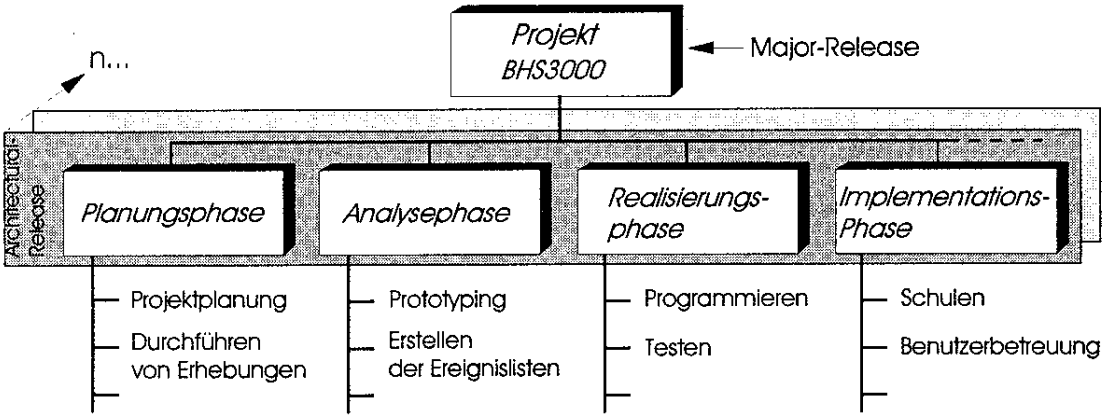
- Was ist ein Projekt und durch welche Merkmale ist ein Projekt gekennzeichnet?
  - zeitlich begrenztes Vorhaben
  - klare Ziele
  - einmalig
  - komplex mit verschiedenen Methoden/Techniken
  - erfordert oft interdisziplinäre Zusammenarbeit
  - besondere Risiken
  - beschränktes Bugdet
  - erzeugt Druck auf Beteiligte

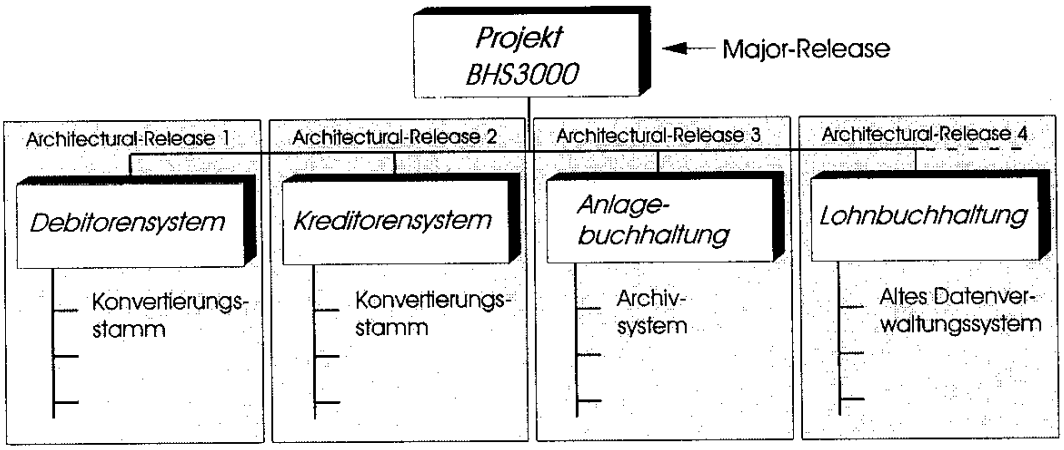
- Was versteht man unter Projekterfolg? Durch welche Rahmenbedingungen wird der Projekterfolg gefördert? Geben Sie für jede Rahmenbedingung zumindest ein konkretes Beispiel an.
  - Projekterfolg wird gleichgesetzt mit Erreichen des vorgegebenen Zieles, ohne den vorgegebenen Zeit- und Kostenrahmen zu überschreiten.
  - Rahmenbedingungen
    - entwicklungsbezogen (Vorhandensein von benötigten Tools [IDEs, Editoren, ...]
    - projektbezogen (Entwicklungszeit, Arbeitsaufteilung)
    - firmenbezogen (Unternehmensstrategien, Wirtschaftlichkeitsvorgaben)
    - personal- und anwendungsbezogen (Sozialverhalten, Ausbildungsgrad)
    - produktbezogen (Komplexität, Modularität)

- Was verstehen Sie unnter einem Projektstrukturplan (Work Breakdown Structure)? Aus welchen Ebenen besteht eine WBS?
  - Ebenen
    1. Projektname/Projektbezeichnung
    2. Unter- bzw. Teilsysteme (vgl. Architectural Releases)
    3. einzelne Arbeitspakete

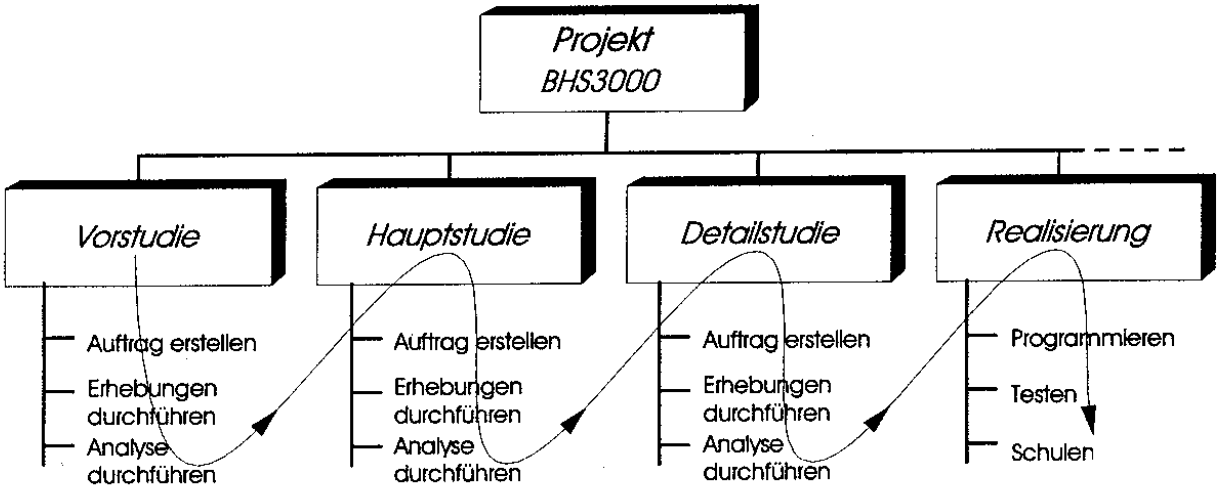
- Skizzieren und erläutern Sie einen funktions- bzw. aufgabenorientierten Projektstrukturplan und einen objektorientierten Projektstrukturplan. Worin liegen die wesentlichen Unterschiede und wofür werden sie eingesetzt?
  - funktions- bzw. aufgabenorientiert:



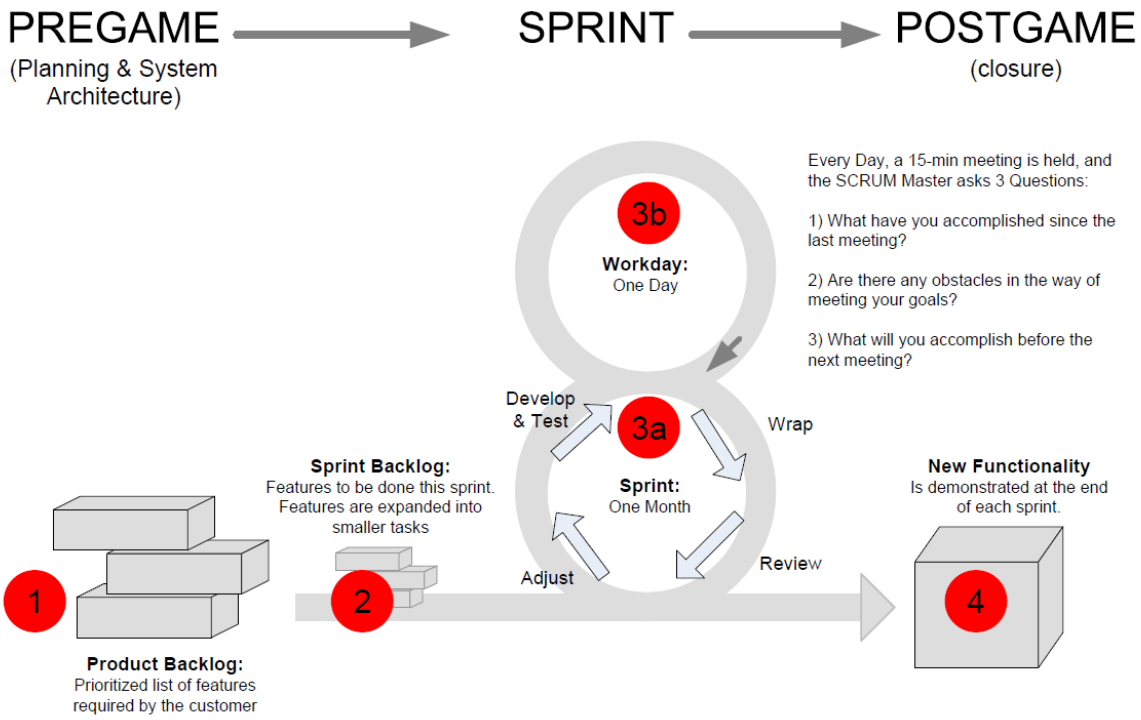
  - objektorientiert:



- ablauforientiert:



- Erläutern Sie anhand einer Skizze das Grundkonzept von SCRUB Sprints.



- Erklären Sie den Unterschied zwischen Produkt- und Sprint-Backlog.
  - Produkt-Backlog beinhaltet alle zu erledigenden Features, der Sprint-Backlog nur die, die im aktuellen Sprint zu erledigen sind.

- Was versteht man unter einem Burn-Down-Chart?
  - Eine viuselle Darstellung des Projektfortschritts. Einfach ein Graph, der (idealerweise gegen Null geht; x-Achse = Zeit, y-Achse = # an Tasks, die zu erledigen sind; eventuell Gegenüberstellung geplant/real)

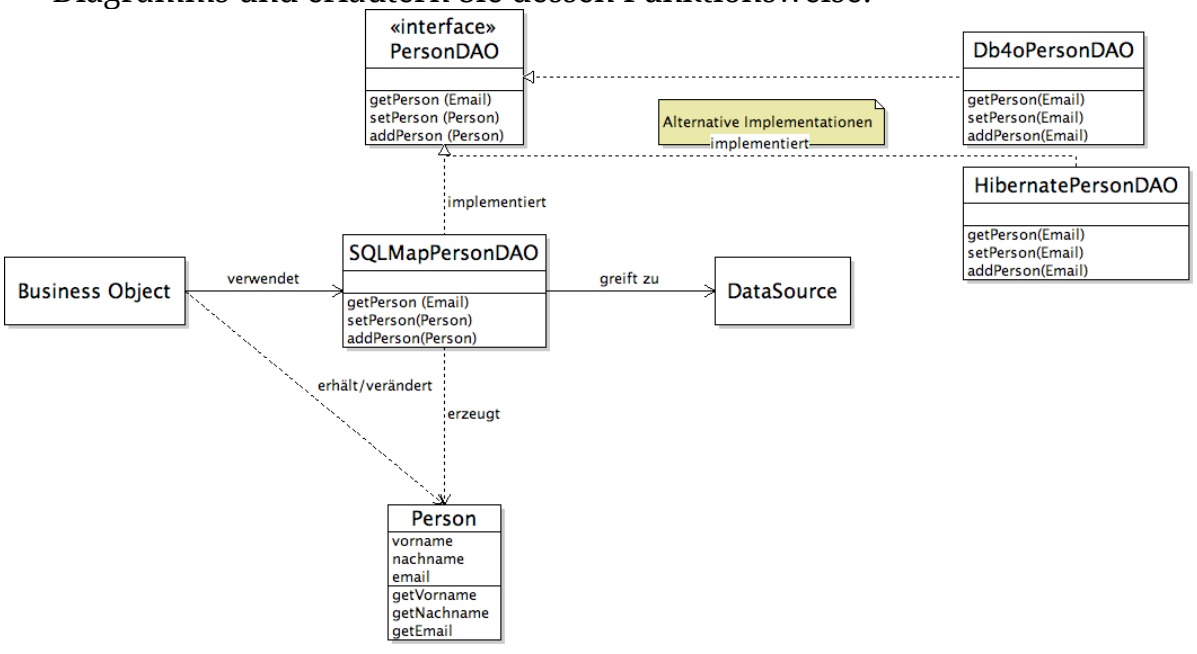
- Vervollständigen Sie die Tabelle um die Wartungskategorien, geben Sie jeweils ein Konkretes Beispiel an.

	Correction	**Enhancement**
Proactive	Preventive	Perfective
Reactuve	Corrective	Adaptive

- Welche Phasen umfasst der Wartungsprozess?
  - Beginnt mit Inbetriebnahme, dann Wartung (siehe Kategorien), endet mit kontrollierter Außerbetriebnahme bzw. Ersetzung durch anderes System. Wartung → Evolution → Retirement

- Nennen + beschreiben Sie zwei Techniken zur Wartung von Software.
  - Produktverständnis herstellen (fremden Code lesen + verstehen, sehr teuer/aufwändig, nur bei guter Dokumentation)
  - Reengineering (Überarbeiten/Überprüfen von Code, potentiell teuer/aufwändig)
  - Reverse Engineering (Analyse eines fertigen Systemes, zB UML aus Code generieren lassen)

- Beschreiben Sie das Data Access Object Patter (DAO) anhand eines UML-Diagramms und erläutern Sie dessen Funktionsweise.



  -

- Wann kommt dieses Pattern typischerweise zum Einsatz?
  - Trennung von Interface/Implementation (mehrere Implementationen möglich, Dependency Injection Systeme (Spring)

- Was verstehen Sie unter Traceability im Zusammenhang mit Softwareprojekten?
  - Informationen bzw. Entscheidungen und Anforderungen müssen zeitlich und hierarchisch durch den Lebenszyklus des Projektes zurückverfolgbar sein.

- Beschreiben Sie kurz die unteschiedlichen Arten von Traceability und welche Vorteile sich in einem Softwareprojekt daraus ergeben. Wie kann Traceability in einem Softwareprojekt umgesetzt werden?
  - zeitliche T. (Nachvollziehbarkeit über Releases/Milestones hinweg)
  - vertikale T. (Beziehungen innerhalb eines Artefakttyps (System → Subsystem → Komponente))
  - horizontale T. (Beziehungen zwischen unterschiedlichen Artefakten)
  - Umsetzung in Software etwa durch Informationen in Headerblocks (z.B. automatische Verarbeitung in IDEs)

- Erläutern Sie den Begriff der Systemintegration.
  - Integration beschreibt den Prozess, in dem verschiedene Artefakte/Module zu einem größeren Konstrukt zusammengefügt werden.

- Skizzieren und erläutern Sie die vorgestellten Integrationsstrategien und welche Vor- und Nachteile damit verbunden sind.
  - Bing Bang: Alles auf einmal. Auftretende Fehler sind schwer auffindbar.
  - Top Down: Mocking der unteren Schichten, Vorteil: Lauffähiges Gesamtsystem schnell verfügbar; Nachteil: Aufwand für Mocking.
  - Bottom Up: Integration von unten nach oben, Vorteil: kein Aufwand für Mocking; Nachteil: Gesamtsystem erst spät lauffähig.
  - Build Integration: Use-Case-basierte Integration, Vorteil: frühe Verfügbarkeit von priorisierten Funktionen, Nachteil: Regressionstests erforderlich, um Fehler durch neue Funktionen zu finden.

- Was ist bei der Systemintegration im Hinblick auf die Qualitätssicherung zu beachten?
  - Ausreichendes Testen? v.a. Regressionsstests bei Build Integration

- Was sind Software Reviews und wozu werden sie typischerweise eingesetzt?
  - Reviews überprüfen die Korrektheit und Konsistenz von Artefakten an wohldefinierten Punkten des Entwicklungsprozesses.

- Welche Arten von Reviews kennen Sie? Erläutern Sie die wesentlichen Aspekte der einzelnen Arten und geben Sie jeweils ein Beispiel an.
  - Inspektion (Code nicht von Autor gelesen)
  - Walkthrough (Code wird vom Autor gelesen/erklärt)
  - Audit (externer Review, Autor wird später informiert)

- Welche Rollen finden sich typischerweise bei Reviews und welche Aufgaben nehmen die unterschiedlichen Rollen wahr?
  - Moderator (leitet Review)
  - Leser (liest den Code)
  - Gutachter (kommentiert den Code)
  - Schreiber (schreibt Protokoll)
  - Autor (klärt offene Fragen, keine Rechtfertigungen/Kommentare)

- Was versteht man unter Test Driven Development (TDD)?
  - Tests werden zuerst anhand der Spezifikation/Interfaces erstellt. Konkrete Implementation erst danach, Tests sollten nie wieder fehlschlagen.
  - Unit-, Modul-, Integrations- und Systemtests
  - Blackbox: I/O-Test (v.a. Unittest), Whitebox: logische Analyse (v.a. Integrationstest)

- Diskutieren Sie die Begriffe Verfikation und Validierung.
  - Verfikation = Überprüfen, ob Produkt technisch richtig entwickelt
  - Validierung = Überprüfen, ob Produkt nach Anforderungen erstellt

- Äquivalenzklassenzerlegung
  - Beispiel: Inputanforderung: $6 < x < 9 \rightarrow$ 3 Äquiv-Klassen
    - Testen mit einem Wert <= 6, einem gültigen und einem >= 9.

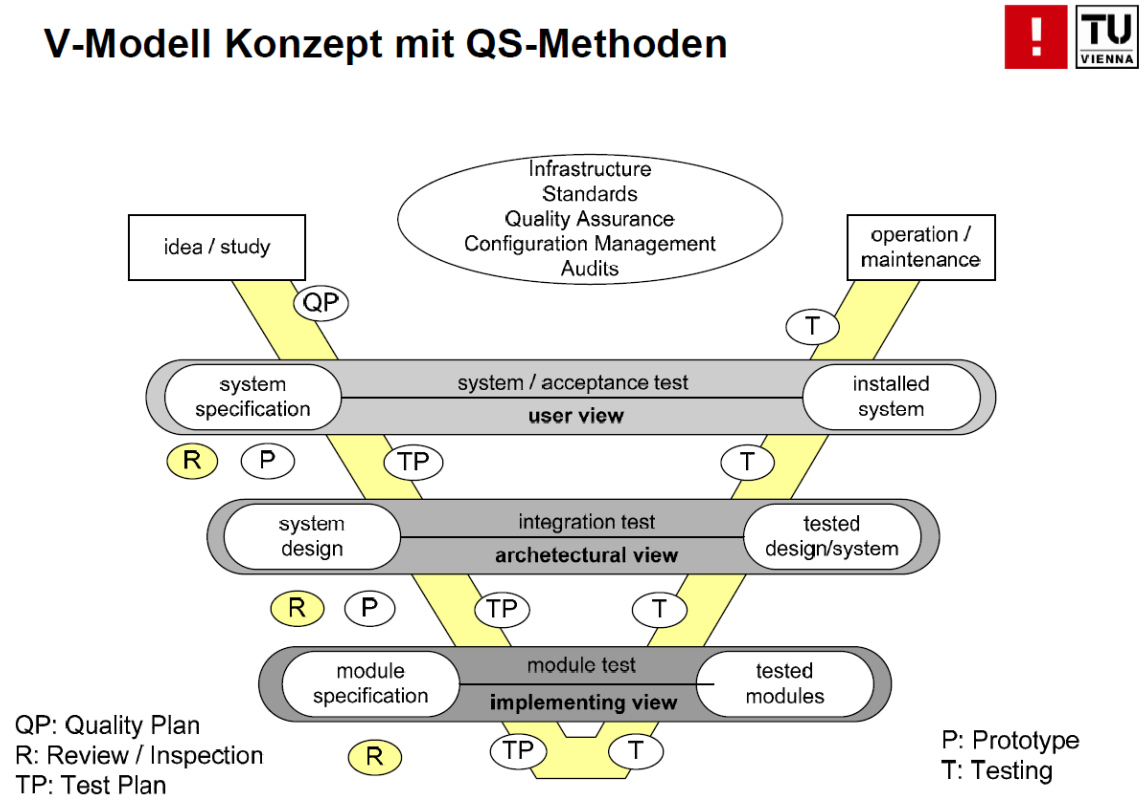
- Grenzwertanalyse
  - genaues Testen von Grenzfällen bei Parameterwerten

- Aus welchen Komponenten besteht eine Testfalldokumentation? Beschreiben Sie die notwendigen Komponenten und geben Sie ein konrektes Beispiel an.

Nr.	Typ	Beschreibung	Vorbedingungen	Eingabewerte	Aktionen	Erwartete Ergebnisse	Ergebnisse	OK / NOK
34	NF	Bestellung von Menü durchführen	(Programm gestartet) (in die Bestellungs-übersicht gewechselt) Min. ein Menü mit preis<250 Euro muss vorhanden sein. Menü muss bestellbar sein.	-Bestellung. take_away -Bestellung. fürDatum -Menu.id -Menü Anzahl = 1	-Neue Bestellung erzeugen. Angabe wann die Bestellung konsumiert wird und ob "take away". -Menü wird aus einer Liste ausgewählt -Anzahl angeben -Bestellung speichern	Bestellung ist in der Datenbank gespeichert und enthält genau ein Menü. Das Attribut Bestellung.storniert ist auf den Booleanwert ‚false' gesetzt, der Primärschlüssel von Bestellung wird von der DB automatisch belegt.	Bestellung ist in der DB gespeichert, jedoch wird bei der Abfrage dessen kein Primärschlüssel mitgeliefert.	NOK

- Welche 4 Kernformen der Arbeitsstruktur (Projekt/Teamorganisation) wurden in der Vorlesung besprochen?
  - klassische hierarchische Organisation
  - typische Matrix-Organisation
  - Chef-Programmierer-Team
  - offen strukturierte Team
  - das SWAT-Team
  - das XP-Team

- V-Modell: Skizze und grundlegende Funktionsweise
  - Verwendung bei öffentlichen Projekten mit hohem Qualitätsanspruch



  - Nachteil: Hoher Dokumentationsaufwand, klare Anforderungen sehr wichtig, kritisch bei unklaren/veränderten Anforderungen
  - Vorteile: Abstraktionssichten (user/archetectural/implementing), Kontext von Tests und Produkten, ständige Tests/Reviews – dadurch frühe Erkennung von Fehlern

- Erläutern Sie das wesentliche Konzept von *Process Customizing* und des *Process Tailoring*. Wozu werden diese Konzepte benötigt, wann werden sie eingesetzt?
  - Benötigt, wenn das Prozessmodell nicht 100%ig auf die Anforderungen passt. Für Customizing/Tailoring ist ein erfahrener Projektleiter notwendig
  - Customizing: Anpassung von Prozessmodellen an firmeninterne Anforderungen
  - Tailoring: Anpassung an konkrete Projekte

- Welche Rolle spielen Requirements im SE Life-Cycle? Wie können Requirements klassifiziert werden?
  - Funktionale (Use-Cases) und Nichtfunktionale (keine direkte Funktion, beeinflusst diese aber; z.B. Performance, Plattformunabhängigkeit, Wartbarkeit, ...) Anforderungen
  - Requirements sollten möglist genau und vollständig spezifiziert werden. Agile Modelle sind auf sich verändernde Anforderungen ausgelegt. Bei V-Modell zB kann das kritisch werden.

- Wie sieht eine moderne 3-tier Architektur aus? Beschreiben Sie Thin-Client/Fat-Client und skizzieren Sie beide Architekturvarianten.
  - 3-tier = Data/Logic/Presentation