

Software Engineering

SS 2005

Prof. Dr. Barbara Paech, Jürgen Rückert



Institut für Informatik
Im Neuenheimer Feld 326
69120 Heidelberg
<http://www-swe.informatik.uni-heidelberg.de>
paech@informatik.uni-heidelberg.de



RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG



4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



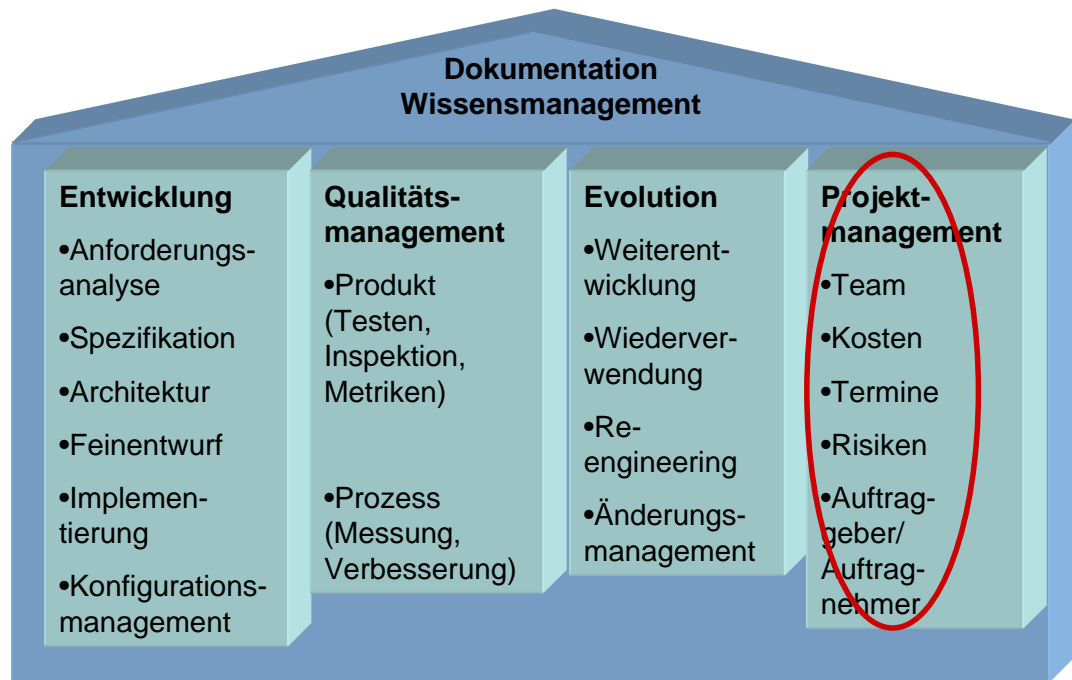
1.3.1. Aufgabenbereiche des Engineering

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 3

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5. Management

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ 5.1. Überblick
- ◆ 5.2. Aufgaben des Projektmanagement
- ◆ 5.3. Mitarbeiterführung
- ◆ 5.4. Organisation
- ◆ 5.5. Planung und Kontrolle

Folie 4

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

DIN 69901

◆ Projekt

- Vorhaben, das im Wesentlichen durch **Einmaligkeit** der **Bedingungen** in ihrer Gesamtheit gekennzeichnet ist, wie z.B.
 - Zielvorgabe
 - Zeitliche, finanzielle oder andere Begrenzungen
 - Abgrenzungen gegenüber anderen Vorhaben
 - Projektspezifische Organisation

PMBOK

◆ Project

- A **temporary** endeavor undertaken to **create** a unique product, service, or result

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

DIN 69901

◆ Projektmanagement

- Gesamtheit von **Führungsaufgaben, -organisation, -techniken und -mittel** für die Abwicklung eines Projektes

PMBOK

◆ Project Management

- The application of **knowledge, skills, tools, and techniques** to project activities to achieve the project requirements

5.1. Welche Inhalte umfasst das Projektmanagement?

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Management von
 - Terminen (Projektplan)
 - Kosten
 - Risiken
 - Beteiligten
 - Team
 - Auftraggeber/Auftragnehmer

- Funktionalität
- Qualität

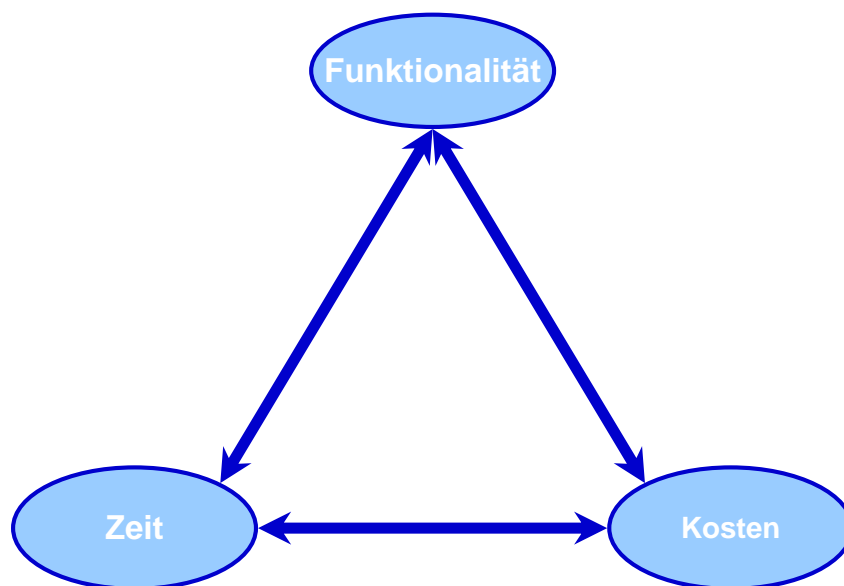
5.1. Magisches Dreieck

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



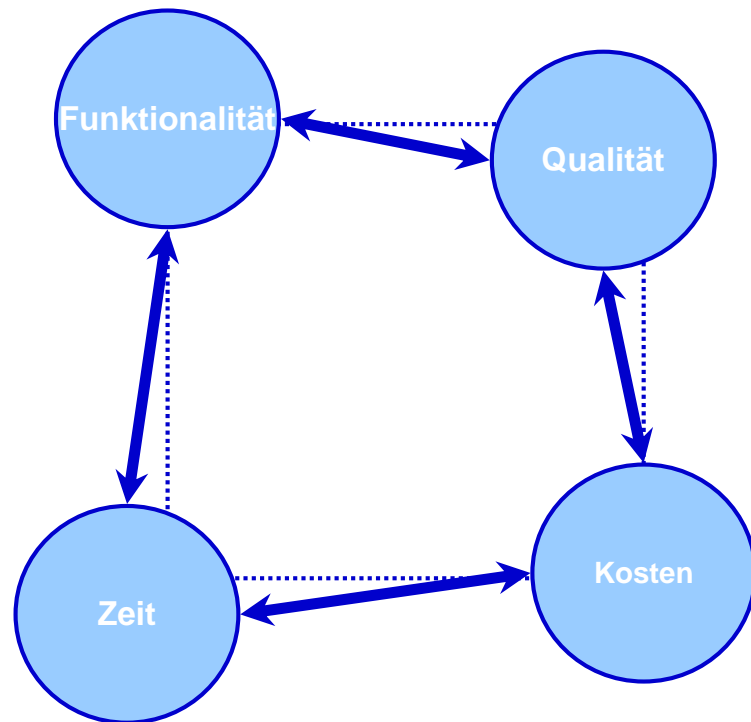
5.1. Magisches Viereck

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 9

Barbara Paech

Vorlesung - Software Engineering - SS 2005

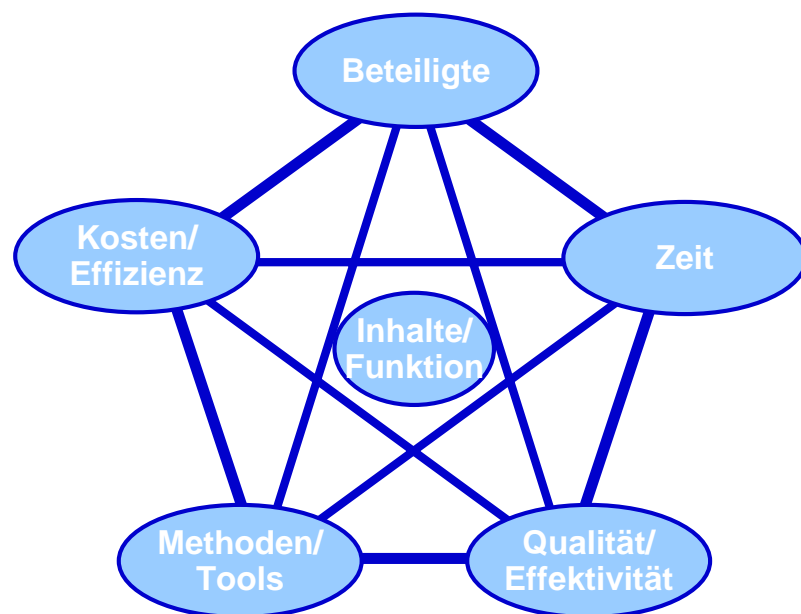
5.1. Magisches Fünfeck (1)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 10

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5.1. Magisches Fünfeck (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Projektinhalt

- Ziele, Projektgröße, Änderungswünsche
- Projektbeschreibung

◆ Qualität/Effektivität

- Effektivität: „die **richtigen Dinge** tun“
- Qualität: „so gut wie erforderlich“
 - NICHT „so gut wie möglich“
- Operationalisiert/messbar

◆ Kosten/Effizienz

- Effizienz: „die **Dinge richtig** tun“
- Software-Entwicklung immateriell:
 - Kosten = Personalkosten
- Outsourcing

5.1. Magisches Fünfeck (3)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Zeit

- Brook'sches Gesetz (überprop. Koordinationsaufwand)
- Angemessene Zeitvorgaben
 - Nicht zu lange Intervall
 - Realistisch/keine Überstunden
- Parkinson's Gesetz („man braucht soviel Zeit, wie man hat“)
- Überschreitung → Umplanung

◆ Beteiligte

- Lenkungsausschuss
- Projektleiter
- Projektteam
- Auftraggeber

◆ Methoden/Tools

- Projektspezifisch
- Standards verwenden

5.2. Welche Aufgaben hat der Projektleiter?

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 13

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5.2. Aufgaben des Projektmanagers (1)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Planung (5.5)**
 - Ziele setzen
 - Realistisch, widerspruchsfrei, eindeutig, verständlich
 - Termine, Spielregeln festsetzen
- ◆ **Organisation (5.4)**
 - Rollen, Verantwortlichkeiten festlegen
 - organisatorische Strukturen festlegen
 - Ablauf-, Aufbauorganisation
 - Kommunikation festlegen
- ◆ **Mitarbeiterentwicklung (5.3.)**
 - Personal auswählen
 - Weiterbildung durchführen
 - Personal beurteilen, entlassen

Folie 14

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5.2. Aufgaben des Projektmanagers (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Leitung (5.3.)

- Mitarbeiter führen
- Aktivitäten koordinieren
- Aufgaben delegieren
 - Aufgabe + Verantwortung + Befugnisse/Rechte
- Entscheidungen treffen
 - Häufiger Fehler: zu späte, nicht nachvollziehbare Entscheidungen

◆ Kontrolle (5.5.)

- Berichts- und Kontrollwesen etablieren
- Projekt Überwachen
 - „Planung ohne Kontrolle ist sinnlos“
- Risiken erkennen
- Korrigieren/Umplanen, wenn notwendig

5.2. SW-Projektprinzipien (1)

Spezifische Erfahrungen aus Softwareprojekten:

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ **Inhalt/Qualität:** Gestaltung von Technik und Arbeit

- Abgleich von Anforderungen und Möglichkeiten
- Benutzerbeteiligung

◆ **Beteiligte: Lernprozess, Konsensfindung**

- Zwischen Entwickler und Anwender
- Während der Entwicklung
- Interessenskonflikte lösen

5.2. SW-Projektprinzipien (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Organisation: Entscheidungsprozess**
 - Inkrementelle Entscheidungsfindung
- ◆ **Organisation: Vielschichtiger Produktionsprozess**
 - Koordination

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Organisation: kontinuierlicher, projektübergreifender Prozess**
 - Bezüge zum Projektumfeld
 - Projektübergreifender Wissenstransfer

5.2. SW-Projektprinzipien (3)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ **Planung: Reduzierung von Unbestimmtheiten**
 - Schrittweise Reduzierung
 - Periodische Neubestimmung von Zielen
- ◆ **Kontrolle: Selbststeuerungsprozess**
 - Freiraum für "stille Leistungen"

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Kontrolle: Spannungsfeld zwischen Offenheit und Verbindlichkeit**
 - Kontrollierte und gesteuerte Korrektur von Zielen

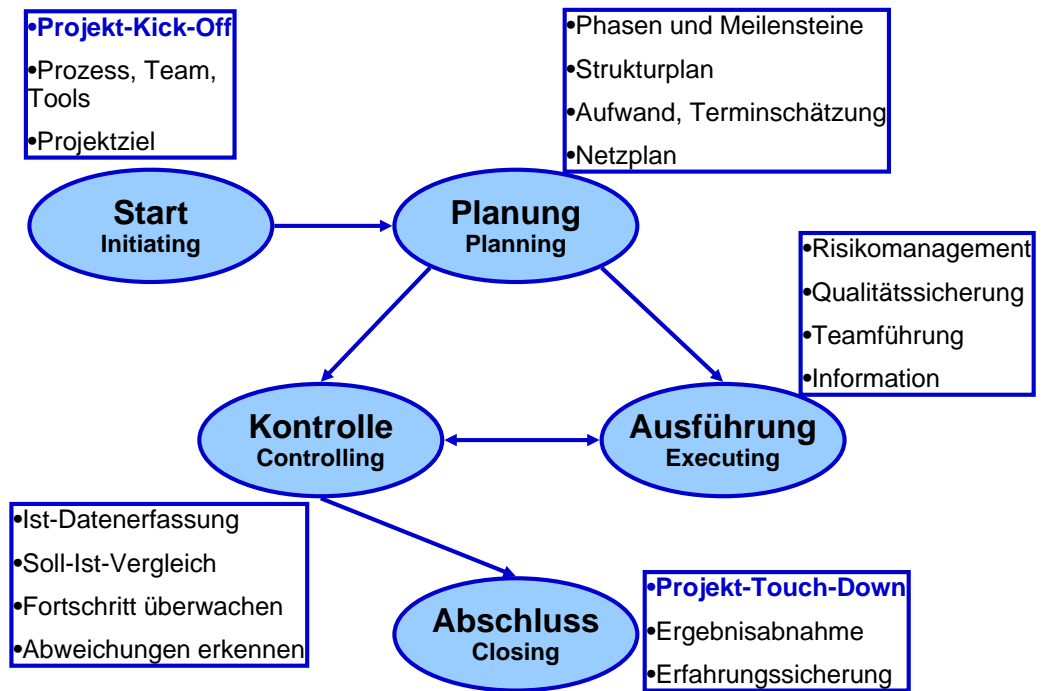
5.2. Phasen des Projektmanagement (1)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 19

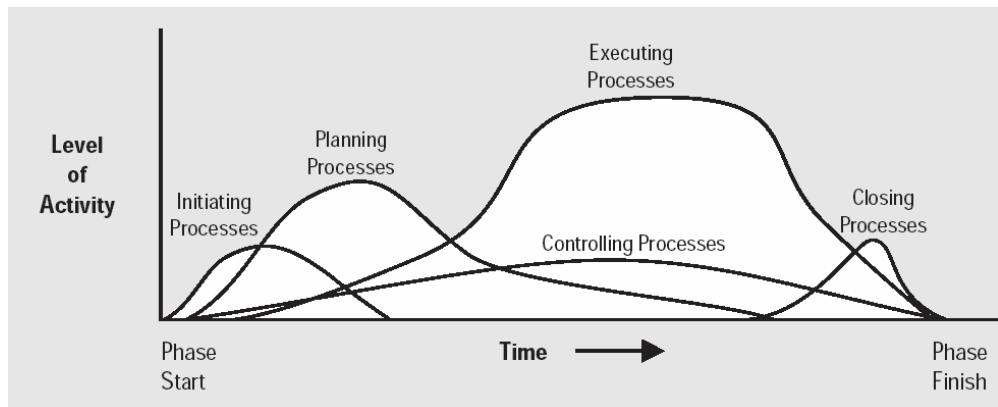
5.2. Phasen des Projektmanagement (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 20

5.3. Die entscheidende Ressource: Menschen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Four Essentials of Good Management

- ◆ Get the right people
- ◆ Match them to the right jobs
- ◆ Keep them motivated
- ◆ Help their teams to jell and stay jelled
(all the rest is Administrativa)

Tom DeMarco, The Deadline
Dorset House, 1997

Folie 21

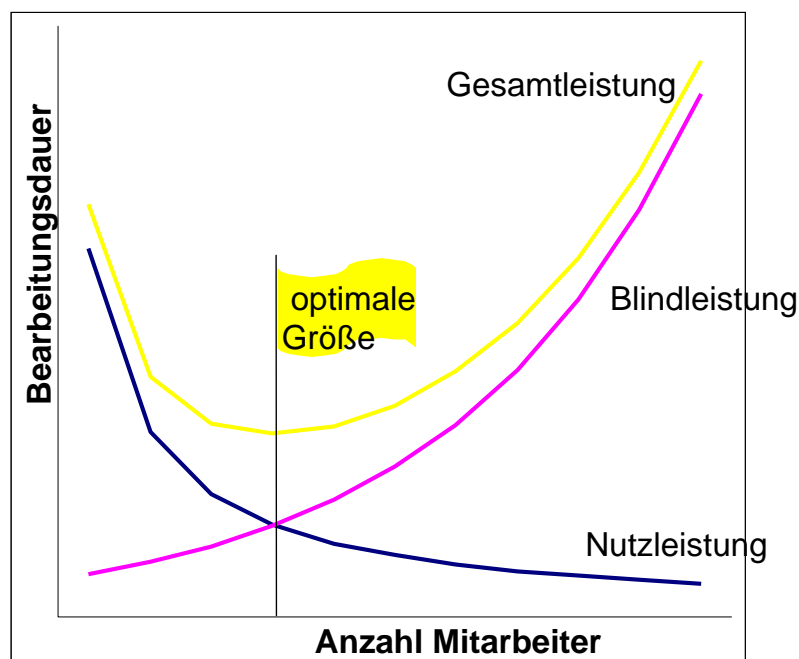
5.3. Teamgröße

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Den Zeitverzug in einem Projekt kann man NICHT durch
Hinzunahme neuer MitarbeiterInnen ausgleichen!

Folie 22

5.3. Fachliche Qualifikation

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Die benötigten Qualifikationen müssen mit den vorhandenen abgeglichen werden:

◆ Hilfsmittel: **Profile**

➤ **Stellenprofile**

welche Fähigkeiten werden für die Arbeit benötigt

➤ **Mitarbeiter-Qualifikationsprofile**

welche Fähigkeiten hat eine MitarbeiterIn

5.3. Soziale Qualifikation

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Die fachliche Qualifikation ist nicht allein entscheidend

➤ **Stresstoleranz**

➤ **Flexibilität**

➤ **Soziabilität**

➤ **Ausgewogenheit zwischen Bescheidenheit/Selbstbehauptung**

➤ **Selbstdisziplin**

➤ **Abstraktionsvermögen**

Personality
Is more important than
Intelligence
(Weinberg, 1971)

5.3. Gute Motivation und Teambildung (1)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Erfolgschance

- Das Team muss eine Chance haben, die Projektziele zu erreichen:
Ziele müssen realistisch sein

◆ Gemeinsames Ziel

- Projektziele müssen allen bekannt, und von allen verstanden sein

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- ▶ 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Klare Kompetenzen

- Die Teammitglieder müssen wissen, was sie dürfen:
 - fachliche, disziplinarische Weisungsbefugnis
 - Spielregeln
- Spielregeln gemeinsam erarbeiten

5.3. Gute Motivation und Teambildung (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Bedeutung des Einzelnen

- Jedes einzelne Teammitglied muss eine klare, wichtige Rolle im Projekt spielen
(kein ersetzbares, kleines Rädchen)

◆ Berücksichtigung des Menschen

- Auf die – über die Arbeit hinausgehenden – Bedürfnisse muss eingegangen werden, z.B. persönliche Probleme

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- ▶ 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Leichte Überforderung

- Dauernde leichte Überforderung steigert Arbeitsqualität und Zufriedenheit

5.3. Gute Motivation und Teambildung (3)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Konsensentscheidungen

- Wichtige Entscheidungen möglichst im Konsens treffen
- Entscheidungen ggf. erklären

◆ Offene Kommunikation

◆ Regelmäßigen Rückmeldungen

- Rückmeldungen bei guten, wie auch bei unbefriedigenden Leistungen

◆ Konflikte lösen

- Konflikte frühzeitig erkennen, ansprechen und lösen

Folie 27

5.4. Organisation

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Struktur, in der einzelne Stellen wohldefinierte Aufgaben haben

- notwendig, um zielgerichtet arbeiten zu können
- abhängig von Unternehmensgröße

◆ Aufbauorganisation

- hierarchische Untergliederung, Weisungsbefugnis

◆ Ablauforganisation

- Festlegung der Arbeitsabläufe

Folie 28

5.4. Organisation bei Software-Projekten

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Ablauforganisation

- Geregelt durch Prozesse (V-Modell, RUP, etc)

◆ Aufbauorganisation

- **Einbettung** von Projekten in Unternehmensorganisation

- Linie
 - Matrix
 - Projekt
- } Organisation

- Organisation **innerhalb** eines Projektes

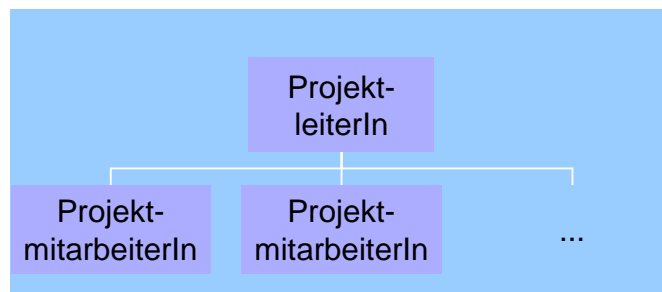
5.4. internes Organisationsbeispiel: kleines Projekt

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



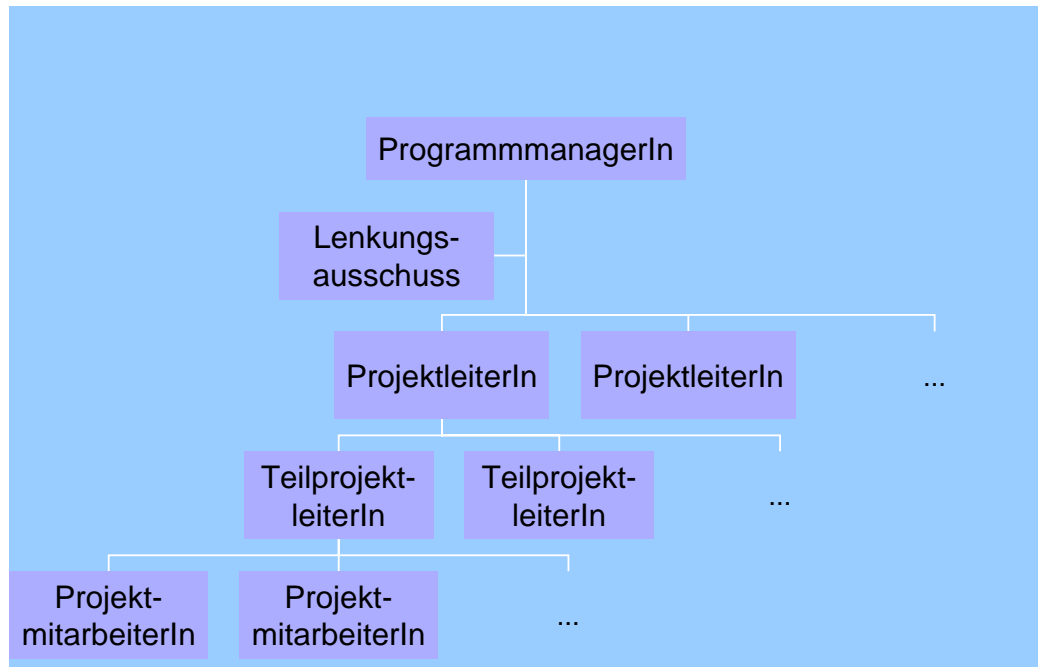
5.4. interne Organisationsbeispiel: großes Projekt

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Folie 31

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5.5. Planung und Kontrolle

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ 5.5.1. Projektstrukturplan
 - **WAS** ist zu tun?
- ◆ 5.5.2. Aufwand und Kostenschätzung
 - Wie viel **RESSOURCEN** sind nötig?
- ◆ 5.5.3. Ablaufplanung
 - **WANN** wird etwas gemacht?

Folie 32

Barbara Paech

Vorlesung - Software Engineering - SS 2005

5.5.1. Projekt-Strukturplan

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Definition

- alle zur Erreichung des Projektzieles notwendigen Aufgaben
- Dargestellt in einer hierarchischen Anordnung

◆ Abkürzung: PSP, WBS (work breakdown structure)

◆ Zweck

- Zerlegung des Projektes in überschaubare Einheiten
 - intern
z.B. Zuweisung der Verantwortung, Überprüfung auf Vollständigkeit, Schaffung eines gemeinsamen Verständnisses
 - extern, z.B. Review, Präsentation
- Grundlage aller
 - Planung
 - Projektmanagement-Aktivitäten

5.5.1. Aufbau eines Projekt-Strukturplans

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Gliederung nach **Objekten (Teilsystemen)** oder **Funktionen (Phasen)**

- Oft Mischform: obere Ebene objekt-orientiert, untere Ebene funktions-orientiert
- Nicht auf einer Ebene mischen!

◆ Meist 3-5 Ebenen, Max 7 Töchter per Knoten

◆ Wiederkehrende Aufgaben mit einem Knoten darstellen

◆ Blätter des PSP: Arbeitspakete

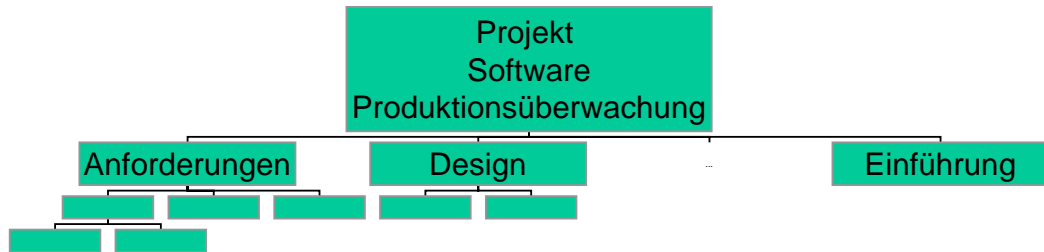
5.5.1. Phasenorientierter PSP

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



5.5.1. Teilsystemorientierter PSP

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Produktstrukturplan Software Produktionsüberwachung

1. Sensor
 - a. Optische Lageerkennung
 - b. Optische Untersuchung
2. Kommunikationssystem
 - a. ...
3. Server
4. Melde-/Alarmsystem

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Festlegen von Arbeitsschritten mit

- Klar umrissenem **Ergebnis**
 - Abgeschlossen
 - Überprüfbar
- Klarer **Verantwortung**
- Festlegbarem **Aufwand und Terminen**

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- ▶ 5.5. Planung/Kontrolle

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- ▶ 5.5. Planung/Kontrolle

| | | |
|--------------------------------|----------------------------|----------------|
| ARBEITSPAKET | | Datum: |
| AP-Nr.: | <u>AP-Beezeichnung:</u> | AP-Leiter: |
| Projektnummer: | <u>Projektbezeichnung:</u> | Projektleiter: |
| Ziele/Ergebnisse: | | |
| Durchführung/Aufgaben: | | |
| 1. | | |
| 2. | | |
| 3. | | |
| 4. | | |
| Voraussetzungen/Zulieferungen: | | |

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Maße:

- Personenjahre (person year)
- Personenmonat (person month)
- Personentage (person day)

◆ Vorsicht:

- Personentag ungleich Arbeitszeit für Projekt
- Mitarbeiter/in
 - Arbeitet an Projekt
 - Arbeitet an anderen Projekten
 - Nimmt teil an Firmenereignissen teil (z.B. Abteilungstreffen)
 - Bildet sich weiter
 - Hat Urlaub
 - Ist krank

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Kostenfaktoren:

- Personalkosten sind Hauptkosten bei SW-Entwicklung !
- Weitere Kosten:
 - Rechner, Tools
 - Zielplattform
 - Projekt-spezifische Weiterbildung
- **Vorsicht:**
 - Kosten unterschiedlich für verschiedene MitarbeiterInnen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Personalzuordnung
- ◆ Reihenfolge der AP
- ◆ Dauer des Projektes

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- ▶ 5.5. Planung/Kontrolle

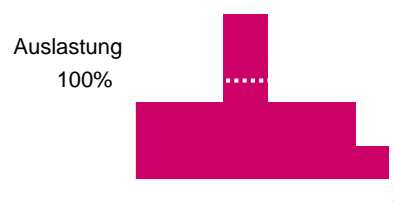
4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

1. Zuordnung von MitarbeiterInnen zu AP
 - Wer arbeitet
 - Mit welchem Anteil an Gesamtarbeitszeit
2. Reihenfolge-/Dauerplanung
3. Optimierung der Personaleinsatzplanung
 - Auslastung des Projektteams gleichmäßig
 - Auslastung einzelner Mitarbeiter gleichmäßig

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- ▶ 5.5. Planung/Kontrolle



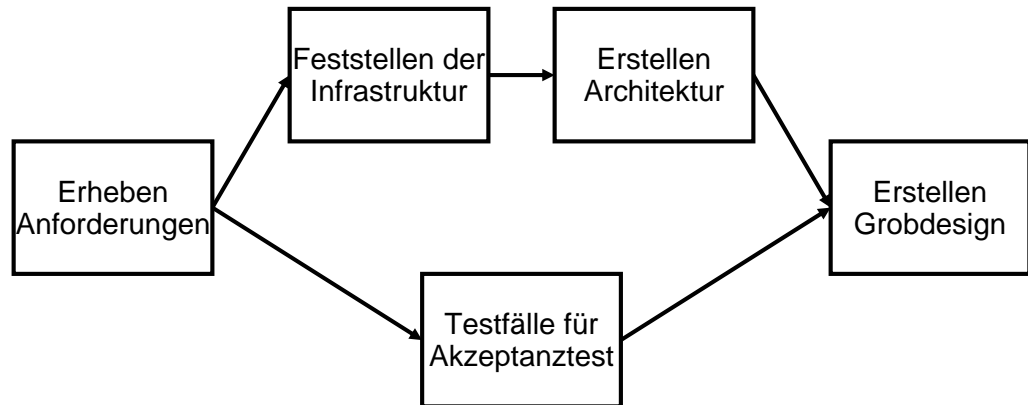
4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Beschreibung der **Abhängigkeit der Arbeitspakete**
- ◆ **Terminrechnung** durch Angabe von frühester / Spätester Starttermin (**FST / SST**), Frühester / spätester Endtermin (**FET / SET**) und Dauer (**D**)



4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **AP1-AP2: Anfang-Anfang:** AP1 in (AA2)
 - AP2 darf erst beginnen, nachdem AP1 begonnen wurde
- ◆ **AP1-AP2: Ende-Anfang:** AP1 in (EA2)
 - AP2 darf erst beginnen, nachdem AP1 beendet wurde
- ◆ **AP1-AP2: Anfang-Ende:** AP1 in (AE2)
 - AP2 darf erst enden, nachdem AP1 begonnen wurde
- ◆ **AP1-AP2: Ende-Ende:** AP1 in (EE2)
 - AP2 darf erst enden, nachdem AP1 beendet wurde
- ◆ Kann mit **Zeitabstand** konkretisiert werden
 - Z.B. $ZEA_{ij} = 1$, $ZAA_{ij} = -4$

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Ablaufplanung ermöglicht
 - **Vorwärtsrechnung:** Wann ist der früheste Starttermin / Endetermin eines APj?

$$FST_j = \text{Maximum von } (FET_i + ZE_{Aij}: AP_i \text{ in } (EA_j)) \text{ und } (FST_i + ZA_{Aij}: AP_i \text{ in } (AA_j))$$

$$FET_j = \text{Maximum von } (FST_j + D_j) \text{ und } (FST_i + ZE_{Eij}: AP_i \text{ in } (AE_j)) \text{ und } (FET_i + ZEE_{ij}: AP_i \text{ in } (EE_j))$$
 - **Rückwärtsrechnung:** Wann ist der späteste Starttermin / Endetermin eines APj?

$$SET_j = \text{Minimum von } (SST_i - ZE_{Aij}: AP_j \text{ in } (EA_i)) \text{ und } (SET_i - ZEE_{ij}: AP_j \text{ in } (EE_i))$$

$$SST_j = \text{Minimum von } (SET_j - D_j) \text{ und } (SET_i - ZAE_{ij}: AP_j \text{ in } (AE_i)) \text{ und } (SST_i - ZAA_{ij}: AP_j \text{ in } (AA_i))$$

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Projektmanagement ist **Balance zwischen Verwaltung und Motivation**
- ◆ **Verwaltung:** Aufbau- und Ablauforganisation, Projektstrukturplan (Arbeitspakete), Netzplan (Aufwand, Kosten, Termine)
- ◆ **Motivation:** Qualifikation, Inhalt, Spaßfaktor, Teambildung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

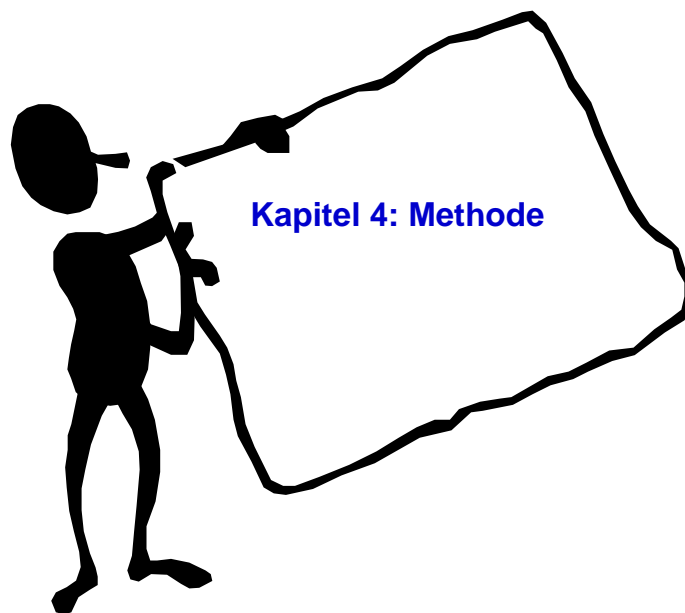
- ◆ Project management institute, PMBOK-A guide to the project management body of knowledge, 2000
- ◆ DIN 69901
- ◆ F. Weltz and R.G. Ortman. *Das Softwareprojekt - Projektmanagement in der Praxis*, Campus, 1992
- ◆ Karol Fruehauf, Jochen Ludewig, Helmut Sandmayr: *Software-Projektmanagement und – Qualitätssicherung*, VDF Verlag 2002
- ◆ Ian Sommerville, *Software Engineering*, Addison Wesley, 2003

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



▶ 4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ 4.1. Allgemeines
- ◆ 4.2. Vorgehen Requirements Engineering
- ◆ 4.3. Vorgehen Architektur
- ◆ 4.4. Vorgehen Entwurf
- ◆ 4.5. Vorgehen Testen

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4. Methode

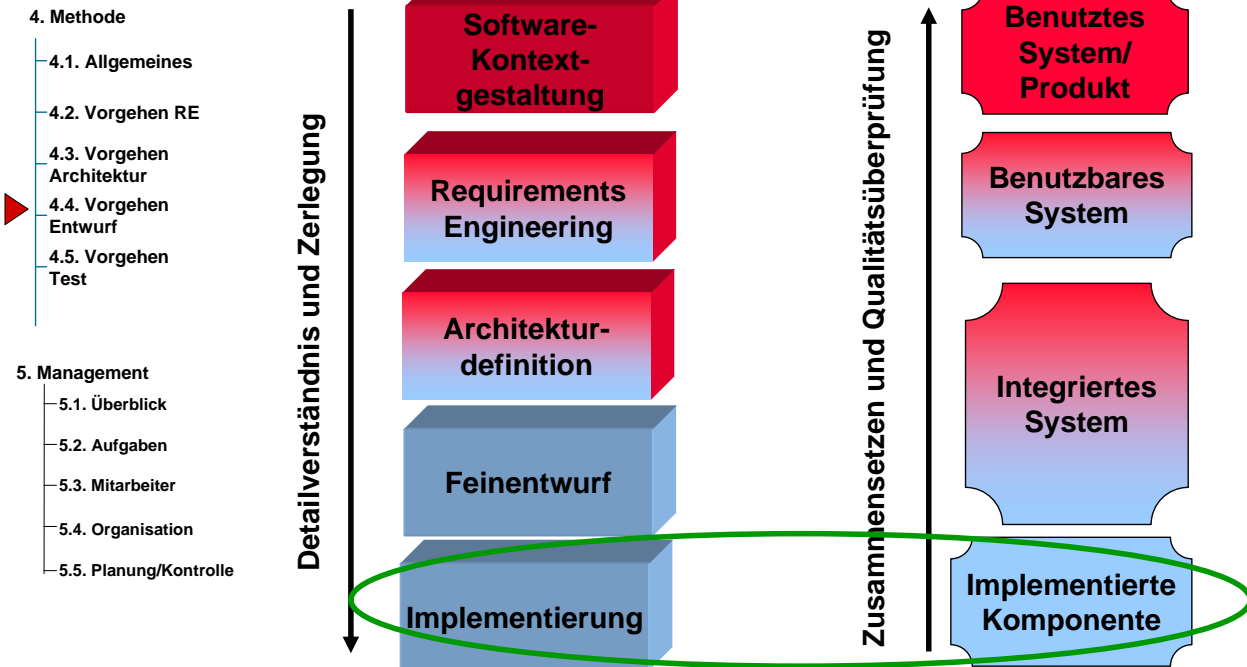
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ 4.4.1. Überblick
- ◆ 4.4.2. Von Use Cases zu Analyseklassen
- ◆ 4.4.3. Verfeinerung der Analyseklassen
- ◆ 4.4.4. Feinentwurf und Implementierung (Folien z.T. von Heinrich Hussmann und Bernhard Rumpe)

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

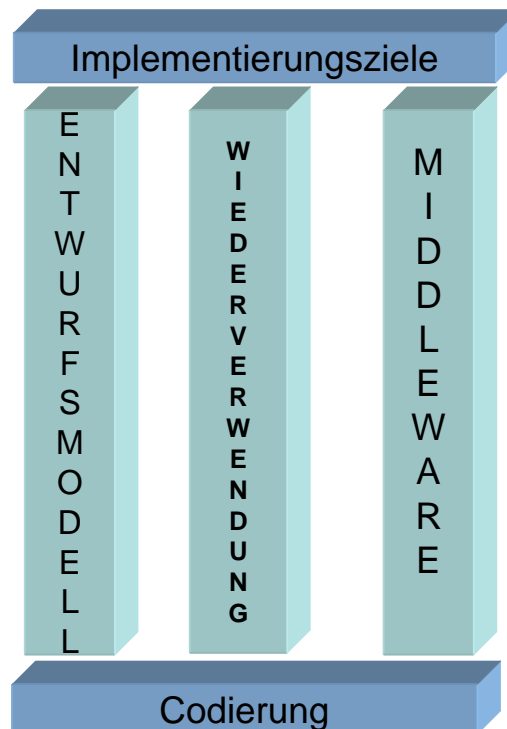
1.3.2. Aktivitäten und Ergebnisse der Entwicklung



Folie 51

4.4.4. Welche Inhalte umfasst der Feinentwurf und die Implementierung?

- 4. Methode**
- 4.1. Allgemeines
 - 4.2. Vorgehen RE
 - 4.3. Vorgehen Architektur
 - 4.4. Vorgehen Entwurf
 - 4.5. Vorgehen Test
- 5. Management**
- 5.1. Überblick
 - 5.2. Aufgaben
 - 5.3. Mitarbeiter
 - 5.4. Organisation
 - 5.5. Planung/Kontrolle



Folie 52

4.4.4. Implementierungsziele

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur

4.4. Vorgehen
Entwurf

4.5. Vorgehen
Test

◆ Legen fest

- Programmiersprache (4.4.4.1.)
- **Wiederverwendung / Weiterentwicklung** durch (4.4.4.3.)
 - Frameworks
 - Klassenbibliotheken
 - Komponenten
- Verwendete **Middleware** für
 - Datenbankbindung
 - Workflow
 - Verteilung

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

◆ Konkretisieren Architekturdefinition

4.4.4. Entwurfsmodell

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur

4.4. Vorgehen
Entwurf

4.5. Vorgehen
Test

◆ Verfeinert die Modelle aus der Analysephase (4.4.4.2)

- Aufteilung auf Komponenten: je ein Modell pro Komponente
- Detailspezifikation der Klassen: Vervollständigung bzgl. Operationen, Datentypen, Sichtbarkeit, Invarianten
- Detailspezifikation der externen Schnittstellen

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

◆ Verwendet Entwurfsmuster

4.4.4. Wiederverwendung / Weiterentwicklung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Definition eines Konzeptes bzgl. Wiederverwendung und Weiterentwicklung

- Welche **existierenden** Codeteile sollen verwendet werden?
 - Klassenbibliotheken
 - Komponenten
 - Framework
- Welche zu entwickelnden Codeteile sollen **zukünftig wiederverwendet** werden?
 - Klassenbibliotheken
 - Komponenten
 - Framework

4.4.4. Middleware

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Stellt Standardlösungen für Technologiefragen zur Verfügung (Implementierung der „Pfeile“ in der Architektur)
 - Datenpersistenz durch Anbindung an eine Datenbank
 - Workflow
 - Verteilung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Implementierung der Entwurfsmodelle
- ◆ Sollte **Codierungs-Richtlinien** berücksichtigen

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Sprachparadigma
- ◆ Welche technische Umgebung ist gegeben?
 - Legacy-System? Gibt es eine Vorgabe des Unternehmens?
- ◆ Welche Bibliotheksfunktionen werden benötigt?
- ◆ Wie gut ist die Werkzeugunterstützung?
 - Compiler, Debugger, IDE, ...
- ◆ Welche Kenntnisse/Vorlieben besitzen die Programmierer?

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



4.4.4.1. Funktionale Programmiersprachen

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ Beispiele: Gofer, Haskell, ML
- ◆ Seiteneffektfrei und dadurch leicht verstehbar
- ◆ Sehr mächtiges Typsystem
- ◆ Patternmatching auf Argumenten
- ◆ Effiziente Definition von Datenstrukturen und Funktionen
 - `data Tree = Leaf(Int) | Node(Tree, Int, Tree)`
- ◆ Funktionen höherer Ordnung (Funktionen auf Funktionen anwendbar)
 - `twice f x = f(f(x))`
- ◆ Kompakte Formulierung
- ◆ **Fazit:**
 - Effektive Programmierung aber langsamere Ausführungszeiten.
 - Geeignet für schnelle Hacks, skaliert nicht für große Programmsysteme
 - Schwierigkeiten mit interaktiven Systemen (z.B. GUI) umzugehen



4.4.4.1. Prozedurale Programmiersprachen

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ Beispiele: Modula-2, Pascal, C, Fortran
- ◆ Ideen zum Modul-Konzept teilweise vorhanden
- ◆ Komfortable Definition von Datenstrukturen
- ◆ Trennung von Datenstruktur und Funktionen macht Wartbarkeit schwieriger
- ◆ **Fazit:**
 - Für kleinere und mittlere Systeme geeignet
 - Prozedurale Programmierung ist in der objektorientierten Programmierung subsumiert und wird deshalb kaum mehr in Reinform verwendet

4.4.4.1. Objektorientierte Programmierung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Beispiel: Java, C++, Oberon, Modula-3, Smalltalk, C#, ...
- ◆ Die Merkmale der OO:
 - Objekte (Klassen) als Kapseln von Daten und Funktionen
 - Objekte dynamisch erzeugen: Objektidentität
 - Vererbung (in ihren verschiedenen Ausprägungen)
- ◆ OO Sprachen subsumieren prozedurale Programmierung
- ◆ Es erfordert aber eine wesentlich andere Vorgehensweise, um Vorteile der OO zu nutzen:
 - Vererbung als Mechanismus zur Anpassung und Verbesserung der Wiederverwendung
 - „Gutes Design“, um Wartbarkeit und Erweiterbarkeit zu stützen
 - Codingstandards für Lesbarkeit
- ◆ **Fazit:**
 - OO ist heute das Mittel der Wahl für große Projekte, effizienter geht es aber oft mit Spezialsprachen

4.4.4.1. Spezialsprachen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Logikprogramme:** Prolog
 - Logische Aussagen in Hornklauselform als Programm
- ◆ **Visuelle Programmierung**
 - a) Komposition des Programms aus Bausteinen
 - b) Modellierung z.B. mit ausführbaren Statecharts
- ◆ **Programmiersprachen mit inhärentem Verteilungskonzept**
 - für massiv verteilte Systeme
- ◆ **Skriptsprachen**
 - Beispiel: Python, Tcl/Tk, Perl
 - Meist kein festes Typsystem
 - effektive Module zur Textbearbeitung (z.B. reguläre Ausdrücke)
 - Neu: immer bessere Integration mit anderen Sprachen (z.B. Python + Java)
- ◆ Weitere Spezialsprachen: HTML, JSP, XML, SQL, ...

4.4.4.1. Fazit: Programmiersprachen

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur

4.4. Vorgehen
Entwurf

4.5. Vorgehen
Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ Die **Kooperationsfähigkeit** der Sprachen erlauben die Anwendung der jeweils besten Sprache:
 - Corba, .NET, Embedded SQL (z.B. in Java)
 - Java Server Pages (JSP) = Java + HTML
 - Modul-Import über API's (Python in Java)
- ◆ Fehlende Spracheigenschaften werden durch **standardisierte Klassenbibliotheken** abgedeckt:
 - I/O wurde in C als erstes durch Bibliotheksfunktionen standardisiert
 - Threads/Nebenläufigkeit wird in Java über die Bibliothek realisiert
 - Kommunikation, Datenspeicherung wird nicht über Sprachprimitive, sondern Bibliotheksfunktionen angeboten
 - Security (z.B. Java Sandbox oder Verschlüsselung)

4.4.3. Entwurfs- vs. Analysemodell

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur

4.4. Vorgehen
Entwurf

4.5. Vorgehen
Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

| Analyse-Modell | Entwurfs-Modell |
|--|--|
| <p>Objekte: Fachgegenstände</p> <p>Klassen: Fachbegriffe</p> <p>Vererbung: Begriffsstruktur</p> <p>Annahme perfekter Technologie</p> <p>Funktionale Essenz</p> <p>Meist projektspezifisch</p> <p>Grobe Strukturskizze</p> | <p>Objekte: Softwareeinheiten</p> <p>Klassen: Schemata</p> <p>Vererbung: Programmableitung</p> <p>Erfüllung konkreter Implementierungsziele</p> <p>Gesamtstruktur des Systems</p> <p>Ähnlichkeiten zwischen verwandten Projekten</p> <p>Genaue Strukturdefinition</p> |
| | Mehr Struktur & mehr Details |

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Erweiterung des fachlichen Kerns: Mehr Details als im Analysemodell

- Vollständige Listen der Attribute und Operationen
- Festlegung von Datentypen, Sichtbarkeit
- Spezifikation der Operationen (z.B. Vor- und Nachbedingungen)
- Assoziationen/Aggregationen: Detaillierung, z.B. ordered
- Keine Mehrfachvererbung

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Zusätzliche Klassen/Pakete:

- Einbindung in Infrastruktur, Altsysteme etc.
- Anpassungs- und Entkopplungsschichten für gewählte Technologien (z.B. Datenzugriffsschicht, CORBA-Schnittstellen, XML-Anschluss...)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Beispiele (Klasse Besprechungsraum):

+ freienRaumSuchen
(plaetze: int, start: Date, dauer: int=60, wunschraum:
Besprechungsraum): _____ Besprechungsraum

– istFrei(beginn: Date, dauer: int):boolean

+ reservieren (für: Termin):boolean;

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4.4.4.2. Reduktion von komplexen Multiplizitäten

4. Methode

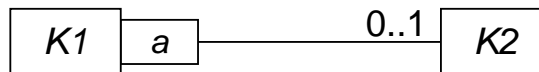
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Verwendung von Qualifikation

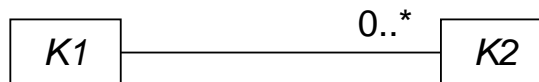
- ◆ **Definition:** Eine **Qualifikation (Qualifier)** ist ein **Attribut a** für eine Assoziation zwischen Klassen K1 und K2, durch das die Menge der zu einem K1-Objekt assoziierten K2-Objekte *partitioniert* wird.

Zweck der Qualifikation ist direkter Zugriff unter Vermeidung von Suche.

◆ Notation:



als Detaillierung von:



Folie 67

Barbara Paech

Vorlesung - Software Engineering - SS 2005

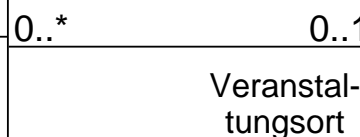
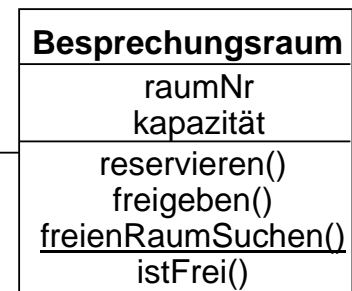
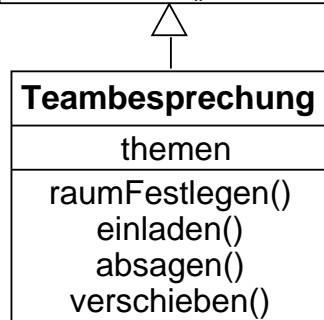
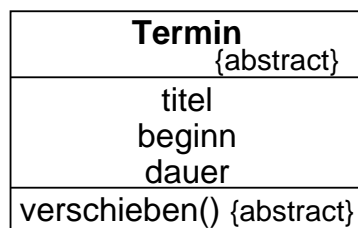
5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4.4.4.2. Qualifizierte Assoziation: Beispiel (1)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Raum12.istFrei(start=04.05.02 10:00, dauer=60min);

führt zu einer Suche über alle assoziierten Teambesprechungen !

Folie 68

Barbara Paech

Vorlesung - Software Engineering - SS 2005

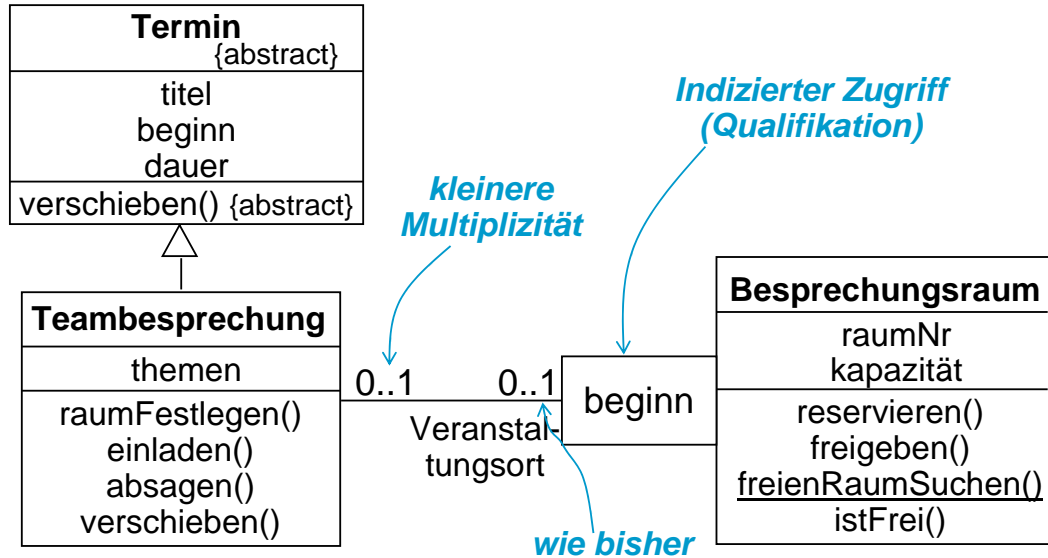
4.4.4.2. Qualifizierte Assoziation: Beispiel (2)

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Raum12.istFrei(start=04.05.02 10:00, dauer=60min);
kann direkt nach Datum abfragen, ob eine Assoziation besteht

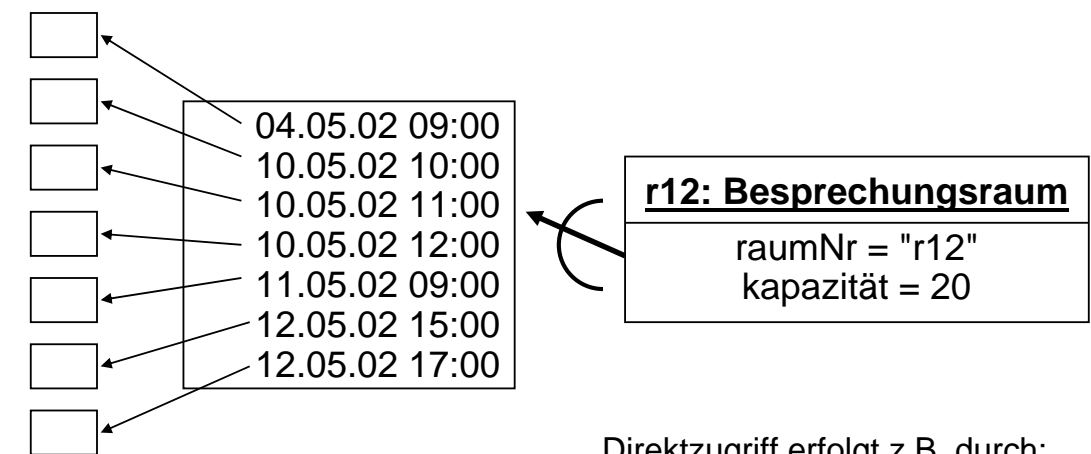
4.4.4.2. Realisierung einer qualifizierten Assoziation

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Teambesprechungs-Objekte

Direktzugriff erfolgt z.B. durch:

Hashfunktion
(Berechnung des Indexwerts aus gegebenem Datum)

Sortierte Baumstruktur

4.4.4.2. Konkretisierung von Datenstrukturen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

◆ Z.B: Festlegung einer Ordnung

- Es besteht eine feste Reihenfolge, in der die assoziierten Objekte durchlaufen werden können → Oft ist Zugriff über Listen, Iteratoren möglich
- Mehrfachvorkommen eines Objekts sind verboten

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



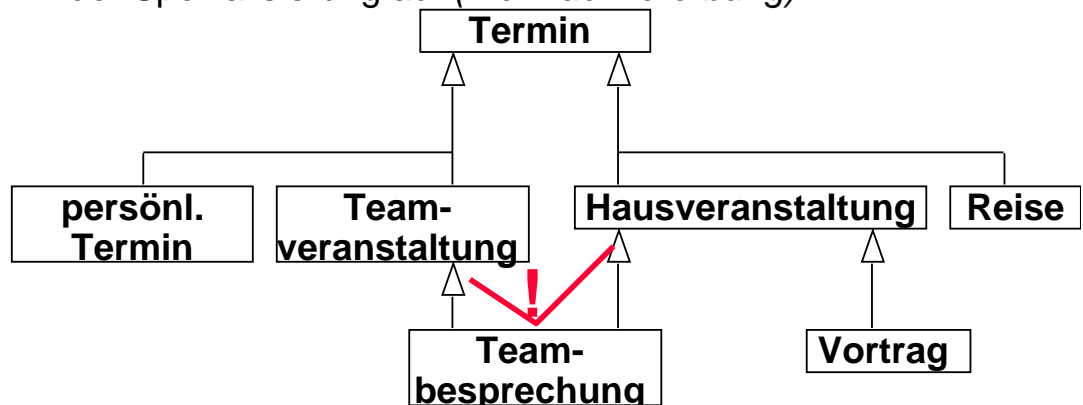
Folie 71

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4.2. Auflösung von Mehrfachvererbung

- ◆ In Analysemodellen treten oft unabhängige Dimensionen der Spezialisierung auf (*Mehrfachvererbung*).



- In Entwurfsmodellen sollten solche Mehrfachvererbungen beseitigt werden. Techniken dazu sind:

- Ersatz von Vererbung durch Komposition
- Definition von Schnittstellen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Folie 72

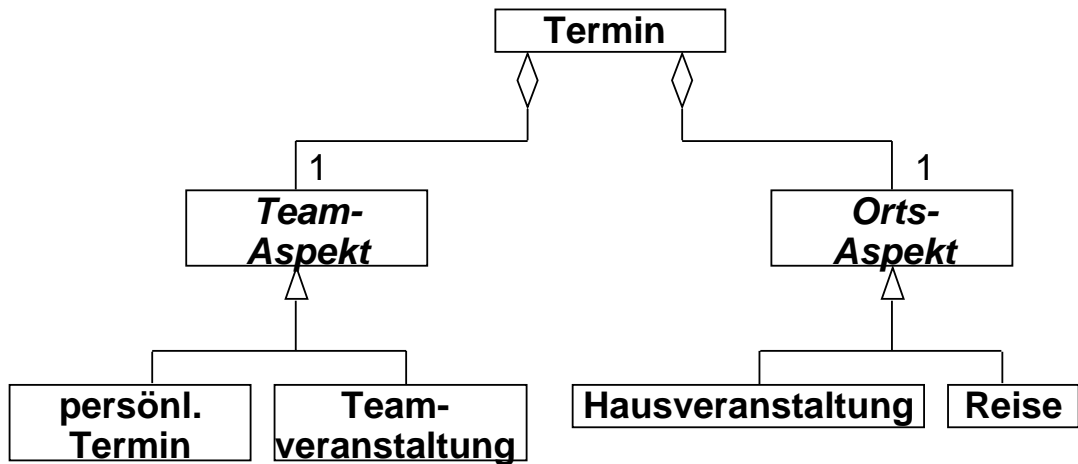
Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4.2. Ersatz von Vererbung durch Komposition

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test



5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

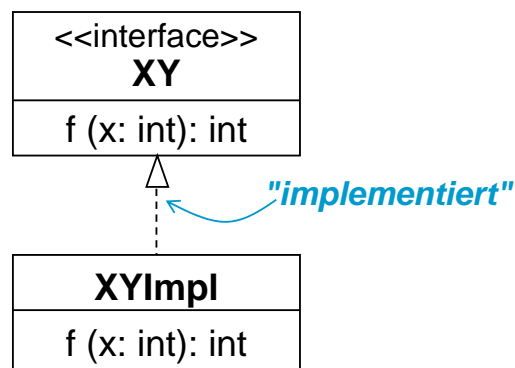
4.4.4.2. Reduktion von Komplexität durch Schnittstellen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Definition: Eine **Schnittstelle** ist eine abstrakte Klasse, die keine Attribute und keine Operationsrümpfe (Implementierungen) enthält.
 - **Sammlung von Operations-Signaturen**

UML:



5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

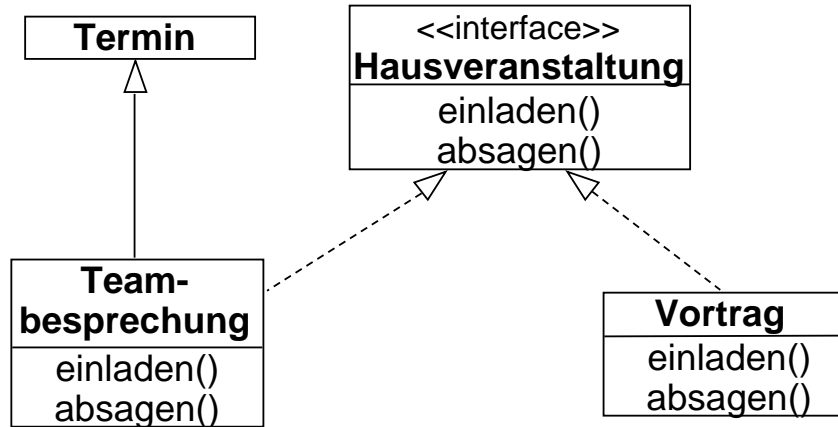
4.4.4.2. Einfache Vererbung durch Schnittstellen

4. Methode

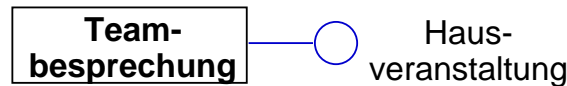
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



Hinweis: "Lutscher"-Notation (*lollipop*) für Schnittstellen ist weit verbreitet und in UML ebenfalls zulässig (und gleichwertig):



4.4.4.2. Schnittstellen und abstrakte Klassen

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

Abstrakte Klasse

Enthält Attribute und Operationen
 Kann Default-Verhalten festlegen
 Default-Verhalten kann in Unterklassen redefiniert werden

Schnittstelle

- Enthält nur Operationen (und ggf. Konstante)
- Kann kein Default-Verhalten festlegen
- Unterklassen müssen Verhalten definieren
- Schnittstelle ist eine spezielle Sicht auf eine Klasse

4.4.4.3. Wiederverwendung / Weiterentwicklung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

- ◆ Bausteinsysteme (*componentware*)
- ◆ Frameworks
- ◆ Architekturmuster und Referenzarchitekturen
- ◆ Entwurfsmuster
- ◆ Klassenbibliotheken

Abnehmende Festlegung der Architektur

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

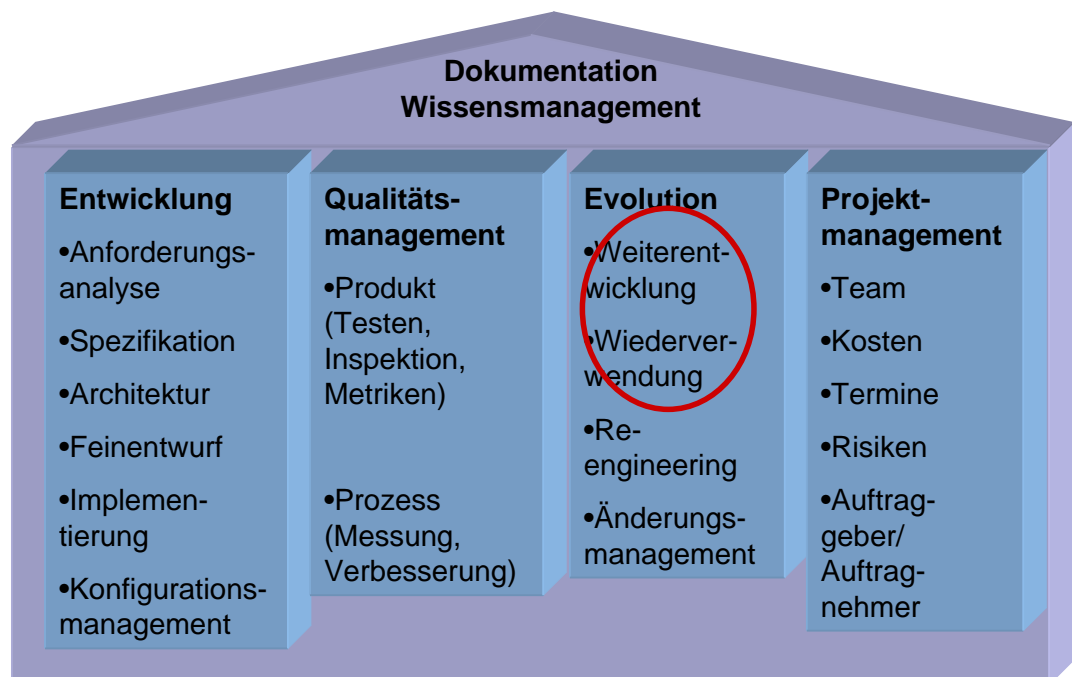
1.3.1. Aufgabenbereiche des Engineering

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



4.4.4.3. Wiederverwendung vs. Neuentwicklung

4. Methode

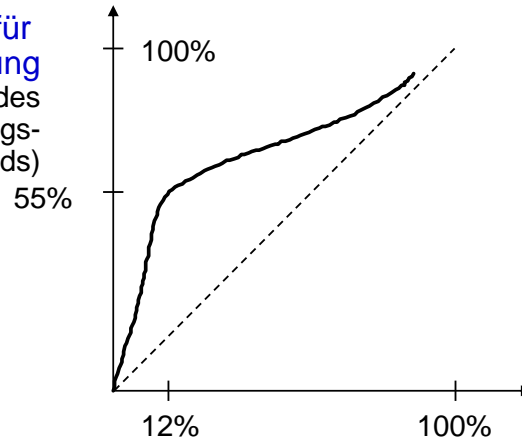
- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Typische Programmiereransicht:
"Wiederverwendung kostet genauso viel Aufwand wie neu schreiben."
- ◆ Diese Ansicht ist **fast richtig, wenn nicht für Wiederverwendung entworfen wird**

Aufwand für Wiederverwendung (als Anteil des Neuentwicklungsaufwands)



Quelle: NASA-Studie 1994

Umfang der Anpassungen (an wiederverwendeter Komponente)

Folie 79

4.4.4.3. Frameworks

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Definition** (nach Pomberger/Blaschek): Ein **"framework"** (Rahmenwerk, Anwendungsgerüst) ist eine Menge von zusammengehörigen Klassen, die einen abstrakten Entwurf für eine Problemfamilie darstellen.
- ◆ **Ziele:**
 - Wiederverwendung von Code, Architektur und Entwurfsprinzipien
 - Wiederverwendung von Verhaltensschemata einer Gruppe von Klassen
 - Homogenität unter verschiedenen speziellen Anwendungssystemen für eine Problemfamilie (z.B. ähnliche, ergonomische Bedienung)

Folie 80

4.4.4.3. Vergleich Klassenbibliothek-Framework

Klassenbibliothek:

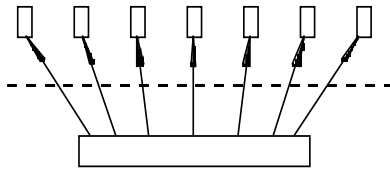
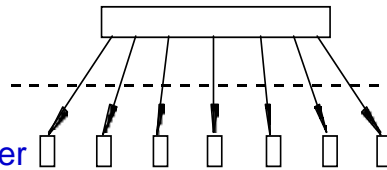
Framework:

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

Eigener Code:

Wieder-verwendeter Code:



"Don't call us, we call you"

(„Hollywood-Prinzip“)

Anpassung nur durch Instanziierung

Anpassung in Subklassen

Ablaufsteuerung nicht vordefiniert

Ablaufsteuerung im Wesentlichen vordefiniert (Übergabe von aufzurufenden Objekten)

Die Grenze ist fließend, siehe z.B. Java AWT !

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4.4.4.3. Klassifikation nach Einsatzart

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Bereichsspezifisches Framework (Domain Framework):**
 - Beinhalten Expertenwissen zu speziellem Anwendungsbereich
 - z.B. Framework für Anlagensteuerungen
 - z.B. Framework für betriebswirtschaftliche Anwendungen
 - z.B. Multimedia-Framework
- ◆ **Anwendungs-Framework (Application Framework):**
 - Rahmen für eine ganze Anwendung
 - Beinhalten Expertenwissen zur Systemarchitektur, das auf ähnlich geartete Programme anwendbar ist
 - z.B. GUI-Frameworks
 - "micro-framework" = Rahmenlösung für Teilsystem (z.B. Datenbankanbindung)
- ◆ **Infrastruktur-Framework (Support Framework):**
 - Bereitstellung von Systemdiensten
 - z.B. Framework zur Anpassung von Gerätetreibern

4.4.4.3. Klassifikation nach Architektur

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Architektur-getriebenes Framework (*architecture-driven*):

- Anpassung durch Vererbung und Redefinieren
- Komplexe Klassenhierarchien und Muster
- Relativ viel neuer Code zu schreiben
- Anpassung erfordert sehr hohen Einarbeitungsaufwand

◆ Daten-getriebenes Framework (*data-driven*):

- Anpassung durch Objektkonfiguration und Parametereinstellung
- Weitergeleitete Objekte bestimmen Verhalten (z.B. Ereignis-Objekte)
- Relativ einfach anzupassen
- Eingeschränkte Flexibilität

◆ Möglicher und sinnvoller Kompromiss:

- Zwei-Schichten-Architektur

daten-getrieben

architektur-getrieben

4.4.4.3. Vor- und Nachteile von Frameworks

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

◆ Vorteile:

- Weitergabe von Expertenwissen
- Durchdachtes Design führt zu *langfristiger* Aufwandsersparnis
- Wartungsaufwand wird reduziert
- Gute Möglichkeiten für systematische Tests
- Prinzipiell sehr hohe Produktivität möglich
- Erleichtert Integration und Konsistenz verwandter Anforderungen

◆ Nachteile:

- Erstellung von Frameworks aufwändig
- Einarbeitung in Frameworks aufwändig
- Zusätzlicher Dokumentations- und Wartungsaufwand
- Fehlersuche erschwert durch Overhead des Frameworks
- Kombination von verschiedenartigen Frameworks sehr schwierig



4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4.4.4.3. Bausteinorientierte Programmierung (*Component-Ware*)

- ◆ Idealzustand der Softwareentwicklung ist die Konstruktion aus vorgegebenen Bausteinen ("Software-ICs")
- ◆ **Generische Bausteine (*components*)**
 - anpassbar
 - zusammensetzbar ("Verdrahtung")
- ◆ **Einfache Kompositionsmechanismen:**
 - werkzeuggestützt
 - graphisch
- ◆ **Infrastruktur zur Komponenteninteraktion: "*object bus*"**
- ◆ Beispiele:
 - JavaBeans, Enterprise JavaBeans (EJBs)
 - CORBAcomponents, Microsoft COM+, DCOM
 - Web Services
- ◆ **Praxis: Oft projektspezifische Komponenten-Entwicklung**
 - **Wiederverwendung der Infrastruktur, nicht der Komponenten!**

Folie 85

Barbara Paech

Vorlesung - Software Engineering - SS 2005



4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

4.4.4.3. Definition Komponenten

- ◆ **Definition:** „A **software component** is a unit of composition with **contractually specified interfaces** and **explicit context dependencies** only. A software component can be deployed independently and is subject to composition by third parties.“
- ◆ ECOOP 1996, Workshop on Component-oriented Prog. 1997 & Clemens Szyperski

Folie 86

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4.3. Definition Komponenten

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Schnittstellen (*interfaces*):**
 - Explizit definierte Interaktionen mit Komponenten und Umgebung
- ◆ **Kontextabhängigkeiten (*context dependencies*):**
 - benötigte Komponenten-Infrastruktur
 - Systemressourcen
- ◆ **Unabhängige Einsetzbarkeit (*independent deployment*):**
 - Alle Bestandteile enthalten (Archiv-Dateien), als Ganzes eingesetzt
- ◆ **Komposition durch Dritte (*composition by third parties*):**
 - Endbenutzer, Komponenten-Hersteller und Komponenten-Integrator
 - Meist nur als kompilierter Code verfügbar, nicht als Quellcode

4.4.4.3. Sind Komponenten Objekte?

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- ▶ 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ 2 verschiedene Sichten auf Komponenten
 - Als **Modul**, d.h. Code, der in bestimmter Weise strukturiert ist
 - Wichtig für statische Zusammensetzbarkeit
 - Als **Objekt**, d.h. aktive Einheit zu Laufzeit
 - Wichtig für Interaktion zur Laufzeit
- ◆ In jedem Fall: Komponente ist komplexer als einzelnes Objekt

4.4.4.3. Arten von Komponenten

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ Komponenten für eigenständige Anwendungen und Clients:
 - orientiert auf **graphische Benutzungsoberfläche**, „Sichtbar“
 - **Fachlogik** ohne Berücksichtigung von Mehrbenutzer-Betrieb
- ◆ Komponenten für Client/Server-Anwendungen:
 - Komponentenbasierte "Application-Server"-Software (Server-seitig), "Unsichtbar"
 - **Sicherstellung von Eigenschaften für Server-Software:**
 - Performance
 - Sicherheit
 - Mehrbenutzerbetrieb
 - Fließender Übergang zur Middleware

4.4.4.3. Beispiel: JavaBeans

4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen Architektur

4.4. Vorgehen Entwurf

4.5. Vorgehen Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ *JavaBeans* ist das Komponentenmodell von Java (ab Version 1.1).
- ◆ **Definition:** Eine **Bean (Bohne)** ist eine Sammlung von Java-Klassen und anderen Ressourcen, bei der die Klassen bestimmten Konventionen genügen.
Eine Bean exportiert zu ihrer Umgebung:
 - **Eigenschaften (*properties*)**,
d.h. Bestandteile des lokalen Zustands, auf die über get- und set-Methoden zugegriffen werden kann;
 - **Ereignisse (*events*)**,
d.h. Ereignisklassen und "EventListener"-Klassen nach den Konventionen des Ereignismodells von Java/AWT
 - **Methoden (*methods*)**,
d.h. öffentlich bekannte Methoden des Objekts

4.4.4.3. Wann ist eine Java-Klasse eine JavaBean?

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Es gibt keine Wurzelklasse aller JavaBeans und kein spezifisches Interfaces für alle JavaBeans!
- ◆ Eine Klasse wird zur JavaBean(-Klasse) durch Einhaltung bestimmter **Namens-Konventionen**:
 - Konventionen für Eigenschaften (Properties):
 - `public setPropertyName (PropertyType p);`
 - `public PropertyType getPropertyName();`
- ◆ Bean-basierte Entwicklungswerkzeuge unterstützen:
 - Instanzieren von Bean-Objekten
 - Manipulation von Properties
 - Ereignisbehandlung
 - Verwendung von "Meta-Programmierung" (Reflektion)
- ◆ Die graphischen Elemente der Java-GUI-Toolkits AWT und Swing erfüllen die JavaBean-Konventionen.

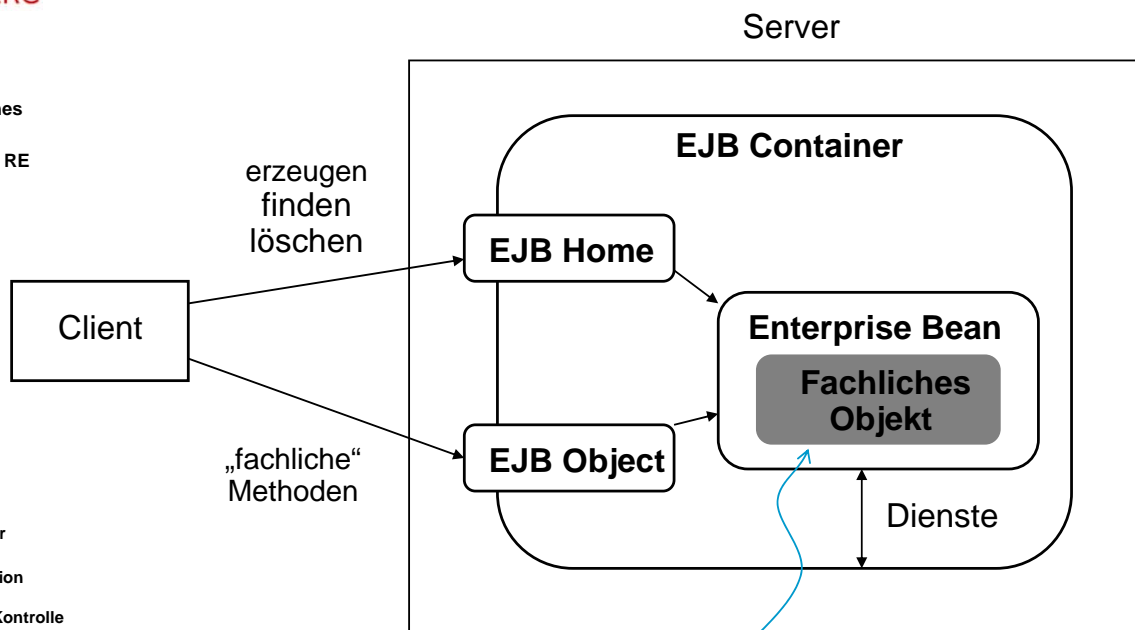
4.4.4.3. Beispiel: EJB

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle



*Eigentliche EJB-Komponente:
Der Rest wird generiert oder ist immer gleich*

4.4.4.3. Eigenschaften EJBs

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Alles außerhalb des „Fachlichen Objekts“ wird entweder fertig bereitgestellt oder automatisch generiert.
- ◆ Jede EJB-Komponente hat ein „Home Interface“, über die Instanzen auf dem Server erzeugt, initialisiert, gelöscht und lokalisiert werden können.
- ◆ Für die Erzeugung und das Finden von Server-seitigen Beans wird das Java Naming and Directory Interface (JNDI), einen standardisierten Verzeichnisdienst, eingesetzt.
- ◆ Das Interface „EJB Object“ ist als notwendige Ergänzung der normalen Fachmethoden einer EJB-Klasse gedacht. Die (wenigen) Operationen hier dienen z.B. zur eindeutigen Identifikation eines Objekts.
- ◆ Das Home- und das Object-Interface, in dem die fachliche Schnittstelle eingebettet ist, werden auf der Client-Seite, über sogenannte „Stubs“ (Entwurfsmuster Proxy) zugänglich gemacht.

Folie 93

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4.3. Arten von Enterprise Java Beans

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ **Entity Bean:**
 - Repräsentiert persistente Information aus einer Datenbank
 - Typischerweise mit mehreren Transaktionen assoziiert
 - Mehrbenutzerzugriff möglich
- ◆ **Session Bean:**
 - EJB-Instanz, die mit einem einzelnen Client assoziiert ist
 - Typischerweise nicht persistent
 - Beispiele:
 - Benutzersitzung innerhalb einer Web-Site
 - Buchungsvorgang
 - „Stateful Session Bean“
 - Dialogzustände innerhalb der Session
 - „Stateless Session Bean“
 - keine Dialogzustände, hohe Effizienz

Folie 94

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4.3. Beispiele: Realisierung von Komponenten

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Eine Komponente ist eine geschlossene Einheit:
 - Möglicherweise viele Klassen
 - Dokumentation
 - Mechanismen zur Anpassung

- ◆ JavaBeans:
 - Bean wird ausgeliefert in Java-Archiv-Datei (JAR-Datei)
 - Metainformation in der JAR-Datei („Manifest“) beschreibt, welche Klassen Beans sind
 - Archiv enthält alle möglicherweise zusätzlich notwendigen Ressourcen
 - Z.B. Daten, Medienobjekte

- ◆ Enterprise JavaBeans:
 - Bean wird ausgeliefert in Java-Archiv-Datei (JAR-Datei)
 - Deployment Descriptor zur Konfiguration der EJB
 - Archiv enthält alle möglicherweise zusätzlich notwendigen Ressourcen
 - z.B. Stubs und Skeletons zur CORBA-Kommunikation

Folie 95

Barbara Paech

Vorlesung - Software Engineering - SS 2005

4.4.4. Zusammenfassung Feinentwurf / Implementierung

4. Methode

- 4.1. Allgemeines
- 4.2. Vorgehen RE
- 4.3. Vorgehen Architektur
- 4.4. Vorgehen Entwurf
- 4.5. Vorgehen Test

5. Management

- 5.1. Überblick
- 5.2. Aufgaben
- 5.3. Mitarbeiter
- 5.4. Organisation
- 5.5. Planung/Kontrolle

- ◆ Für Feinentwurf und Implementierung heute viele **technologiespezifische Erfahrung** (Muster, Codierungs-Standards)
- ◆ Aber immer noch eine große Fehlerquelle
- ◆ **Werkzeuge nutzen**
- ◆ **Feedback sehr wichtig** (Pair programming!)

Folie 96

Barbara Paech

Vorlesung - Software Engineering - SS 2005



4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur4.4. Vorgehen
Entwurf4.5. Vorgehen
Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

- ◆ Gamma/Helm/Johnson/Vlissides: Design Patterns, Addison-Wesley 1994 (= „Gang of Four“, „GoF“)
- ◆ Vlissides: Pattern Hatching, Addison-Wesley 1998
- ◆ Buschmann/Meunier/Rohnert/Sommerlad/Stal: A System of Patterns, Wiley 1996
- ◆ W. Pree: Komponentenbasierte Softwareentwicklung mit Frameworks, dpunkt 1997
- ◆ C. Szyperski: Component Software - Beyond Object-oriented Programming, Addison-Wesley 1997
- ◆ Vermeulen et al.: The Elements of Java Style
- ◆ Scott Ambler: <http://www.ambyssoft.com/javaCodingStandards.html>



4. Methode

4.1. Allgemeines

4.2. Vorgehen RE

4.3. Vorgehen
Architektur4.4. Vorgehen
Entwurf4.5. Vorgehen
Test

5. Management

5.1. Überblick

5.2. Aufgaben

5.3. Mitarbeiter

5.4. Organisation

5.5. Planung/Kontrolle

Mehr über solche neue Technologien,
insbesondere Web Services und
Erstellung mobiler Komponente im
nächsten Semester

SWE IIb: Moderne Anwendungen mit
innovativen Technologien