareentwicklung in kleinen bis mittelgroßen Teams, deren Arbeit vagen, sich rasch ändernden Anforderungen unterliegt
Hauptziel: termingerechtes Abliefern auftragsgemäßer Software
Merkmale: eigenes Wertesystem, viele Prinzipien, System von sich gegenseitig unterstützenden Techniken

Prinzipien:

- Kommunikation zwischen allen Beteiligten
- Einfachheit
- Feedback
- Mut (Offenheit, Transparenz, Änderungen)
- Respekt (vor sich selbst, dem Team, dem Produkt)

Techniken: Kunde vor Ort, Kurze Release-Zyklen, Testen, Einfaches Design, Refactoring, Pair Programming, Gemeinsame Verantwortlichkeit, 40-Stunden-Woche, Programmierstandards
Rollen:Entwickler, Kunde, XP-Trainer, Tracker, Tester, Berater, Big Boss

Was ist Kanban?

Keine Rollen, Keine Meetings, Kanbanboard, Work In Progress Limit
The Rest is up to You

Erkläre das V-Modell 97

Entwickelt um alle Ansätze unter einen Hut zu bringen.
Ausgerichtet auf Durchführungsphase
Es dient

- als Vertragsgrundlage bei der Auftragsvergabe
- als Arbeitsanleitung für die SE
- als Kommunikationsbasis zw den Parteien

Entwicklungsphilosophie:

- inkrementell
- zu Beginn Gesamtfunktionalität abgegrenzt und die einzelnen Funktionen werden priorisiert
- einzelne Funktionen werden nach der Priorität den Ausbaustufen zugeordnet und umgesetzt
- Ergebnis einer Stufe -> Endanwender
- Somit wächst der Umfang des Systems ausgehen von einer Basisfunkt. stetig an

Wie unterscheidet sich das V-Modell XT vom V-Modell 97?

Extreme Tailoring / extendable

- Bessere Anpassungs- und Anwendungsmöglichkeit bei unterschiedlichen Projekten und Organisationen
- Skalierbarkeit hinsichtlich verschiedener Projektgrößen
- Bessere Änderungs- und Erweiterungsmöglichkeiten des Modells selbst
- Anpassung an die aktuellen Normen und Vorschriften
- Betrachtung des gesamten Systemlebenszyklus
- Unterstützung der Einführung sowie Verbesserung des Modells selbst und Anpassung an organisationsspezifische Gegebenheiten

Eigenschaften

- Ziel- und ergebnisorientierte Vorgehensweise
- Produkte stehen im Mittelpunkt des V-Modells und stellen die zentralen Projektergebnisse dar
- Die Reihenfolge der Produktfertigstellung und damit die grundlegende Struktur des Projektverlaufes werden durch Projektdurchführungsstrategien und Entscheidungspunkte vorgegeben
- Auf Basis der Bearbeitung und Fertigstellung von Produkten wird die detaillierte Projektplanung und -steuerung vorgenommen

ISO/IEC 12207 - Prozesse

Primary Processes: Funktionen die von beteiligten Parteien ausgeführt werden können

Supporting Pro.: können von primären oder anderen supporting Prozessen verwendet werden
Organizational Pro.: werden eingesetzt um Aufgaben, die organisatorische bzw. projektübergreifende Aspekte betreffen,umzusetzen und andere Prozesse zu etablieren, zu kontrollieren und zu verbessern

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

weder das eine noch das andere stellen eine “Silver Bullet” dar.
agile für SP mit raschen time-to-market
Teamstruktur (Personnel) Boehm und Thuner nehmen in ihrem Modell an, dass die Qualifikation des Teams bzw. der beteiligten Personen beim Einsatz agiler Methoden höher sein muss als beim Einsatz plangetriebener Methoden.
Dynamik der Anforderungen (Dynamism) Je häufiger Anforderungen geändert werden, desto einfacher muss dies möglich sein.
Entwicklungskultur (Culture) Je nach organisatorischem Umfeld bzw. Entwicklungskultur können agile Methoden eingesetzt werden.
Teamgröße (Size) Es wird oftmals argumentiert, dass agile Methoden in kleinen bis mittelgroßen Teams effizient einsetzbar sind.
Kritikalität (Criticality) Wenn Menschenleben von einem Softwaresystem abhängig sind, werden natürlich viel höhere Maßstäbe an die rigorose Prüfung und Nachvollziehbarkeit auf allen Ebenen gelegt.

Projekt- und Risikomanagement

Was ist ein Projekt, wie kann man es klassifizieren?

„Vorhaben, das im Wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, wie z.B. Zielvorgabe, zeitliche, finanzielle, personelle oder andere Begrenzungen, Abgrenzung gegenüber anderen Vorhaben und projektspezifische Organisation.“
Art: Forschungs-, Entwicklungs-, Planungs-, Organisations-, Rollout-, Investitionsprojekte
Größe: Personal-, Finanzaufwand, Dauer, # Mitarbeiter, eigenes Budget ?

Welche Tradeoffs gibt es beim Projektamangement?

2 aus Qualität, Kosten, Zeit Qualität vs Kosten Umfang vs Zeit

Wie ist ein Projekt definiert?

Machbarkeitsstudie: **Was** wird **wie** getan?
Projektumfeldanalyse: Projekttyp, Funktionalitäten, Qualität, mögl. Risiken, unterstützende Faktoren
Anforderungskatalog: Erste Planungsunterlage, erfasst Projektziele
Pflichtenheft: Vereinbarung zwischen AG und AN, erweitert Katalog, fachliches Grobkonzept
Leistungsbeschreibung: Fachliches Feinkonzept, technisches Grobk., legt Inhalt verbindlich fest
Wirtschaftlichkeitsbetrachtung:

STATISCHE VERFAHREN	DYNAMISCHE VERFAHREN	BEWERTUNGSVERFAHREN
Kostenvergleichsrechnung	Kapitalwertmethode	Kosten-Nutzen-Analyse
Rentabilitätsrechnung	Annuitätenmethode	Nutzwertanalyse
Amortisationsrechnung	Interne-Zinsfußmethode	

Wie kann ein Projekt organisiert werden?

Reine Projektorganisation

- Projektleiter trägt Gesamtverantwortung, volle Weisungsbefugnis
- **Vorteile**: eindeutige Verantwortung, starke Identifikation der Mitarbeiter mit dem Projekt, hohe Motivation
- **Nachteile**: schwierige Akquirierung der Mitarbeiter, problematische Wiedereingliederung der Mitarbeiter am Ende des Projekts,

Einfluss-Projektorganisation

- Projektleiter nur Projektkoordinator, kaum Weisungsbefugnis
- **Vorteile**: geringe organisatorische Veränderungen, hohe personelle Flexibilität, Kommunikation mit Fachabteilung bleibt erhalten
- **Nachteile**: keine klare Verantwortlichkeit, Fehlende Autorität des Projektleiters, hohes Konfliktpotential, fehlender Teamgeist

Matrix-Projektorganisation:

- Projektleiter trägt Gesamtverantwortung, keine volle Weisungsbefugnis
- Vorteile: Projektverantwortung durch Projektleiter, Sicherheitsgefühl der Mitarbeiter, flexibler Personaleinsatz
- Nachteile: Mitarbeiter Diener zweier Herren, Konfliktpotential Linie/Projekt, hoher Koordinationsaufwand

Was ist ein Projektauftrag?

Schriftliche Vereinbarung zwischen AG und AN, wichtigste Eckdaten als Zielvereinbarung
Vertragscharakter, Unterzeichnung von beiden Parteien => Geburtsschein des Projekts

Wie geht man bei der Projektplanung vor, welche Faktoren sind wichtig?

Vorgehen: Projektstrukturplan -> Arbeitspaketdef. -> Aufwandsschätzung -> Ablauf u Terminplanung -> Ressourcenplanung -> Kostenplan -> Optimierung des Gesamtplans
Faktoren: Motivation (warum?), Leistung, Qualität, Kosten, Zeit

Was ist ein Projektstrukturplan, wie wird er dargestellt?

Übersichtliche Darstellung des Projektinhalts zur Erkennung von Schwerpunktaufgaben, systematische Erfassung aller Aktivitäten
Entweder Top-Down (Zerlegung der Gesamtaufgabe) oder Bottom-Up (Sammlung von Aufgaben, Strukturierung nach Kriterium, Aufbau einer Hierarchie)
Dargestellt als Organigramm oder Liste

Was ist ein Arbeitspaket?

kleinste Einheit des Projektstrukturplan, Umfang sehr unterschiedlich
Kritierien: Konkretes Ergebnis, kalkulierbar, keine Überschneidung, klare Schnittstellen, eindeutige Verantwortlichkeit

Nennen und erklären Sie eine Methode zur Aufwandsabschätzung

Information durch Erfahrungen bewertet und in Aufwand umgerechnet, erfordert gute Wissensbasis
Delphi Methode

- Moderator + mehrere Experten
- zu jedem Arbeitspaket gibt jeder einen Schätzwert ab
- Mittelwert der Schätzwerte falls alle Schätzwerte rund um MW ±20% (sonst diskussion und neue Schätzung)
- Zuschläge zu den Einzelschätzungen (10-15% Aufwand PM)

3-Experten-Konzept

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

Wie funktioniert Terminplanung, was ist GANTT Diagramm, die 5 Beziehungen?

Terminplanung: Elemente des PSP in logische Reihenfolge, Ermittlung der Abhängigkeiten, Zusammenfassung zu Sammelvorgängen, Festlegen von Meilensteinen, Darstellung der Vorgänge
GANTT Diagramm: Älteste und am weitesten verbreitete grafische Methode, Aufgaben werden zur Terminplanung dargestellt.

- Ende-Anfangsbeziehung
- Anfangs-Anfangsbeziehung
- Ende-Endebeziehung
- Anfangs-EndeBeziehung
- Überlappung/Verzögerung (Durch Zeit- oder Prozentangabe wird definiert, wann der Nachfolger vor oder nach dem Anfang oder Ende des Vorgängers beginnen kann)

Terminberechnung: Vorwärts: Anfangstermin fest - Rückwärts: Auslieferdatum fest
Kritischer Pfad: nur kritische Vorgänge (Vorgang ohne Pufferzeit)

Wie können Ressourcen und Kosten geplant werden?

Bedarfsermittlung, Ermittlung der Kapazität, Soll/Ist Vergleich, Kapazitätsabgleich
Kostenarten: Personal, Material, Betriebsmittel, Kapital, Finanzierung, Fremdleistungen

Wie kann ein Projektplan optimiert werden?

Entwurf wiederholt überarbeiten, Alternativen erstellen, optimale Lösung wählen
Termine: Konzentration auf kritischen Pfad, **Ressourcen**: bessere Auslastung erzielen, **Kosten**: Teure Ressourcen durch billigere ersetzen. **Qualität** nicht vergessen!

Welche Methoden zum Projectcontrolling gibt es?

Projektüberwachung

Ziel: Rechtzeitiges Erkennen von Abweichungen gegenüber Plandaten
Vorraussetzungen: Detailierte, realistische Planung; Zeitnahe Kenntnis der Ist-Daten
Vorgehen: Berichtswesen (Ist-Daten); Soll/Ist-Vergleich (Kontrolle oder Trendanalysen); Analyse der Abweichungen (Ursache und Bewertung)
Beispiel: Meilenstein-Trendanylse - Datum der Meilensteine zu Berichtszeitpunkten plotten => Kurven

Projektsteuerung

soll Projekt auf Kurs halten.
Korrektive Maßnahmen: Angleichung Ist an Soll - Motivation verbessern, Neuverhandlungen, Kostenkontrolle verstärken
Reaktive Maßnahmen: Anpassung Planung an Situation - Termine verschieben, Budget erhöhen, Projektteam erweitern, Funktionen reduzieren

Wie funktioniert Projektkommunikation, was für Fehler gibt es?

Ziel: einheitliches Besprechungs- und Präsentationswesen
Kommunikationsplanung - Informationsverteilung - Fortschrittsberichte - Administrativer Abschluss
Fehler

- Informationen nur gefiltert an Projektteam weitergegeben
- Unternehmensbereiche zu Beginn nicht über das Projekt informiert
- Informationen über Projektfortschritt fließen nur spärlich in Linienbereiche
- Informationen über Projektfortschritt fließen nur spärlich /gar nicht an den Auftraggeber

Projektbesprechungen

- Regelmäßige Projektbesprechungen zu fest vereinbarten Terminen
- Ergebnisgesteuerte Besprechungen - Entscheidungsgremium über Start/Schluss von Phasen
- Ereignisgesteuerte ~=~ - bei unerwarteten Ereignissen, zb Krisensitzung

Wozu dienen die verschiedenen Dokumentationsarten, was sollen sie enthalten?

=> Festhaltung aller wesentlichen Projektereignisse
Projektdokumentation: Besprechungsprotokolle, Kostenaufstellung, Pflichtenheft und Fachkonzept, Testprotokolle, Projektabschlussbericht, Produktdokumentation, Offene Punkte und Erweiterungsmöglichkeiten, Erfahrungsbericht
Prozessdoku: Projektantrag, -auftrag, -planung, -berichte, -abschlußbericht
Benutzerdokumentation: Schulungsunterlagen, Benutzeranleitung
Wartungs- u Weiterentwicklungsdoku: Feinkonzept, Programmlistings, Testpläne
Betriebsdoku: Installationsbeschr., Betriebshandbuch

Was ist für einen erfolgreichen Projektabschluss notwendig?

Abschluss der Projektdokumentation

- Projekt-Abschlußbericht - Zusammenfassung der wichtigsten Ergebnisse des Projektes
- Adressaten: Auftraggeber/Sponsor und Gremien

Abschlusspräsentation

- Dient der Projektabnahme (intensive Vorbereitung)
- Darstellung der Ziele und Ergebnisse, zusätzlicher Nutzen des Projektes (besondere Erfolge)
- Durchgeführte Arbeiten (grober Überblick)

Projektabnahme durch den Auftraggeber (Abnahmeerklärung)

- Abnahmeverfahren („Big Bang“ oder schrittweise Abnahme)
- Projektleiter übergibt formal die Ergebnisse
- Auftraggeber prüft, ob Projektziele erreicht wurden

Freigabe der Projektmitglieder und –ressourcen

- Auflösung des Projektteams (mögliche Konflikte), sonstige Ressourcen (Räume, Rechner,...)
- Projektende und Auflösung gemeinsam planen

Nachbetrachtung der Projektarbeit (Review)

- Abschlussanalyse
    - Nachkalkulation und Kostenabweichungsanalyse
    - Wirtschaftlichkeitsanalyse
- Erfahrungssicherung (Lessons Learned)
    - Essentielle Erfahrungen für neue Projekte sammeln, auswerten und sichern
    - Was hat jeder Teilnehmer aus dem Projekt gelernt?
    - Welche Ergebnisse sind für die Gesamtorganisation wichtig?
    - Welche positiven Erfahrungen sind relevant für andere Projekte?
    - Was soll bei künftigen Projekten anders gemacht werden?
- Abschlussbesprechung (inkl. Social Event)

Was ist Risikomanagement, welche Begriffe gehören dazu?

Risiko: unsicheres Ereignis, von dem nicht bekannt ist, ob es eintreten wird und welchen Schaden es verursachen wird
Risikofaktor = Risikowahrscheinlichkeit * Schadenshöhe
Risikomanagement: systematischer Prozess zur Identifizierung, Analyse, Behandlung und Kontrolle der Projektrisiken
Risikoidentifikation: Ziel: vollständige Liste konkret formulierter Risiken; Workshops mit Kreativitätstechniken; Checklisten, Fragebögen; Erfahrungen aus vorangegangenen Projekten
Risikoanalyse: Ziel: qualitative und quantitative Bewertung der Risiken; Eintrittswahrscheinlichkeit je Risiko; Schadensausmaß je Risiko; Risikomatrix zur Priorisierung
Risikobehandlung und -kontrolle: Ziel: Erstellung eines Maßnahmenplans; Verhältnismäßigkeit Schadenshöhe/Eintrittswahrscheinlichkeit – Aufwand/Kosten; Maßnahmenplan:

Welche Projektmanagement Software gibt es, was kann sie?

Funktionen: Aufgaben, Termin-, Ablauf-, Ressourcen-, Kosten-, Finanzplanung, Aufgabenverteilung, Workflow & Koordination, Informationsverteilung, Risikoanalyise, Projektcontrolling, Berichte
PM-Software für Teilaufgaben: eingeschränkt, Terminplaner und Hilfsprogramme
kleine u. mittlere PM-Software: typischer PM-SW Funktionsumfang, Grenze zu PM-System fließen
große PM-Systeme: speziell für Ressourcen-, Kostenplanung, Projektcontrolling, Multiprojektmang.
+ übersichtlich, einheitlich, Zeitersparnis, flexible Aufgabenerled, höhere Qualität durch besser Analyse, Automatisierung, erweitertes Controlling, transparent für Team, berechnet kritische Pfade
- kein Garant für Erfolg, hoher Aufwand, oft überdimensioniert, Beteiligte “verlassen” sich auf Tools
Beispiele: GanttProject, MS Project, Artemis, Planview, Open Workbench, Superproject, …

Welche Rolle spielt der Faktor Mensch beim Projektmanagement?

Management von “Human Resources” Kernaufgabe von PM, entscheidende Bedeutung für

Erfolg/Misserfolg, Fehler meist extrem negativ für das Projekt
Team: mehrere Personen vollbringen etwas in gegenseitiger Abhängigkeit; aufgabenorientierte Arbeitsgruupe mit starken Kontakt und direkter Kommunikation
Gliederung nach Ort, Größe, Bekanntheit der Teammitglieder untereinander
+ mehr Ideen, produktiver Wettbewerb, beschleunigte Infoverteilung, Gesamtproblemsicht, Identifikation mit Gesamtlösung, Gruppenentscheidung meist besser, Teamgeist
- Komm. besonders zeitaufwendig, Diskussionen manchmal unproduktiv, keine gemeinsame Plattform in zu heterogenen Gruppen
Rollen von Mitgliedern: Fachspezialisten, Teamarbeiter, Botschafter - sollte ausgewogen sein

Wie funktioniert Teambildung, wie sieht ein erfolgreiches Team aus?

Forming: Team kommt zusamm, Aufgaben verteilt
Storming: Normen u Ziele hinterfragt, Strukturen ändern sich, Aufgaben wechseln, Personen gehen
Norming: Team legt Regeln fest, Führung und Strukturen bilden sich
Performing: zusammengeschweißtes Team arbeitet effizient
Merkmale erfolgreicher Teams: hoher Zusammenhalt, Engagement, fachlich-sozial ausgewogen, Anerkennung von außen, klare Aufgaben- u Rollenverteilung, Teamleiter nicht autoritär, jeder Beitrag aufgenommen, Konflikte offen angesprochen, Motivationssystem, Entscheidungskompetenz

Welche Rollen muss ein Projektleiter einnehmen?

Architekt, Stratege, Führungskraft, Controller, Moderator, Konfliktmanager, Motivator, Psychologe, Sündenbock

Welche Führungsstile gibt es?

= zeitlich überdauerndes, konsistentes Führungsverhalten von Vorgesetzten gegenüber Mitarbeitern
Dimensionen: Teilnahme am Entscheidungsprozess, Aufgaben- vs Personenorientierung
Autokratischer Führungsstil: Projektleiter entscheidet über Inhalt, wenig Kritikbereitschaft, genau Anweisungen u Kontrolle, keine kreativen Einbringunsmöglichkeiten
Kooperativer Führungsstil: Beiteiligung, Delegation von Aufgaben, Transparenz bei Entscheidungen, Ergebniskontrolle, Betonung der Eigenverantwortlichkeit
Demokratischer Führungsstil: Entscheidung durch Gruppendiskussion, persönliche Präferenzen berücksichtigt, Projektleider schlägt nur vor
Situativer Führungsstil: unterschiedliche Stile ja nach Situation, zb GRID-Führungsmodell mit Orientierung an Menschen und der Sache
Authentischer Führungsstil: anders als übliche Modelle, sehr erfolgreich, individuelle Persönlichkeit steht im Vordergrund. **Grundwerte**: Akzeptanz, Respekt, Vertrauen, Toleranz, Offenheit für Neues

Welche Faktoren spielen bei der Motivation mit, wie kann sie gefördert werden?

Maslow Pyramide: Körperliche Grundbed., Sicherheit, Beziehung, Anerkennung,Selbstverwirklichung
Förderung: gemeinsames Ziel anstreben, Bedürfnisse berücksichtigen, gemeinsame Aktivitäten, laufendes Feedback, Weitergabe von Anerkennung, an kleinen Fortschritten aufrichten, Motivation des Projektleiters muss hoch bleiben, Projektleiter muss Können unter Beweis stellen

Qualitätssicherung

Welche Arten der Softwarequalitätssicherung gibt es?

Statische analytische QS: stat. analyse / Code metriken, Review/Walkthrough/Inspection
Dynamische analytische QS: Testen, Dyn. Analyse
Organisatorische QS: Templates/Checklisten, Wissensmgmt, Qualitätsmgmtstandards

Was sind Softwarequalitätsfaktoren, welche gibt es?

ermöglichen Aussagen über die Qualität der Software zu treffen
ISO/IEC 9126: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit

Beschreibe Softwarefehler

Softwarfehler -> Softwarefehlerfall -> Softwarefehlverhalten
häufig werden Fehlerfälle gar nicht oder nur selten ausgeführt
Ursachen: fehlerhafte Definitionen in / falsche Interpretation der / logische Designfehler auf Grund von Anforderungen, Fehlinterpretation des Desigdokuments, Sprachliche Fehler in der ProgSprache, Fehler in Entwicklungswerkzeugen oder bei der Datenauswahl, Nichteinhaltung von Code- oder Dokumentationsrichtlinien
Behebung: je früher (Phase) desto billiger

Welche Kosten hat die Qualitätssicherung?

Setzen sich aus den Herstellungskosten und Qualitätskosten zusammen
Ziel: vermeidung von Fehlern und deren Folgekosten
Prüfkosten: QS Prüfkosten, Fehlervermeidungskosten
Fehlerkosten: Interne Fehlerkosten, Externe Fehlerkosten

Beschreibe die Statische QS

- Analytische Prüfung eines Objekts, ohne dass dieses dyn. durchgeführt wird
- Prüfung bereits sehr früh in Projektlebenszyklus möglich, noch bevor Code zur Verfügung
- Prüfung einzelner Programmteile bereits vor der Integration
- Mit Hilfe von Strukturanalyse: innere Qualität kann ermittelt werden (Basis: Abhängigkeitsgraphen und Q.metriken).
- Prüfung der Einhaltung der Richtlinien durch syntaktische überprüfung
- Fehlermusteranalyse prüft den Code gegen typische Fehlermuster

Warum werden Softwaremetriken in der Qualitätssicherung verwendet?

Ermöglichen Softwarequalität zu interpretieren
Prozessmetriken: quantitative Daten über Entwicklungsprozess
Produktmetriken: messen die Software oder Teile davon (ohne Entwicklung oder Zustand)
Umfangsmetriken: Lines of Code
OOMetriken: berücksichtigen die Beziehungen zw Objekten und deren Struktur:

- CBO coupling between objects
- DIT depth of inheritance tree
- NOC number of children
- RFC response for class
- WMC weigthed methods per class
- LCOM lack of cohesion in methods

Beschreibe den Review-Prozesses nach IEEE. Wieso sind Reviews so wichtig?

Review - formell organisierte Zusammenkunft von Personen zur Überprüfung eines Produktteils nach vorgegebenen Prüfkriterien und -listen (IEEE-Std 1028).
Review-Prozess: Planungsphase -> Initialisierungsvorgang -> (Vorbereitung - Sitzung - Nacharbeit) -> Freigabe in der klammer: eigtl Review
im Anschluss aussagen über die Effektivität des Reviews (über Metriken und Analysen)
VT: Reviews noch bevor man testen kann -> Latenzzeit eines Fehlers in der Entwicklung verkürzt -> geringere Kosten (kompensiert Reviewaufwand)

Rollen:

- **Manager** - PL, der den Auftrag gegeben hat und verantwortlich für die Freigabe ist
- **Moderator / Review-Leiter** - plant, organisiert und leitet das Review.
- **Schreiber** - notiert Erkentisse ins Protokoll
- **Autor** - Urheber des Testsobjekts
- **Gutachter / Reviewer** - begutachtet das zu prüfende Material vorab und berichten darüber
- **Leser** - Bereit zu prüfendes Material auf und verantw. für die zeitlich angemessene Durchf.

Welche Metriken werden bei Reviews verwendet?

Über Schlüsselmetriken:

- Total noncoment lines for source code inspected, in thousand (KLOC)
- Average... lines of code inspected, preparation rate, inspection rate,effort per KLOC, effort per fault detected, faults detected per KLOC
- Percentage of reinspection
- Defect-removal efficiency

Welche Review-Verfahren gibt es?

- Stellungnahme - Kopien gelesen -> kommentiert retourniert -> Analyse -> möglw. Korrektur
- Walkthrough - regen Diskussion an
- Technisches Review - Beurteilung durch ein qualifiziertes Team (fast Inspektion)
- Inspektion - Formellste und wirkungsvollste
- Round Robin Review - Umkehrung -> Gutachter suchen nach etwas positiven
- Peer Review

Beschreibe die Dynamische QS

Prüfen und Überwachen währen der Laufzeit
System mit Testdaten ausgeführt; Testen
Dynamische Analyse: Interaktion zw Komponenten, Debugger, Programm-Trace, wo treten Unregelmäßigkeiten auf?

Was ist die Organisatorische QS?

Bereitstellung von Infrastrukturkomponenten zur Vermeidung von Fehlern (oder Verringerung)

- **Wissensmgmt**: Weiterentwicklung von Ideen, Umgang mit Ressource Wissen (Fähigkeiten, Erfahrung, Fertigkeiten, Normen, Routinen)
- **Konfigurationsmgmt**: Verwaltet Spezifikation, Dokumentation. Änderung der Anforderungen; Ablauf: Objekt wird eingreicht -> Identifiziert,versionisiert und archiviert -> zur Prüfung freigegeben -> freigegeben -> Änderung wird eingereicht -> veranlasst.
- **Templates**: Dokumenten werden auf Vollstädnigkeit und QUalität gesichert
- **Checklisten**: in Frageform gekleidete Richtlinien und Hypothesen von vermuteteten Schwachstellen (ja oder nein)

Was sind Qualitätsmanagementstandards?

- beinhaltet aufeinander abgestimmte Tätigkeiten für Leiten & Lenken einer Organisation bzgl Q
- sollen die Entwicklung eines UN hin zu einen hohen Qlevel und zu Kontinuität unterstützen
- verbessern das wechselseitige Verständnis und Koordination unterschiedlicher Teams
- Kunde kann von einem Qualitätsniveau ausgehen -Zertifizierungs- und Assesmentsstandards

Was sind ISO 9001 und 9000-3?

Zertifikat für Qualitätsstandard: Prozessgedanke in Vordergrund, Erfüllung der Kundenbedürfnisse
Prinzipien: Kundenorientierung, Prozessorientierung, Mitarbeiterführung, Zuständigkeit der

Mitarbeiter, Kunden-Lieferanten-Verhältnis, Kontinuierliche Verbesserung, Betrachtung von Prozessen als System, Sachliche Herangehensweise bei der Entscheidungsfindung
Anforderungen: Qualitätsmanagementsystem, Verantwortung des Managements, Ressourcenmgmt, Produktrealisierung, Messungen, Analysen und Verbesserungsmaßnahmen
Zertifizierungsprozess: Entscheidung für Zertifizierung -> Planung der Prozesse -> Entwicklung des QS-Systems -> (Review, Anforderungen eingehalten?) ja -> Umsetzung des QS Systems -> (Audit, wenn nicht erfüllt verbessern) -> Zertifikat

Beschreibe das Capability Maturity Model (CMM)

entwickelt von US Verteidungsmin.
Ziel: Vergeleichbarkeit und Qualität von SP im Rahmen der Vergabe an Lieferanten verbessern
Verbreitung -> CMMI entwickelt

Was ist das Capability Maturity Model Integrated (CMMI)?

Analyse des Ist-Zustandes (kein international anerkanntes Zertifikat)
CMM-Assesment beurteilt die Fähigkeit einer Organisation Software unter bestimmten Rahmenbedingungen erstellen zu können
Prinzipien und Konzepte:

- Quantitative Managementmethoden erhöhen die Fähigkeit eines Unternehmens, die Qualität zu kontrollieren und die Produktivität zu steigern
- Die Anwendung des Fünf-Ebenen-Capability Maturity Model ermöglicht die Bewertung der Leistungen und bestimmt die Aufwände, die zur Erreichung der nächsten Stufe notwendig sind
- Generische Prozessgebiete, die das „was“ - nicht „wie“ definieren, ermöglichen die Anwendung des Modells auf eine breite Palette von Organisationsumsetzungen

Reifegrade:

- initial - Keine Anforderungen
- wiederholbar
- definiert
- geführt
- optimiert

Wozu dient SPICE? Was sind die 6 Reifegrade?

Software **P**rocess **I**mprovement for **C**apability D**e**termination, ähnlich CMMI -> Konkurrenz
zweidim Referenzmodell: Prozessdimension <-> Fähigkeitsdimension
Prinzipien und Ziele

- Zusammenführung verschiedener Assessment-Methoden zu zusammenhängendem Systemmodell
- Universelle Einsetzbarkeit
- Hohe Professionalität
- internationale Akzeptanz, weltweiter Standard

Zielsetzungen

- Verstehen des Entwicklungsstandes der eigenen Prozesse -> Verbesserungen
- Eignung der eigenen Prozesse in Bezug auf Anforderungen
- Eignung der Prozesse anderer Organisationen in Bezug auf Verträge

Reifegradstufen

0. unvollständig
1. vorhanden: Process Performance
2. geleitet: Performance Management, Work Product Management
3. gesichert: Process Definition and Tailoring, Process Resource Availability

4. vorhersehbar: Process Measurement, Process Control
5. optimiert: Process Change, Continuous improvement

Erfüllungsgrade: zb nicht / teilweise / weitestgehend / vollständig erreicht
Prozesskategorien: Engineering, Customer-Supplier, Management, Support, Organisation

Unterschied Verifikation und Validation

- **Validation.** The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with *verification*. Erfüllt Bedürfnisse?
- **Verification**. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with *validation*."
  Erfüllt Spezifikation?

Zusammenfassung QS

- Organisatorische Maßnahmen schaffen die notwendigen Rahmenbedingungen und sorgen dafür, dass analytische Maßnahmen stattfinden können
- Statische analytische Maßnahmen (z.B. Reviews) und dynamische analytische Maßnahmen (z.B. Tests) hingegen versuchen, Fehler und Mängel zu erkennen und zu lokalisieren
- Da es nicht möglich ist, fehlerfreie Software herzustellen, muss das Verhältnis zwischen Fehlerkosten, Fehlervermeidungskosten und Qualität optimiert werden.
- Kostenoptimale Qualität wird erreicht, wenn die Kosten für die Vermeidung und Prüfung von Fehler genauso hoch sind wie die Kosten für deren Behebung. Das ist auch jener Punkt, an dem die Gesamtkosten am geringsten sind.
- Weitere Vermeidungs- und Prüfmaßnahmen sind nicht mehr wirtschaftlich, da mit steigendem Aufwand die Fehlerfindungsrate deutlich sinkt. Es wird also immer schwieriger und aufwändiger, Fehler zu finden.
- ISO 9000-3: Primär steht hier der Prozessgedanke im Vordergrund. Jeder Prozess orientiert sich an der Erfüllung der Bedürfnisse des Kunden. Im Vergleich zu anderen Qualitätsmanagementsystemen steht ein Zertifikat im Vordergrund, das den Nachweis eines Unternehmens liefert, normenkonform zu sein.
- Capability Maturity Modell Integrated (CMMI): Dieses Modell liefert eine Analyse des Ist-Zustandes eines Unternehmens und ist kein international anerkanntes Zertifikat wie ISO 9000-3. Ein CMM-Assessment beurteilt die Fähigkeit einer Organisation, Software unter bestimmten Rahmenbedingungen erstellen zu können.
- SPICE (Software Process Improvement and Capability Determination): Dieses Modell ist dem CMM-Standard sehr ähnlich.SPICE stellt auch ein spezifisches Framework für Assessments von Softwareprozessen für alle Phasen und Bereiche der Softwareherstellung bereit.

Security

Was sind SQL-Injection Angriffe? Wie kann man sich davor schützen?

Usability