

Martin Glinz Harald Gall

Software Engineering

Wintersemester 2005/06

Kapitel 15

Software-Projektführung



Universität Zürich
Institut für Informatik

15.1 Projektplanung

15.2 Projektkontrolle und -lenkung

15.3 Projektabschluss

15.4 Software-Risikoführung



Ist Software-Projektführung etwas Besonderes?

Besonderheiten von Software-Projekten?

Software ist **immateriell**

- ⇒ Führung durch **genaues Hinsehen** funktioniert **nicht**
 - ⇒ sorgfältige **Planung**, fortlaufende **Überprüfung**, **Lenkung**
 - ⇒ Beachtung der **Projektrisiken**
- In dieser Vorlesung:
 - Konzentration auf softwarespezifische Fragen
 - Keine Behandlung allgemeiner Probleme der Projektführung

Software-Projektplanung

- Was ist zu planen?
 - Prozessmodell
 - Organisationsstruktur
 - Personal und Personaleinsatz
 - Termine und Kosten
 - Dokumente und Verfahren
- Wie planen?
 - Sach- und zielorientiert
 - Anspruchsvoll, aber realistisch
 - Aufgaben und Ressourcen im Gleichgewicht



Software-Projektorganisation

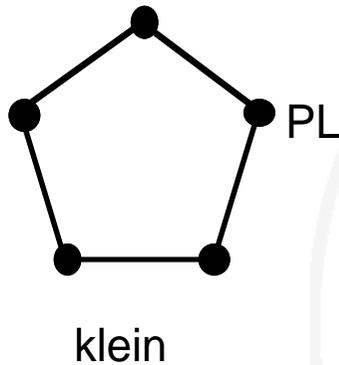
- Grundsätzlich wie bei jedem Projekt
- Besonders beachten
 - Rolle der **Projektleiterin** / des **Projektleiters**
 - **Beziehungen** der **Projektbeteiligten** untereinander

Rolle der Projektleitung

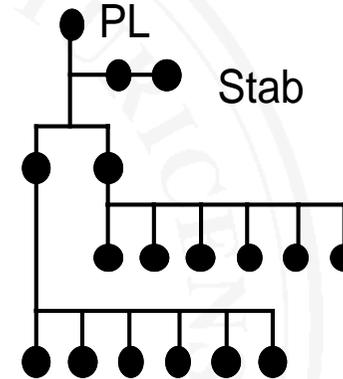
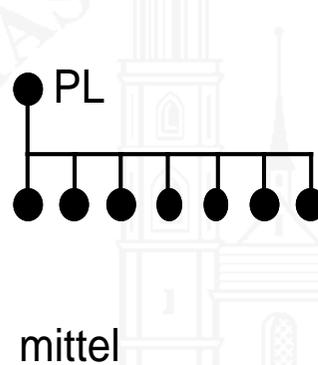
- ProjektleiterIn ist **Schlüsselfigur**
- Führung durch **Zielsetzung**
- **Kompetenzen**, **Verantwortung** und **Ressourcen**
- **eigenverantwortliches Handeln**
- **Berichten** und **informieren**

Beziehungen der Projektbeteiligten untereinander

Demokratisches Team



Hierarchisch organisiertes Team



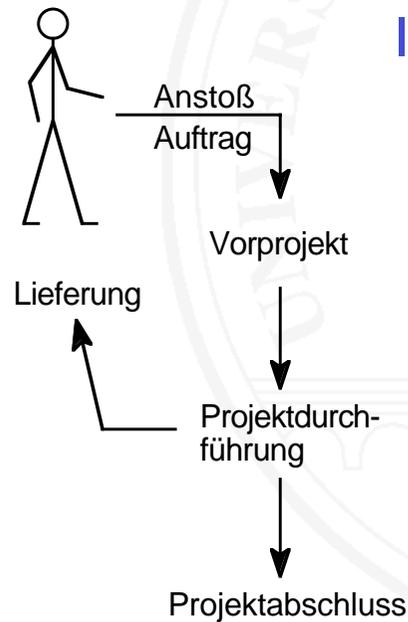
PL: Projektleiter

- Idealfall: pro Person nur ein Projekt gleichzeitig
- sonst: Umschaltverluste

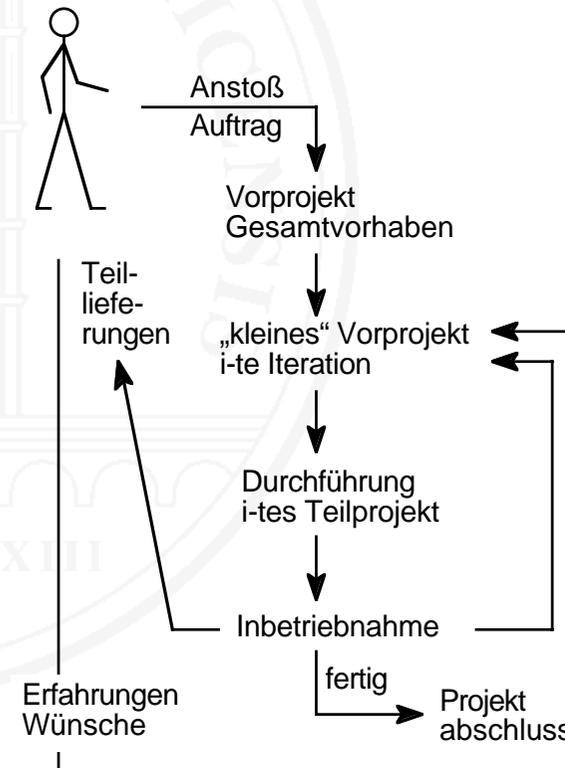
Der Projektstart

- Kritische Phase ⇨ **Vorprojekt**
- Unterschiedlich je nach Projekttyp (intern, extern, Produkt)
- Je nach Prozessmodell eine oder mehrere Planungsphasen

Linear:

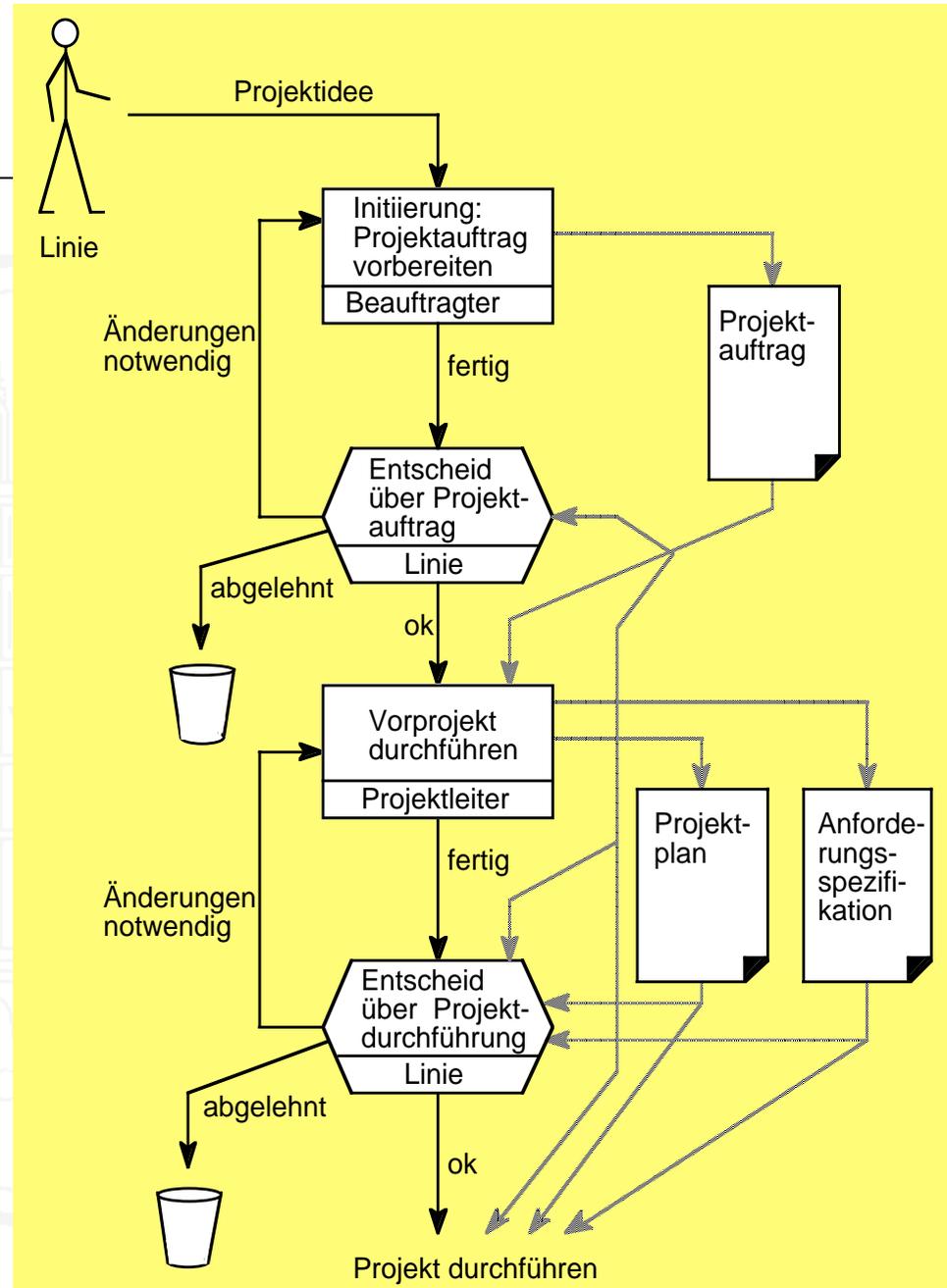


Inkrementell:



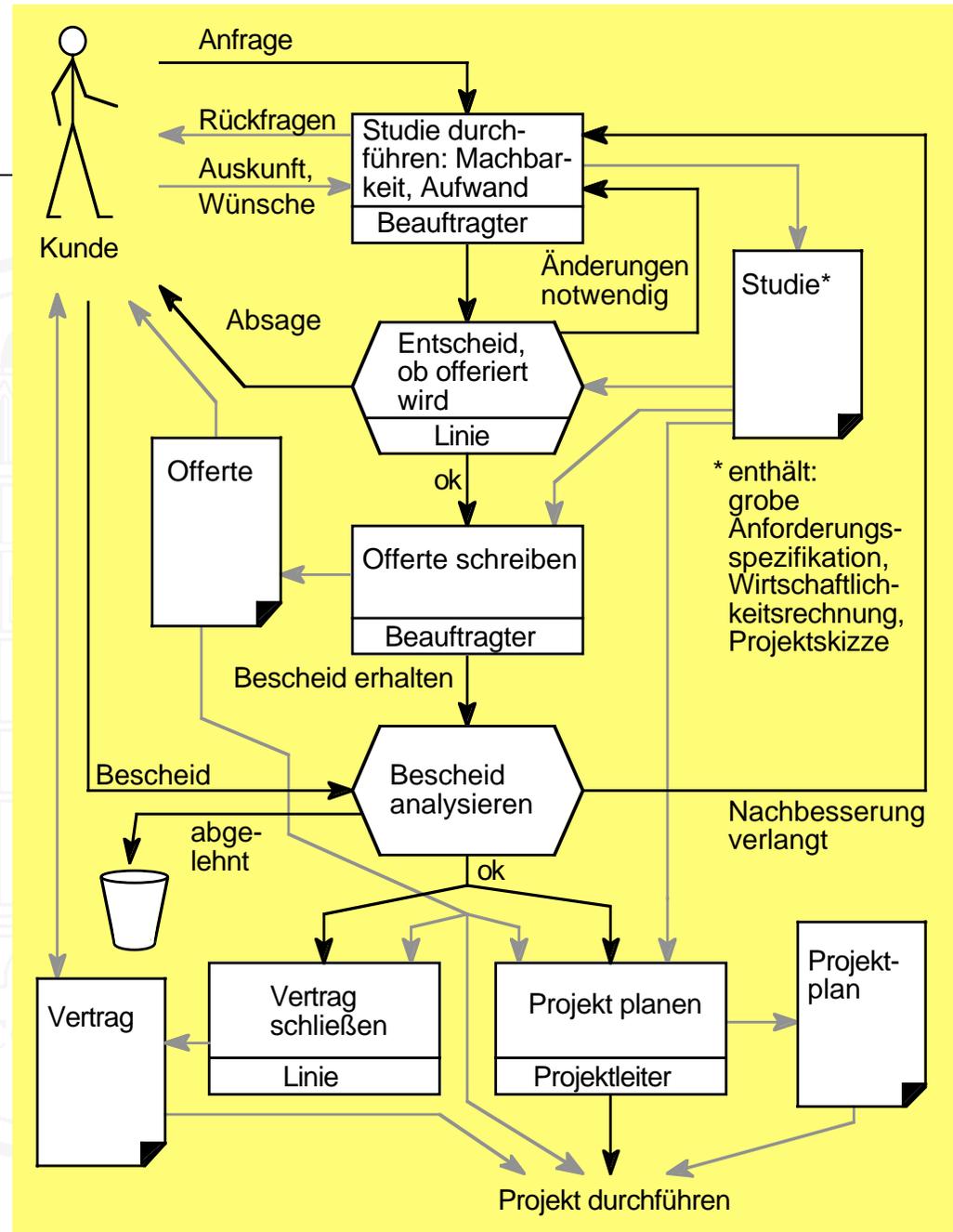
Prozessmuster – 1

Unternehmensinternes Projekt



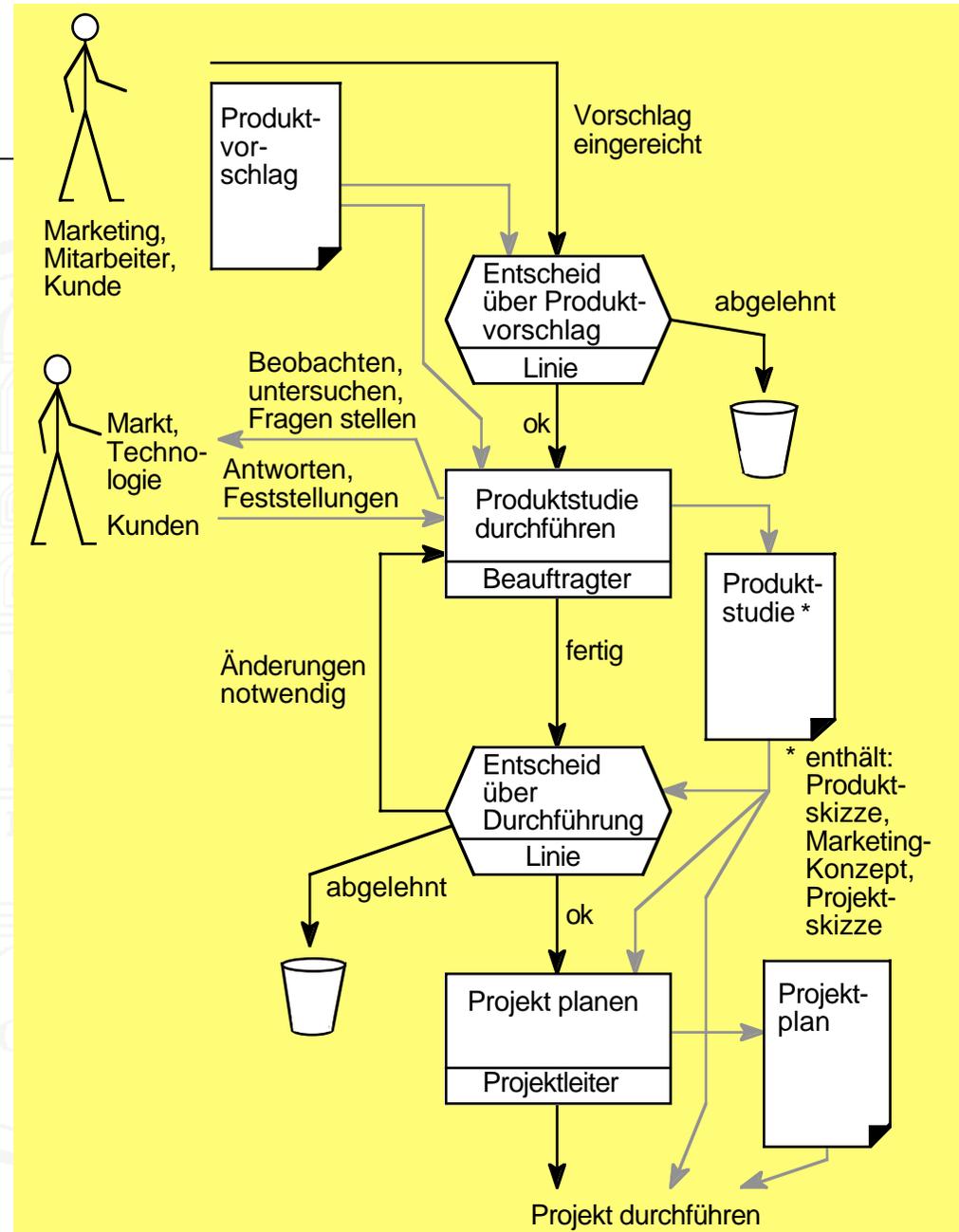
Prozessmuster – 2

Externes Auftragsprojekt



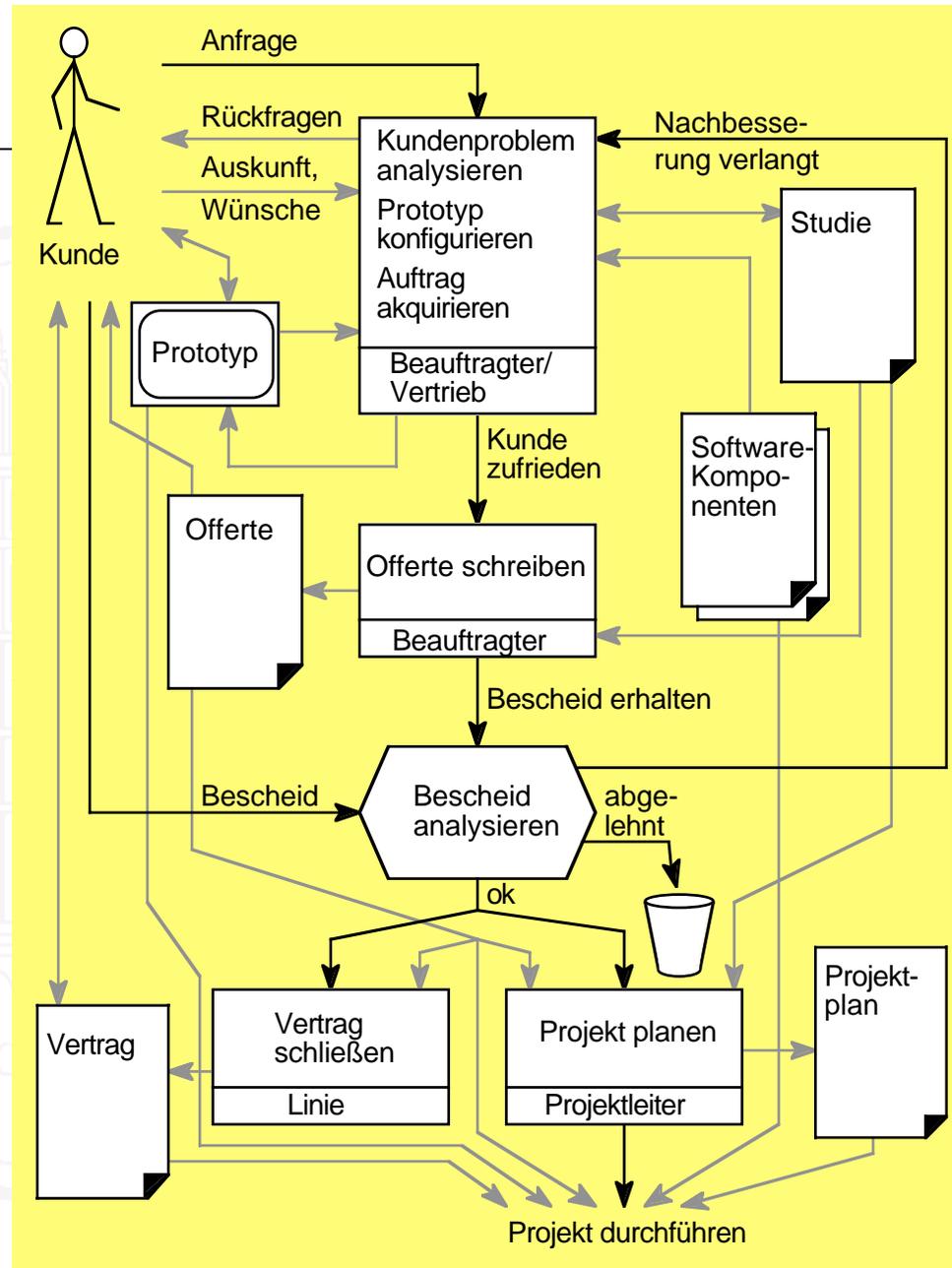
Prozessmuster – 3

Produktentwicklungsprojekt



Prozessmuster – 4

Konfigurationsprojekt

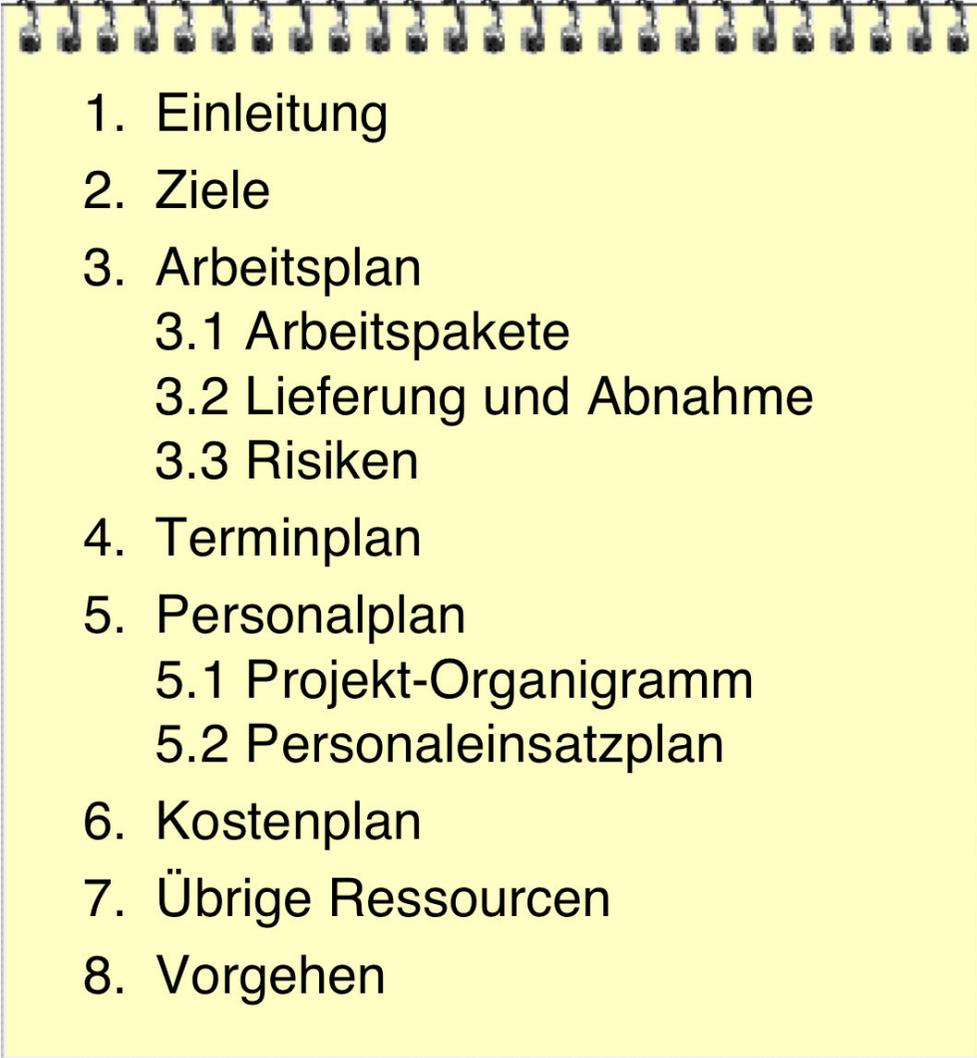


Der Projektplan

- Dokumentiert
 - alle Ergebnisse der **Planung**
 - **Planfortschreibungen**

- Gibt Antwort auf **sechs W-Fragen**:
 - **WARUM**: **Veranlassung** und **Projektziele**
 - **WAS**: die zu liefernden **Resultate** (Produktziele)
 - **WANN**: die geplanten **Termine**
 - **DURCH WEN**: **Personen** und ihre **Verantwortlichkeiten**
 - **WOMIT**: die zur Verfügung stehenden **Mittel** (Geld, Geräte, Software...)
 - **WIE**: die **Vorgehensweise** und die Maßnahmen zur Sicherstellung des Projekterfolgs

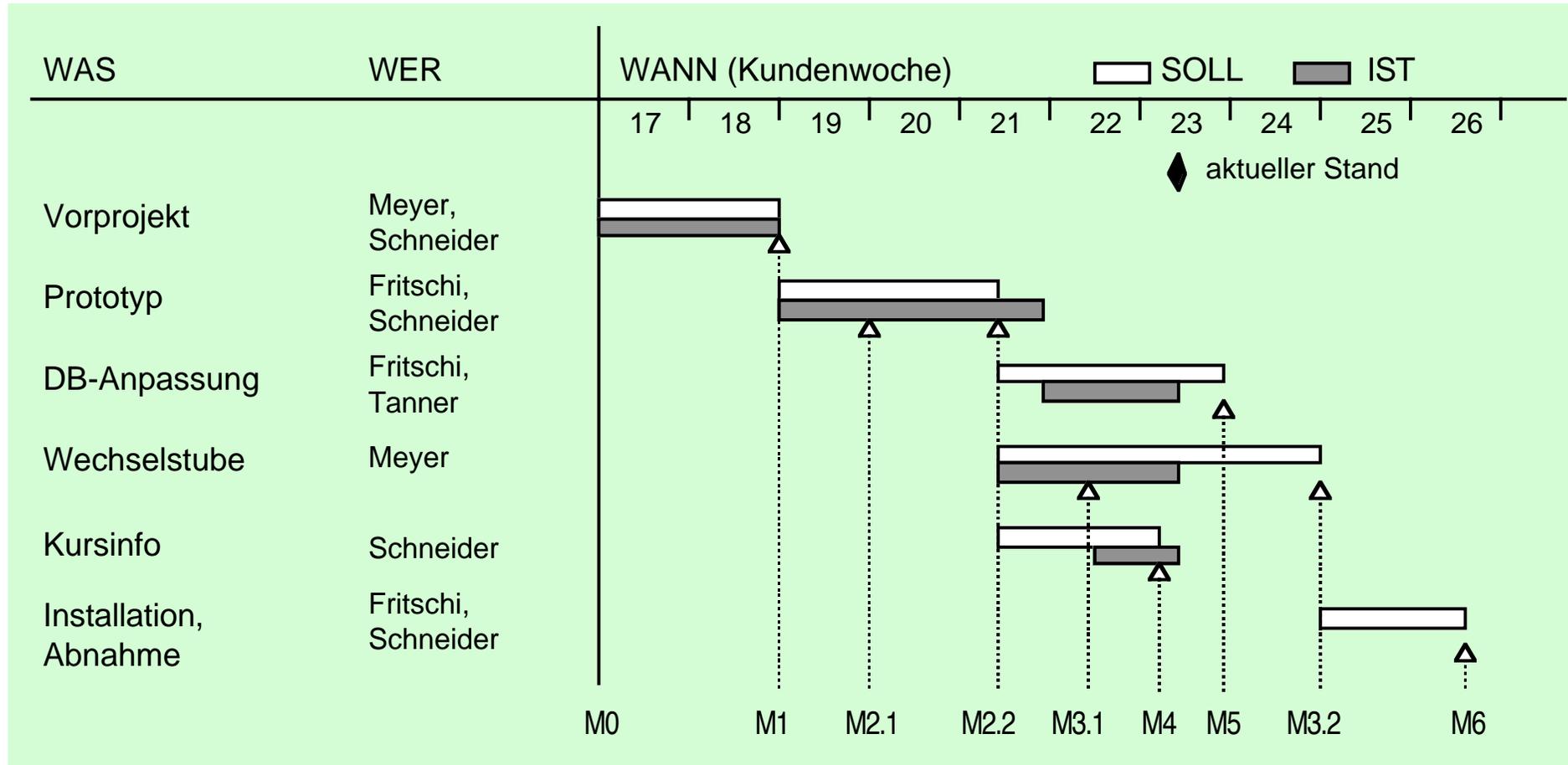
Mögliche Gliederung eines Projektplans

- 
1. Einleitung
 2. Ziele
 3. Arbeitsplan
 - 3.1 Arbeitspakete
 - 3.2 Lieferung und Abnahme
 - 3.3 Risiken
 4. Terminplan
 5. Personalplan
 - 5.1 Projekt-Organigramm
 - 5.2 Personaleinsatzplan
 6. Kostenplan
 7. Übrige Ressourcen
 8. Vorgehen

Planungshilfsmittel

- Terminpläne, Kostenpläne, Arbeitspläne, Personaleinsatzpläne graphisch in Diagrammen
- Immer **SOLL** und **IST**
- Möglichst mit Hilfe von **Werkzeugen**
- Geplante und tatsächliche Aufwendungen genau ermitteln und dokumentieren
 - ⇒ Grundlage für **Schätzungen** in neuen Projekten

Kombinierter Arbeits-/Personaleinsatz-/Terminplan



15.1 Projektplanung

15.2 Projektkontrolle und -lenkung

15.3 Projektabschluss

15.4 Software-Risikoführung



Was und warum

“Plan the flight and fly the plan” (B. Boehm)

- Fortschrittskontrolle ist notwendig
- Ergebnisse müssen messbar sein, sonst droht das 90% Syndrom
- Bei Abweichungen: Lenkung notwendig

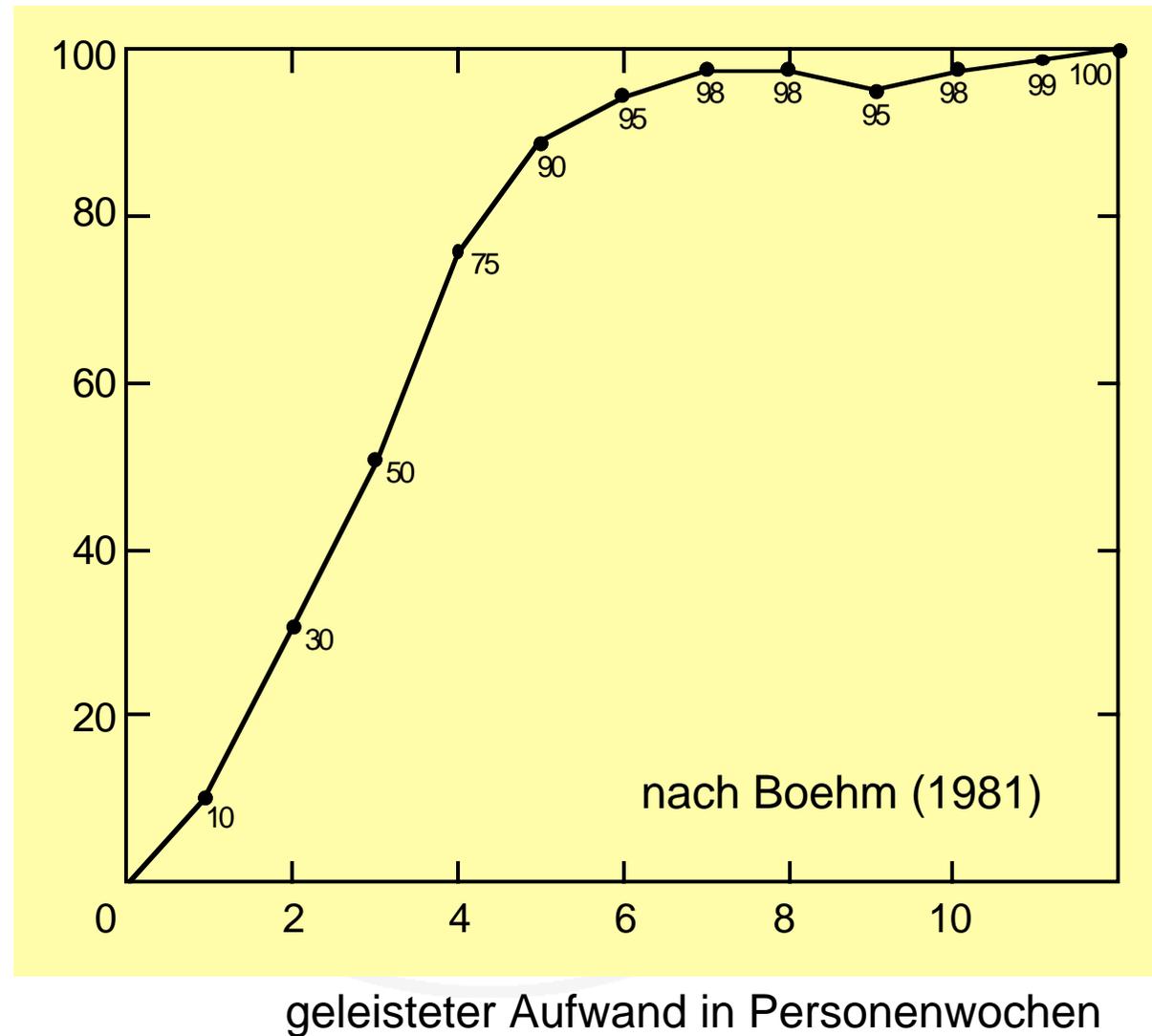
- Terminverfolgung
- Sachzielverfolgung
- Kostenverfolgung
- Risikoverfolgung

Zu 90% fertig...



Das 90%-Syndrom

geschätzter
Fertigstellungs-
grad in Prozent

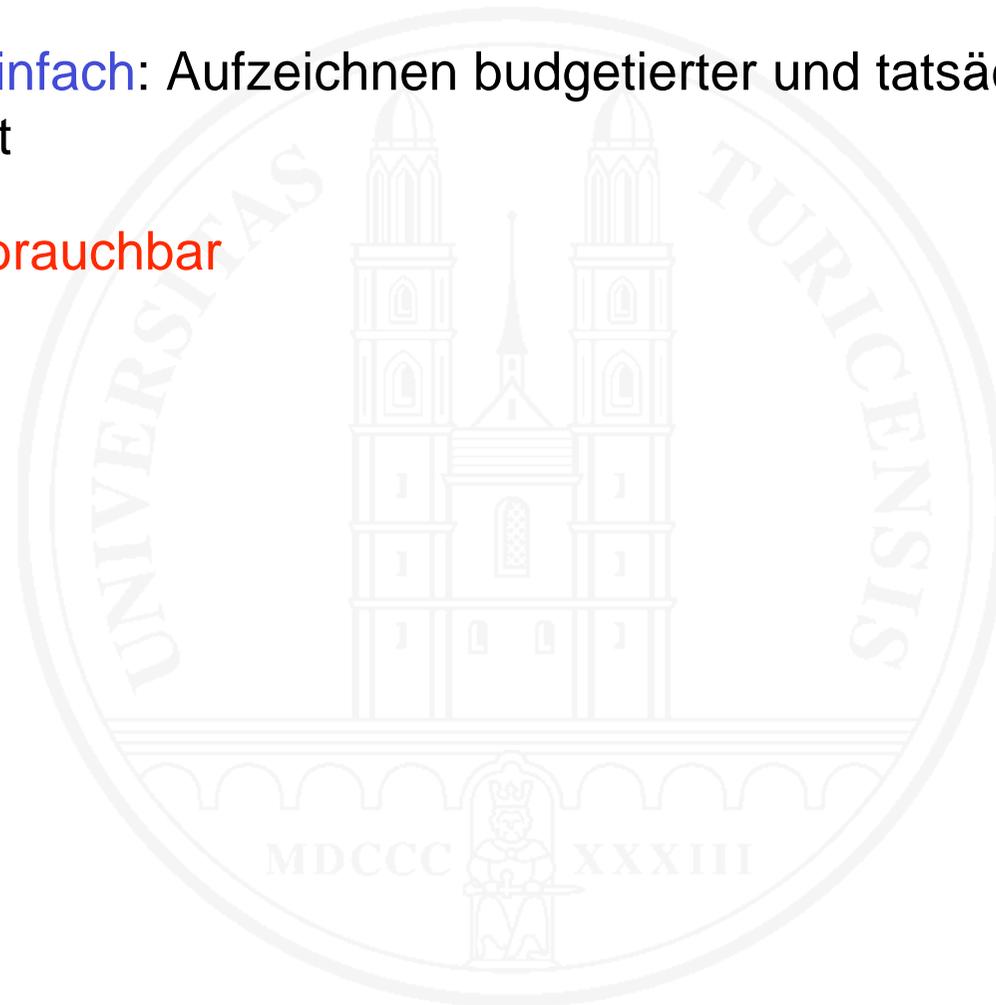


Sachziel- und Terminverfolgung

- Mit Hilfe der **Meilensteine**
- **Planung:**
 - Meilensteine strukturieren die Sachziele in Teilziele
 - Festlegung der SOLL-Termine für alle Meilensteine
- Wenn Meilenstein **erreicht:**
 - **Zwischenziel** nachweislich **erreicht**
 - **gesicherte quantitative Aussage** der **Terminlage** durch Soll-Ist-Vergleich
- Meilenstein am geplanten Termin **nicht erreicht:** **Schätzung** der **Terminlage** durch Schätzung des verbleibenden Aufwands

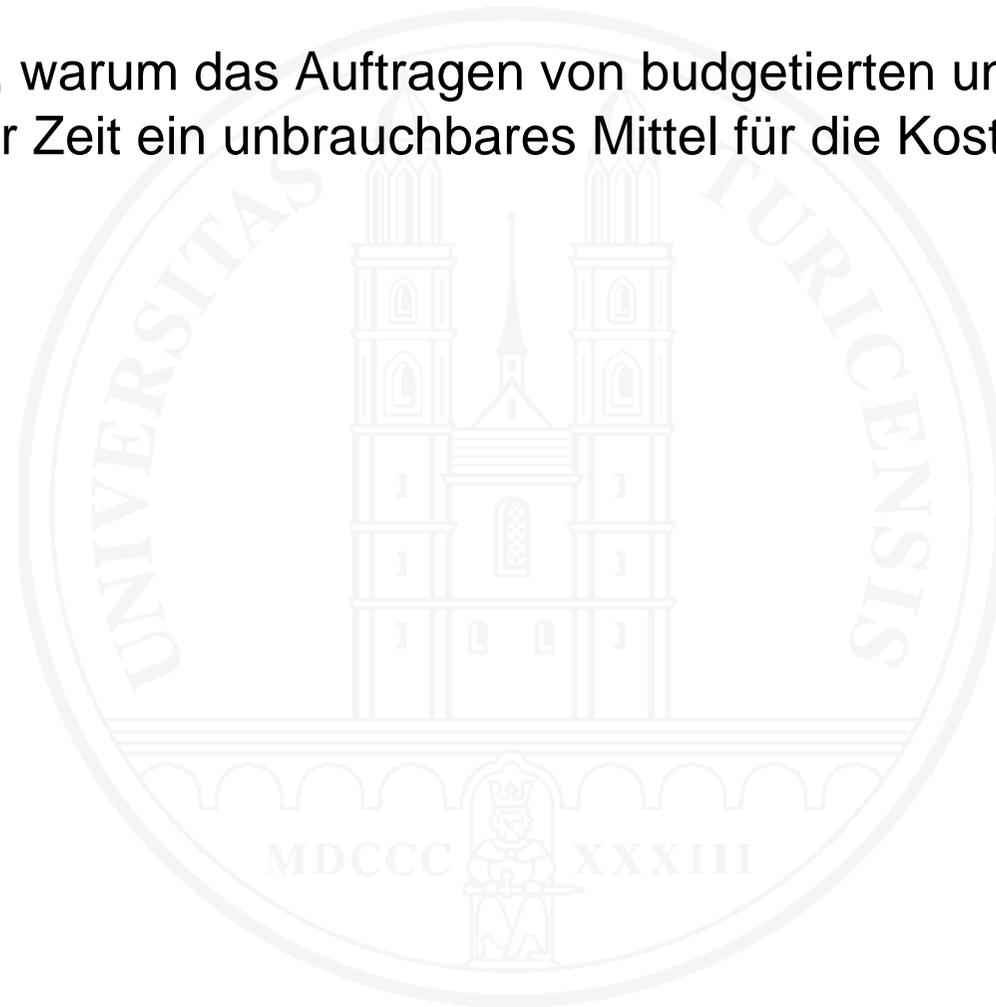
Kostenverfolgung – 1

- **Scheinbar einfach**: Aufzeichnen budgetierter und tatsächlicher Kosten über der Zeit
- De facto **unbrauchbar**



Mini-Übung 15.1 (Aufgabe 4.2 im Skript)

Begründen Sie, warum das Auftragen von budgetierten und tatsächlichen Kosten über der Zeit ein unbrauchbares Mittel für die Kostenverfolgung in Projekten ist.



Kostenverfolgung – 2

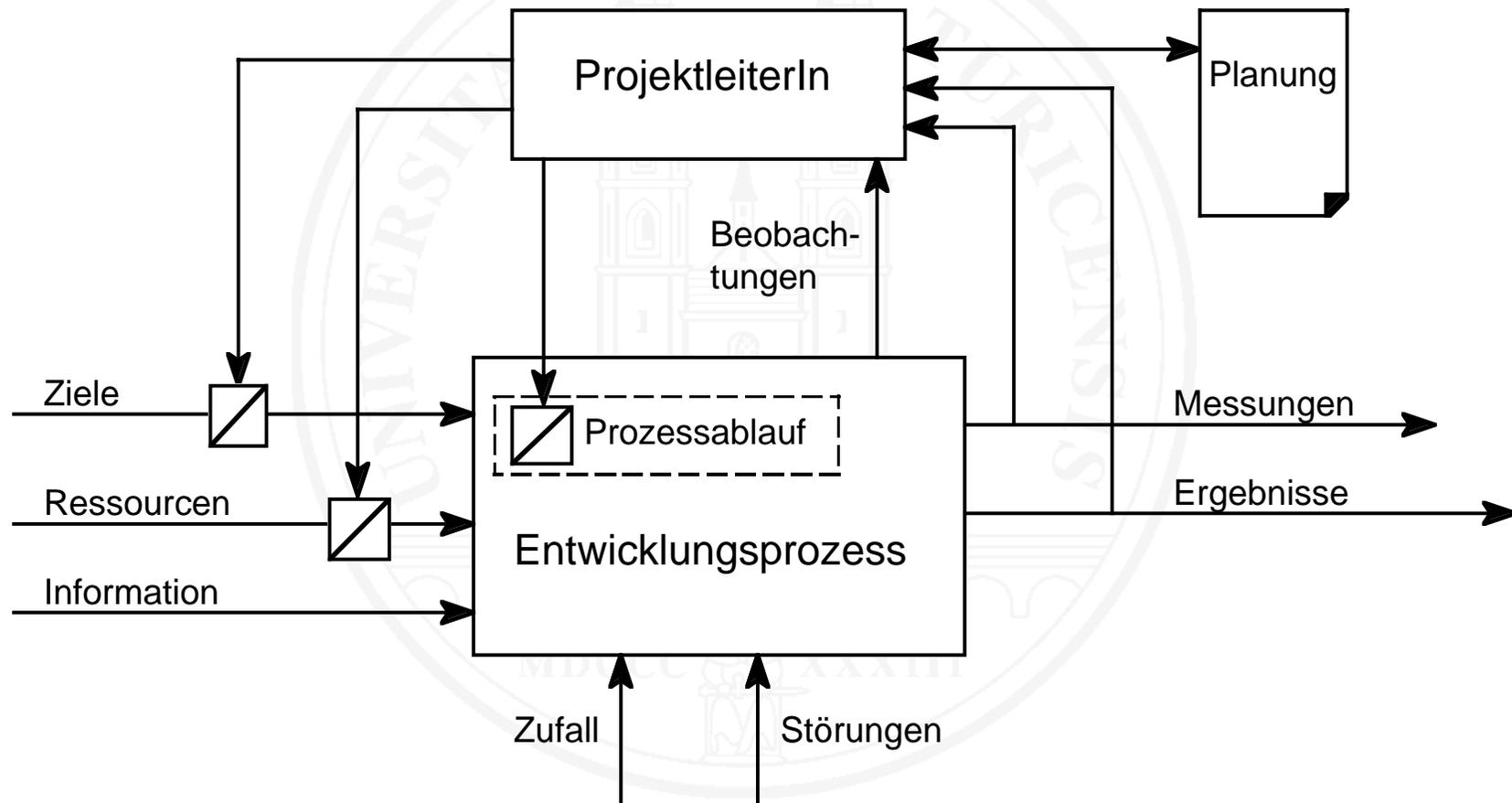
- Welche brauchbaren Möglichkeiten zur Kostenverfolgung gibt es?
- Zwei Alternativen:
 - **Kosten** und fiktive **Erträge** auf einer **Zeitachse**
 - **Kosten** auf einer **Meilensteinachse**: Soll-Kosten am geplanten Meilenstein-Termin mit Ist-Kosten bei Erreichung des Meilensteins vergleichen

[Risikoverfolgung: später]

Verantwortlichkeiten und Berichtswesen

- **Verantwortlichkeiten** und **Kompetenzen** aller Beteiligten klar geregelt
- Keine Übertragung von Verantwortung ohne die dafür notwendigen Kompetenzen und Ressourcen
- **Stufengerechtes** Umgehen mit **Problemen**
- **Berichtswesen**
 - Nicht als bürokratische Schikane...
 - ...sondern als **Frühwarnfunktion**
- **Arbeitspaket-Ordner** als Organisationsmittel

Projektlenkung

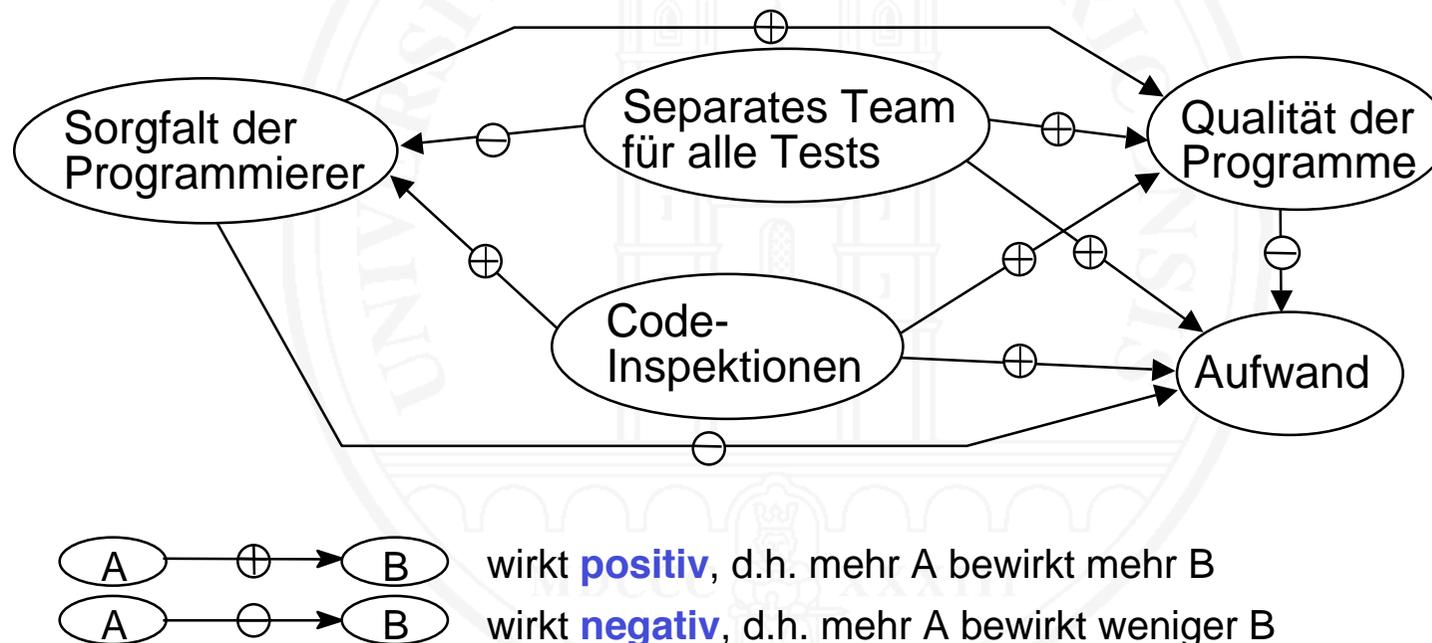


Maßnahmen bei Abweichung

- Abweichungen ⇨ Gegenmaßnahmen
- Beispiel: Terminüberschreitung ⇨
 - Bereitstellung zusätzlicher Ressourcen
 - Befreiung von Projektmitarbeitern von anderen Verpflichtungen
 - Anordnung von Überstunden
 - Vergabe von Teilaufträgen an Dritte
 - Abstriche bei den zu erreichenden Sachzielen
 - Etappierung der Sachziele (nach der Art eines Wachstumsmodells)
- **Vorsicht:** Software-Projekte sind nichtlineare Systeme
- Falls Gegenmaßnahmen versagen oder nicht möglich sind:
 - ⇨ Planung anpassen

Software-Projekte sind nichtlineare Systeme

- Wirkungen, Nebenwirkungen und Auswirkungen studieren
- Beispiel: Maßnahmen zur Lenkung der Qualität der erstellten Programme in einem Software-Projekt

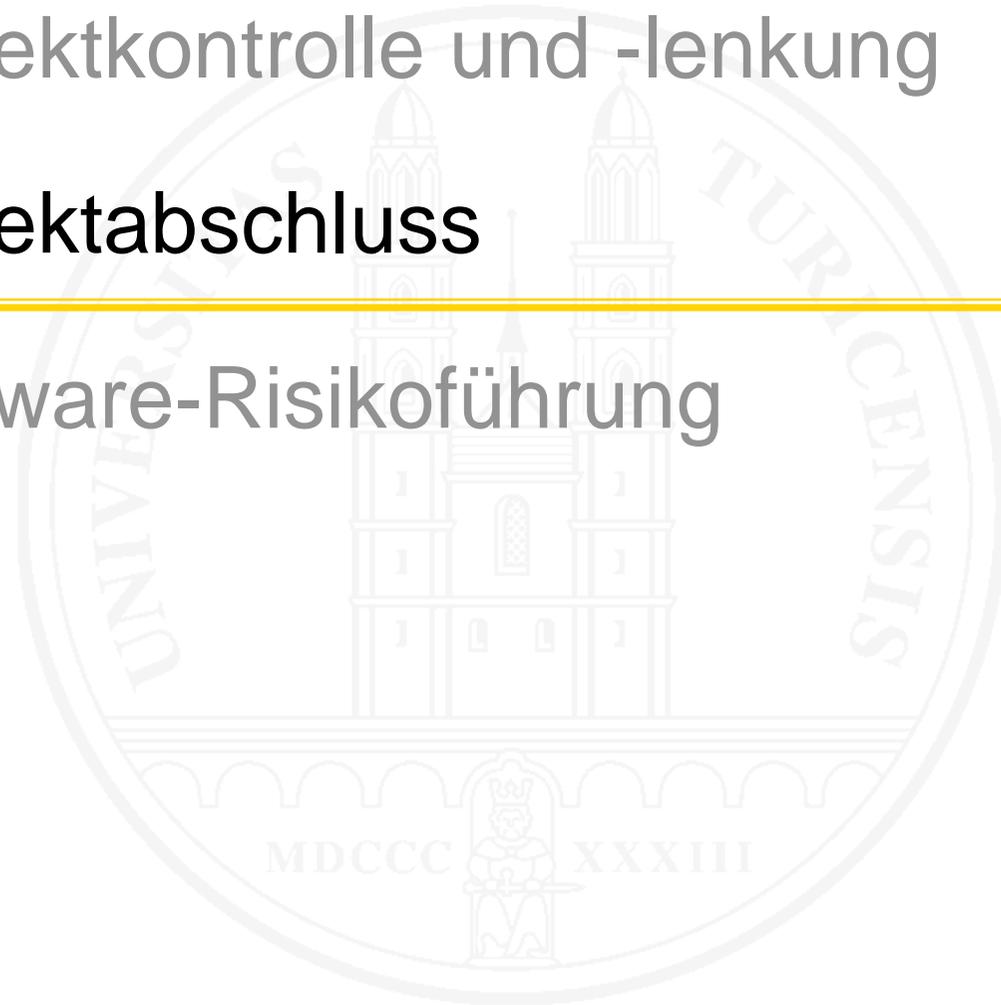


15.1 Projektplanung

15.2 Projektkontrolle und -lenkung

15.3 Projektabschluss

15.4 Software-Risikoführung



Projektabschluss

- ⇒ Produkt geordnet in die **Pflege überleiten**
- ⇒ Entstandenes **Wissen sichern**

- **Dokumente abschließen** und **archivieren**
- **Messungen**
 - abschließen
 - **Gesamtgrößen berechnen** (zum Beispiel Gesamtaufwand, totale Durchlaufzeit)
 - Messwerte **archivieren**
- **Projektgeschichte dokumentieren**
 - SOLL und IST für Kosten, Termine, Sachziele, Personaleinsatz
 - Erfahrungen und Lehren

Lehren: Die Geschichte der Elchjäger in Kanada



15.1 Projektplanung

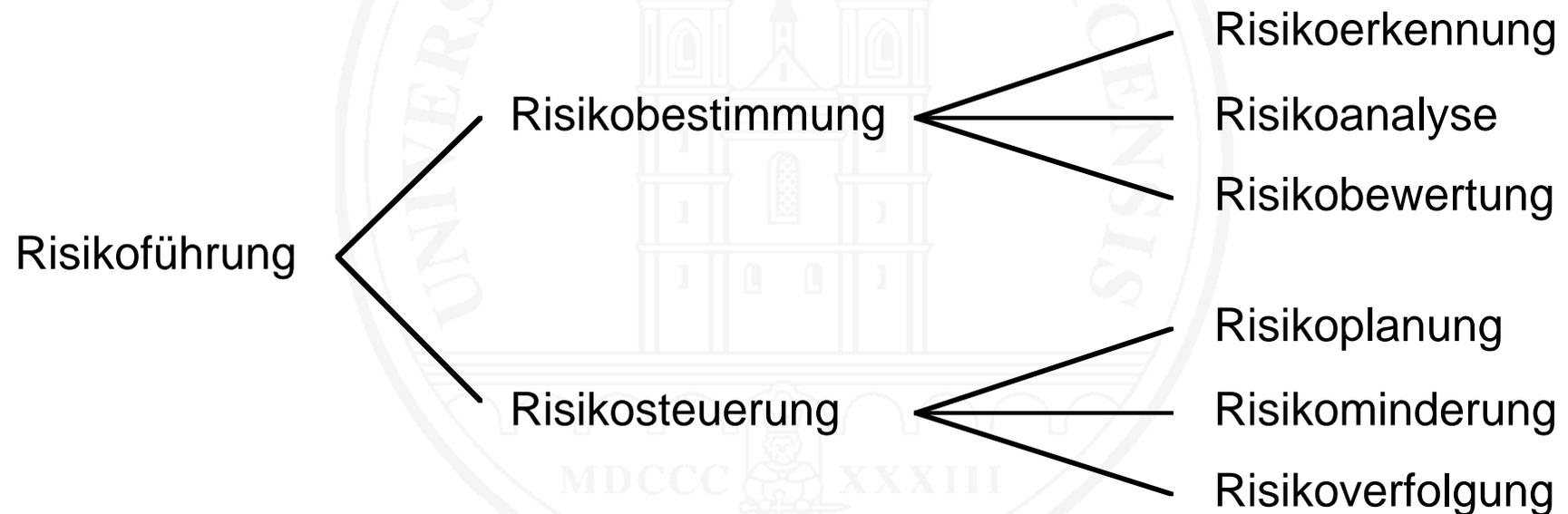
15.2 Projektkontrolle und -lenkung

15.3 Projektabschluss

15.4 Software-Risikoführung

Risikoführung

Risiko – Ereignis, welches den **sachlichen** oder **wirtschaftlichen Erfolg** eines Projekts **bedroht**.



Die 10 häufigsten Software-Risiken

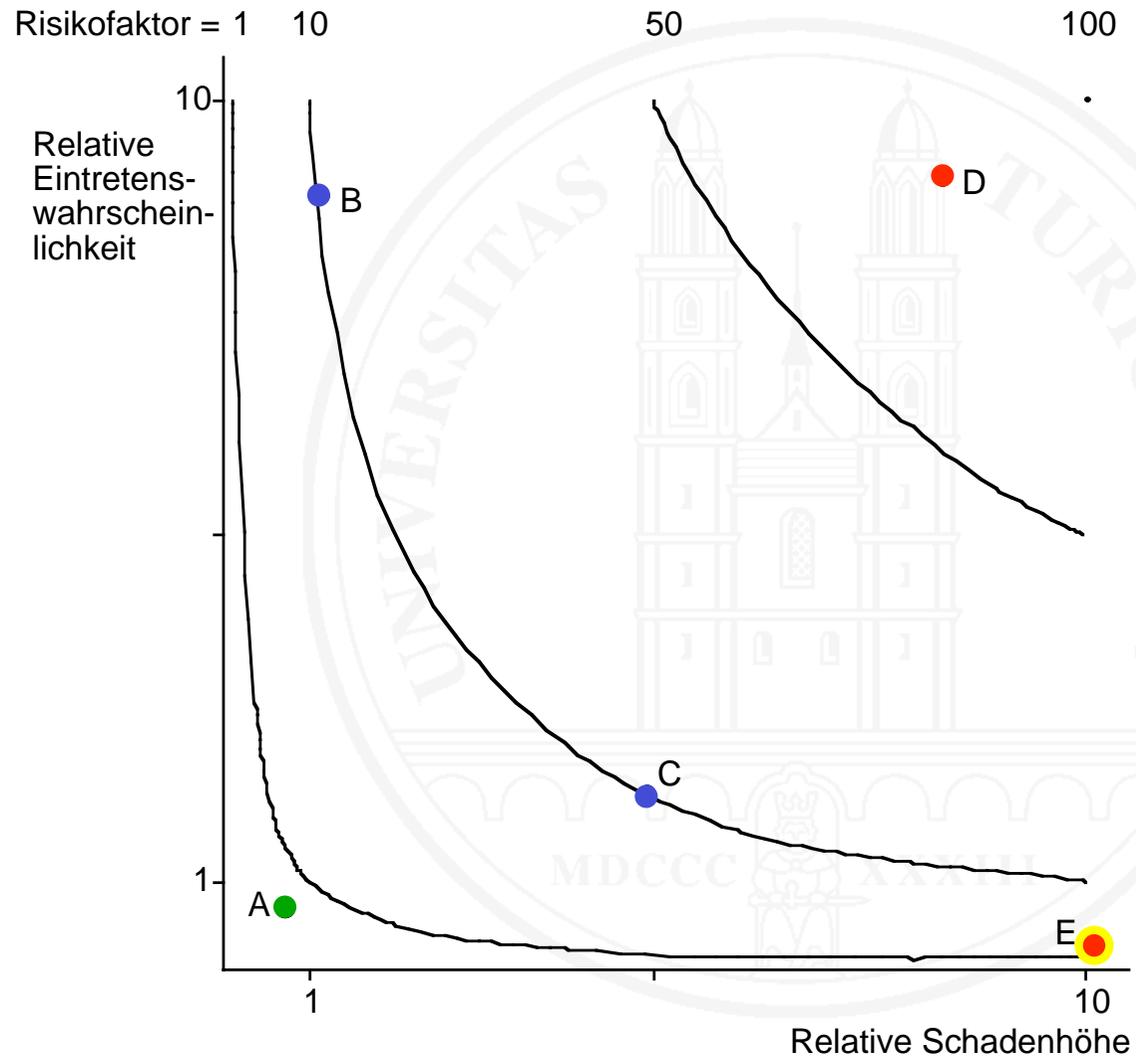
- 1 Zu wenig Leute
- 2 Unrealistische Kosten- und Terminpläne
- 3 Falsche Funktionalität
- 4 Falsche Benutzerschnittstelle
- 5 Vergoldung (überflüssiger Luxus)
- 6 Sich ständig verändernde Anforderungen
- 7 Probleme mit zugekauften Komponenten
- 8 Probleme mit extern vergebenen Aufträgen
- 9 Nichterreichen der verlangten Leistungen (z.B. Reaktionszeit)
- 10 Überforderung der Mitarbeiter in Bezug auf ihr softwaretechnisches Können

Quelle: Boehm (1991)

Risikoanalyse – 1

- Bestimmung der **Gefährlichkeit** der Einzelrisiken und von Risikokombinationen
- Gefährlichkeit:
 - Eintretenswahrscheinlichkeit $p(\text{Risiko})$
 - Schadenhöhe $s(\text{Risiko})$
 - ⇒ Bewertung mit **Risikofaktor**: $f(\text{Risiko}) = p(\text{Risiko}) * s(\text{Risiko})$
- Risiken mit gleichem Risikofaktor sind etwa gleich gefährlich
- Problem der **Restrisiken**

Risikoanalyse – 2



Risikosteuerung

- Pläne und Maßnahmen für die **Beherrschung** der **großen Risiken**:
 - Ist das Risiko **vermeidbar**?
 - Gibt es Maßnahmen zur **Minderung**?
 - Wie soll das Risiko im Projekt **verfolgt** werden?
 - Kann das Risiko auf Dritte **abgewälzt** werden?
- **Risikoverfolgung**: Die Risiken während der Projektabwicklung im Auge behalten
- Zum Beispiel durch regelmäßig aktualisierte Rangliste der Risiken

Die 10 häufigsten Software-Risiken: Maßnahmen – 1

Risiko	mögliche Maßnahmen
Zu wenig Leute	Gute Leute einstellen, vorhandene Leute ausbilden, Motivation und Arbeitsklima fördern, den richtigen Leuten die richtigen Aufgaben geben
Unrealistische Kosten- und Terminpläne	Sorgfältige Aufwandschätzung, Entwicklung mit Wachstumsmodell, Anforderungen reduzieren, kostenorientierte Entwicklung
Falsche Funktionalität	Quantifizierte Ziele, sorgfältige Spezifikation, Prototypen, Beteiligung des Auftraggebers
Falsche Benutzerschnittstelle	Prototypen, Einbezug der Endbenutzer (oft nicht identisch mit den Auftraggebern!)
Vergoldung (überflüssiger Luxus)	Kosten-Nutzen-Analyse, Setzen von Prioritäten in den Zielen, kostenorientierte Entwicklung

Die 10 häufigsten Software-Risiken: Maßnahmen – 2

Risiko	mögliche Maßnahmen
Sich ständig verändernde Anforderungen	Setzen von Wichtigkeits-Schwellwerten (unterhalb derer nicht geändert wird), änderungsfreundlicher Entwurf (z.B. durch Information Hiding), Entwicklung mit Wachstumsmodell
Probleme mit zugekauften Komponenten	Sorgfältige Auswahl (z.B. mit Benchmarks), Eingangs-Qualitätskontrolle
Probleme mit extern vergebenen Aufträgen	Überprüfung des Auftragnehmers vor Auftragsvergabe, klar formulierte Aufträge, Zwischeninspektionen während der Abwicklung, Abnahmeinspektion, Aufträge mit Erfolgshonorar
Nichterreichen der verlangten Leistungen (z.B. Reaktionszeit)	Abschätzung in Review, Simulationen, Prototypen, Messung und Optimierung
Überforderung der Mitarbeiter in Bezug auf ihr softwaretechnisches Können	Aufgabenanalyse, Ausbildung, Reduktion der Anforderungen, Entwicklung mit Wachstumsmodell

Literatur

Siehe Literaturverweise im Kapitel 4 des Skripts.

Im Skript [M. Glinz (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich] lesen Sie Kapitel 4.

Im Begleittext zur Vorlesung [S.L. Pfleeger, J. Atlee (2006). *Software Engineering: Theory and Practice*, 3rd edition. Upper Saddle River, N.J.: Pearson Education International] lesen Sie Kapitel 3 mit Ausnahme des Unterkapitels 3.3.

