

---

# Vorlesung „Methoden des Software Engineering“

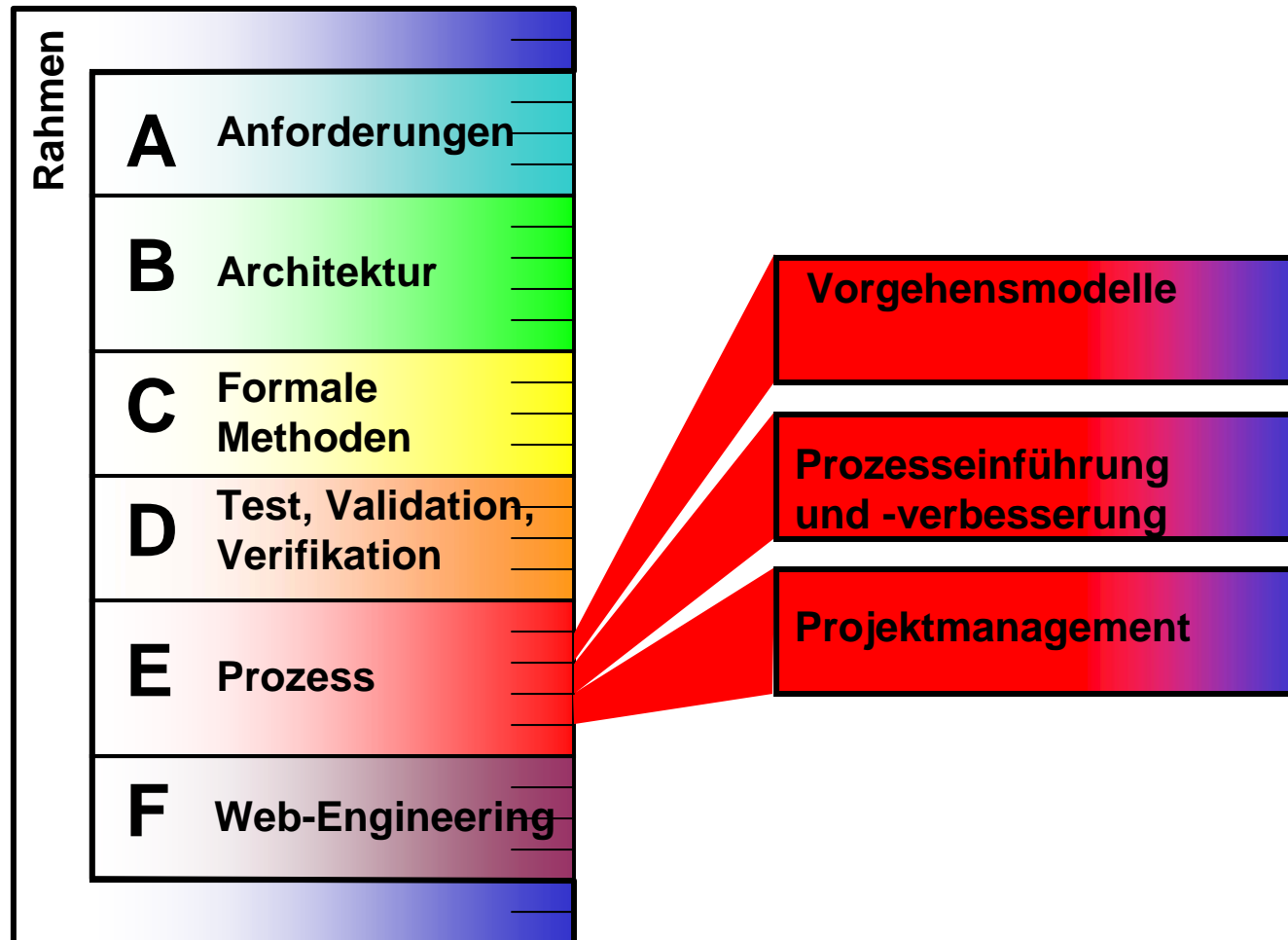
Block E „Software-Prozess und Projektmanagement“

## **Projektmanagement**

Martin Wirsing

Einheit E.3, 25.1.2005

# Gliederung Block E: Software-Prozess



# Ziele

---

- Abgrenzung und prinzipiellen Ablauf von Projekten verstehen
- Aufgaben des Projektmanagement kennen lernen
- Einblick in Projektplanung, Schätzverfahren und Risikomanagement erhalten

# Was ist ein Projekt?

---

- Ein Projekt ist ein Vorhaben, das im wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, z.B.
  - Zielvorgabe
  - zeitliche, finanzielle, personelle und andere Begrenzungen
  - Abgrenzung gegenüber anderen Vorhaben
  - projektspezifische Organisation.

( DIN 69 901)

- Was heißt es, ein Projekt zu "machen"?
  - Schaffen einer stabilen, akzeptierten und funktionierenden "temporären Organisation,,
  - Abgrenzungen gegenüber Umgebung nötig

# Abgrenzung eines Projekts

---

- **Zeitliche Abgrenzung**
  - Zeitplan, Termine,
  - Vor- und Nachprojekt-Phase
- **Sachliche Abgrenzung**
  - Ziele, Hauptaufgaben, Nicht-Ziele
  - zu anderen Projekten, Tätigkeiten
- **Soziale Abgrenzung**
  - Projektleiter, Projektauftraggeber, Projektteam
  - Umwelt-/Umgebungsanalyse

# (Unerwünschter) Projektablauf

---

1. Enthusiasmus
2. Desillusionierung
3. Panik
4. Suche nach den Schuldigen
5. Bestrafung der Unschuldigen
6. Ruhm & Ehre für die Unbeteiligten

# Projektmanagement

---

- **Projektmanagement umfasst alle Aufgaben bei der Durchführung von Projekten hinsichtlich**
  - Vorbereitung (Struktur und Personal)
  - Planung
  - Kontrolle
  - Steuerung
- **Dazu gehören auch projektübergreifende Aufgaben:**
  - Projektabschluss und Ergebnisdokumentation
  - Prozessverbesserung
  - Personalführung
  - Interaktion mit dem Kunden und Koordination von Zulieferern.

# Projektverlauf

---



- **Projektvorbereitung**
  - Festlegung der Projektziele
  - Zusammenstellung der groben Projektplanung  
...in Koordination mit dem Auftraggeber!
  - Zusammenstellung und Nutzen von Know-How aus früheren Tätigkeiten und Projekten
  - Möglichst Durchführung eines Start-Workshops gemeinsam mit Auftraggeber und Projektteam
- **Projektabschluss**
  - Zusammenstellung und Präsentation der Projektergebnisse
  - Gemeinsamer inhaltlicher und emotionaler(!) Abschluss des Projektes
  - Sicherung des Know-Hows und der "learned lessons" für nachfolgende Projekte
  - Prozessverbesserung

# Projektverlauf

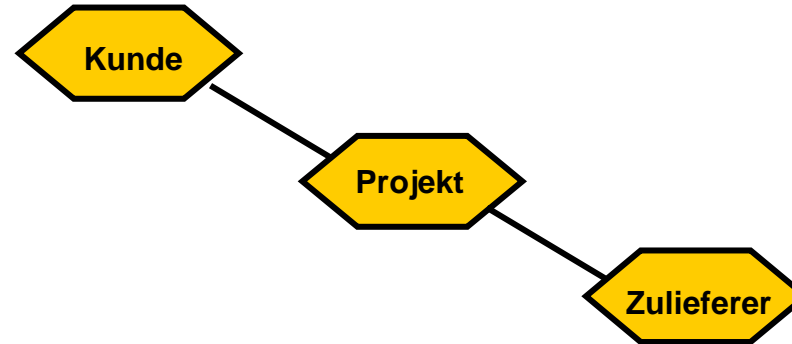
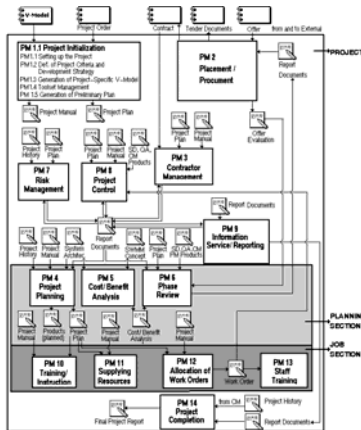
---



- **Projektdurchführung**
  - **Projektkontrolle**
    - Feststellung des Projektstatus
    - Feststellung von Planabweichungen
    - Techniken:
      - Fortschrittsanalyse
      - Risikoanalyse
      - Qualitätssicherungsmaßnahmen
  - **Projektsteuerung**
    - Durchführungsentscheidungen
    - Korrektivmaßnahmen
    - Techniken:
      - Risikomanagement
      - Qualitätsmanagementmaßnahmen
      - Konfigurationsmanagement

# Projektverlauf

## Prozesse im Submodell PM des VM'97



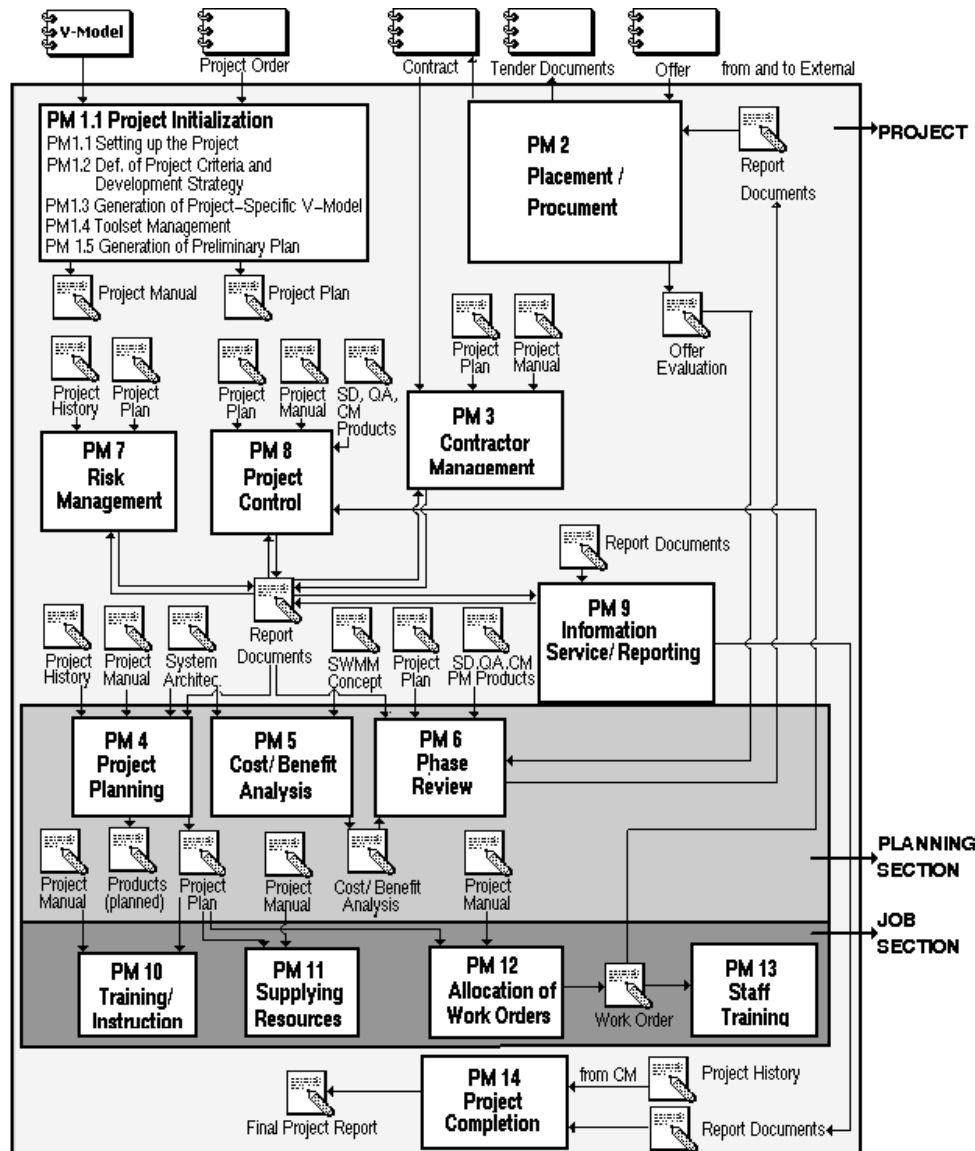
### Prozesse nach VM'97 (Submodell PM)

- PM1:** Projektinitialisierung
- PM2:** Vergabe/Beschaffung
- PM3:** Auftragnehmer-Management
- PM4:** Feinplanung
- PM5:** Kosten-/Nutzenanalyse
- PM6:** Durchführungsentscheidung
- PM7:** Risikomanagement

- PM8:** Projektkontrolle und -steuerung
- PM9:** Informationsdienst/Berichtswesen
- PM10:** Schulung/Einarbeitung
- PM11:** Bereitstellung der Ressourcen
- PM12:** Vergabe von Arbeitsaufträgen
- PM13:** Einweisung der Mitarbeiter
- PM14:** Projektabschluss

# Projektverlauf

## Prozesse im Submodell PM des VM'97



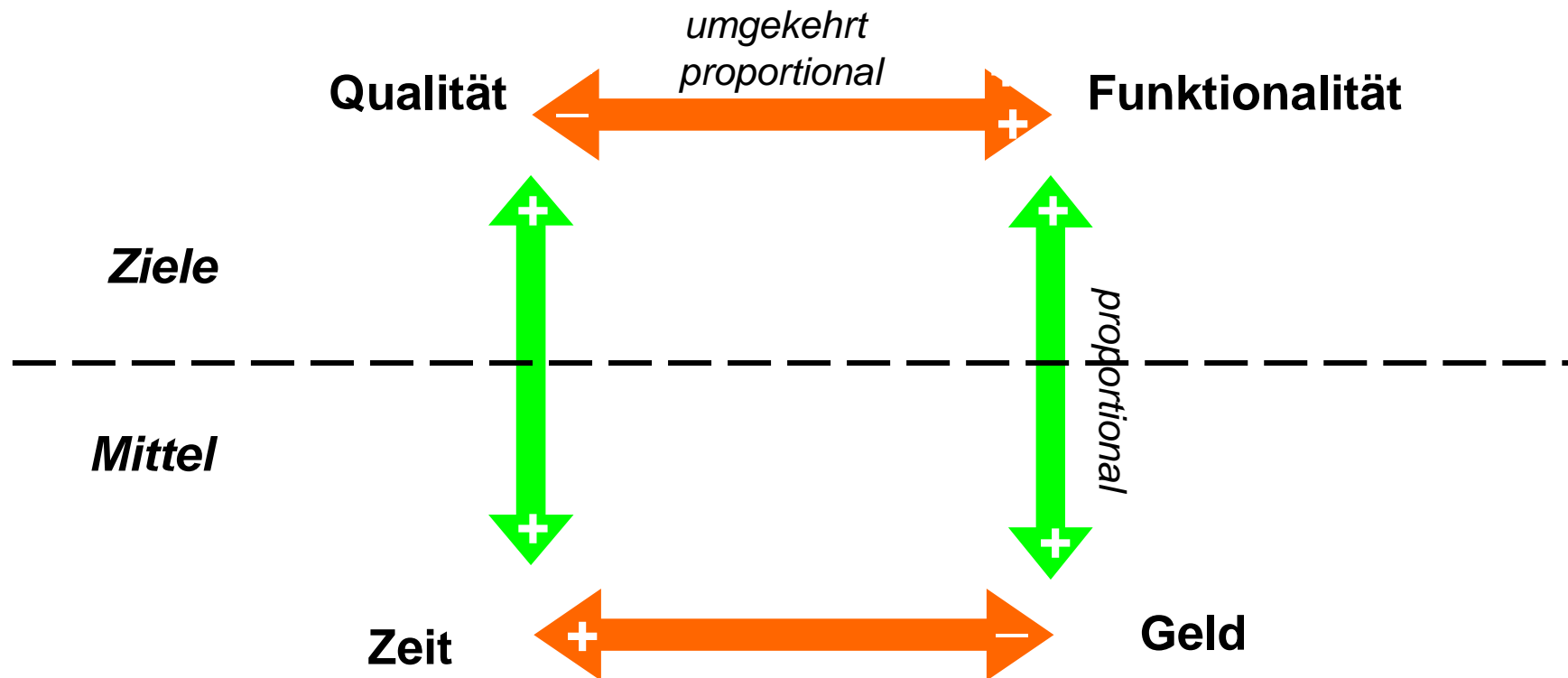
# Spezielle Aspekte des Projektmanagement

---

1. Projektziele
2. Techniken zur Planung und Kontrolle
3. Schätzverfahren
4. Risikomanagement

# 1. Projektziele

Die Projektziele werden charakterisiert durch das „magische Viereck“ bestehend aus Zeit, Geld, Qualität und Funktionalität:



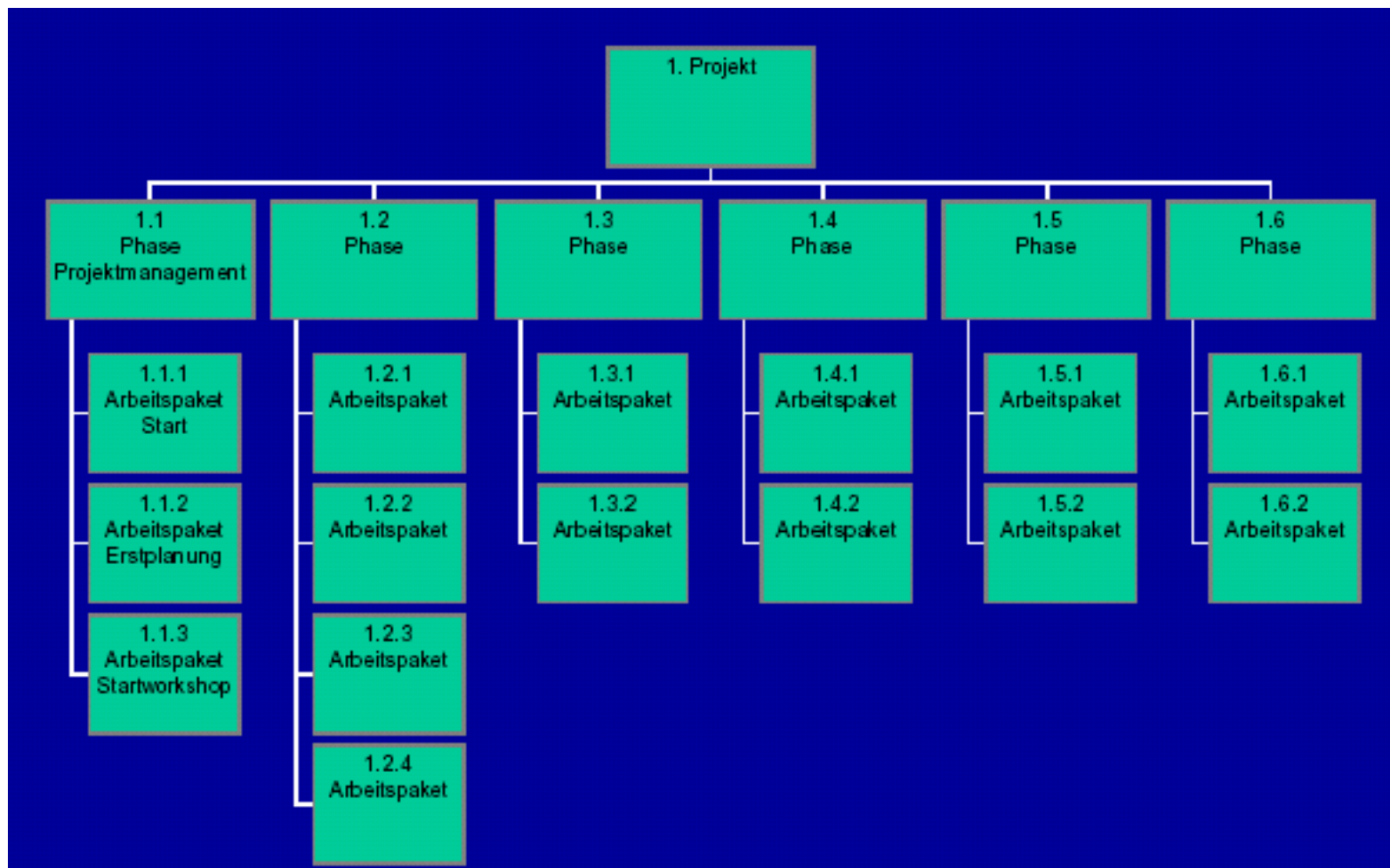
## 2. Projektplanung

---

- **Ziele**
  - Vorbereitung Projektdurchführung
  - Überwachung und Steuerung Durchführung
- **Inhalte**
  - Definition der Aufgaben: Was ist zu tun?
  - Definition der Vorgaben/Hilfsmittel: Wie ist es zu tun?
  - Definition der Termine: (Bis) Wann ist es zu tun?
  - Definition der Verantwortlichkeiten: Wer hat es zu tun?
- **Durchführung der Planung**
  - Planung bei Projektinitiierung
  - (Wiederholte) Neuplanung bei
    - Detaillierung der Produktstruktur
    - Änderung der Terminlage
    - Änderung der Personallage
    - Änderung der Ressourcenlage
- **Methoden der Projektplanung**
  - **Projektstrukturplan**
  - **Terminplanung**
  - Ressourcen- und Kostenplanung
  - Projektdokumentation
  - Computergestütztes PM?

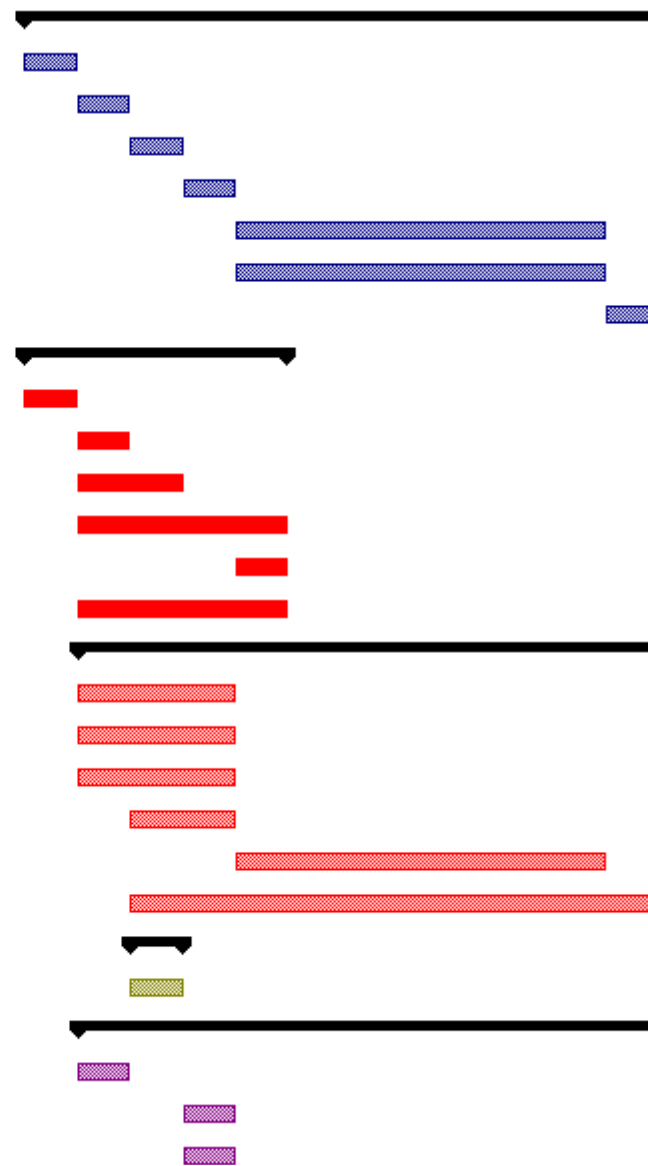
# Projektstrukturplan

- Ziele
  - Gliederung des Projekts in Arbeitspakete
  - Vorbereitung der Aufwands- und Zeitplanung



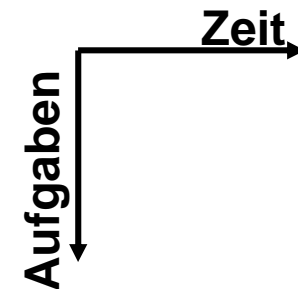
# Zeitplanung

<b>Project Management</b>
Conceive New Project
Evaluate Project Scope and Risk
Develop Software Development Plan
Plan for Next Iteration
Manage Iteration
Monitor and Control Project
Reevaluate Project Scope and Risk
<b>Business Modeling</b>
Assess Business Status
Identify Business Processes
Refine Business Processes
Design Business Process Realizations
Refine Roles and Responsibilities
Explore Process Automation
<b>Requirements</b>
Analyze the Problem
Understand Stakeholder Needs
Define the System
Manage the Scope of the System
Refine the System Definition
Manage Changing Requirements
<b>Test</b>
Plan Test
<b>Environment</b>
Prepare Environment for Project
Prepare Environment for an Iteration
Prepare Guidelines for an Iteration

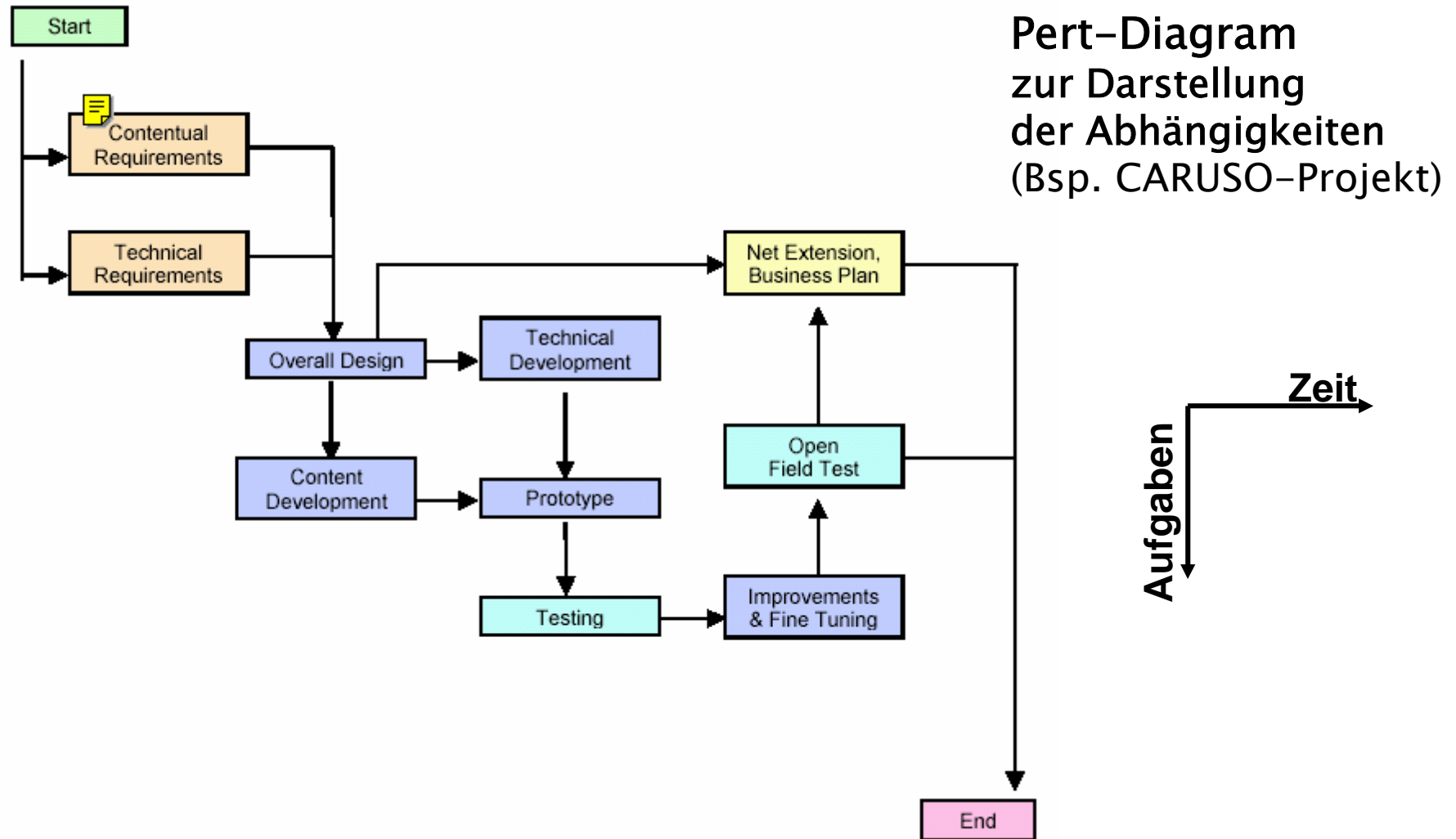


**Gantt-Diagramm aus MS Project**

**Core Workflows nach RUP**



# Darstellung der Abhängigkeiten



# 3. Schätzverfahren

---

„Was man nicht misst, das kann man nicht steuern.“  
[Tom de Marco, *Controlling Software Projects*, 1982.]

- **Messung von Softwaresystemen:**
  - Zu Projektbeginn: Projektplanung (Schätzung)
  - Bei Durchführung: Projektstatus/Projektsteuerung
- **Ziel des Schätzens:**
  - Bestimmung des Entwicklungsaufwands
    - Realisierungsaufwand
    - Realisierungszeit
  - Abhängig von
    - Systemkomplexität
    - Produktivität
  - Möglichst vor Systemrealisierung
- **Prinzipielle Strategie beim Schätzen: per Analogie**

# Messen von Systemkomplexität

---

- **Komplexitätsfaktoren**
  - Größe: Anzahl der Bausteine
  - Diversität: Unterschiedlichkeit der Bausteine
  - Vernetzung: Abhängigkeit der Bausteine
- **Grundlagen der Softwareschätzung**
  - Messung/Schätzung von Systemkomplexität
  - Komplexität, Aufwand, Laufzeit
- **Softwaremetrik Code**
  - Ansatz: Software = Programmcode
  - Einheit: Anzahl der Programmzeilen (Lines of Code – LoC)
  - Messverfahren: Softwareumfang = Anzahl Programmzeilen
  - Vorzug: Einfache Messung
- **Verschiedene Maße in verschiedenen Phasen**
  - Implementierung: Codezeilen
  - Feinentwurf: Abstrakter Code
  - Analyse: Funktionspunkte

# Lines-Of-Code (LoC)

## Varianten und abgeleitete Maße

---

- **Lines-of-Code (LoC)**
  - zähle Zeilen, subtrahiere leere Zeilen
- **Non-Comment-Lines-of-Code (NCLoC)**
  - zähle LOC, subtrahiere reine Kommentarzeilen
- **Kommentardichte (KD)**
  - abgeleitet, daher kein eigenes Messverfahren nötig
  - $KD(P) = CLoC(P) / NCLoC(P)$
- **Verteilung Deklarationen/ausführbare Befehle**
  - unterscheide NCLoC in DecLoC und ExecLoC
  - z.B. Deklarationsanteil(P) =  $DecLoC / (DecLoC + NCLoC)$

# Lines-Of-Code (LoC) Produktivitätsparadoxon

---

- Die Benutzung einer höheren Programmiersprache scheint eine geringere Produktivität nach sich zu ziehen: für das gleiche Geld/in der gleichen Zeit bekommt man weniger Ergebnis.

Sprache	<u>Assembler</u>		<u>Cobol</u>	
Resultat	10.000	LoC	3.000	LoC
Aufwand	500	PT	300	PT
Kosten	125.000	€	75.000	€
Produktivität	12,50	€/Zeile	25,50	€/Zeile
	0,05	PT/Zeile	0,1	PT/Zeile
	20	Zeile/PT	10	Zeile/PT

- Die Abstraktionsebene der Programmiersprache geht in diese Berechnung nicht ein – in den Nutzen für den Anwender aber schon.
- Daher das LoC-Maß nur für relative Produktivitätsbetrachtungen geeignet, also z.B. bei der Prozessverbesserung.

# Alternativen zu Lines-Of-Code (LoC)

---

- **Non-Comment-Source-Statements (NCSS)**
  - baue abstrakten Syntaxbaum auf, zähle geeignete Knotentypen
  - z.B. im Eclipse-Plugin
- **Volumen**
  - Wird gemessen in Bit
  - kein eigenes Messverfahren, Wert wird abgeleitet:  $V=N \cdot \log_2 \eta$
  - wobei
    - $N$ =Größe des Vokabulars
    - $\eta$ =benutzte Symbole
- Es zeigt sich aber empirisch:  
Für viele realistische Programme  $p$  gilt:  $V(p) = O(\text{LoC}(p))$ .
- Für eine Reihe weiterer Größenmaße gilt das gleiche.

# Funktionspunkte (Function Points – FP)

---

- Eingeführt 1979 von Alan Albrecht (IBM).
- Gegenstand der ISO-Standards 10926 und 14143-1:1998.
- Vorgehen:
  - Zähle gewisse Elemente eines Entwurfs und gewichte sie nach definierten Erfahrungswerten, dies ergibt unangepasste Function Points.
  - Wende eine Reihe von Korrekturfaktoren an, dies ergibt (angepasste) Function Points.

# Unangepasste FP (UFP)

---

- **Messverfahren: zähle**
  - Eingaben                    Parameter wie Dateinamen, Kommandos, Menüeinträge
  - Ausgaben                Nachrichten, Protokolle, Wertelisten
  - Interaktionen            Benutzerinteraktion
  - ext. Zugriffe            Zugriffe auf Schnittstellen zu externen Systemen
  - DB-Zugriffe            Zugriffe auf Schnittstellen zu untergeordneten Systemen & Dateien
  
- **Bewerte jedes Element als *einfach*, *mittel* oder *komplex* und bilde die gewichtete Summe nach folgender Tabelle:**

<u>Item</u>	<u>einfach</u>	<u>mittel</u>	<u>komplex</u>
- Eingaben	3	4	6
- Ausgaben	4	5	7
- Interaktionen	3	4	6
- ext. Zugriffe	7	10	15
- DB-Zugriffe	5	7	10

# Angepasste FP

---

$$FP = UFP \cdot TK$$

wobei der Faktor Technische Komplexität (TK) gebildet wird durch

$$TK = 0.65 + 0,01 \sum_{i=1..14} F_i$$

und

- |  |                                    |
|--|------------------------------------|
| F <sub>1</sub> Reliable back-up and recovery | F <sub>2</sub> Data communications |
| F <sub>3</sub> Distributed functions         | F <sub>4</sub> Performance         |
| F <sub>5</sub> Heavily used configuration    | F <sub>6</sub> Online data entry   |
| F <sub>7</sub> Operational ease              | F <sub>8</sub> Online update       |
| F <sub>9</sub> Complex interface             | F <sub>10</sub> Complex processing |
| F <sub>11</sub> Reusability                  | F <sub>12</sub> Installation ease  |
| F <sub>13</sub> Multiple sites               | F <sub>14</sub> Facilitate change  |

haben je einen Wert zwischen 0 und 5 (*nicht bzw. sehr stark vorhanden*).

# Technische Korrekturfaktoren

## Übersetzung und Erläuterung

---

<b>Faktor</b>	<b>Originalname</b>	<b>Erläuterung</b>
F1	Reliable back-up & recov.	Anforderungen an die Datenintegrität
F2	Data communications	Datenaustausch mit Umsystemen
F3	Distributed functions	Applikationen auf mehr als einem Rechner
F4	Performance	Leistungsanforderungen
F5	Heavily used config.	Starke Konfigurationsabhängigkeiten
F6	Online data entry	Interaktive Benutzung
F7	Operational ease	Interaktionsqualität
F8	Online update	ummittelbare Aktualisierung (WYSIWYG)
F9	Complex interface	komplexe Benutzerschnittstelle
F10	Complex processing	algorithmisch komplexe Verarbeitung
F11	Reusability	geplante Wiederverwendung/Wartung
F12	Installation ease	Installationskomfort
F13	Multiple sites	mehrere (unterschiedliche) Installationen
F14	Facilitate change	Unterstützung der Anpassung in der Wartung

# Berechnung am Beispiel „Ausleihmaske“

**Ausleihsystem**

**Stammdaten**

Lesernummer

Name

Vorname

Geburtsdatum

**Kontodaten**

Titel	Signatur	Frist	

Mögliche Interaktionen:

- 1) Ausweis scannen
- 2) Lesernummer eintippen
- 3) Stammdaten eingeben

Entspricht 2 Funktionalitäten

- 1) Vorbereitung Ausleihe
- 2 & 3) Beauskunftung

# Berechnung am Beispiel „Ausleihmaske“

---

## 1.) Elemente zählen

	(1)	(2)	(3)
Eingaben	1 x einfach	1 x einfach	1 x mittel
Interaktionen	-	-	-
ext. Zugriffe	1 x einfach	-	-
DB-Zugriffe	-----1 x einfach + 1 x komplex-----		
Ausgaben	-----1 x einfach + 1 x komplex-----		

# Berechnung am Beispiel „Ausleihmaske“

## 2.) UFP berechnen

	(1)	(2)	(3)
Eingaben	1x 3	1x 3	1x 4
Interaktionen	-	-	-
ext. Zugriffe	1x 7	-	-
DB-Zugriffe	-----1x 5	+	1x 10-----
Ausgaben	-----1x 4	+	1x 7 -----
<hr/>			
UFP	36	29	30

# Berechnung am Beispiel „Ausleihmaske“

## 3.) Technische Faktoren schätzen

2	F <sub>1</sub> Reliable back-up & recovery
1	F <sub>2</sub> Data communications
2	F <sub>3</sub> Distributed functions
1	F <sub>4</sub> Performance
0	F <sub>5</sub> Heavily used configuration
3	F <sub>6</sub> Online data entry
5	F <sub>7</sub> Operational ease
3	F <sub>8</sub> Online update
2	F <sub>9</sub> Complex interface
0	F <sub>10</sub> Complex processing
1	F <sub>11</sub> Reusability
1	F <sub>12</sub> Installation ease
4	F <sub>13</sub> Multiple sites
1	F <sub>14</sub> Facilitate change

### Technische Komplexität

$$TK = 0,65 + 0,01 \sum_{i=1..14} F_i$$

hier also

$$\begin{aligned} TK &= 0,65 + 0,01 * 26 \\ &= 0,169 \end{aligned}$$

### Also für die Interaktionsarten

	(1)	(2)	(3)
UFP	6,1	4,9	5,1

...aber Überschneidungen...

# Ist das Produktivitätsparadoxon behoben?

Sprache	LoC / FP			LoC / Mannmonat relativ konstant.
Komplexität	niedrig	mittel	hoch	=> In einer höheren Programmiersprache schafft ein Programmierer mehr Funktionalität pro Zeiteinheit.
Assembler	200	320	450	=> Es gibt Produktivitätsunterschiede zwischen Programmiersprachen.
C	60	128	170	
Fortran	75	107	160	
Cobol	65	107	150	
C++	30	53	125	
Smalltalk	15	21	40	

**Damit ist ein Teil des Produktivitätsparadoxons behoben, aber es gibt noch andere Einflußgrößen: Das Verhältnis LoC/FP hängt z.B. auch von der Systemkomplexität insgesamt ab. Aber FPs sind schon deutlich besser als LoC.**

**Erfahrungswert: Die Erstellung eines Function Points kostet etwa 1.500 €**

# Entwurfsmetriken

---

- OO-Metriken:
  - Attribute, Methoden, Anzahl Vererbungsstufen, Abhängigkeiten
- Graph-basierten Maße (gut bei UML-Modellen)
  - Zusammenhalt (auch: Kohäsion) eines Teilgraphen G: Anteil der Kanten innerhalb von G im Vergleich zur Anzahl aller Kanten, an denen G beteiligt ist.
  - Kopplung zwischen zwei Teilgraphen G1 und G2 eines Graphen: Anteil existierender Verbindungen zwischen Knoten von G1 und G2 im Vergleich zur Anzahl aller möglichen Verbindungen
  - McCabe's Cyclomatic Complexity (bei Flussgraphen): Anzahl der linear unabhängigen Pfade ( $= \#Kanten - \#Knoten + 2$ )

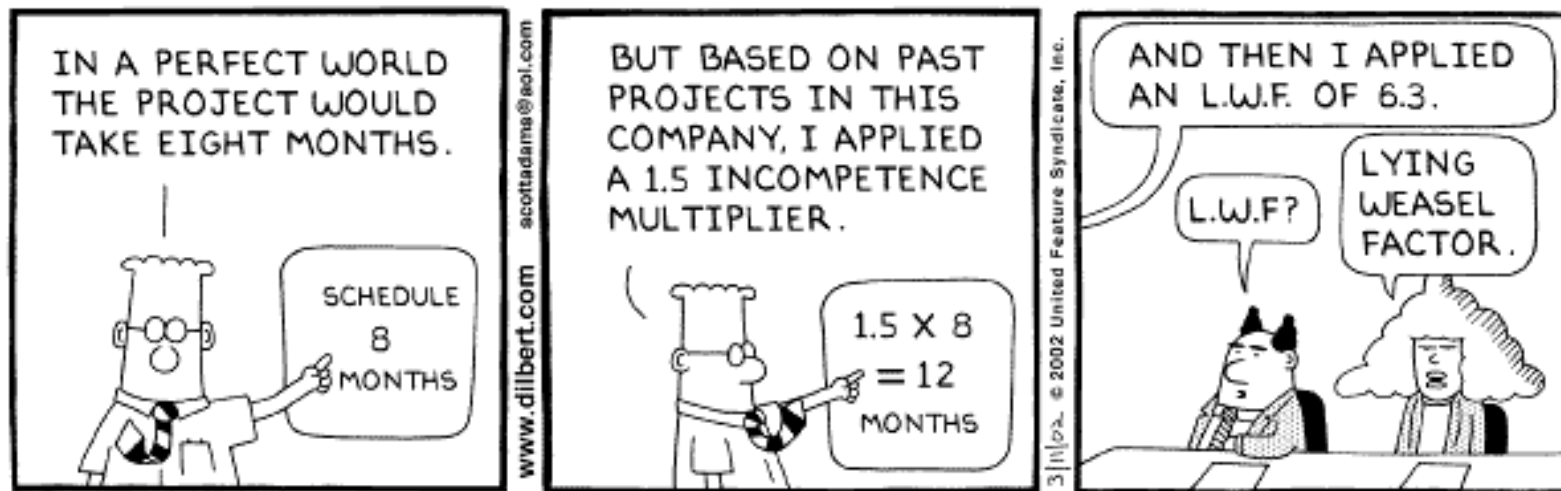
# Schätzklausur

---

- **Phase 1: Auftragsdefinition**
  - Auftraggeber, Kunde, Verantwortliche und Durchführende festlegen (5-Rollen-Bild)
  - Ziel definieren
  - Aufgaben analysieren (Prozesszerlegung)
- **Phase 2: Schätzungsvorbereitung**
  - Geeignete Leute einladen
  - Fachleute, Verantwortliche, Durchführende
  - 2 Stunden ansetzen
  - Excel-Sheet vorbereiten
- **Phase 3: Schätzung**
  - Projekt und Projektplan vorstellen, Aufwands- und Zeitplan austeilen, ggf. korrigieren (lassen)
  - Jeder schreibt seine Schätzungen auf Ansage auf
  - Jeder sagt seine Schätzungen auf Ansage an
  - Extrema diskutieren, ggf. entfernen
  - Mittelwerte bilden
  - Summe bilden

# Schätzklausur

**Meistens gibt es anschließend noch eine Phase 4, in der die Schätzung interessengeleitet verändert wird.**



Copyright © 2002 United Feature Syndicate, Inc.

# Schätzklausur

**Sind Schätzungen notorisch ungenau – wieso?**

**In wohlgeordneten Prozessen sind Schätzungen von einem zum nächsten Mal vergleichbar.**

**Man kann also aus Fehlern lernen.**

**Nicht so bei ungeordneten Prozessen.**

**Hier werden Fehler versteckt oder verdeckt.**



# 4. Risikomanagement

---

- Ein Risiko beschreibt die Möglichkeit, dass eine Aktivität unerwünschte Folgen hat (ein potentiell Problem)
- Aufgabe des Risikomanagements: Risiken bewerten und beherrschen
- Bewertung der Risiken
  - Einzelfaktoren bestimmen (z.B. per Schätzklausur)
    - Eintretenswahrscheinlichkeit
    - Auswirkungen (i.d.R. Kosten des Eintretens eines Risiko)
  - $\text{Kosten} = \text{Eintretenswahrscheinlichkeit} * \text{Auswirkungskosten}$
- Entscheidungsfindung

# Fehler und Risiken

---

## 1. Menschen machen Fehler

- Ungeeignete Mitarbeiter
- Unzureichende Anwenderbeteiligung

## 2. Prozesse sind oft mangelhaft

- Übermäßig optimistischer Zeitplan
- Scheitern von Subunternehmern
- Mangelhafter Entwurf

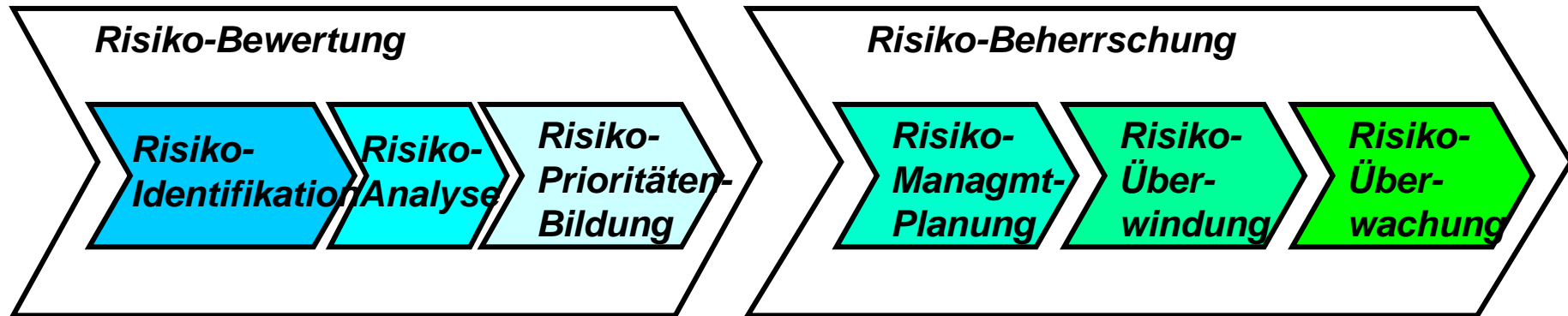
## 3. Produkt ist nicht klar definiert / wandelt sich

- „Feature creep“
- „Goldene Wasserhähne“

## 4. Technikgläubigkeit („Silver-Bullet“)

- Überschätzung neuer Tools oder Methoden
- Wechsel der Infrastruktur mitten im Projekt
- Verzicht auf Versionsverwaltung / Änderungsmanagement

# Risikotechniken



- Checklisten
- Vergleich mit Erfahrungen

- Leistungs- u. Kostenmodelle
- Analyse der Qualitätsanforderungen

- Risikofaktoren u. -Wirkung bestimmen

- Risikoplan erstellen, in Projektplan integrieren

- Prototypen erstellen
- Anforderungen lockern
- ...

- Meilensteine u. Top10 Risiken verfolgen
- Risiko neu bewerten u. planen

# Beispiel: Risikobewertung

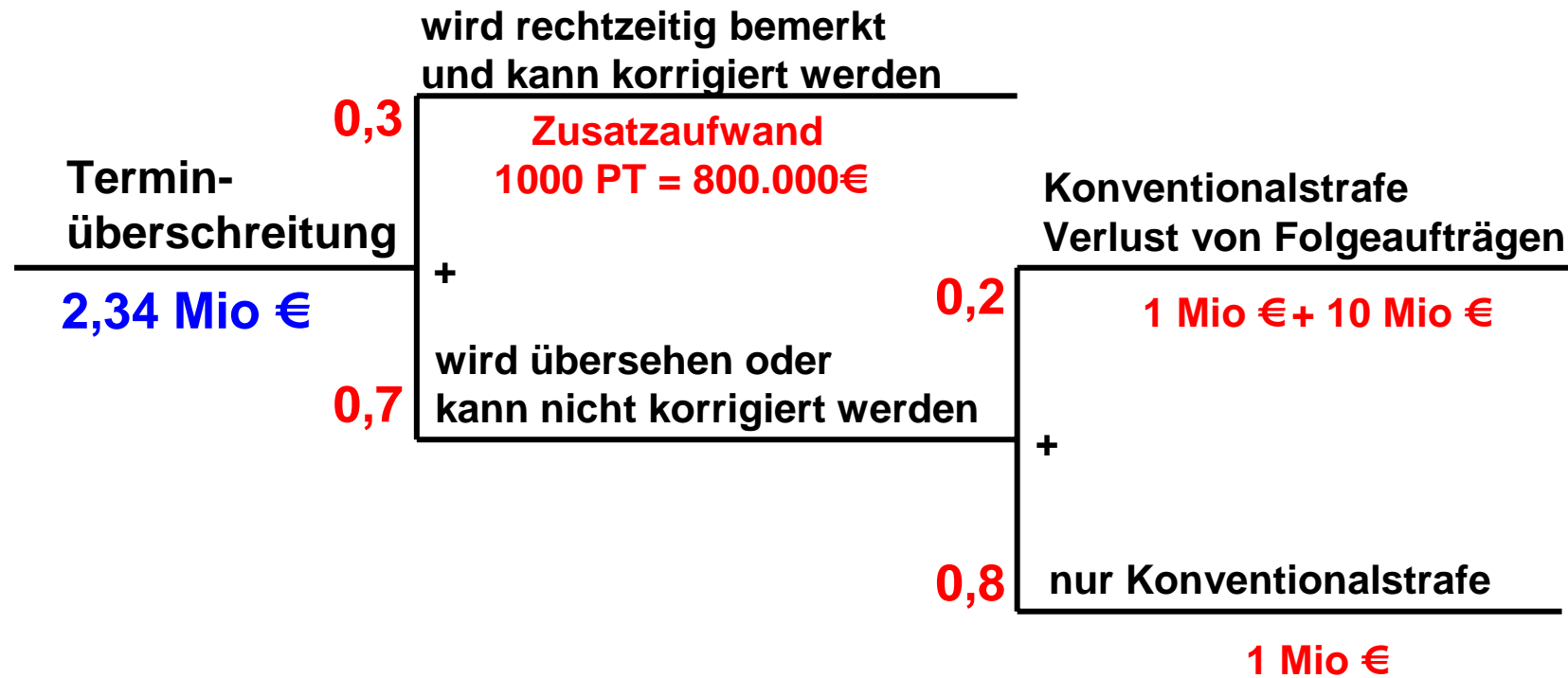
---

## Pendenzenliste zur Priorisierung von Risiken

lfd.Nr.	Beschreibung	Wahrschein.	Auswirkung	Gewicht	Gegenmaßnahme	verantwortlich
1	Zulieferer fällt aus	30%	100 T	30 T	2. Zulieferer	Störrle
2	Terminüberschreitung	11%	2,34 M€	257 T€	keine	Wirsing

# Beispiel: Risikoanalyse

## Auswirkungen mit Ereignisbaum abschätzen

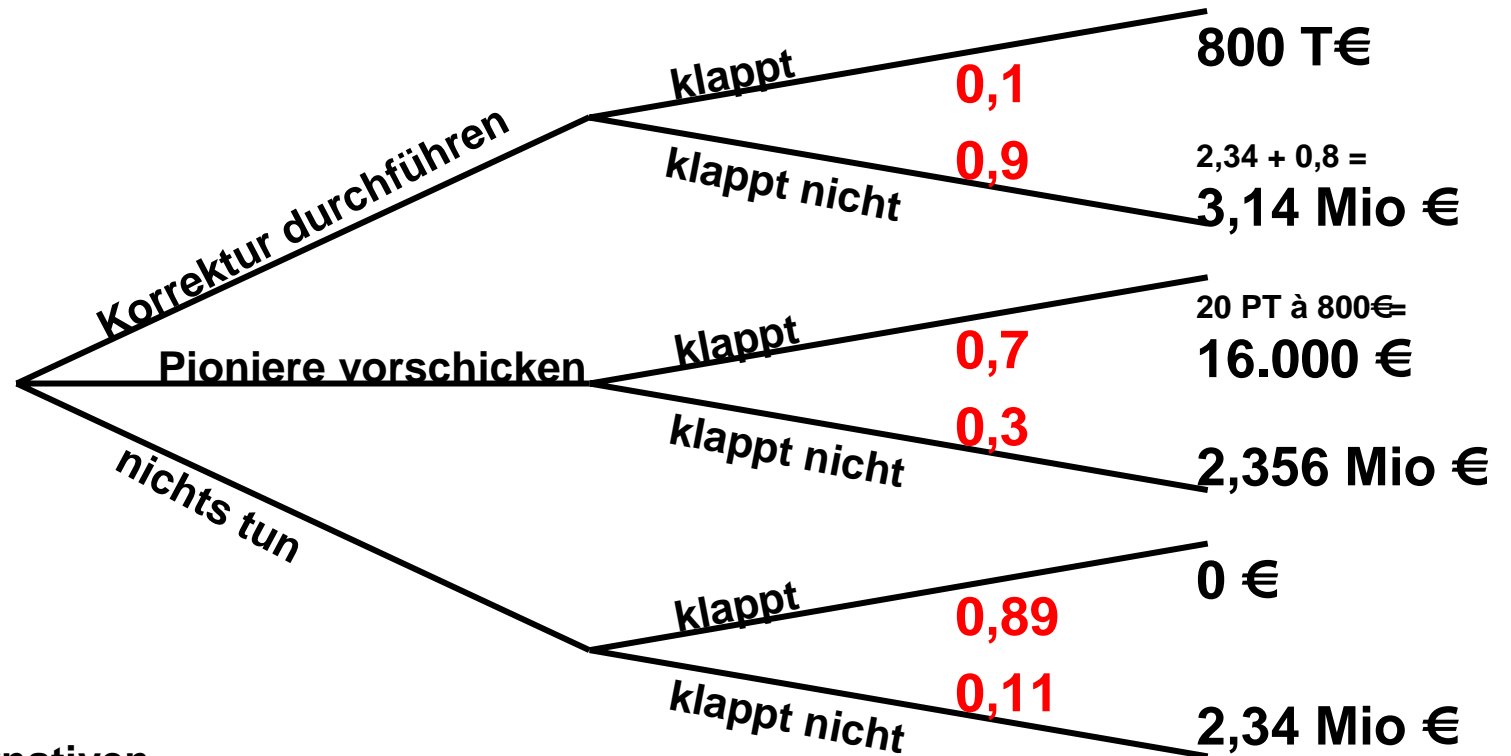


$$0,3 * 800.000 € + 0,7 * (0,2 * 11 \text{ Mio €} + 0,8 * 1 \text{ Mio €}) = 2,34 \text{ Mio €}$$

**errechnet**  
**geschätzt**

# Beispiel: Risikoprioritätenbildung

## Maßnahmen durch Entscheidungsbaum vergleichen



### Vergleich der Alternativen

nichts tun:	$0,11 * 2,34 \text{ Mio€}$	$= 257.400 \text{ €}$
Pioniere:	$0,7 * 16,000 \text{ €} + 0,3 * 2,356 \text{ Mio€}$	$= 718.000 \text{ €}$
Korrektur:	$0,1 * 0,8 \text{ Mio €} + 0,9 * 3,14 \text{ Mio€}$	$= 2,906 \text{ Mio €}$

# Zusammenfassung

---

- **Kernaufgaben des Projektmanagements sind**
  - Abschätzung von Risiken und Aufwänden,
  - Planung von Ergebnissen, Aktivitäten, Meilensteinen und Terminen,
  - Organisation und Controlling der Umsetzung,
  - Interaktion mit dem Kunden und Koordination von Zulieferern.
- **Die zentralen Kennzahlen von Projekten werden im „magischen Viereck“ beschrieben.**
  - Seine Ecken sind Zeit, Geld, Qualität und Funktionalität.
- **Typische Hilfsmittel im Projektmanagement sind**
  - Gantt- und PERT-Diagramme, Fehlerbäume und Schätzklausuren,
  - Statusberichte und -Meetings, Ablagestrukturen, und Pendenzenlisten.
- **Schätzmaße für Softwareprodukte sind:**
  - Typische Größenmaße sind Lines-Of-Code (LoC) und Non-Comment-Source-Statements.
  - Typische Komplexitätsmaße sind z.B. Function Points, Zyklomatische Zahl, Kopplung/Kohäsion, Vererbungstiefe.
- **Aufgabe des Risikomanagements ist es Risiken zu bewerten und beherrschen.**

# Literatur für E.3 – einige Klassiker

---

- De Marco, Lister: **Wien wartet auf Dich!** Hanser, 1991. (original: **Peopleware**)
- Brooks: **The mythical Man-Month.** Addison-Weseley, 1975
- Brooks: **No Silver Bullet – Essence and Accidents of Software Engineering.** IFIP '86
- Boehm: **Software Engineering Economics.** Prentice Hall, 1981
- Jones: **Programming Productivity.** McGraw-Hill, 1986

# Literatur für E.3 – spezielle Themen

---

- **Organisation allgemein**
  - Steinbuch: Organisation. Kiehl, 11.Aufl. 2000
  - Weidner: Organisation in der Unternehmung. Hanser, 2000
- **Neuere ökonomische Betrachtungen zum SE**
  - Davis: The Art of Requirements Triage. IEEE Computer March 2003
  - Boehm, Huang: Value-Based Software Engineering: A Case Study. IEEE Computer March 2003
- **Industrielle Fertigungstechnik**
  - Womack, Roos, Jones: Die zweite Revolution in der Automobilindustrie Campus, 1991
  - Hammer, Champy: Business Process Reengineering, Campus, 1995
- **Individuelle Arbeitsorganisation**
  - Malik: Führen, Leisten, Leben DVA, 2001
  - Krause: Die Kunst des Krieges für Führungskräfte (nach Sun Tzu) Uebereuther, 1995