

# Kapitel 7 - Schätzmethoden

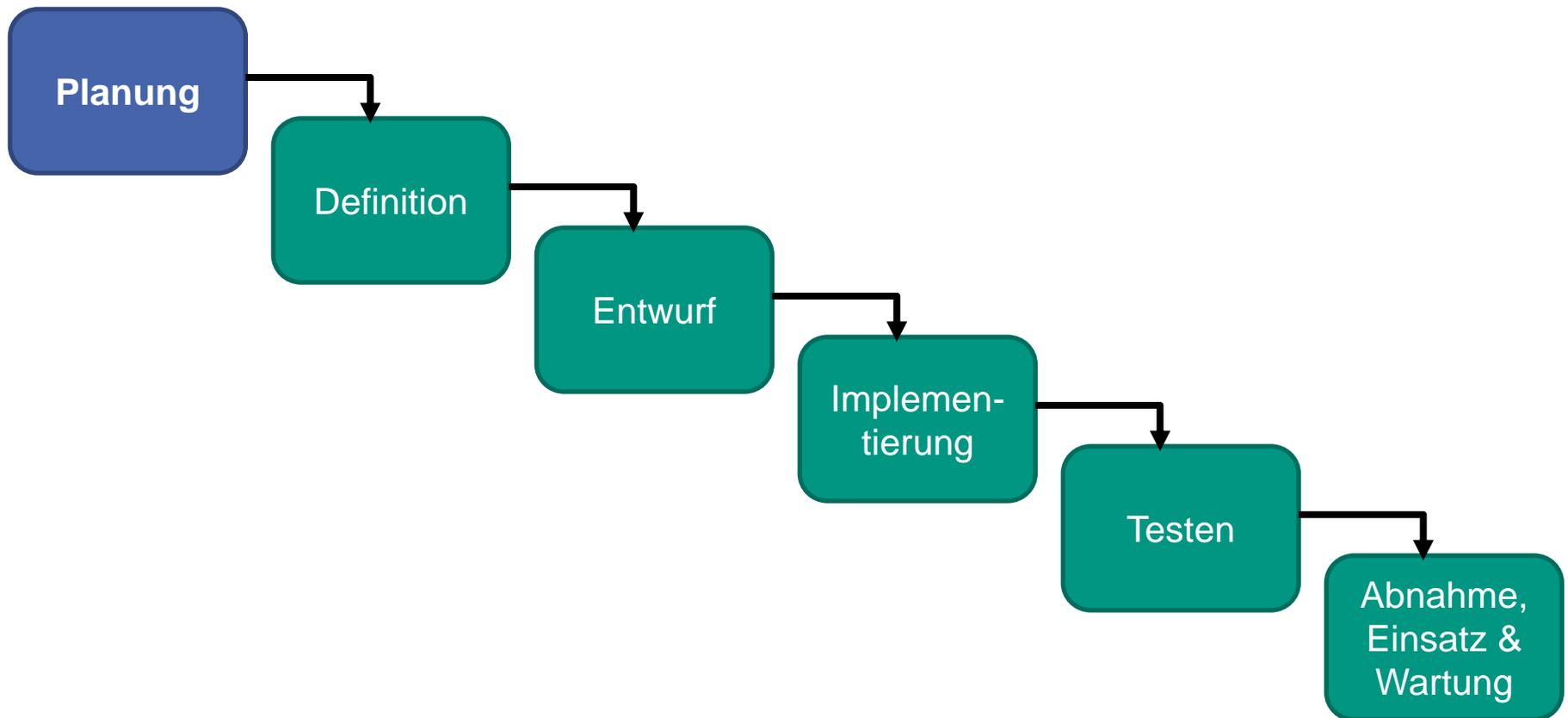
SWT I – Sommersemester 2010

Walter F. Tichy, Andreas Höfer, Korbinian Molitorisz

IPD Tichy, Fakultät für Informatik

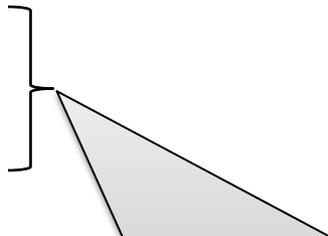


# Wo sind wir gerade?



# Lernziele

- Die aufgeführten Schätzmethoden **wiedergeben** und auf Beispiele **anwenden** können
- Inhalt
  - Einflussfaktoren der Aufwandsschätzung
  - Basismethoden der Aufwandsschätzung
  - COCOMO II



Nach: **Balzert, H.**, *Lehrbuch der Software-Technik - Softwareentwicklung*, 2. Auflage, 2000, Spektrum, Kapitel 1, S. 74ff.

# Aufwandsschätzung

- Die Aufwandsschätzung ist eine der **unbeliebtesten** Tätigkeiten in der Softwaretechnik, doch sie ist **entscheidend** für den wirtschaftlichen Erfolg eines Softwarehauses.
- Wirtschaftlichkeit eines Produktes:
  - Gewinn (Verlust) = Deckungsbeitrag \* geschätzte Menge - einmalige Entwicklungskosten
  - wobei
    - Deckungsbeitrag = Preis - laufende **variable** Kosten

# Aufwandsschätzung

- Sicht des Software-Herstellers bzw. des Auftragnehmers:
  - Entwicklungskosten eines Software-Systems:
    - Hauptanteil der Entwicklungskosten: **Personalkosten**
    - Anteilige Umlegung der **Rechner-** und **Softwarekosten** für die Produktentwicklung
    - Anteilige Kosten für andere Dienstleistungen, **Schulungen**, **Büromaterial**, Druckkosten, Dokumentation, Reisekosten, usw.

# Aufwandsschätzung

- Methoden zur Kosten- und Termschätzung
  - Die meisten Modelle basieren auf dem geschätzten Umfang des zu erstellenden Software-Produktes in Anzahl der Programmzeilen bzw. in **Lines of Code** (LOC) oder **1000 Lines of Code** (KLOC)
    - Bei höheren Sprachen werden z.B. alle Vereinbarungs- und **Anweisungszeilen** geschätzt (ohne Kommentarzeilen)
    - Der geschätzte Umfang wird durch einen **Erfahrungswert** für die **Programmierproduktivität** (in LOC) eines Mitarbeiters pro Jahr oder Monat geteilt
    - Ergebnis: geschätzter Aufwand in Personenjahren (PJ) oder Personenmonaten (PM) (auch Mitarbeiterjahre (MJ) bzw. Mitarbeitermonate (MM))
    - 1 PJ = 9 PM oder 10 PM.

# Aufwandsschätzung

- Ein **Personenmonat (PM)** ist die Menge an Arbeit, die eine Person durchschnittlich in einem Monat schafft.
  - Ein PM besteht aus 4 Personenwochen (PW)
  - Eine PW besteht aus 5 Personentagen (PT)
  - Ein PT besteht aus 8 Personenstunden
- In einem Kalendermonat arbeitet eine Person also  $4 \text{ PW/PM} * 5 \text{ PT/PW} * 8 \text{ h/PT} = 160 \text{ h}$
- Bei einer Produktivität von ca. 3.500 LOC/PJ und 10 PM/PJ ergeben sich **350 LOC/PM**, also ca. **2,12 LOC/Personenstunde**.

# Einflussfaktoren

- Der ermittelte Aufwand wird durch die nach der Terminvorgabe zur Verfügung stehende Entwicklungszeit geteilt
  - Ergebnis: Die Zahl der für die Entwicklung einzusetzenden, parallel arbeitenden Mitarbeiter
- Faustregeln
  - Eine durchschnittliche Software-Entwicklung liefert ungefähr **350 getestete Quellcodezeilen** (Kommentarzeilen nicht gezählt) pro Ingenieurmonat
  - Dabei umfasst die Ingenieurzeit alle Phasen von der Definition bis zur Implementierung.

# Einflussfaktoren

## ■ Beispiel

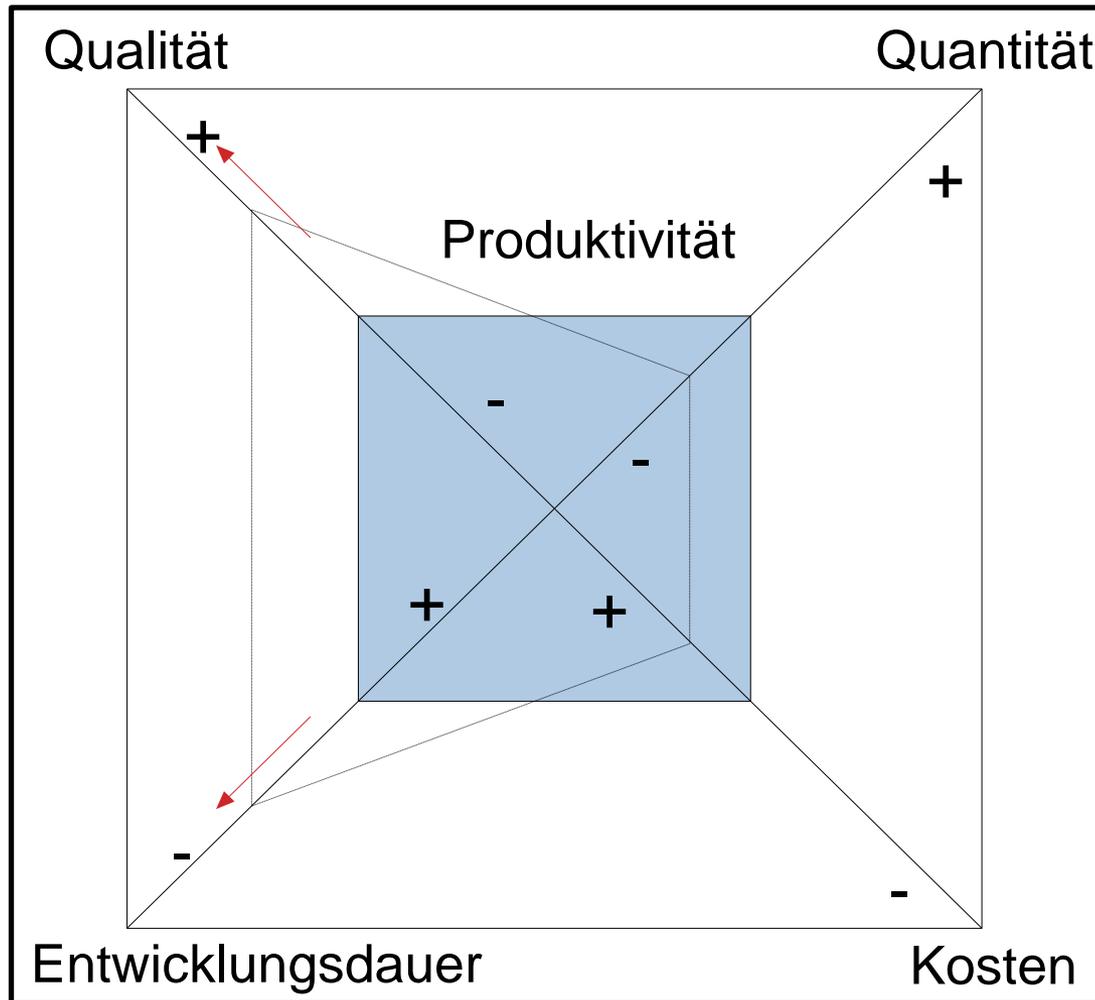
- Es soll ein Software-Produkt mit geschätzten 21.000 LOC realisiert werden
- Durchschnittliche Produktivität pro Mitarbeiter: 3.500 LOC/Jahr
- 6 Mitarbeiterjahre werden benötigt
- Also:
  - Arbeiten 3 Mitarbeiter im Team zusammen, dann werden 2 Jahre bis zur Fertigstellung benötigt
  - Muss das Produkt innerhalb von 6 Monaten fertiggestellt werden, sind dafür 12 Mitarbeiter nötig

Ist die Korrelation zwischen Aufwand und Entwicklungsdauer linear?

# Einflussfaktoren

- Wichtige Einflussfaktoren
  - Quantität
  - Qualität
  - Entwicklungsdauer
  - Kosten
- Beeinflussen sich gegenseitig – veranschaulicht im „Teufelsquadrat“

# Das „Teufelsquadrat“



Man kann an den Ecken des inneren Vierecks ziehen, dabei muss aber seine **Fläche gleich** bleiben

# Einflussfaktoren: Quantität

- Umfang
  - Maß „Anzahl Programmzeilen“ (KLOC)
  - lineare oder überproportionale Beziehung zwischen KLOC und dem Aufwand
  - Maß abhängig von Programmiersprache
- Komplexität
  - qualitative Maße „leicht“, „mittel“ und „schwer“
  - Abbildung auf Zahlenreihe
    - Beispiel: Noten zwischen 1 und 6.

# Einflussfaktoren: Qualität

- Je höher die **Qualitätsanforderungen**, desto größer ist der **Aufwand**
- Es gibt nicht „die Qualität“, sondern es gibt verschiedene Qualitätsmerkmale
- Jedem Qualitätsmerkmal lassen sich Kennzahlen zuordnen

# Einflussfaktoren: Entwicklungsdauer

- Soll die Zeit verkürzt werden, dann werden mehr Mitarbeiter benötigt
- Mehr Mitarbeiter erhöhen den **Kommunikationsaufwand** innerhalb des Entwicklungsteams
- Der höhere Kommunikationsanteil reduziert die Produktivität
- Kann die Entwicklungsdauer verlängert werden, dann werden weniger Mitarbeiter benötigt
  - Der Kommunikationsanteil sinkt
  - Die Produktivität jedes Mitarbeiters steigt
- Der Zusammenhang zw. **Aufwand** in PM und **Entwicklungsdauer** ist also nur **annähernd** linear.

# Einflussfaktoren: Produktivität

- Wird von vielen verschiedenen Faktoren beeinflusst
  - **Lernfähigkeit** und Motivation der Mitarbeiter
  - Verwendete **Programmiersprachen**, Methoden und Werkzeuge
  - **Ausbildung/Vertrautheit** mit Anwendungsgebiet
  - Arbeitsklima

# Basismethoden: Analogiemethode

- Vergleich der zu schätzenden Entwicklung mit bereits abgeschlossenen Produktentwicklungen **anhand von Ähnlichkeitskriterien**
- Kriterien
  - gleiches oder ähnliches Anwendungsgebiet
  - gleicher oder ähnlicher Produktumfang
  - gleicher oder ähnlicher Komplexitätsgrad
  - gleiche Programmiersprache
  - gleiche Programmierumgebung

# Analogiemethode: Beispiel

- Pascal-Übersetzer = 20 PM
- Modula-2-Übersetzer = ?
  - 20% neue Konstrukte
  - 50% des Codes wiederverwendbar
  - 50% müssen überarbeitet werden
- Schätzung
  - 50% leicht modifiziert:  $10 \text{ PM} * 0,25 = 2,5 \text{ PM}$
  - 50% völlige Überarbeitung: 10 PM
  - 20% zusätzliche Neuentwicklung hoher Komplexität:
    - $4 \text{ PM} * 1,5$  (Komplexitätszuschlag) = 6 PM
- Schätzung Modula-2-Übersetzer = 18,5 PM.

# Basismethoden: Analogiemethode

- Vorteile:
  - Relativ einfache und intuitive Schätzmethode
- Nachteile:
  - intuitive, globale Schätzung aufgrund individueller Erfahrungen, **nicht übertragbar**
  - fehlende allgemeine Vorgehensweise

# Basismethoden: Relationsmethode

- Das zu schätzende Produkt wird direkt mit **ähnlichen Entwicklungen** verglichen
- Aufwandsanpassung erfolgt mit **Erfahrungswerten**
- Im Gegensatz zur Analogiemethode stehen für die Aufwandsanpassung **Faktorenlisten** und **Richtlinien** zur Verfügung

# Relationismethode: Faktoren

## ■ Faktoren:

Programmiersprache		Programmiererfahrung	
C	= 100	5 Jahre	= 80
COBOL	= 120	3 Jahre	= 100
ASSEMBLER	= 140	1 Jahr	= 140

Dateiorganisation	
Sequentiell	= 80
Indexsequentiell	= 120

- Werte geben an, in welcher Richtung und wie stark die einzelnen Faktoren den Aufwand beeinflussen
- Wenn ein vergleichbares, etwa gleich großes Projekt existiert, ergeben die Faktoren Ab- oder Aufschläge

# Relationismethode: Beispiel

- Ein neues Produkt soll in C realisiert werden
  - Das Entwicklungsteam hat im Durchschnitt 3 Jahre Programmiererfahrung
  - Es ist eine indexsequentielle Dateioorganisation zu verwenden
- Zum Vergleich: Ähnliche Entwicklung in Assembler programmiert
  - Sequentielle Dateioorganisation
  - Team mit 5 Jahren Programmiererfahrung

- Wenn alle 3 Faktoren den Aufwand gleichgewichtig beeinflussen

Assembler zu C:	140 zu 100	= -40 Punkte
5 Jahre zu 3 Jahre:	80 zu 100	= +20 Punkte
sequentiell zu indexsequentiell:	80 zu 120	= +40 Punkte

- Es ergibt sich ein Mehraufwand von 20 Prozent
- Wenn das neue Projekt vom Umfang her größer oder kleiner ist, kann man aufgrund des Größenunterschiedes einen weiteren Auf- oder Abschlag ausrechnen.

# Basismethoden: Multiplikatormethode

- Das zu entwickelnde System wird soweit in Teilprodukte zerlegt, bis jedem Teilprodukt ein bereits **feststehender** Aufwand zugeordnet werden kann (z.B. in LOC)
- Der Aufwand pro Teilprodukt wird meist durch **Analyse** vorhandener Produkte ermittelt
- Oft werden auch die Teilprodukte bestimmten Kategorien zugeordnet wie
  - Steuerprogramme
  - E/A-Programme
  - Datenverwaltungsroutinen
  - Algorithmen usw.

## Basismethoden: Multiplikatormethode

- Die Anzahl der Teilprodukte, die einer Kategorie zugeordnet sind, wird mit dem Aufwand dieser Kategorie multipliziert
- Die erhaltenen Werte für eine Kategorie werden dann addiert, um den Gesamtaufwand zu erhalten
- Auch „Aufwand-pro-Einheit-Methode“ genannt.

# Multiplikatormethode: Beispiel

- Die Aufteilung eines zu schätzenden Produkts in Teilprodukte hat folgendes ergeben:

Kategorie	Teilprodukte LOC	Summe LOC	Aufwands- faktor	LOC bewertet
Steuerprogramm	$1 \cdot 500$	500	1,8	900
E/A-Programme	$1 \cdot 700 + 2 \cdot 500$	1700	1,5	2550
Datenverwaltung	$1 \cdot 800 + 2 \cdot 250$	1300	1,0	1300
Algorithmen	$1 \cdot 300 + 5 \cdot 100$	800	2,0	1600
<b>Summe</b>				<b>6350</b>

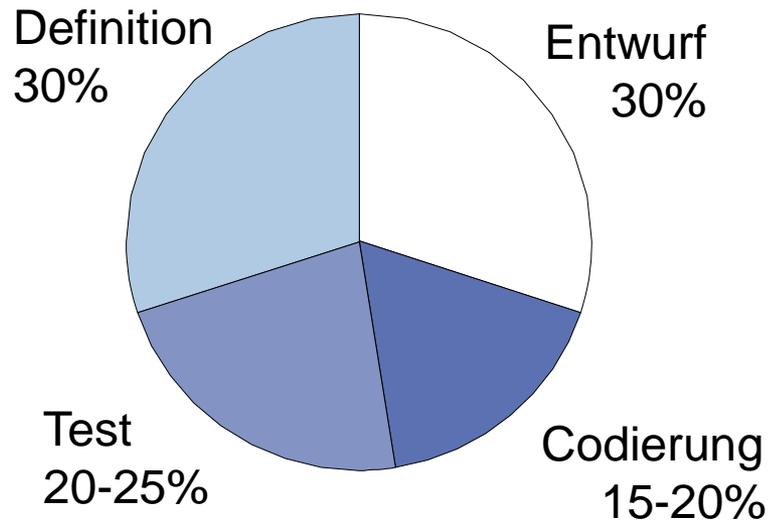
# Multiplikatormethode: Bewertung

- Es ist eine **umfangreiche**, **empirische** Datensammlung und -auswertung erforderlich, um die zu berücksichtigenden Faktoren unternehmensspezifisch zu bestimmen.
- Außerdem ist eine **Umrechnung** der bewertete LOC in PM erforderlich.
- Die Faktoren und die Aufteilung in Kategorien müssen **permanent** überprüft werden, um den **technischen Fortschritt** zu berücksichtigen

# Basismethoden: Phasenaufteilung

- Aus **abgeschlossenen Entwicklungen** wird ermittelt, wie sich der Aufwand auf die einzelnen **Entwicklungsphasen** verteilt hat.
- Bei **neuen Entwicklungen** wird
  - eine Phase entweder zunächst **vollständig abgeschlossen** und aus dem Ist-Aufwand dann anhand der Aufwandsverteilung der Aufwand für die restlichen Phasen geschätzt, oder
  - es wird eine **detaillierte Schätzung** einer Phase durchgeführt und hieraus dann auf den Gesamtaufwand geschlossen

# Basismethoden: Phasenaufteilung



Wenn hier die Definitionsphase z.B. **6 Monate** erforderte, wird bei dieser Verteilung ein Gesamtaufwand von **20 Monaten** angenommen.

- Vorteil: Diese Methode kann bereits **frühzeitig** eingesetzt werden, wenn der Aufwand für mindestens eine Phase bestimmt wurde.
- Nachteil: Empirische Untersuchungen haben gezeigt, dass der prozentuale Anteil der Phasen von Projekt zu Projekt stark variiert. Daher ist diese Methode **kaum brauchbar**.

# Basismethoden: Bewertung

- **Keine** der aufgeführten Basismethoden allein ist ausreichend
- Je nach **Zeitpunkt** und **Kenntnis** von aufwandsrelevanten Daten sollte die eine oder andere Methode eingesetzt werden
- Für frühzeitige, **grobe** Schätzungen benutze die
  - Analogiemethode
  - Relationsmethode
- Sind die **Einflussfaktoren während** der **Entwicklung** dann genauer bekannt benutze die
  - Multiplikatormethode

# Die Schätzmethode COCOMO II

- Berechnet aus der geschätzten **Größe** und 22 **Einflussfaktoren** die Gesamtdauer eines SW-Projektes in Personenmonaten.
- Die Größe wird in KLOC oder unjustierten **Funktionspunkten** geschätzt.
- COCOMO II ist Nachfolger des 1981 entwickelten, ersten COCOMO
  - COCOMO: *Constructive Cost Model*

# Schätzformel von COCOMO II

$$PM = A \cdot (Size)^{1,01+0,01 \cdot \sum_{j=1}^5 SF_j} \cdot \prod_{i=1}^{17} EM_i$$

- *PM*: Anzahl Personenmonate
- *A*: Konstante für die Kalibrierung des Modells (z.B. für LOC oder Funktionspunkte)
- *Size*: geschätzter Umfang der Software in KLOC oder unjustierten Funktionspunkten
- *SF<sub>j</sub>*: Skalierungsfaktoren (engl. *scale factors*)
- *EM<sub>i</sub>*: multiplikative Kostenfaktoren (engl. *multiplicative cost drivers*)
- Durch den Exponenten > 1 wächst der Aufwand in PM etwas überproportional im Umfang.

# COCOMO II Skalierungsfaktoren

Scale Factors (SF <sub>i</sub> )	Very Low (5)	Low (4)	Nominal (3)	High (2)	Very High (1)	Extra High (0)
Precedentedness	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Architecture / risk resolution	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Team cohesion	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
Process maturity	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

# COCOMO II Multiplikative Kostenfaktoren

- **Produktfaktoren**
  - Required Software Reliability, Data Base Size, Product Complexity, Required Reusability, Documentation match to life-cycle needs
- **Plattformfaktoren**
  - Execution Time Constraint, Main Storage Constraint, Platform Volatility, Computer Turnaround Time
- **Personalfaktoren**
  - Analyst Capability, Programmer Capability, Applications Experience, Platform Experience, Language and Tool Experience, Personnel Continuity
- **Projektfaktoren**
  - Use of Modern Programming Practices, Use of Software Tools, Multisite Development, Required Development Schedule, Classified Security Application
- Für jeden der 17 Faktoren ist ein Nominalwert von 1 voreingestellt.

# COCOMO II – Beispielrechnung (1)

## ■ Annahmen:

- A hat den Wert 3
- Alle Kostenfaktoren haben den Wert 1 (Nominalwert)
- Die Skalierungsfaktoren sind:
  - SF(Very Low): Alle SF haben den Wert 5  $\rightarrow B = 1,26$
  - SF(Nominal): Alle SF haben den Wert 3  $\rightarrow B = 1,16$
  - SF(Extra High): Alle SF haben den Wert 0  $\rightarrow B = 1,01$
- Die Größe der Projekte liegt zwischen 10.000 LOC (10 KLOC) und 10.000.000 (10.000 KLOC)

$$\prod_{i=1}^{17} EM_i = 1$$

$$B = 1,01 + 0,01 \cdot \sum_{j=1}^5 SF_j$$

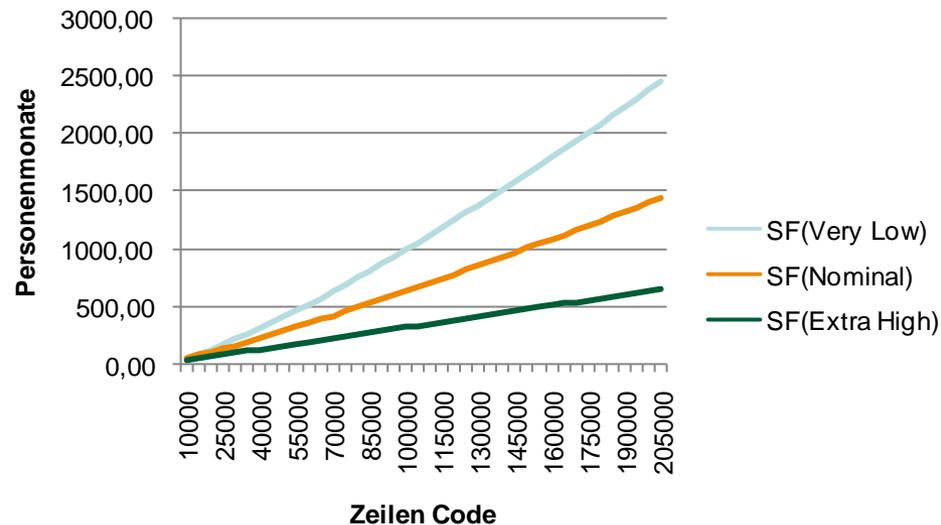
$$PM = A \cdot (Size)^{1,01+0,01 \cdot \sum_{j=1}^5 SF_j} \cdot \prod_{i=1}^{17} EM_i$$

$$\rightarrow PM = 3 \cdot (Size)^B \cdot 1 = 3 \cdot (Size)^B$$

# COCOMO II – Beispielrechnung (2)

$$PM = 3 \cdot (Size)^B$$

	10000	15000	20000	25000	30000	...	20500
<b>SF(Very Low)</b>	54,59	90,99	130,74	173,19	217,92	...	2454,32
<b>SF(Nominal)</b>	43,36	69,40	96,90	125,53	155,09	...	1441,30
<b>SF(Extra High)</b>	30,70	46,24	61,82	77,45	93,11	...	648,62



Einheit:  
Personenmonate

## COCOMO II with Heuristic Risk Assessment

**Model:** Post-architecture  
**Calibration:** COCOMOII.2000  
[Current rule base implementation](#)

### Size

	<u>SLOC</u>	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	<input type="text"/>						
Reused	<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text"/>	<input type="text"/>		
Modified	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Rate each cost driver below from Very Low (VL) to Extra High (EH). For **HELP** on each cost driver, select its name.

Very Low (VL)    Low (L)    Nominal (N)    High (H)    Very High (VH)    Extra High (EH)

### Scale Drivers

- [Precedentedness](#)     VL     L     N     H     VH     XH
- [Development Flexibility](#)     VL     L     N     H     VH     XH
- [Architecture/Risk Resolution](#)     VL     L     N     H     VH     XH
- [Team Cohesion](#)     VL     L     N     H     VH     XH
- [Process Maturity](#)     VL     L     N     H     VH     XH

### Product Attributes

- [Required Reliability](#)     VL     L     N     H     VH

# COCOMO II – Quelle und Anwendung

- Webseite:  
[http://csse.usc.edu/csse/research/COCOMOII/cocomo\\_main.html](http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html)
- COCOMO II Rechner:  
[http://sunset.usc.edu/research/COCOMOII/expert\\_cocomo/expert\\_cocomo2000.html](http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html)
- Barry Boehm et al., “Cost Models for Future Software Life Cycle Processes: COCOMO 2.0”, in Annals of Software Engineering Special Volume on Software Process and Product Measurement, 1995
- Sunita Chulani, Brad Clark, Barry Boehm, “Calibrating the COCOMO II Post Architecture Model”, International Conference on Software Engineering, 1998