

**Hochschule Osnabrück**  
University of Applied Sciences

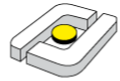


# Softwareprojekt

Prof. Dr. Klaus Dallmüller

WS 2016/17

# Warum ausgerechnet diese Veranstaltung?



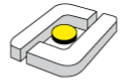
Hochschule Osnabrück  
University of Applied Sciences

- In den letzten Semestern haben Sie viel gelernt:

- (Objektorientierte) Programmierung
- Datenbanken
- Objektorientierte Analyse und Design
- Projektmanagement
- Verteilte Systeme

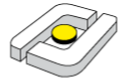
*Nun ist es Zeit dieses Wissen zu vernetzen – und anzuwenden*

- Softwareentwicklung in der Praxis ist ganz überwiegend Projektarbeit
- Nicht nur Softwareentwicklung ist Projektarbeit – auch die Einführung eines neuen Produkts, die Veränderung eines Geschäftsprozesses, die Fusion von Unternehmen – all dies und mehr wird als „Projekt“ abgewickelt.



**Die Bearbeitung eines Softwareentwicklungsprojektes in einem Team  
ist eine zentrale Aufgabe eines  
Informationsmanagers / Wirtschaftsinformatikers**

# Diese Veranstaltung ist keine normale Vorlesung



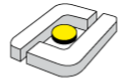
Hochschule Osnabrück  
University of Applied Sciences

## Organisation

### ■ Stattdessen

- betreute Projektarbeit in Gruppen (Gruppen mit jeweils 4 Teilnehmern)
  - jeder Teilnehmer übernimmt eine bestimmte Rolle (PL, FCD, TCD, QB).
  - Lehrender übernimmt 2 Rollen: Kunde und Coach
- plus regelmäßige Zwischenpräsentationen der Projektgruppen
- plus Abschlusspräsentation

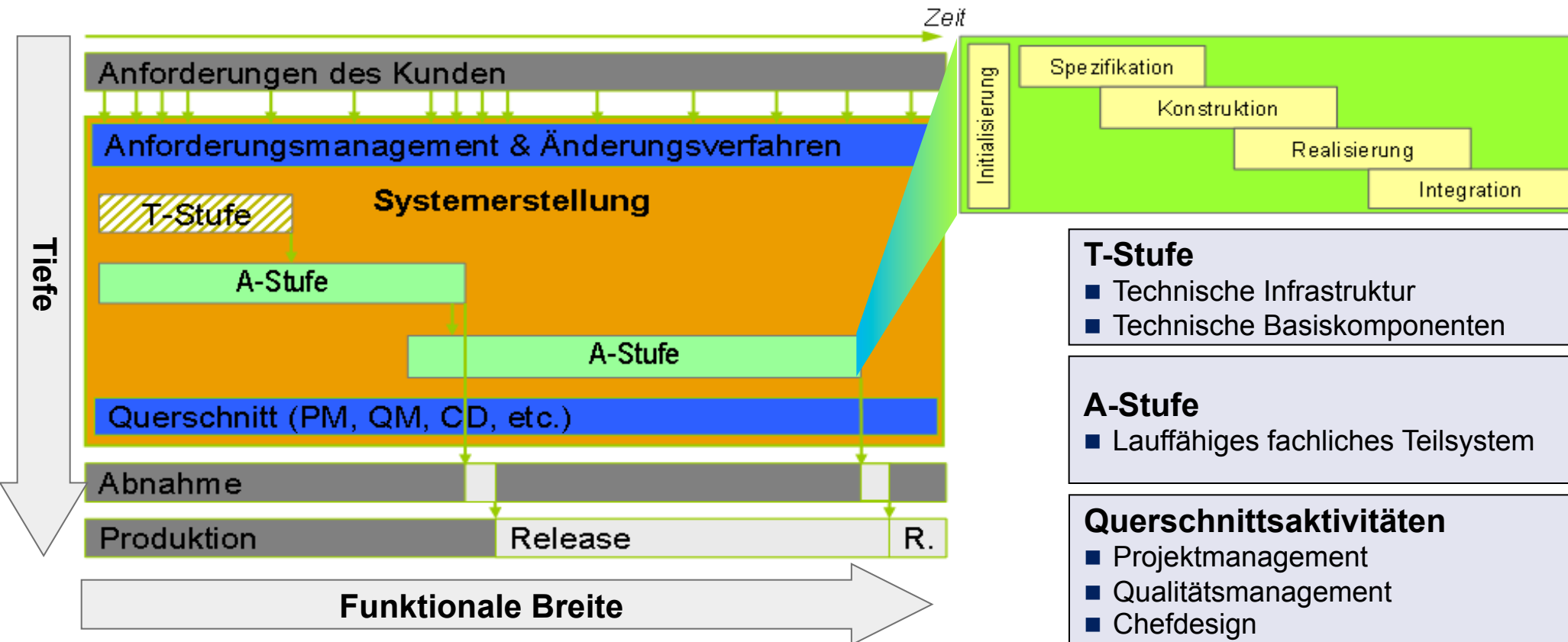
Wir arbeiten wie in einem „richtigen“  
Softwareentwicklungsprojekt – es geht also um viel mehr als nur  
ums Programmieren!



Hochschule Osnabrück  
University of Applied Sciences



# Wir orientieren uns am Projektmodell von Capgemini (sd&m) – einer individuellen Ausgestaltung von RUP



## T-Stufe

- Technische Infrastruktur
- Technische Basiskomponenten

## A-Stufe

- Lauffähiges fachliches Teilsystem

## Querschnittsaktivitäten

- Projektmanagement
- Qualitätsmanagement
- Chefdesign
- Änderungsverfahren
- Konfigurationsmanagement

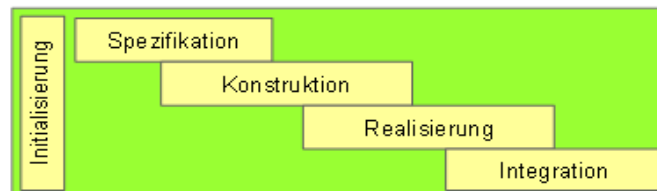
aus [Taubner]

Da unsere Projekte klein und die Projektlaufzeit kurz ist, wählen wir das verzahnte Wasserfassmodell



## Ausgestaltung der A-Stufe je nach Projekttyp

### Verzahntes Wasserfallmodell



#### Motivation / Einsatz

- Kleines Projekt
- Klarer, überschaubarer Funktionsumfang
- Frühe Gesamtspezifikation erforderlich

#### Besonderheiten

- Ist Spezialfall einer Stufe mit nur einem Inkrement
- Funktionsumfang früh definiert und weitgehend fix

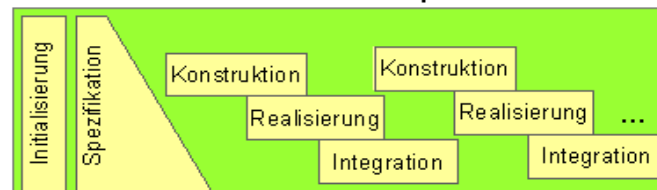
### Inkrementelles Vorgehen



- Schnelle Ergebnisse & schnelles Lernen – auch bei komplexer Funktionalität
- Risiko reduzieren durch „Wichtigstes zuerst“

- Frühes Feedback durch schnell lauffähiges Teilsystem
- Schrittweises Verfeinern

### Inkrementell mit Vorspezifikation



- Gesamtspezifikation zu Beginn der Stufe
- Gesamtaufwand leichter planbar als bei inkrementellem Vorgehen

- Mischform der beiden obigen Modelle
- Schwerpunkt ab zweitem Inkrement auf Realisierung (weniger Konstruktion)

# Wir erstellen die folgenden Dokumente



- Folgende Dokumente werden – neben dem eigentlichen Programmcode – erstellt:
  - *Liste mit Geschäftsanwendungsfällen, Anforderungsliste*
  - *Fachkonzept*
  - *DV-Konzept*
  - *QM-Plan und Nachweis durchgeführter Maßnahmen*
  - *Projektplan*
- Das ist schon Einiges – aber wir lassen auch Vieles weg, was in einem realen Projekt noch hinzukäme: Projekthandbuch, Projekttagbuch, Risikoliste, Programmierrichtlinien, technische Querschnittskonzepte, Entwicklerhandbuch, Konfigurationsmanagementhandbuch, Auslieferungshandbuch, Betriebshandbuch, etc.
- Außerdem vereinfachen wir die Dokumente

**(Trotzdem) gilt:  
Ein Softwareprojekt ist mehr als ein Programmierprojekt!**



# Wir arbeiten mit folgenden Rollen



## ■ **Projektleiter**

- verantwortet die Qualität der Ergebnisse, leitet das Projektteam und berichtet über den Projektfortschritt
- verantwortet den Projektplan

1 Person

## ■ **Qualitätsbeauftragter**

- verantwortet Planung und Durchführung von QS-Maßnahmen
- verantwortet den *QM-Plan* und die *Testfälle*

1 Person

## ■ **Fachlicher Chefdesigner**

- verantwortet die Spezifikation als Ganze, also Methodik und Inhalt, insb. die fachliche Architektur
- verantwortet das *Fachkonzept*

1 Person

## ■ **Technischer Chefdesigner**

- Verantwortet die technische Architektur der Anwendung
- Verantwortet das *DV-Konzept*

1 Person

## ■ **Entwickler/Spezifizierer/Tester**

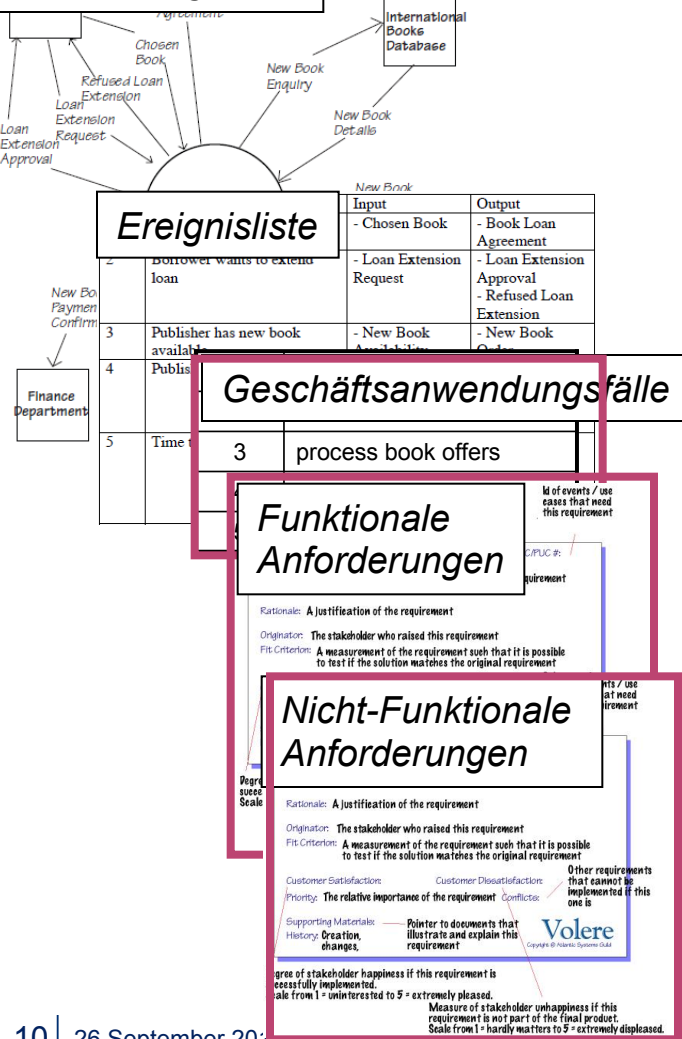
- erstellt die Bestandteile des Softwaresystems

alle

# Für unser Softwareprojekt kürzen wir das Requirements Engineering ab und beschränken uns auf *Geschäftsanwendungsfälle* und *Anforderungen*

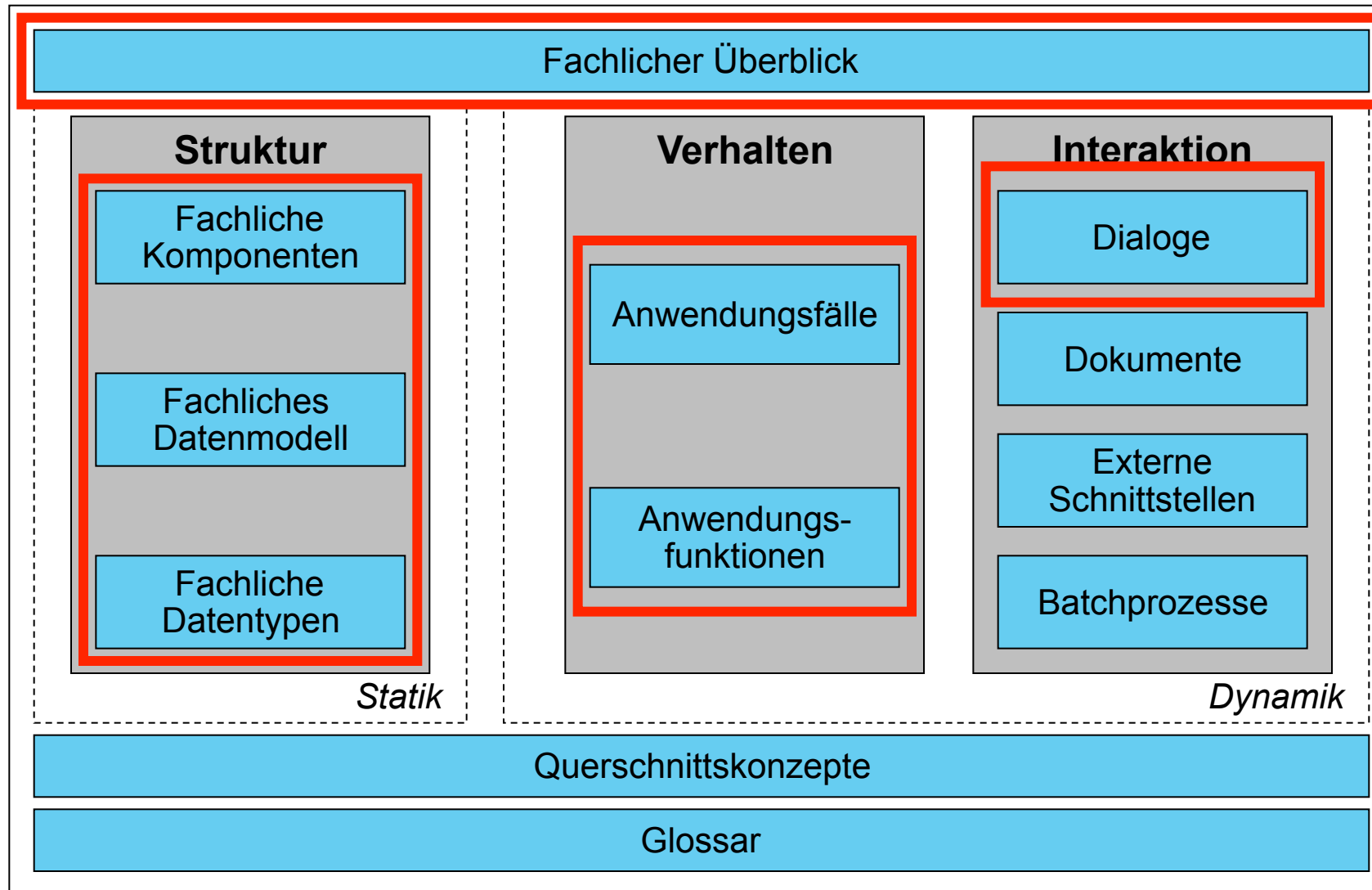


## Kontextdiagramm



- Geschäftsanwendungsfälle werden nur aufgelistet (mit Name und Beschreibung), siehe <Projekt>\_GAFliste.xls
- Funktionale und Nicht-funktionale Anforderungen werden in einer gemeinsamen Liste gepflegt (mit Basisinformationen), siehe <Projekt>\_Anforderungsliste.xls
- Beide Dokumente sind als Ausgangspunkt zu verstehen und können im Projekt weiter entwickelt werden
- Die Anforderungsliste werden wir auch verwenden, wenn sich im Projektverlauf zeigt, dass die Anforderungen geändert werden müssen – auch „Outscoping“ ist also möglich!

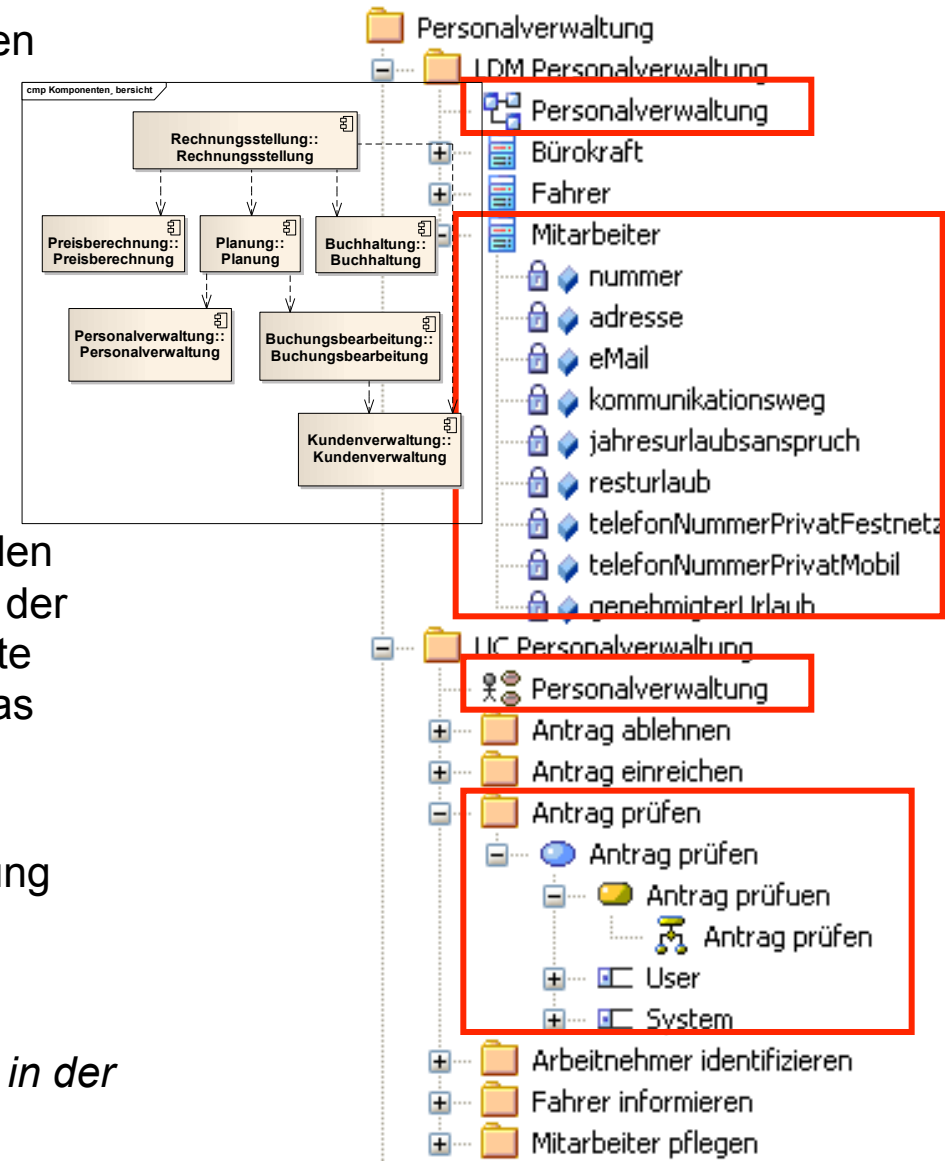
# Für das OOA-Modell erstellen wir alle notwendigen Artefakte



# Die Analysemodell wird vollständig in Enterprise Architect erstellt



- Komponentendiagramm mit fachlichen Komponenten (nur Beziehungen, keine explizite Schnittstellen)
- Zu jeder fachlichen Komponente gehören:
  - Ein Use Case-Diagramm mit allen Anwendungsfällen der Komponente
  - Die Anwendungsfälle mit vollständiger Beschreibung und Aktivitätsdiagramm (evtl. Ausnahme bei einfachen / gleichartigen Anwendungsfällen)
  - Ein (evtl. mehrere) Klassendiagramm(e) mit allen Entitätstypen und allen fachlichen Datentypen der Komponente (evtl. ergänzt um direkt assoziierte Entitätstypen anderer Komponenten, die für das Verständnis wichtig sind)
  - Entitätstypen und fachliche Datentypen mit vollständiger Beschreibung. Auf die Modellierung von Lebenszyklen verzichten wir
  - Dialogentwürfe mit Dialogspezifikation



... **Erinnerung:** Fachliche Komponenten haben schon in der Spezifikation Daten- und Funktionshoheit!

# Arbeiten mit Enterprise Architect in der Analysephase



- Grundsätzlich setzen Sie EA so ein, wie in der Veranstaltung OOAD kennengelernt
- Im Gegensatz zur Veranstaltung OOA arbeiten mehrere Benutzer gleichzeitig mit einem Modell. Wir wählen dazu den „Shared Model“-Ansatz“ (siehe hierzu Folien zu Enterprise Architect)
- Bei Interesse: Detaillierte Informationen in folgenden Whitepapers
  - zu Teamdevelopment: EA\_Deployment.pdf
  - zu Versionskontrolle: EA\_Version\_Control.pdf

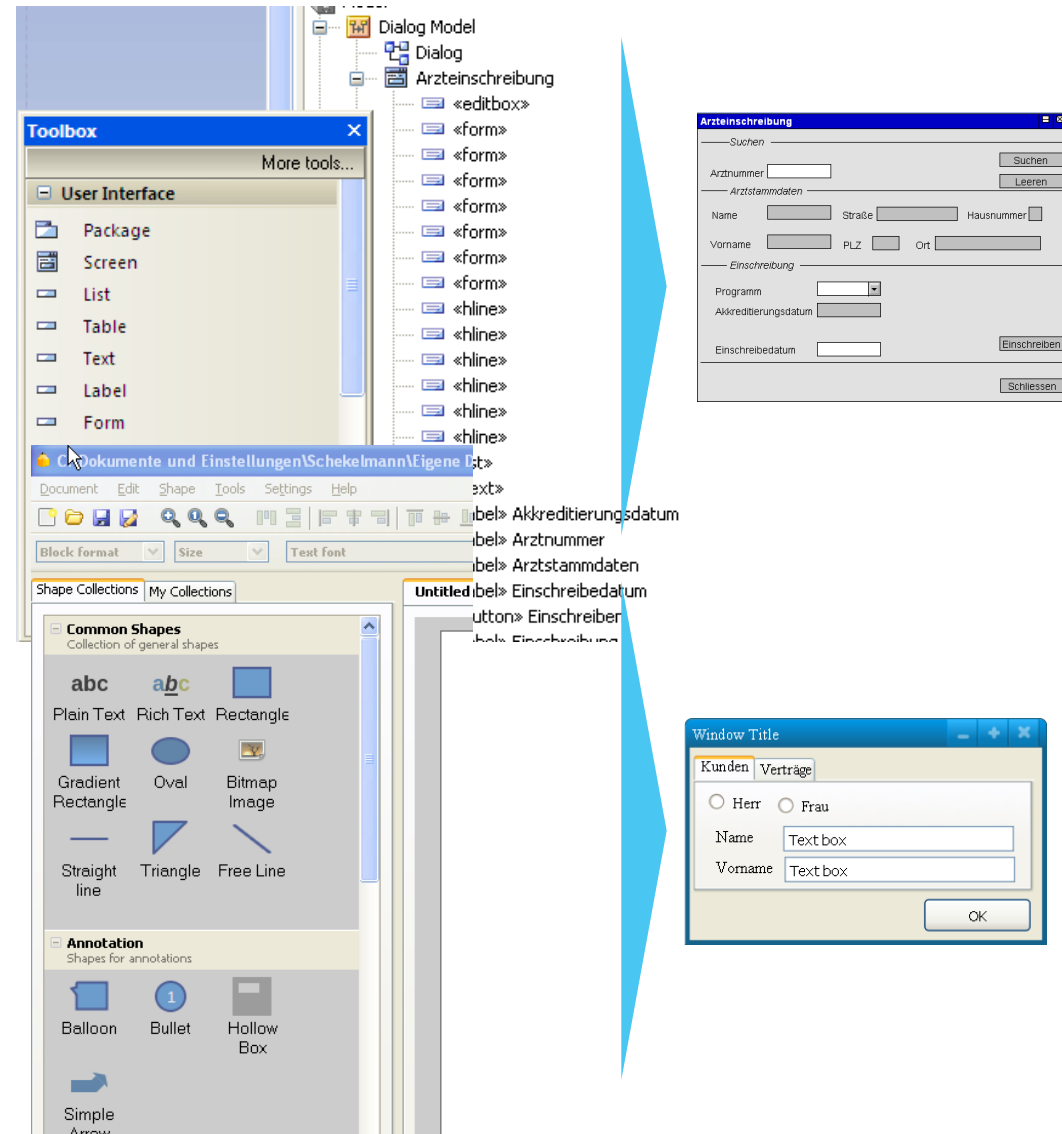
Das physische Dialoglayout kann mit EA  
– oder Powerpoint, Access, etc. erstellt werden.  
Es wird aber in jedem Fall in EA hinterlegt!



Für das Erstellen der Dialogentwürfe gibt es keine konkreten Vorgaben

- Geeignet sind: Enterprise Architect
- Das Freeware Mockup-Tool Evolus Pencil (<http://www.evolus.vn/pencil/>)
- Zur Not geht auch Papier und einscannen
- ...

Hinweis: Die Dialogentwürfe sind *Entwürfe* – sie müssen ordentlich sein, aber nicht perfekt!



# Das Fachkonzept fasst das Analysemodell zusammen und ergänzt einen Überblick



## Gliederung des Fachkonzepts

### 1. Fachlicher Überblick mit

- Problemstellung (=Zusammenfassung der zentralen Anforderungen)
- Lösungsidee
- (evtl. vereinfachtes) Datenmodell mit Kernentitäten
- Fachliches Komponentenbild

### 2. Komponentenbeschreibung je Komponente

- Struktur: Logisches Datenmodell, Liste der Entitätstypen mit Kurzbeschreibung
- Verhalten: Use Case Diagramm, Liste der Use Cases mit Kurzbeschreibung
- Interaktion: Liste aller Dialoge mit physischem Layout

Das Fachkonzept wird als Powerpoint-Foliensatz erstellt:

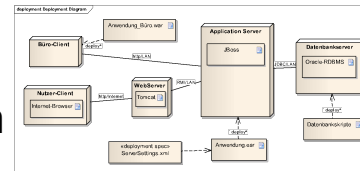
- ➔ weniger Arbeit als ein ausformulierter Text
- ➔ eignet sich gleichzeitig auch für eine Präsentation

In der Konstruktion konzentrieren wir uns auf die *technische* und vor allem auf die *fachliche* Architektur



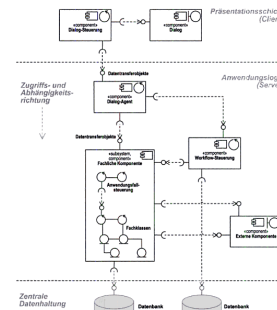
## Architektur der technischen Infrastruktur:

Sie beschreibt die Hardware, die installierte Systemsoftware und das Zusammenspiel von Hardware und Systemsoftware



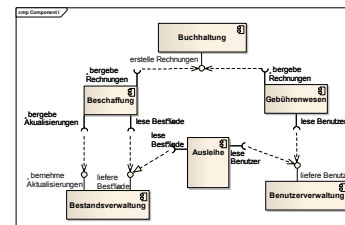
**Wir arbeiten auf den Rechner des Lab4Apps**

**Technische Architektur:** Sie beschreibt die allgemeinen Eigenschaften die Architektur (Schichteneinteilung, vom fachlichen Anwendungsgebiet unabhängigen Komponenten wie Zugriffsschicht, Rahmen für graphische Benutzungsoberflächen, Fehlerbehandlung, etc.)



**Wir verwenden eine Standard 3-Schichten-Architektur**

**Fachliche Architektur:** Sie strukturiert die Software aus Sicht der Anwendung, d.h. aus Sicht des Anwendungsgebiets und der Fachlichkeit



**Wir identifizieren fachliche Komponenten**



Wir modellieren die Artefakte wie in der Veranstaltung OOA in EA



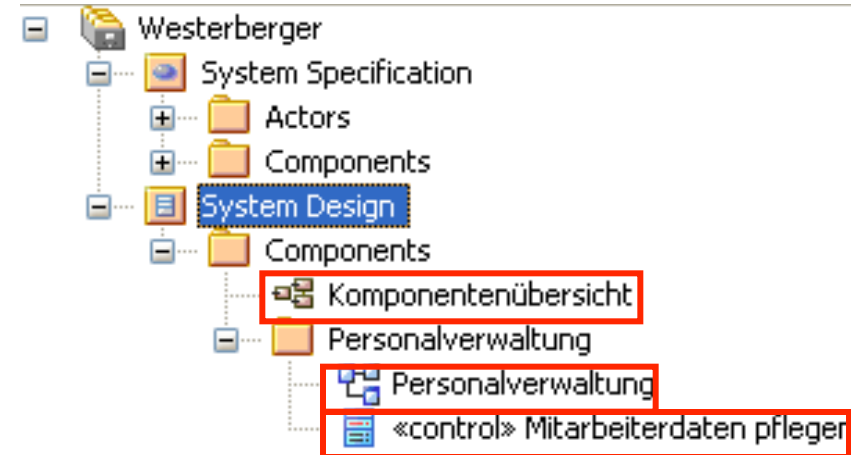
### Folgende Artefakte werden in EA modelliert:

- Klassen (Entitäts- und Kontrollklassen)
- Technische Komponenten

### Zu jeder technischen Komponente gehört:

- Ein Klassendiagramm mit allen Klassen der Komponente
- Die Klassen der Komponente (mit Beschreibung, ohne Methoden)

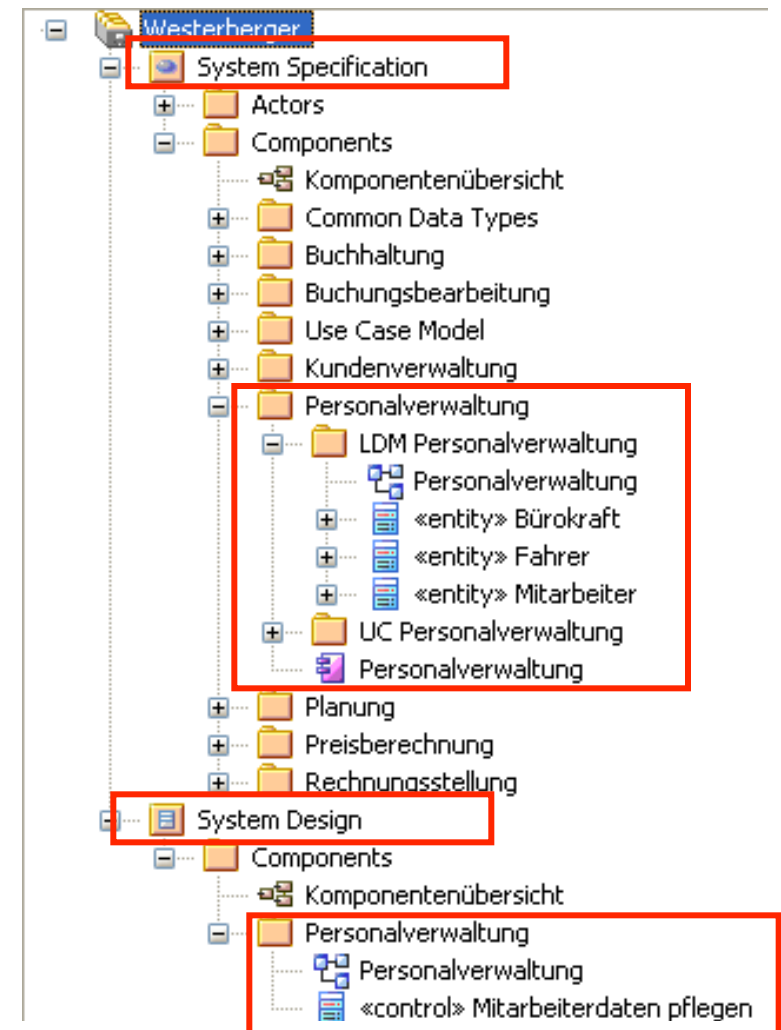
Außerdem eine Komponentendiagramm mit einer Komponentenübersicht



# Arbeiten mit Enterprise Architect in der Designphase



- Empfehlung: Ein gemeinsames Analyse-und Designmodell
  - Das Analysemodell enthält die fachlich motivierten Entitätsklassen
  - Das Designmodell enthält die Kontrollklassen (und ggf. technisch motivierte Entitätsklassen)
- ➔ Die Klassendiagramme im Designmodell enthalten Klassen aus dem Analyse- und dem Designmodell



# Das DV-Konzept fasst das Designmodell zusammen und enthält zentrale Konzepte



## Gliederung des DV-Konzepts

### 1. Technischer Überblick mit

- Komponentenbild
- Dokumentation der wesentlichen Entwurfsentscheidungen zu
  - Oberfläche
  - Anwendungskern
  - Datenhaltung

### 2. Komponentenbeschreibung je Komponente

- Klassendiagramm für jede Komponente

Das DV-Konzept wird als Powerpoint-Foliensatz erstellt:

- ➔ weniger Arbeit als ein ausformulierter Text
- ➔ eignet sich gleichzeitig auch für eine Präsentation



... außerdem setzen wir zwei weitere Werkzeuge zur Projektführung ein: Risikoliste und LOP



	A	B	C	D	E	F	G	H	I	J	K	L
1	Nr	Thema	Frage	Priorität	angelegt	verantwortlich	Zieltermin	geklärt	Status	Ergebnis		
2					von	am		von	am			
3												
4												

## Liste offener Punkte:

Die Liste offener Punkte ist ein wichtiges Arbeitsmedium in Projekten

- Hier werden übergreifende Fragen gesammelt und ihr Status verfolgt
- Für diese Fragen werden Verantwortlichkeiten und Termine festgelegt
- Die Antworten auf die Fragen werden hier (kurz) dokumentiert

## Risikoliste

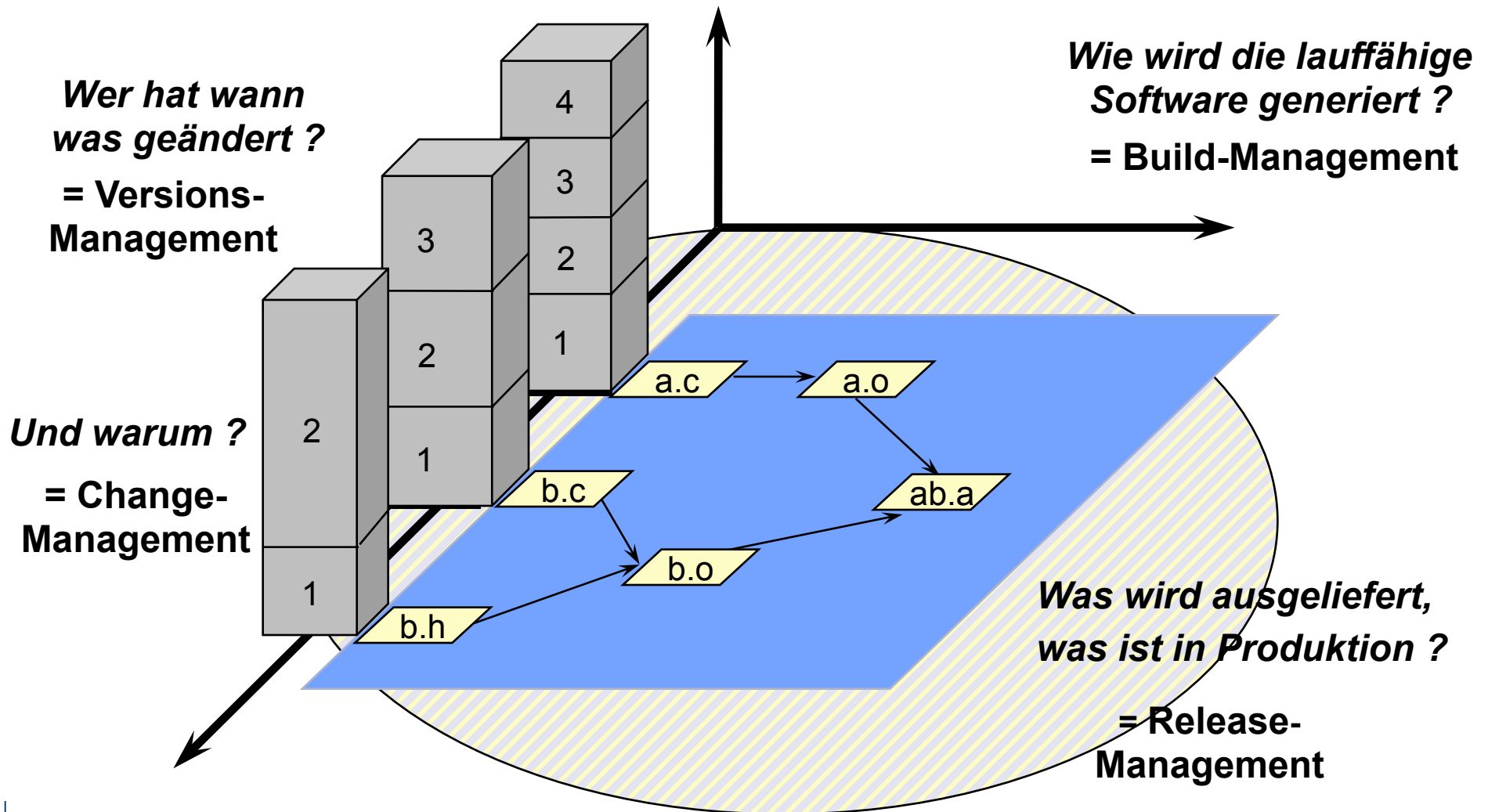
- Die Liste ist Grundlage für das Risikomanagement.
- Sie enthält für jedes Risiko: ID, Beschreibung, Quelle, Kritikalität, Eigentümer, Maßnahmen

# Für das Qualitätsmanagement definieren wir Ziele, Kriterien und Maßnahmen

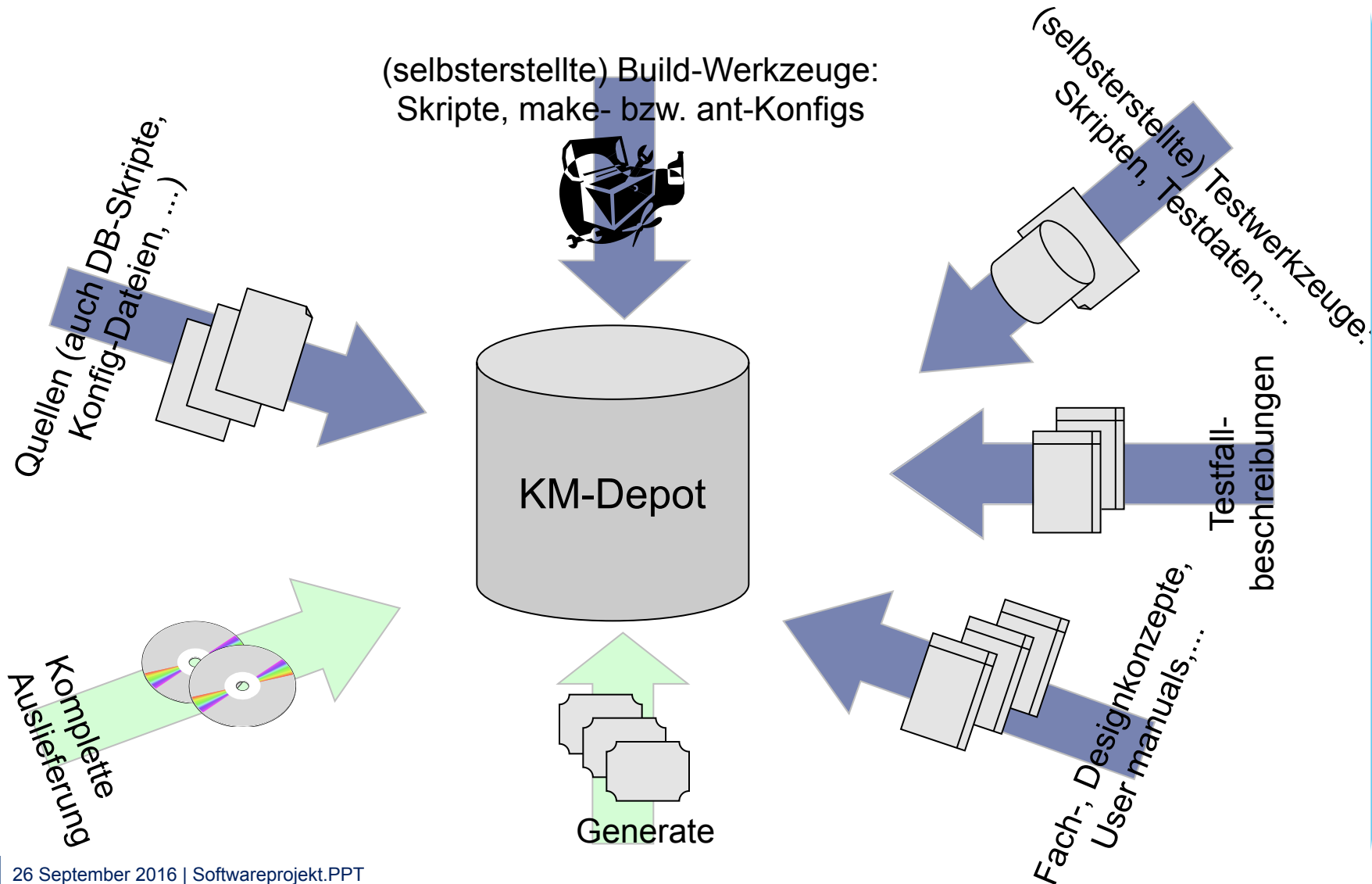


- Der QM-Plan enthält Qualitätsziele, -kriterien und -maßnahmen
  - *Qualitätsziele* für das Projekt
  - *Qualitätskriterien* für die Qualitätsziele
  - Konstruktive und analytische *Qualitätsmaßnahmen* zur Überprüfung der Qualitätskriterien
- Die Durchführung der Qualitätssicherungsmaßnahmen (konstruktiv und analytisch) wird mit in den Projektplan eingeplant
- Die Durchführung der Qualitätssicherungsmaßnahmen sowie Einarbeitung der Ergebnisse wird durch Reviewprotokolle (siehe Vorlage „Reviewprotokoll“) oder Testprotokolle (Testfall\_incl.Dokumentation) dokumentiert

Für das Konfigurationsmanagement setzen wir das Werkzeug Subversion / Git(hub) ein. Dabei konzentrieren wir uns auf Versions- und Releasemanagement



# Wir verwalten alle Projektergebnisse im KM-Werkzeug – bis auf Generiertes



**Wir beschränken uns in unserem Projekt auf  
alles was ins KM-Depot muss!**



In unserem Projekt setzen wir das KM-Werkzeug Subversion ein



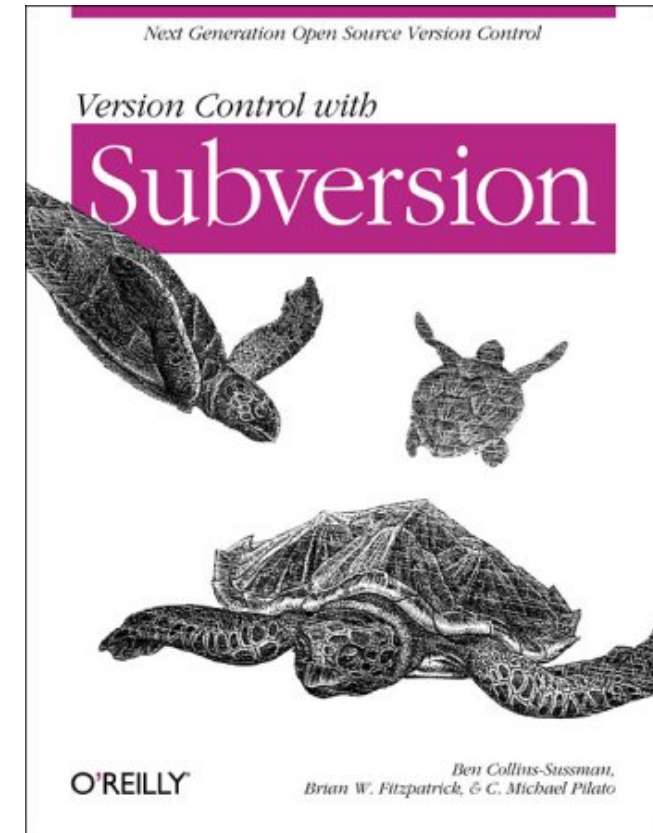
Subversion ist ein freies/Open-Source Versionskontrollsystem. Subversion verwaltet Dateien und Verzeichnisse und die Änderungen an ihnen im Lauf der Zeit.

- Alte Versionen von Dateien können wiederhergestellt werden
  - Geschichte der Änderungen kann verfolgt werden
- ➔ Subversion ist eine Art Zeitmaschine

Subversion kann netzwerkübergreifend arbeiten

### Features von Subversion:

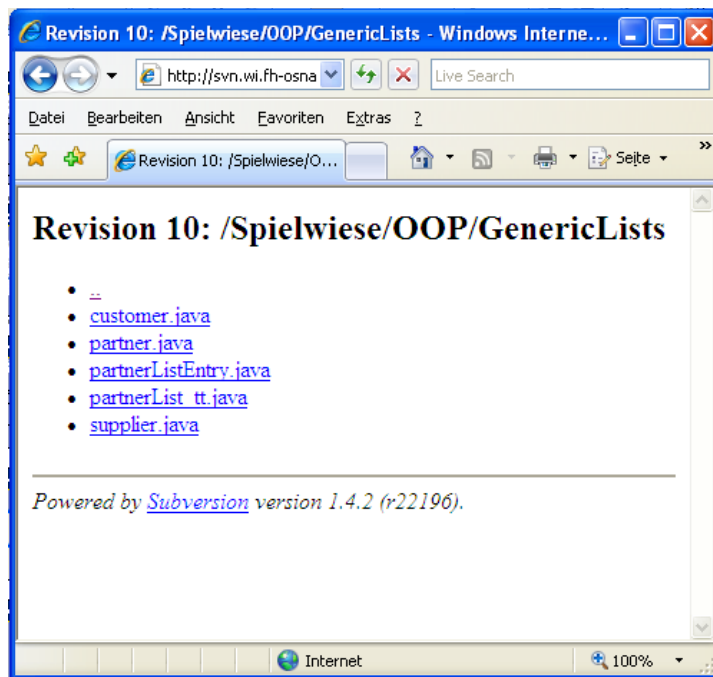
Directories are versioned. Copying, deleting, and renaming are versioned. Free-form versioned metadata ("properties"). Atomic commits. Branching and tagging are cheap (constant time) operations. Merge tracking. File locking. Interactive conflict resolution. Binary files handled efficiently. Costs are proportional to change size, not data size, ...



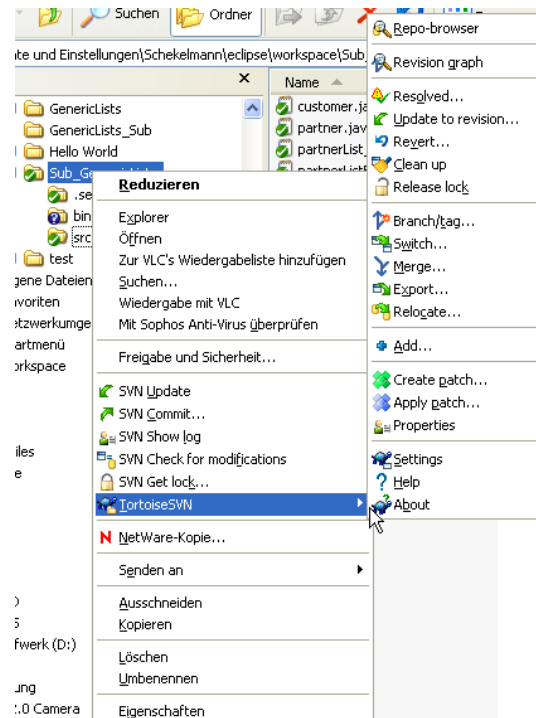
Wir greifen auf Subversion über einen Browser, Explorer oder direkt aus Eclipse zu



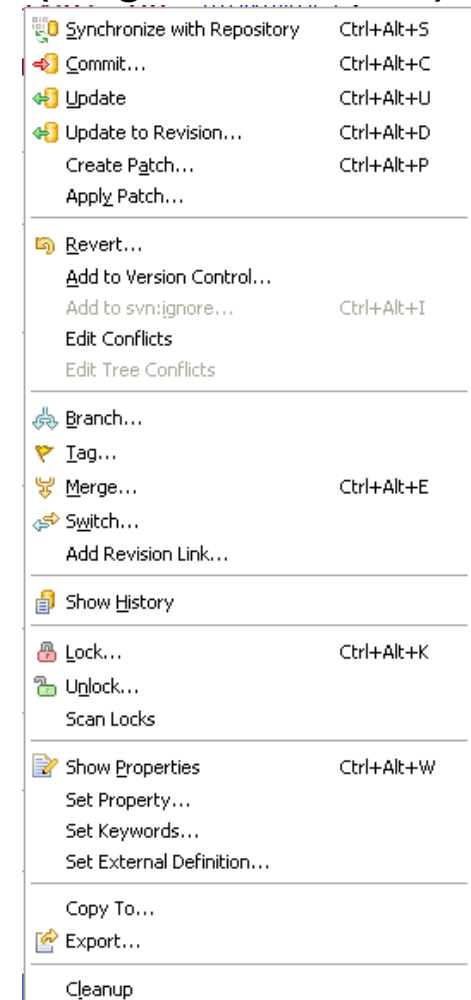
## Zugriff über Browser



## Zugriff über Explorer (Plug-In TortoiseSVN)



## Zugriff über Eclipse (Plug-In Subversive)



# Installation und Einrichtung von TortoiseSVN

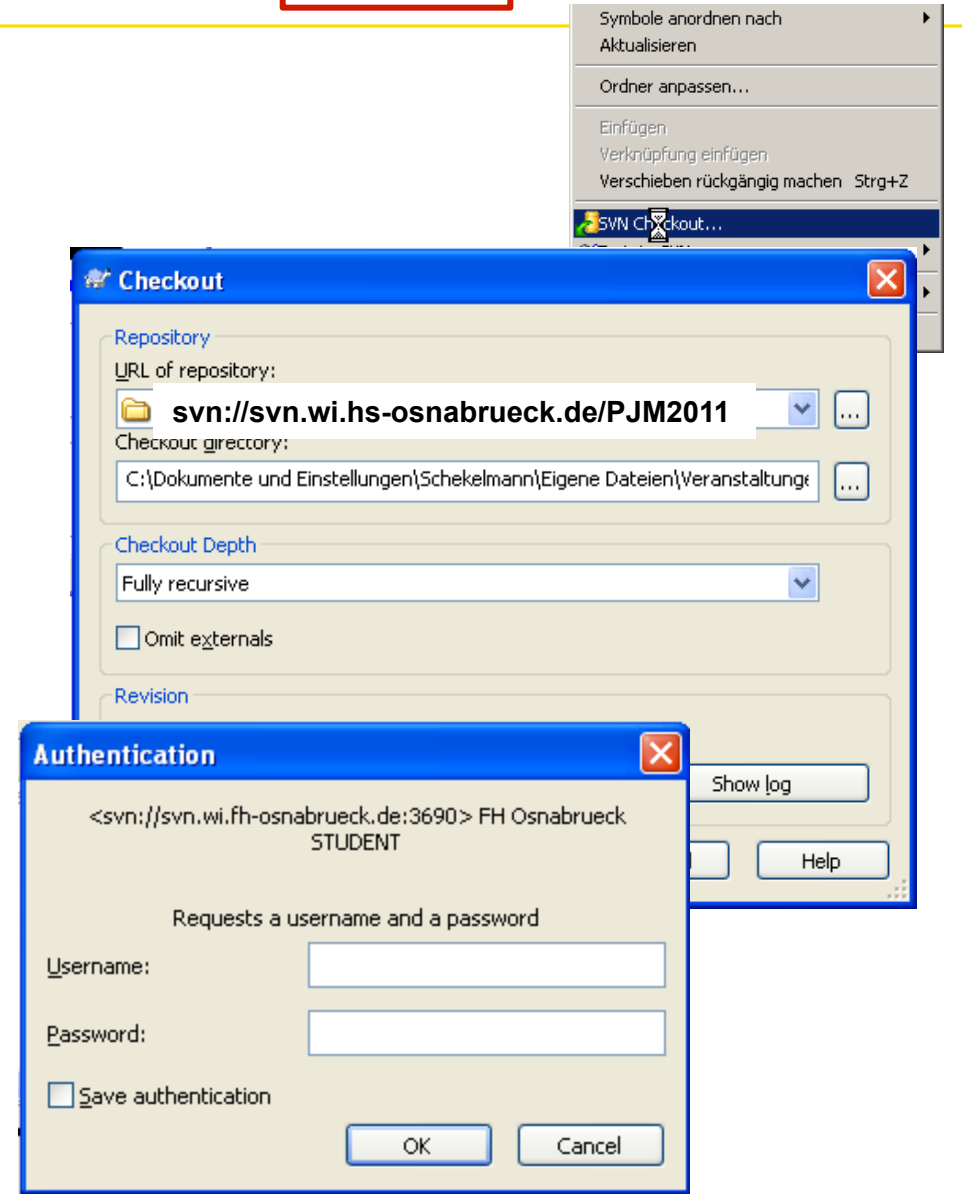


## Installieren (auf eigenem Rechner)

- Verfügbar unter: <http://tortoisesvn.net/downloads>
- Entsprechend der Installationsanweisungen installieren
- Datei C:\Dokumente und Einstellungen\<User>\Anwendungsdaten\Subversion\config durch bereitgestellte Config-Datei ersetzen

## Einrichten

- Lokales Arbeitsverzeichnis (=Entwicklerarbeitsraum!) anlegen
- Datenn initial auschecken: „SVN Checkout ...“ im Kontextmenu
- Repositoryinformationen angeben
- Authentifizierung dauerhaft bestätigen
- Username und Password eingeben



In der Java-Entwicklung greifen wir auf das Repository direkt über Eclipse zu



Um aus Eclipse auf Subversion zuzugreifen, ist das Plug-In Subversive erforderlich,

<http://www.eclipse.org/subversive/>

### Installieren (auf eigenem Rechner)

verfügbar unter <http://www.eclipse.org/subversive/downloads.php>

Installationsanleitung: siehe [Eclipse]

### Einrichten (einmalig auf jedem Rechner)

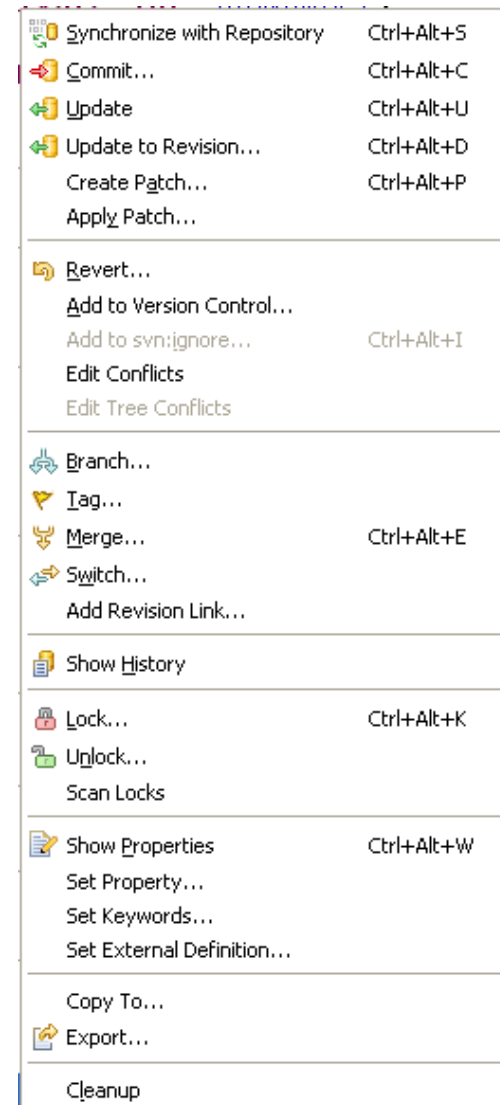
siehe [Eclipse]

### Projekt unter Versionskontrolle stellen (einmalig pro Projekt)

siehe [Eclipse]


### Zugriff auf das Repository aus Eclipse

siehe [Eclipse], Kommandos sind dieselben wie in Subversion



In realen Projekten ist ein automatischer Buildprozess erforderlich. Darauf verzichten wir hier



- Buildmanagement wird in allen Phasen gebraucht.
    - Entwicklung
    - Integration
    - Release
  - Komplexe Abhängigkeiten
  - Generierung verschiedenster Ergebnisse
- 
- Frühzeitig hohen Automatisierungsgrad anstreben
  - Automatisierte Tests zum Check des Buildergebnisses

➔ In unserem Projekt machen wir das Build aus Eclipse heraus

# Release-Management machen wir nur rudimentär: Nur Baselining, kein Branching, kein Merging

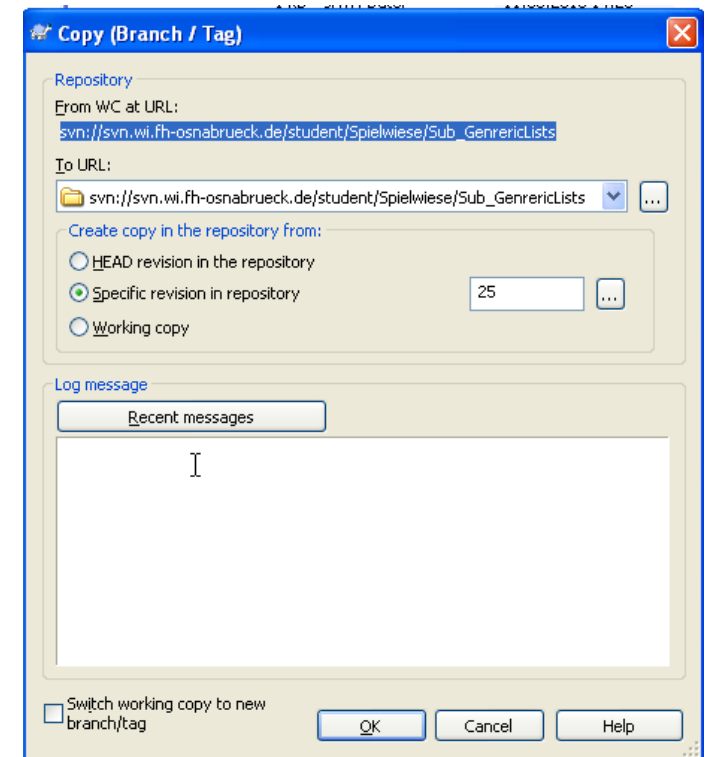


Tags sind in Subversion eigentlich eine Kopie, d.h. ein definierter Stand wird kopiert und an einem neuen Platz zum Repository hinzugefügt.

Dazu werden am einfachsten ganze Verzeichnisse getagged

Es gibt 3 Möglichkeiten

- Die aktuelle Revision im Repository taggen („Head“)
- Eine bestimmte Revision taggen
- Arbeitskopie taggen



**→ In unserem Projekt taggen wir mindestens am Ende des Projekts  
– der getaggte Stand ist der, der bewertet wird.**

# Zusammenfassung: Die erwarteten Ergebnisse im Überblick

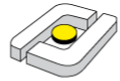


- aktuell gehaltener Struktur-, Aufwands, Team- und Terminplan
- Liste offener Punkte
- Risikoliste
- QM-Plan mit Zielen, Kriterien und Maßnahmen
- Testfälle
- Nachweise über durchgeführte Maßnahmen

Konfig.-Management

- Geschäftsanwendungsfälle
- Anforderungen (funktional und nicht-funktional)
- Analysemodell in EA, gegliedert in Komponenten mit Anwendungsfällen, fachlichen Datenmodell und Dialogmodell
- Zusammenfassendes Fachkonzept als Foliensatz (inkl. fachlichem Überblick)
- Designmodell in EA, gegliedert in Komponenten mit Klassen
- Zusammenfassendes DV-Konzept (inkl. weiterer technischer Konzepte)

- Anwendung, die lauffähig auf einem Rechner in Lab4Apps zur Verfügung gestellt wird
- Programmquellen in Subversion



## **Montag, 31. Oktober 2016 (12:00, CN0105): Erster Überblick (20 min) – PL**

Erwartet wird:

- Ein erster Projektplan, Risikoliste
- Ein erster fachlicher Überblick mit Geschäftsanwendungsfällen und wesentlichen Anforderungen
- Ein erster Überblick über die technische Architektur

## **Donnerstag, 28. November 2016 (12:00, CN0105): Präsentation der Spezifikation (20min) – FCD**

Erwartet wird:

- Aktualisierter Projektplan
- Fachkonzept

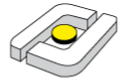
## **Januar/Februar 2017 (Zeit und Ort folgen): Abschlusspräsentation (35min) – TCD / QB**

Erwartet wird

- Präsentation der Anwendung
- Überblick über DV-Konzept
- Überblick über durchgeführte QS-Maßnahmen



... und außerdem

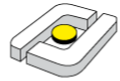


Hochschule Osnabrück  
University of Applied Sciences

## Coaching Termine

- wöchentlich, 30 min pro Gruppe während der regulären Vorlesungszeit
- Terminabstimmung individuell mit dem Betreuer
- Standard-Agenda
  - Was ist seit dem letzten Treffen passiert?
  - Welche Schwierigkeiten sind aufgetreten?
  - Was nimmt sich die Gruppe für den nächsten Zeitraum vor?

*... außerdem sind natürlich bei Bedarf weitere Termine möglich!*

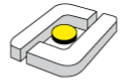


## Basisliteratur

- J. Siedersleben (Hrsg.): Softwaretechnik, 2. Auflage. München, Wien: Hanser 2003.  
insb. Kapitel 2 (Projektmodell), 12 (Konfigurationsmanagement), 13 (Qualitätsmanagement), 14 (Testen), 15 (Projektmanagement), Kopien bei Bedarf für jede Gruppe bei mir
- C. Michael Pilato, et.al: Versionskontrolle mit Subversion, 3.Auflage. O 'Reilly 2009.  
Ältere Ausgabe (ausreichend) on-line verfügbar: <http://svnbook.red-bean.com/>
- Stephan Kleuker: Nutzung der SW-Entwicklungsumgebung Eclipse. Unterlagen zur Vorlesungen in der Fakultät Iul, in Stup.IP

Außerdem wird **Eigeninitiative** erwartet – auch beim eigenständigen Recherchieren!

Und nun ... die Projekte 😊



Hochschule Osnabrück  
University of Applied Sciences

**Projekt 1:** eLearning-Portal für BIM(-Softwareentwicklungsveranstaltungen)

**Projekt 2:** Verteilte Arbeitszeit- und Statistikerfassung

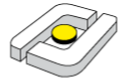
**Projekt 3:** Online-Clicker-Anwendung

**Projekt 4:** Terminfinder-App

**Projekt 5:** Sportwetten-Runden-Portal

**Projekt 6:** Online-Prüfungsportal

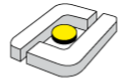
**Projekt 7:** Verwaltung von Arbeitsverträgen



**33%** eigene Leistung, d.h. Bewertung der Ergebnisse der eigenen Rolle

**33%** Gruppenergebnis, d.h. Mittelwert aller eigenen Leistungen

**34%** Software



***“Ein am Projektanfang verlorener Tag tut genauso weh wie ein  
am Projektende verlorener Tag”***

*(Tom DeMarco)*

***“Do the simplest thing that might possibly work”***

*(Extreme Programming)*

***“Premature optimization is the root of all evil”***

*(Donald E. Knuth)*