1-1 Software Engineering Einführung

- Typische Probleme großer SW-Projekte
  - Zeitverzögerung
  - Budget-Überschreitung
  - Mangelnde Qualität
  - Spätes Erkennen von Design Fehlern (late design breakage)
  - Schwierige und teure Wartung
  - SE/QM-Ansätze
    - Formale Methoden: Verifikation, bessere Programmiersprachen
    - Prozessverbesserung: Produkte, Prozesse, Vorgehensmodelle
    - Personen: Training, Motivation, "Kultur guter Arbeit

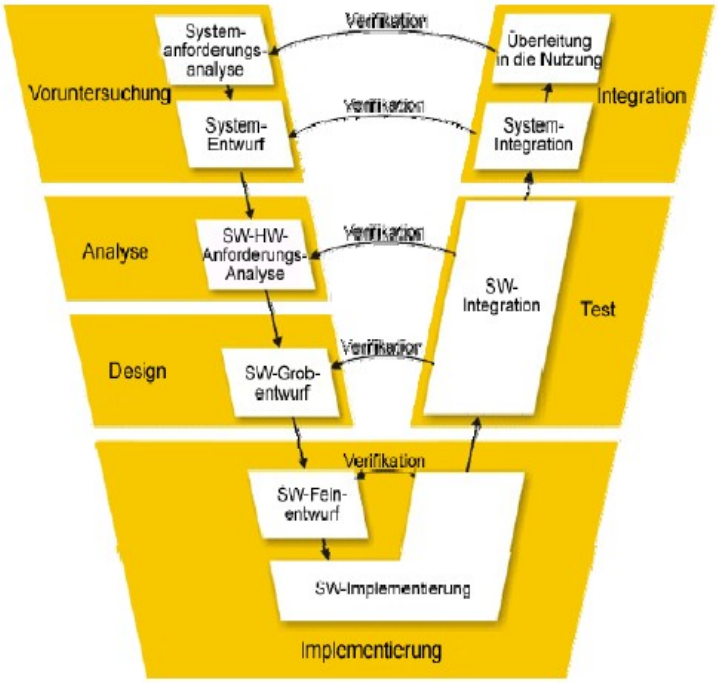
1-2 Projekttypen

- Komplexitätstreiber
  - Größe (PM) – Anzahl beteiligter Personen
  - Dauer und Aufwand (PM) – Anzahl Wochen
  - Verwendete Technologien (SE) – Art, Anzahl, Alter
  - Komplexität (SE) – Anzahl Klassen, Module, Datenbanken; Zeilen Code
- Größe eines Projekts
  - Klein
    - 1-6 Personen
    - 0-8 Monate
    - < 5 Technologien
  - Mittel
    - 10-30 Personen
    - 9-24 Monate
    - 5-12 Technologien
  - Groß
    - 50-100 Personen
    - 25-45 Monate
    - 12-20 Technologien
  - Sehr groß
    - > 100 Personen
    - > 45 Monate
    - > 20 Technologien
- Was ändert sich mit zunehmender Größe
  - Komplexität
  - Bedarf an Flexibilität
  - Bedarf an Organisation, Planung, Überblick
  - Bedarf an Prozessorientierung
  - Human Factors
  - Kommunikationspfade
  - Testaufwand
  - Saubere Dokumentation wichtiger (v.a. Schnittstellen)
  - Versionenverwaltung bzw. Konfigurationsmanagement
- Umfeld des SE-Prozesses
  - Personen
  - Projekt
  - Markt
  - Technologie
- Kauf von Software wenn
  - Anforderungen weitgehend erfüllt

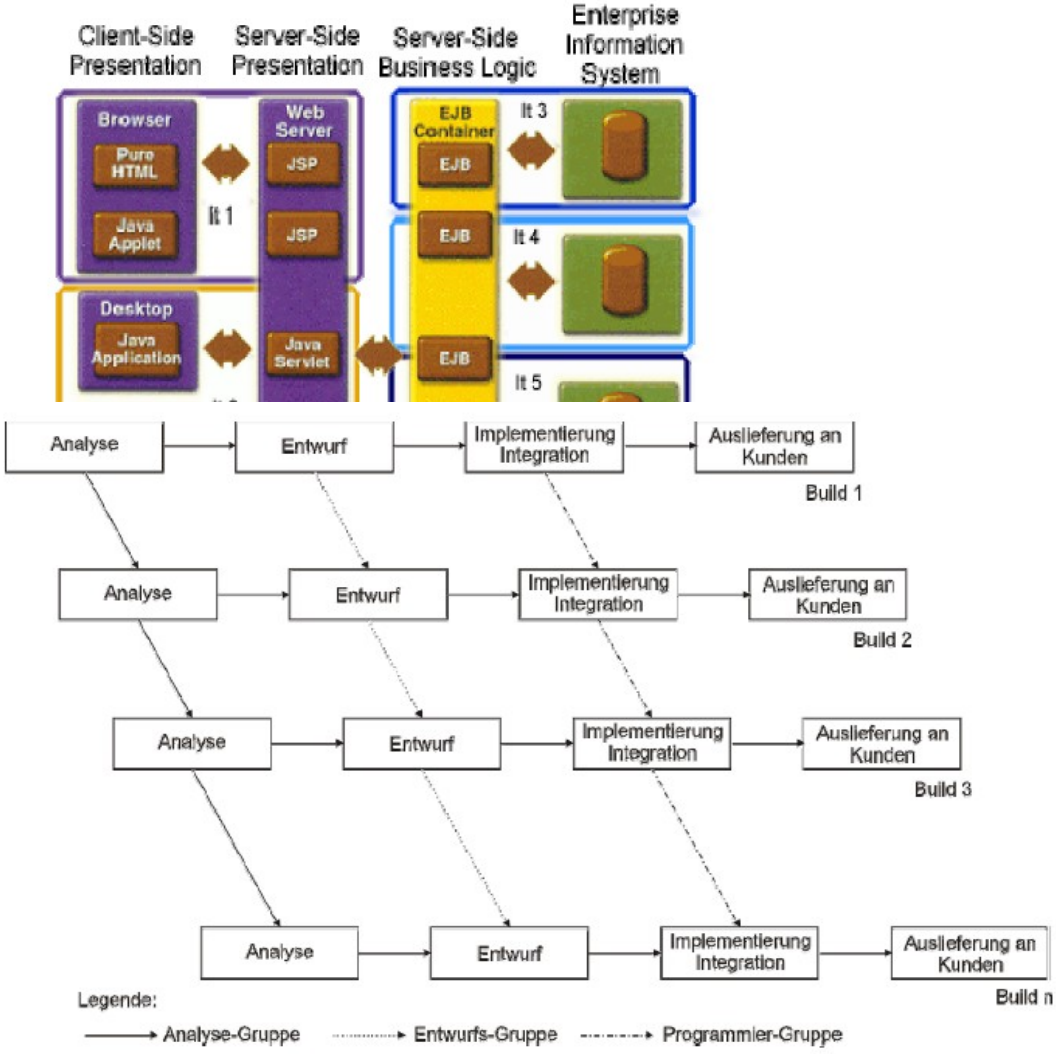
    - Anpassung möglich
    - Kosten geringer als Eigenentwicklung
    - Dokumentation vorhanden
    - Support bei Problemen
- Risikofaktoren (Faktoren für Scheitern des Projekts)
    - Unvollständige Anforderungen
    - Anwender nicht involviert
    - Zu wenig Resourcen
    - Unrealistische Zeit- & Kostenpläne
    - Keine Management Unterstützung
    - Häufige Änderung der Anforderungen
    - Qualitätsmängel bei extern vergebenen Komponenten/Aufgaben
    - Fehlende Planung
    - Projekt wird nicht mehr benötigt
- Faktoren für erfolgreiches Projekt
    - Einbringung & Berücksichtigung der Anwender
    - Adäquates Projektmanagement
    - Anforderungen eindeutig, realisierbar, überprüfbar
    - Flexibler, realistischer Projektplan (berücksichtigt mögliche Verzögerungen)
    - Realistische Kostenschätzung und Budget (inkl. Risikoanalyse)
    - Angemessene Ziele
    - Schlüsselteammitglieder haben genügend Erfahrung
    - Gute Teamarbeit, funktionierende Kommunikation

1-3 SW Prozesse und Produkte

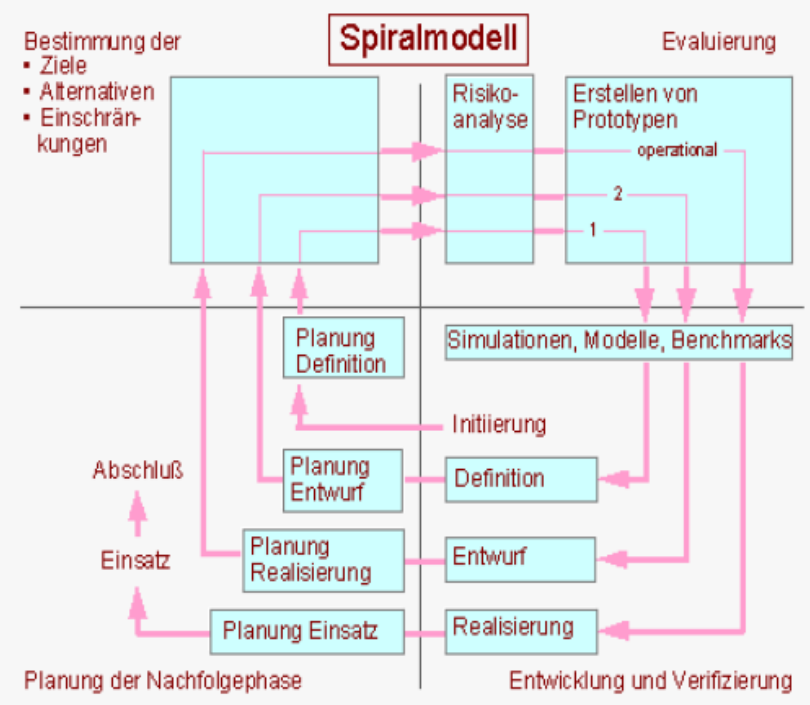
- Produktqualitäten
    - Anforderungen
    - Design
    - Testplan
    - Testbericht
    - Benutzerschnittstelle
    - Datenbank
    - Technische Dokumentation
    - Projektplan
- Verfolgen von Anforderungen
    - Validierung (z.B. mit Prototyp)
    - Anforderungen an Anforderungen
        - Gültigkeit
        - Vollständigkeit
        - Überprüfbarkeit
        - Verfolgbarkeit
        - Konsistenz
        - Realismus
        - Verständlichkeit
        - Anpassbarkeit
    - 4 Richtlinien
        - Value-driven requirements
        - Shared-vision driven requirements
        - Change-driven requirements
        - Risk-driven requirements
- V-Modell



- Implementierung mit Iterationen/Inkrementen (Inkrementelles Modell)



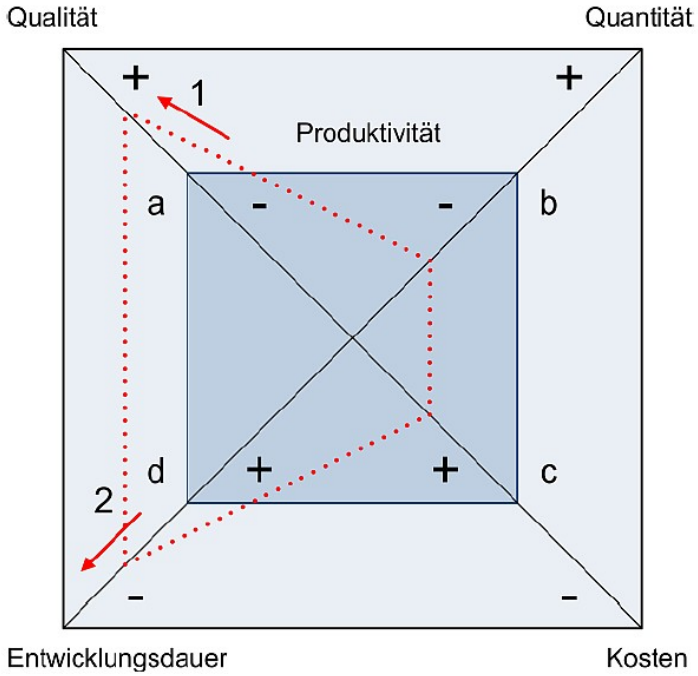
- Spiralmodell



Einführung in Projektmanagement

- Definition Projekt
  - einmaliges Vorhaben
  - definierter Anfang
  - definiertes Ende
  - mehrere beteiligte Personen
- Kennzeichen eines Projekts
  - Abgrenzbarkeit bezüglich Aufgabe, Ergebnis, Ressourceneinsatz, Zeitrahmen
  - Komplexität der Aufgabe
  - Einmaligkeit der Aufgabe
- Definition Projektmanagement
  - Gesamtheit von Methoden zur effizienteren Steuerung der Abwicklung von Projekten
  - Im engeren Sinn Projektleitung
- Warum tun wir uns mit Risikomanagement schwer?
  - Wir denken Zielorientiert
  - Risiko gefährdet Interessen von Stakeholdern
  - Wir tun uns mit Wahrscheinlichkeiten schwer
  - Am Ende des Problems nicht ein bisschen, sondern ganz oder gar nicht
- Ursachen
  - Mangelhafte Anforderungen
  - Mangelhafte Umsetzung der Anforderungen
  - Mängel im Projektmanagement
- Risikoanalyse
  - Spezifische Risiken aus Aufgabenstellung
  - Generelle Risiken (Personalausfall, ...)
  - Eintrittswahrscheinlichkeit
  - Schadenspotential
- Maßnahmen
  - Vorbeugungsmaßnahmen (reduzieren Risiko)
  - Abhilfemaßnahmen (lindern Folgen)
  - Kalkulation möglicher Mehrkosten

- Teufelsquadrat



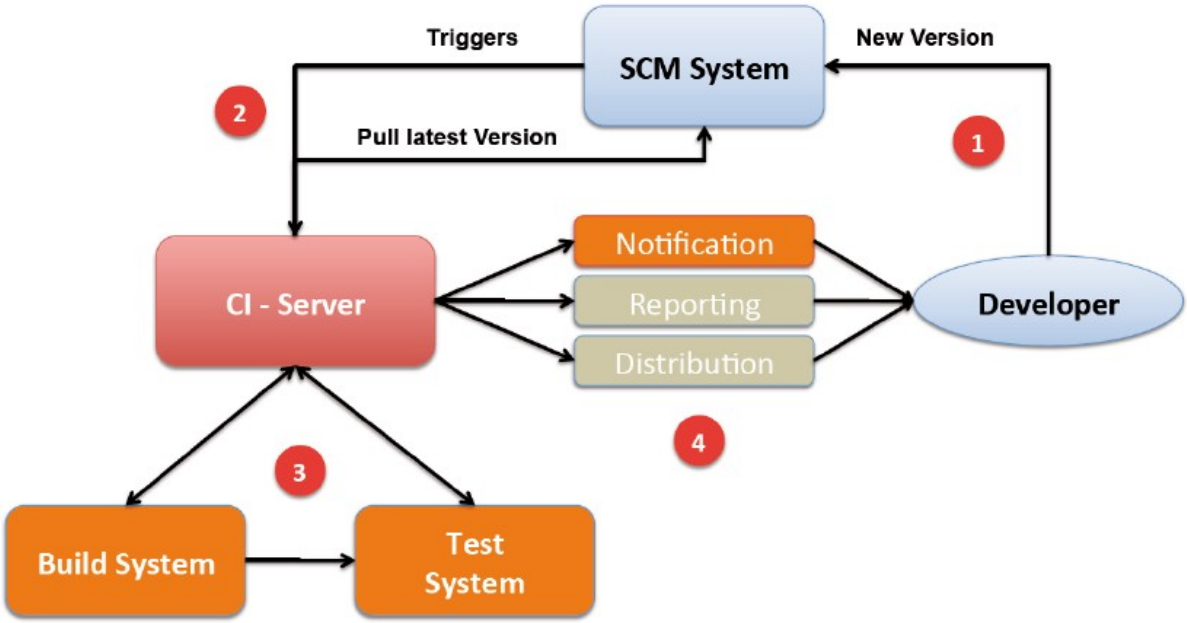
- Aufgaben im PM
  - Aufgabenstellung definieren
  - Projekt planen
  - Projekt(team) organisieren
  - Aufgaben verteilen
  - Fortschritt kontrollieren
  - Bei Bedarf steuernd eingreifen
  - Mit Risiko beschäftigen
  - Entscheidungen vorbereiten/fällen
  - Verwaltungskram
- Grundlagen Projektplanung
  - Vorgaben des Kunden (Aufgabenstellung, Termin-, Kostenrahmen)
  - Realisierungskonzept
  - Möglichkeiten des Projektteams (Resourcen, Know-How, Budget, Termindruck)
- Elemente des PM
  - Projektablauf – was ist zu tun?
  - Arbeitsstruktur – wer ist wofür zuständig?
  - Informationswesen – wer informiert wen/worüber/wann/wie?
  - Dabei zu beachten: Methodenunterstützung, psychologische Aspekte, Umfeldbedingungen
- Projektstart
  - Projektidee
  - Projektziel
  - Projekt-Antrag
  - Genehmigung
  - Projektauftrag
  - Kick-Off
- Vorarbeiten
  - Evaluierung, Vorstudien, Ersteinschätzung Resourcenanforderungen
  - Projektumfeldanalyse
    - Projekttyp

        - Funktionalitäten
        - Qualität
        - Projektunterstützende Faktoren
        - Mögliche Risiken
    - Wozu notwendig
        - Entscheiden, ob Projekt machen
        - Konkretisierung
        - Ziel ist Risikoreduktion
        - Forschung/Entwicklung
        - Vorarbeit kann eigenständiges Projekt sein
- Voraussetzungen für erfolgversprechendes Projekt
    - klare Prioritäten
    - Spielräume für Veränderungen
    - Planungszyklen werden kürzer --> geordnete & flexible Reaktion auf Änderungen
- Rollen im Projekt
    - Auftraggeber und -nehmer
    - Projektleiter
    - Entwicklerteam
        - Software Architekt
        - Dokumentenbeauftragter
        - Testbeauftragter
- Projektauftrag
    - schriftliche Zielvereinbarung zwischen Auftraggeber & Auftragnehmer, ein Projekt zu bestimmten Bedingungen durchzuführen (auch Projektantrag, -erklärung, -definition, -beschreibung)
    - Management-Unterlage – umfasst alle wichtigen Daten für Abwicklung
    - Voraussetzung: Ausarbeitung von Projektvorschlag
- Vorgehensweise
    - Projektstrukturplan
    - Arbeitsplanung & Arbeitspaketdefinition
    - Aufwandsschätzung
    - Ablauf & Terminplanung
    - Resourcenplanung
    - Kostenplanung
    - Optimierung
- Umsetzung
    - Sequenziell
        - Wasserfall
        - V-Modell
    - Iterativ
        - RUP (Rational Unified Process)
    - Agil
        - SCRUM
        - XP

Technik und Werkzeuge

- Sourcecode-Management (SCM) Systeme
    - Revision-Control-Systeme (RCS) – einzelne Dateien
    - Concurrent-Version System (CVS) – zentral, serverbasiert (zb SVN)
    - Verteilte SCM – jeder Entwickler eigenes Repo (zb Git)
- Terminologie
    - Check-out: Lokale Kopie anlegen

    - Check-in (Commit): changeset in Repo übernehmen
    - Tagging: Markieren von bestimmtem Zustand des Systems
    - Merging: Änderungen aus Branch in anderen übernehmen
- Maven
    - Lifecycle
        - validate
        - compile
        - test
        - package
        - integration-test
        - verify
        - install
        - deploy
- Continuous Integration Arbeitsablauf

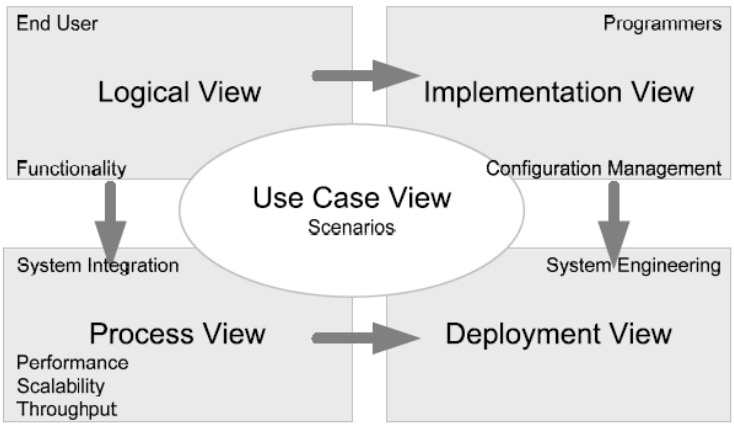


- Komponente
    - klare, stabile Schnittstelle
    - höhere Granularität als Klasse
    - Wiederverwendbarkeit
    - locker verbunden
- Service
    - klar definierte Schnittstelle
    - plattformunabhängig
    - über ein Netzwerk angeboten
- Framework
    - Wiederverwendbarkeit
    - Rahmenbedingungen für Komponenten (z.B. Persistenz-Framework)
- Inversion of Control (IoC) – Spring, Google Guice, etc
    - Abhängigkeiten von einem Container verwaltet
    - DI
    - Hollywood Prinzip "don't call us, we'll call you"
    - Vorteile
        - Wiederverwendbarkeit
        - Einfaches Austauschen von Implementierung

        - Verwalten von verschiedenen Konfigurationene (dev vs deploy)
        - Automatisiertes Verdrahten
        - Verteilung von Aufgaben

SE-Phasen

- Ziel: qualitativ hochwertiges Softwareprodukt
    - Minimale Anzahl Fehler
    - Hohe Kundenzufriedenheit
    - Anwendung Best-Practice Methoden
- Grundlegende Vorgehensweise
    - Systematische & strukturierte Vorgehensweise
    - Konstruktive Methoden
    - Analytische Methoden
- SW Life-Cycle
    - Anforderungen & Spezifikation
        - Aus Kundensicht (A) & Technischer Sicht (S)
    - Planning
        - Projektplan bezügl. Zeit, Dauer, Kosten
    - Entwurf & Design
        - technische Lösungen für Systemanforderungen
    - Implementierung & Integration
    - Betrieb & Wartung
    - Retirement
        - Am Ende kontrolliert aus Betrieb nehmen
    - (Projekt- und Qualitätsmanagement über alle Phasen)
- Anforderungen
    - Arten
        - Funktionale (was umgesetzt? Wie verhalten? Datenformate)
        - Nicht-funktionale (Performance, Usability, Sicherheit, Wartbarkeit, Erweiterbarkeit)
        - Designbedingungen (z.B. Schnittstellen, Plattformen, Entwicklungsumgebung)
        - Prozessbedingungen (Rahmenbedingungen, z.B. Dokumentation)
    - Priorisierung
        - must-be
        - expected
        - nice-to-have
    - Schritte
        - Elicitation
        - Analysis
        - Specification
        - Validation
        - --> Software Requirements Specification (SRS)
- Entwurf & Design
    - 4+1 View Model of Architecture



- Logical: Funktionale Anforderungen, End-User-Sicht
- Implementation: beschreibt Software Komponenten, Implementierung
- Process: Nicht-Funktionale Anforderungen, Systemintegration

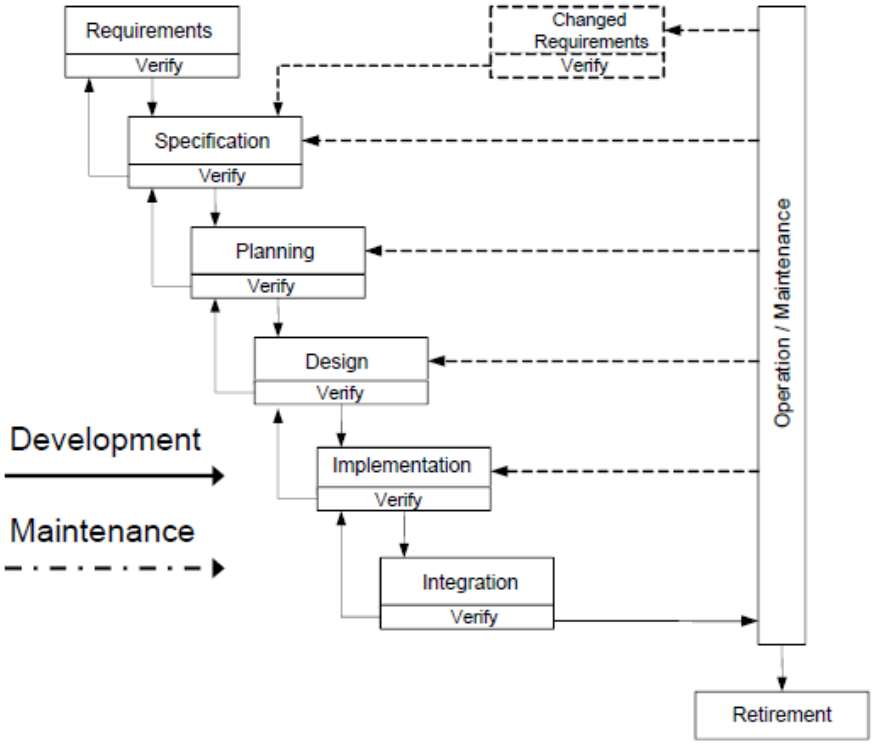
        - Deployment: Ausführbare Applikation, System Engineers
        - Use-Case: Ergänzung, gemeinsamer Nenner; Systemanalyse, Entwurf & Design; Architektursicht, Schlüsselszenarien
    - Design Prinzipien
        - Coupling & Cohesion (Kopplung & Bindung)
            - Kopplung: Abhängigkeit zwischen einzelnen Komponenten
            - Bindung: Innerer Zusammenhalt einer Komponente
                - niedrige Bindung = viel (auch ungenutzte) Funktionalität in Komponente
            - Ziel: Gleichgewicht
        - System Control
            - Wer übernimmt Kontrolle?
            - Stair
                - Verteilte Kontrolle
            - Fork
                - Zentrales Objekt kontrolliert gesamten Use Case
        - Abstraction
        - Decomposition und Modularisierung
        - Encapsulation / Information Hiding
            - zB private variables <> public properties
        - Trennung von Interface und Implementierung
- Implementierung und Integration
    - Standardisierung (intern vs. Extern)
        - Tipps für Entwicklung
            - Namenskonventionen
            - Formatierungsrichtlinien
            - Versionierungen
            - Headerblocks
        - Traceability
            - Nach- bzw Rückverfolgbarkeit von Information durch gesamten Entwicklungszyklus
            - Arten
                - Zeitlich (unterschiedliche Releases; zB durch Konfigurationsmanagement)
                - Vertikal (Beziehungen innerhalb einer Phase und Artefakttyps; zB System – Subsystem – Komponente)
                - Horizontal (Beziehung zwischen Entwicklungsartefakten; zB Anforderungen – Implementierung – Testfälle)
        - Interne Standards --> Unternehmensebene
        - Übergreifende Standards --> von IEEE, ISO, etc.
            - Kommunikationsmethoden
            - Programmiersprachen
            - Plattformen
            - Tools
    - System Integration
        - Big-Bang Integration
            - Alle Komponenten gleichzeitig integriert
            - Vorteil : Kein zusätzlicher Testaufwand für Test-Stubs
            - Nachteile: Fehler sind schwer zu lokalisieren, hohes Risiko bei Integration
            - Anwendung: kleine Projekte
        - Top-Down Integration
            - Schrittweise Integration, ausgehend von den Business Cases
            - Vorteile: früh ausführbares Produkt, Prototypen, Framework für Testfälle
            - Nachteile: Test stubs, Integration von Hardware erfolgt spät (Risiko)

        - Bottom-Up Integration
            - Schrittweise, beginnend bei Hardware
            - Vorteile: Stabiles System, Schrittweise Richtung Business Cases
            - Nachteile: Spät ausführbar, Aufwand für Prototypen, Aufwand für Test Drivers (testen von lower-level Komponenten)
        - Build Integration
            - Schrittweise entsprechend Business Cases, über unterschiedliche Layer
            - Vorteile: Frühe verfügbarkeit funktionaler Anforderung, Prototypen, Priorisierung möglich
            - Nachteile: Wiederverwendung schwierig, Regressions-Tests erforderlich
- QS & Testen
    - Testtechniken
        - Unit Test: Fokus auf einzelne Komponenten
        - Integrationstest: Fokus auf Interfaces & Vereinbarungen zwischen Komponenten
        - Systemtest: Fokus auf Übereinstimmung zwischen funktionalen & technischen Bedingungen dest Gesamtsystems
        - Regressionstest: Testen geänderter Komponenten
        - Akzeptanztest: Übereinstimmung Anforderungen und Zielumgebung
        - Installationstest: Identifizieren von Fehlern während Installation
    - Strategien
        - Black Box Tests (Data-driven)
        - White Box Tests (Logic-driven)
            - Äquivalenzklassen
                - Gleiches verhalten auf einer Menge von Eingabedaten mit selbem Ergebnis
            - Grenzwertanalyse
- Wartung
    - Sichten
        - Activity-View: Änderung des Systems nach Auslieferung und Inbetriebnahme
        - Process-View: Schritte zur Durchführung von Wartungsaufgabe
        - Phase-Oriented-View: Wartungsphase beginnt mit Auslieferung, endet mit Stilllegung
    - Kategorien
        - Reaktiv
            - Korrektiv
            - Adaptiv
        - Pro-Aktiv
            - Produktpflege und Verbesserung (Perfective)
            - Präventiv
    - Aufwand
        - Mehr als 80% für nicht-korrektive Tätigkeiten
        - Abhängig von:
            - Applikationstyp
            - Neuheit
            - Verfügbarkeit von Personal
            - Hardware
            - Qualität von Design, Konstruktion, Dokumentation, Tests
    - Techniken
        - Herstellen von Produktverständnis
        - Reengineering (gravierend und teuer)
        - Reverse Engineering (höheres Abstraktionsniveau)
- Retirement
    - Gründe für Stilllegung

        - Zahlreiche Änderungen während Wartung können Redesign verursachen
        - Laufzeitfehler durch Nebeneffekte
        - Inkonsistenzen durch Änderungen ohne Aktualisierung von Dokumentation
        - Hardwareänderungen

SW Prozesse

- Wasserfall Modell



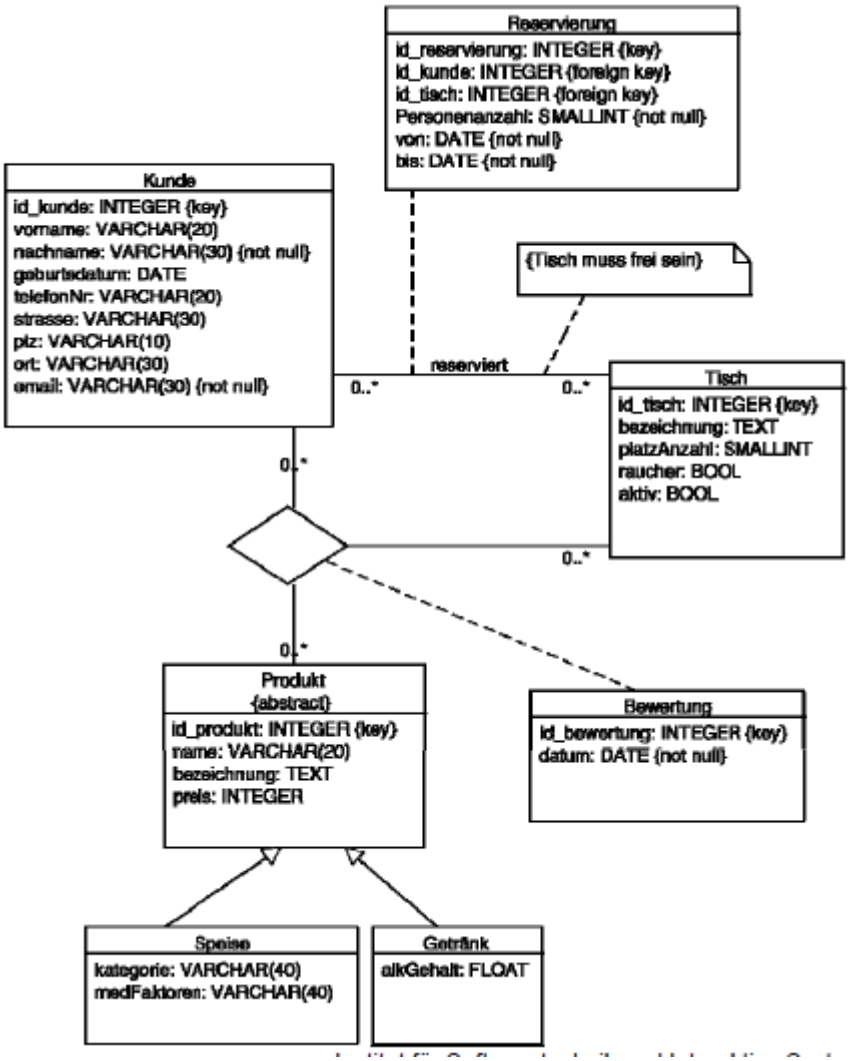
    - Vorteile
        - Backtracking zu früheren Phasen
        - Risikominimierung durch Abschluss einer Phase
        - Bekanntheit und Verbreitung
        - Trennung einzelner Phasen
        - Unterstützung von kleinen Teams
    - Nachteile
        - Alle Tasks einer Phase müssen abgeschlossen werden
        - Starke Auswirkung von Fehlern in früheren Phasen
    - Anwendung
        - No-Surprise Software (Kenntnis Anforderungsdomäne)
        - Klar definierte Anforderungen
- V-Modell
    - Vorteile
        - Spezifikationsphase vs. Realisierung und Testen
        - Kontext von Produkten und Tests
        - Verschiedene Abstraktionslevels
        - Fehlerbehandlung in frühen Phasen (durch Reviews)
    - Nachteile
        - Klare Beschreibung von Anforderungen wichtig
        - Hoher Dokumentationsaufwand
        - Kritisch bei unklaren Anforderungen
    - Anwendung
        - Große Projekte im öffentlichen Bereich

        - Klar definierte Anforderungen
- V-Modell XT
    - Ziel
        - Verbesserung der Unterstützung von Anpassbarkeit, Anwendbarkeit, Skalierbarkeit, Änderbarkeit, Erwiterbarkeit bei V-Modell
        - Berücksichtigung neuer Stand der Technik
        - Kompatibilität zu formalen Richtlinien
        - Erweiterung Anwendungsbereich
    - Komponenten
        - Projekttypen
            - eingeteilt nach: Projektgegenstand (HW, SW, etc) und Projektrollen (Auftraggeber/ -nehmerprojekt) --> 4 Typen
        - Vorgehensbausteine (kapseln Produkte, Aktivitäten, Rollen)
            - V-Modell Kern
            - Einführung und Pflege von organisationsspezifischem Vorgehensmodell
            - Auftraggeber / -nehmer Schnittstelle
            - Tool-Unterstützung
        - Unterstützung der Anpassbarkeit durch Tailoringmechanismen
        - Methoden- und Toolunterstützung
        - Entscheidungspunkte (~ Meilensteine)
            - An Entscheidungspunkten müssen definierte Produkte vorliegen
        - Projektdurchführungsstrategien
            - definierte Abfolge von Entscheidungspunkten (zB inkrementell oder agil)
    - Anwendung
        - flexibel durch Anpassung an Gegebenheiten
- Rational Unified Process (RUP)
    - Inkrementell und iterativ
    - Grundlegende Phasen
        - Inception – Beginn
        - Elaboration – Concept & Design
        - Construction
        - Transition – Auslieferung
    - 6 Engineering Workflows
    - 3 Supporting Workflows
    - Vorteile
        - Real-world Szenarien
        - Werkzeugunterstützung
        - Vordefinierte Liste mit erforderlichen Artefakten
    - Nachteile
        - Komplexität
        - Dokumentationsaufwand
        - Anbieterabhängigkeit
    - Anwendung
        - Große Projekte durch ganzheitliche Prozess-Sicht auf das gesamte Projekt
- Agile Ansätze
    - Individuals and interactions over processes and tools
    - Working software over comprehensive documentation
    - Customer collaboration over contract negotiation
    - Responding to change over following a plan
    - Principles
        - Satisfy customer through early and continuous delivery of valuable software
        - Welcome changing requirements

        - Deliver working software frequently
        - Business people and developers must work together daily
        - Build projects around motivated individuals
        - face-to-face conversation
        - working software is the primary measure of progress
        - agile processes promote sustainable development
        - continuous attention to technical excellence and good design
        - simplicity
        - self-organizing teams
        - reflect on how to become more effective regularly
    - SCRUM
- Process Tailoring / Customization
    - Tailoring
        - Anpassung an individuelle Gegebenheiten
        - Ersetzen einzelner Schritte durch alternative Lösungen
        - Wiederverwendung von Best-Practices
        - Anpassung Projektplan
        - Erfordert erfahrene Projektleiter
        - Berücksichtigung von Tailoring-Kriterien und Produktabhängigkeiten
    - Customization
        - Anpassung an Unternehmensstandards; Domänenabhängige Anpassung
        - Verallgemeinerung von Tailoring
        - Effizienzsteigerung von Tailoring
        - Komplatibilität zu zugrunde liegendem Prozess muss sichergestellt sein

Modellierung von Anwendungsszenarien

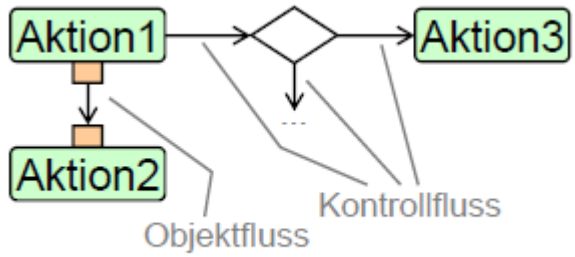
- Typische Fehler
    - Datenmodellierung, Reports
        - Schlüsselwerte nicht einduetig, fehlende Attribute
        - Fehler bei Beziehungen
    - Entwurf: Ergebniswerte von Methoden
        - Typkonversion falsch
    - Iniialisierung von Variablen
        - Fehlende Zuweisung
        - Keyword static falsch verwendet
    - Bedingungen
        - Logische Fehler, falsche Ausdrücke
    - Schleifen
        - Anzahl Durchläufe nicht korrekt
    - Testfälle
        - Unzureichende Überdeckung
        - Überflüssige Testfälle
- Test driven development
    - Testfälle anhand Interface erstellen
    - Assertions ableiten
    - Brauchbare Modellierung unterstützt effiziente Testfälle
- Datenmodellierung
    - Anwendung in Verifikation und Validierung:
        - Datenorientierte Testfälle (Black box)
        - SQLUnit
        - Unit Tests von DAO Pattern via Mocking
    - ER



- OOM
  - Aktivitätsdiagramm

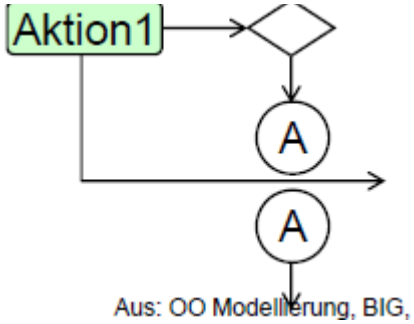
– Start und Ende von Abläufen

- Initialknoten
- Aktivitätsendknoten
- Ablaufendknoten



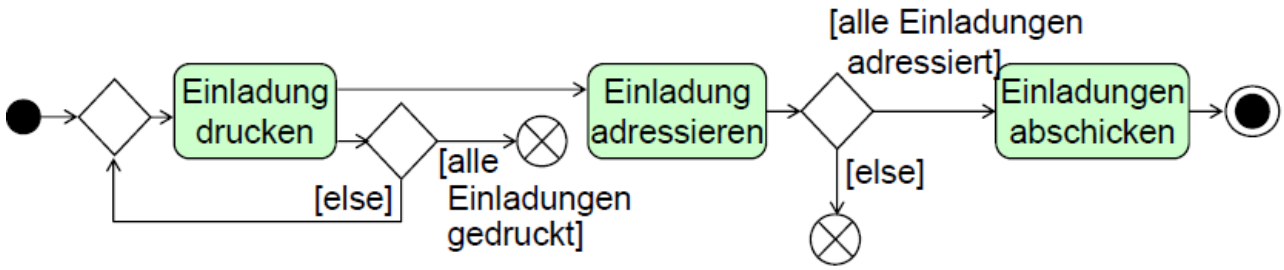
– Alternative Abläufe

- Entscheidungsknoten
- Vereinigungsknoten

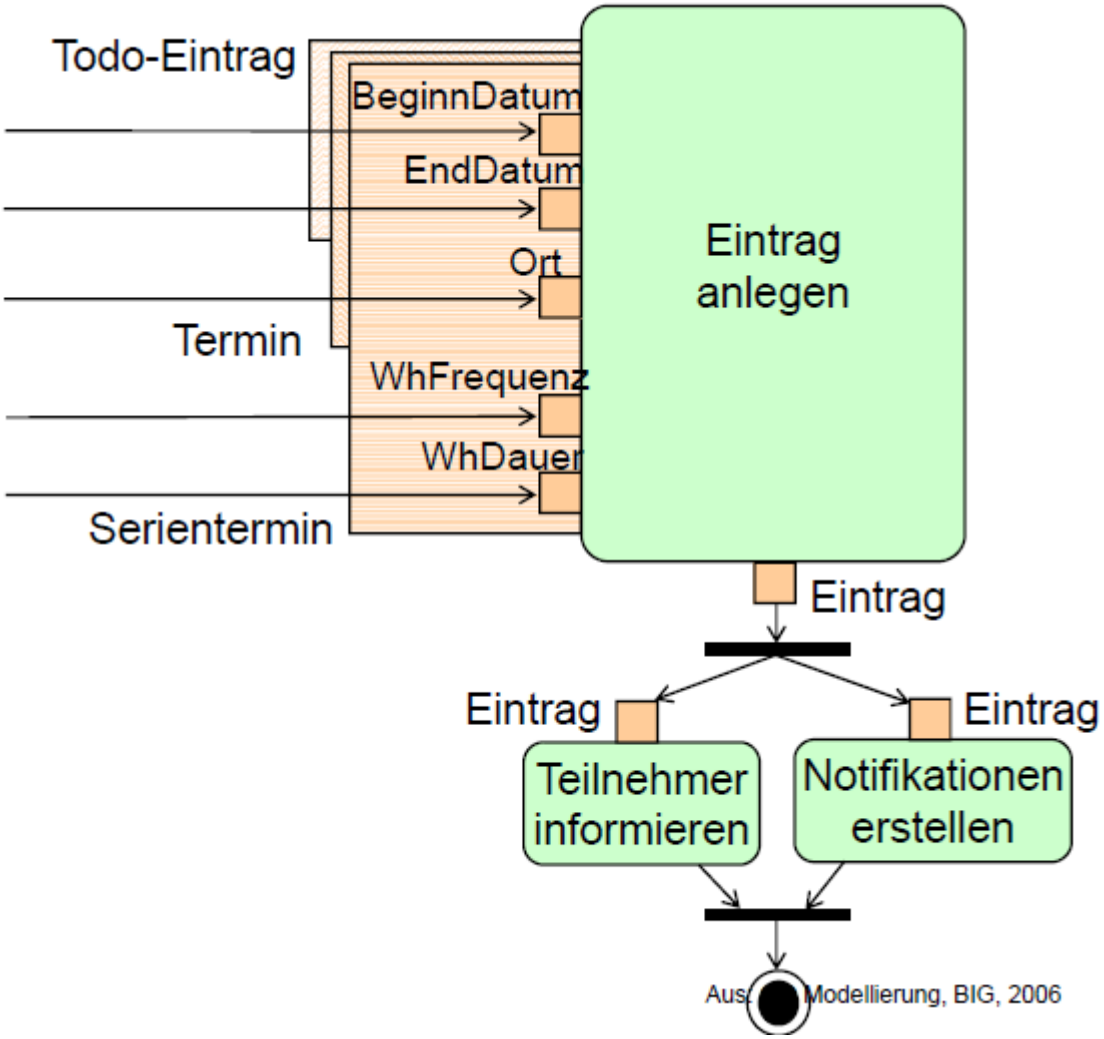


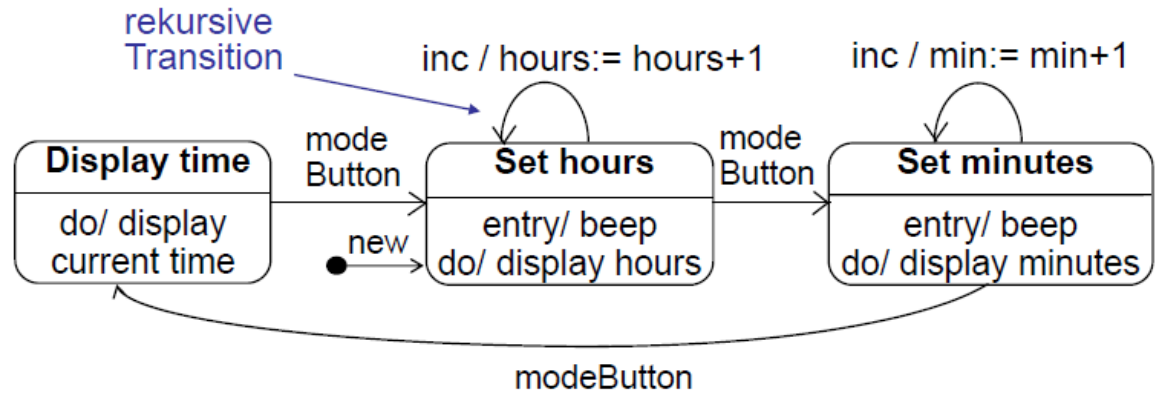
Aus: OO Modellierung, BIG,

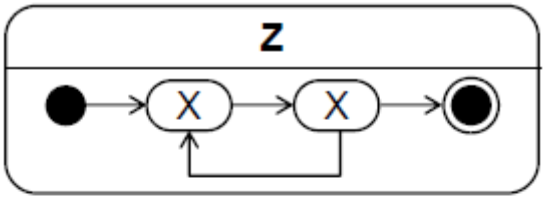
Überwachungsbedingung (Guard) \/

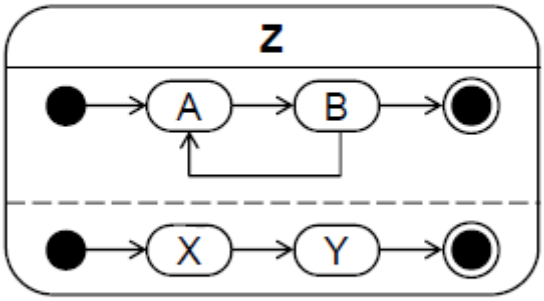


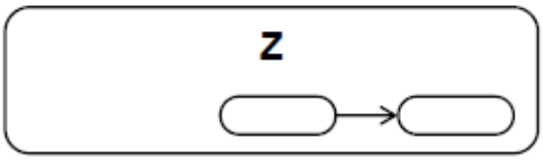
- Zustandsdiagramm











Qualitätssicherung in SEPM

- Qualität = Eignung zur Erfüllung vordefinierter Anforderungen
  - Faktoren zB Korrektheit, Efffizienz, Verwendbarkeit, Testbarkeit, Wartbarkeit
  - keine Eigenschaft, die später hinzugefügt werden kann
  - muss während Entwicklung gesichert werden
- Qualitativ hochwertige SW-Produkte sind
  - termingerecht und im Rahmen des Budgets
  - verwendbar (Anwender)
  - verständlich und änderbar (Professionist)
  - effizient und administrierbar (Betreiber)
- QS = Durchführung von Verifikation & Validierung in jeder Phase der SW-Herstellung

Statische Methoden der QS

- Reviews
  - vor allem für qualitative Beurteilung von Produkten und Prozessen, die schwer quantitativ beurteilt werden können
  - Beurteilung des Produktes, nicht des Autors
  - Definierte Rollen mit zugeordneten Aufgaben
  - Arten
    - Software Requirements Review

        - Preliminary Design Review
        - Critical Design Review
        - In-Process Review
        - Management Review
        - (Fagan) Inspection
            - Behebung von Mängeln (nicht durch Autor)
        - Code Walkthrough
            - Behebung von Mängeln (durch Autor)
        - Technical Review
        - Audit
            - Überblickende Kontrolle (durch externe Person)
    - Rollen
        - Moderator (Leiter)
        - Leser
        - Gutachter (Kommentierung)
        - Schreiber (Protokoll)
        - Autor (keine Kommentierung und Rechtfertigung)
    - Ablauf
        - Planung: Ziele, Kriterien festlegen
        - Vorbesprechung: Vorstellung Prüfobjekt
        - Intensive Einzeldurcharbeitung
        - Durchführung: Gemeinsam, Aufzeigen von Mängeln --> Mängel entdecken, nicht beheben
        - Nachbearbeitung: Mängel korrigieren
        - Bewertung: Überprüfen
        - Wiederholung möglich
        - Checklisten unterstützen Reviews
    - Richtlinien
        - Produkt – nicht Autor
        - Arbeitsplan
        - Diskussionen sachlich und kurz
        - Problembereiche identifizieren – nicht alles gleich lösen
        - Protokoll
        - Teilnehmerzahl begrenzen – gute Vorbereitung
        - Für jedes Produkt passende Checkliste
        - Ausreichend Resourcen
        - Vorab Training für Reviewer
        - Im nachhinein beurteilen

Testansätze, TDD

- Value based testing
    - Requirements-Based: welche Anforderungen bringen den meisten Nutzen
    - Risk-Based: Welche Risiken gefährden Nutzen
    - Test case selection techniques: welche Testfälle addressieren dieses Risiko
- TDD Prozess
    - Think: Spezifikation des Tests
    - Red: Implementierung des Tests – Test schlägt fehl
    - Implementierung der zu testenden Klasse/Komponente – Test ist erfolgreich
    - Refactor: Änderung an Implementierung – Test sollte nie fehlschlagen

Software Patterns

- Managing Complexity

    - Abstraction
    - Decoupling
    - Decomposition
    - Classification
    - Standardization
    - Modeling
    - Transformation
    - Experience
- Classification of Patterns
    - Architectural (structure of systems, subsystems, dependencies, communication)
    - Design (structure and relations at level of classes)
    - Idioms (low-level details, language specific)
    - Protopatterns (particular case)
    - Antipatterns (commonly used but ineffective)
- Types of patterns
    - fundamental (deal with essential concepts of architecture)
    - creational (deal with initializing and configuring classes and objects)
    - structural (deal with decoupling interface and implementation)
    - behavioral (deal with dynamic interactions among objects)
- Fundamental Patterns
    - Interface
        - seperation of interface and implementation
    - Delegation
        - Extension of functionality without inheritance
        - outsource functionality into third class and use its instance via delegation
    - Immutable
        - Provides unchangeable object after initialization
        - Initialize variables in constructor, provide read only access via getters
- Creational
    - Singleton
        - only single instance
    - Factory
        - Method in a derived class creates associates
    - Abstract Factory
        - Factory for building related objects without specifying their concrete classes
- Structural
    - Facade
        - simplifies the interface for a subsystem
    - Adapter
        - Translator adapts a server interface for a client
        - Wrapper
    - Proxy
        - one object approximates another
        - extends concept of delegation pattern
- Behavioral
    - Observer
        - Dependents update automatically when a subject changes
    - Decorator
        - extends an object transperently
    - State
        - Object whose behavior depends on its state
    - Strategy

        - Vary algorithms independently
    - Chain of Responsibility
        - Request delegated to the responsible service provider

Motivation

- extrinsisch (durch äußere Zwänge)
- intrinsisch (durch Anreiz der Aufgabe)
- Prozesstheorien
    - Individuelle Einflussfakturen des Motivationsprozesses
        - Anspruchsniveau
            - Erwartung an eigene Leistung und Belohnung
            - Häufiges Erreichen des Ziels --> Niveau angehoben
        - Persönliche Einstellung
            - Grundhaltung gegenüber Organisation, Problemstellung, Umfeld
            - Negative Einstellungen --> Frustration
        - Subjektive Wahrscheinlichkeit
            - Abschätzung von Chancen, Erfolg zu haben
            - Zusammenhang mit Anspruchsniveau --> eigener Qualitsmaßstab muss erreicht werden können
    - Folgerungen
        - Mitarbeiter als Individuum behandeln
        - Mitarbeiter nicht 1 zu 1 austauschbar
        - Tagesverfassung erkennen und reagieren
        - Mitarbeiter lernen aus Erfahrungen und richten zukünftiges Verhalten danach
- Inhaltstheorien
    - Monothematische Theorien (keine entscheidende Bedeutung)
        - Ein einzelnes Motiv ausschlaggebend für zielgerichtete Dynamik (bsp. Sexualtrieb oder Minderwertigkeit)
    - Polythematische Theorien
        - Mehrere Grundmotive – verschiedene Triebe in unterschiedlicher Rangordnung und Stärke
    - Motivationspyramide nach Maslow
        - Physiologische Bedürfnisse (Schlaf, Nahrung)
        - Sicherheitsbedürfnisse (Schutz des Lebens, Eigentums, Arbeitsplatz, Unfallfolgen)
        - Soziale Bedürfnisse (Eingliederung in Gruppe, Akzeptanz, Freundschaft)
        - Achtung (Selbstachtung und Fremdachtung; Kompetenz, Status, Respekt)
        - Selbstverwirklichung (potentiell gegebene Fähigkeiten entfalten)

Produktionsfaktor Mensch

- Produktivität = Verhältnis erbrachter Leistung zu aufgewendeter Zeit
- Häufige Maßnahmen zur Produktivitätssteigerung:
    - Personal zu mehr Arbeit anhalten
    - Automatisieren
    - Qualitätsmaßstäbe reduzieren
    - Vorgehensweise standardisieren
- langfristig kontraproduktiv:
    - verringern Arbeitszufriedenheit
    - erhöhte Fluktuation
- Fluktuation
    - typischerweise 33-80%
    - Kosten fallen an
- Faktoren ohne Einfluss auf Produktivität

    - Programmiersprache
    - Berufserfahrung
    - Anzahl Fehler
    - Gehalt (schwach)
- Faktoren mit Einfluss auf Produktivität
    - Teampartner – Personen aus gleichem Unternehmen bringen fast gleiche Leistung
    - Qualität des Arbeitsplatzes
- Ruhe am Arbeitsplatz (U-Faktur)
    - ca. 15 Minuten ungestörte Konzentration, um guten Zustand zu erreichen
    - U = ungestörte Stunden / anwesende Stunden
- Arbeitsplatzgestaltung
    - Platz
    - Privatsphäre
    - Beleuchtung
    - Außenraum
    - Innenraum
    - Kommunikationsräume
    - Individualität
    - Flexibilität

Team Management

- Wege, Teambildung zu verhindern
    - Defensives Management – Teammitglieder entmündigt
    - Bürokratie – sinnloses Produzieren von Projektdokumenten
    - Physikalische Trennung
    - Zersplitterung der Zeit – Aufteilung von Mitarbeitern auf mehrere Projekte
    - Qualitätsreduktion der Produkte – geringere Identifikation mit Projekt resultiert
    - Scheintermine – unrealistische Termine --> geringere Identifikation
    - Cliquenkontrolle – Veränderung von Struktur während Projekt & Zerschlagung des Teams nach Projektende
- Formen der Team-Organisation
    - klassisch hierarschisch
    - Matrix-Organisation
    - Chef-Programmierer-Team
    - offen strukturiert
    - SWAT-Team
    - XP-Team
- Richtlinien zur Teamorganisation
    - bessere Leute
    - weniger Leute
    - Aufgaben, die zu Fähigkeiten passen
    - Aufgaben, die zu Interessen passen
    - auf persönliche Entwicklung achten
    - balancierte und harmonische Mischung
    - von Leuten lösen, die nicht zum Team passen