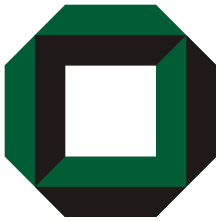


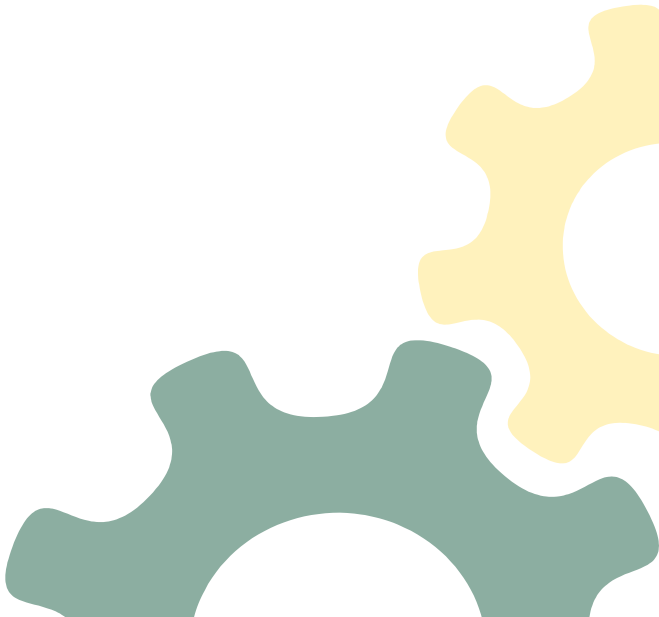
Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

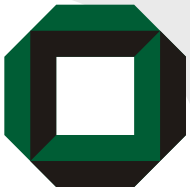
Kapitel 8
Schätzmethoden

SWT I – Sommersemester 2009

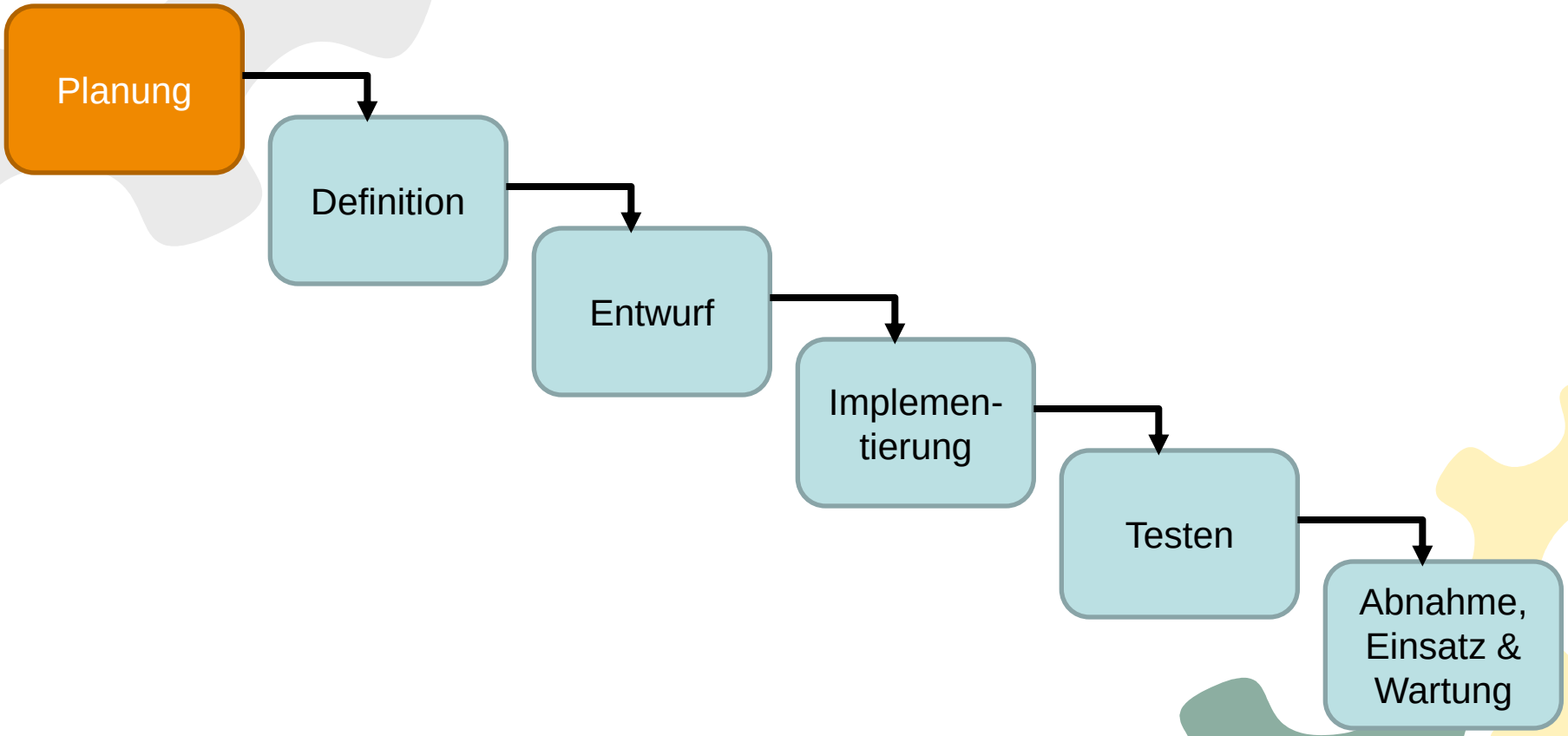
Prof. Dr. Walter F. Tichy

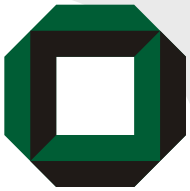
Fakultät für **Informatik**
Lehrstuhl für Programmiersysteme





Wo sind wir gerade?

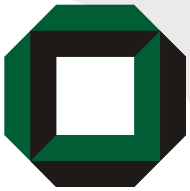




Lernziele

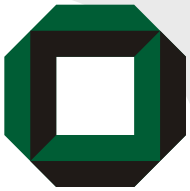
- Die aufgeführten Schätzmethoden wiedergeben und auf Beispiele anwenden können
- Inhalt
  - Einflussfaktoren der Aufwandsschätzung
  - Basismethoden der Aufwandsschätzung
  - COCOMO II

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



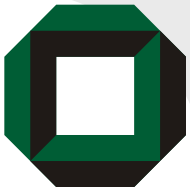
Aufwandsschätzung

- 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



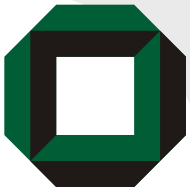
Aufwandsschätzung

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



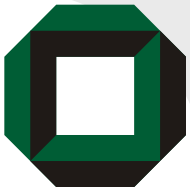
Einflussfaktoren

- Methoden zur Kosten- und Terminschätzung
  - 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.



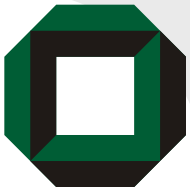
Einflussfaktoren

- 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

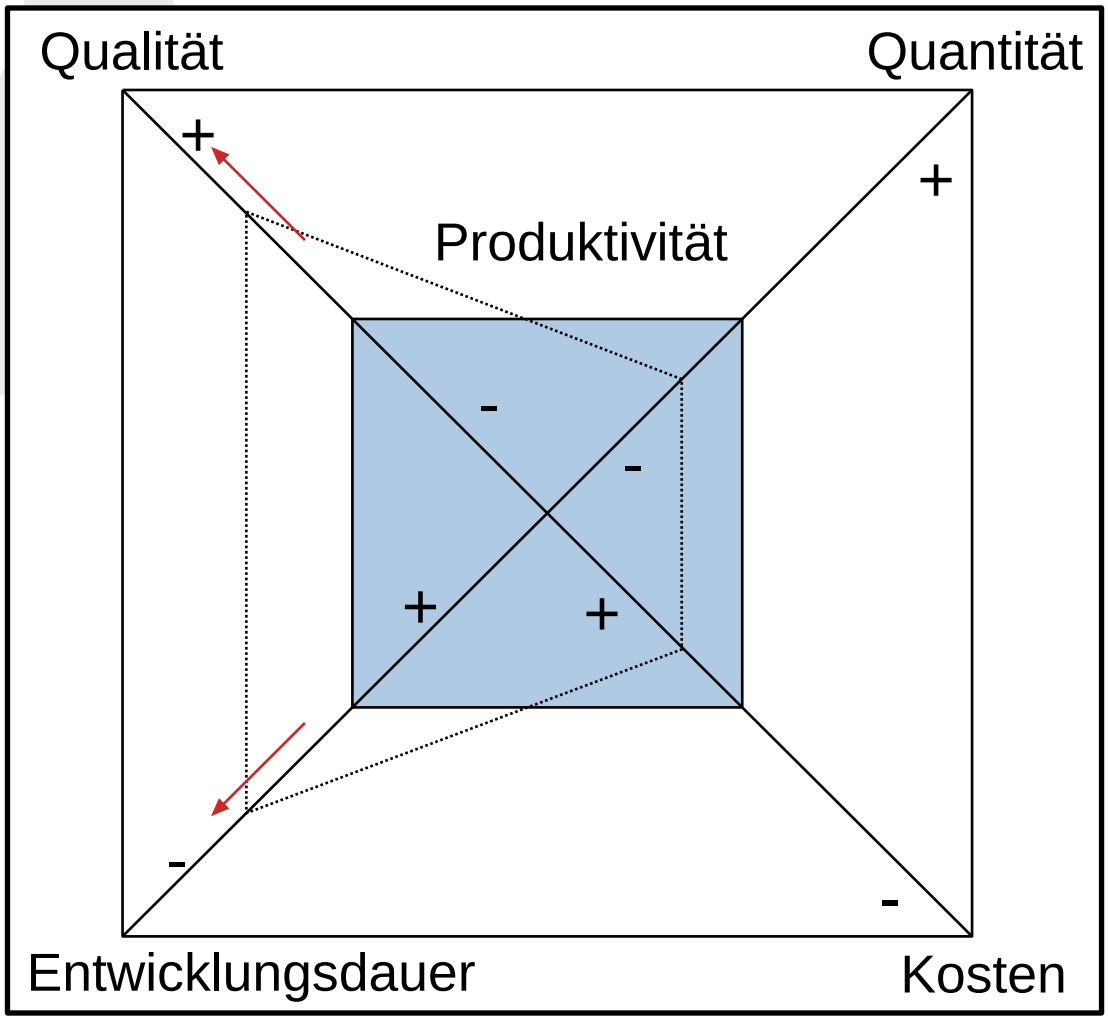
- Beispiel
  - Es soll ein Software-Produkt mit geschätzten 21.000 LOC realisiert werden
  - Durchschnittliche Produktivität pro Mitarbeiter: 3.500 LOC/Jahr
  - 6 Mitarbeiterjahre werden benötigt
  - Also: Arbeiten 3 Mitarbeiter im Team zusammen, dann werden 2 Jahre bis zur Fertigstellung benötigt.



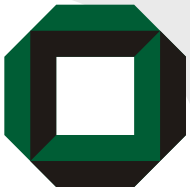
Einflussfaktoren

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

Das „Teufelsquadrat“

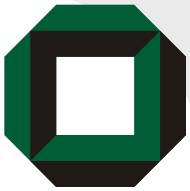


Man kann an den Ecken des inneren Vierecks ziehen, dabei muss aber seine Fläche in etwa gleich bleiben.



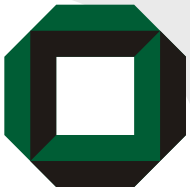
Einflussfaktoren: Quantität

- 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.



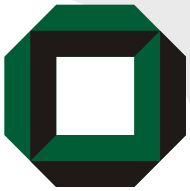
Einflussfaktoren: Qualität

- 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



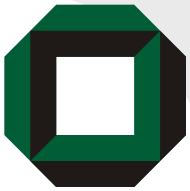
Einflussfaktoren: Entwicklungsdauer

- 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
- D.h. der Zusammenhang zw. Aufwand in PM und Entwicklungsdauer ist nur annähernd linear.



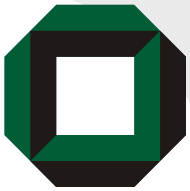
Einflussfaktoren: Produktivität

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



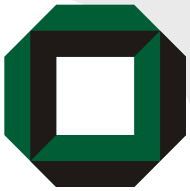
Basismethoden: Analogiemethode

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



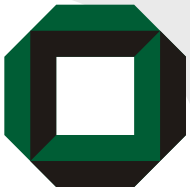
Analogiemethode: Beispiel

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



Basismethoden: Analogiemethode

- 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
- Für die Aufwandsanpassung stehen Faktorenlisten und Richtlinien zur Verfügung

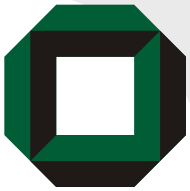
Relationsmethode: Faktoren

- 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

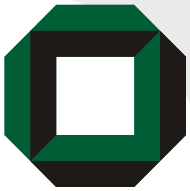


Relationsmethode: Beispiel

- Ein neues Produkt soll in C realisiert werden
  - Das Entwicklungsteam hat im Durchschnitt 3 Jahre Programmiererfahrung
  - Es ist eine indexsequentielle Dateiorganisation zu verwenden
- Zum Vergleich: ähnliche Entwicklung (in Assembler programmiert)
  - sequentielle Dateiorganisation
  - Team mit 5 Jahren Programmiererfahrung
  - Wenn alle 3 Faktoren den Aufwand gleichgewichtig beeinflussen

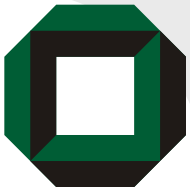
Assembler zu C:	140 zu 100	=	-40 Punkte
5 Jahre zu 3 Jahre:	80 zu 100	=	+20 Punkte
sequentiell zu indexsequentiell:	80 zu 120	=	+40 Punkte

- 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.



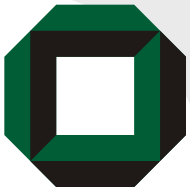
Basismethoden: Multiplikatormethode

- 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

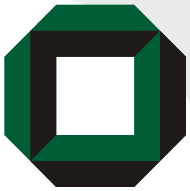
- Die Anzahl der Teilprodukte, die einer Kategorie zugeordnet sind, wird mit dem Aufwand dieser Kategorie multipliziert
- Die erhaltenen Werte für eine Kategorie werden dann addiert, um den Gesamtaufwand zu erhalten
- Auch „Aufwand-pro-Einheit-Methode“ genannt.



Multiplikatormethode: Beispiel

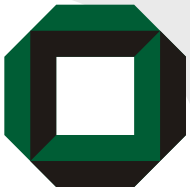
- Die Aufteilung eines zu schätzenden Produkts in Teilprodukte hat folgendes ergeben:

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



Multiplikatormethode: Bewertung

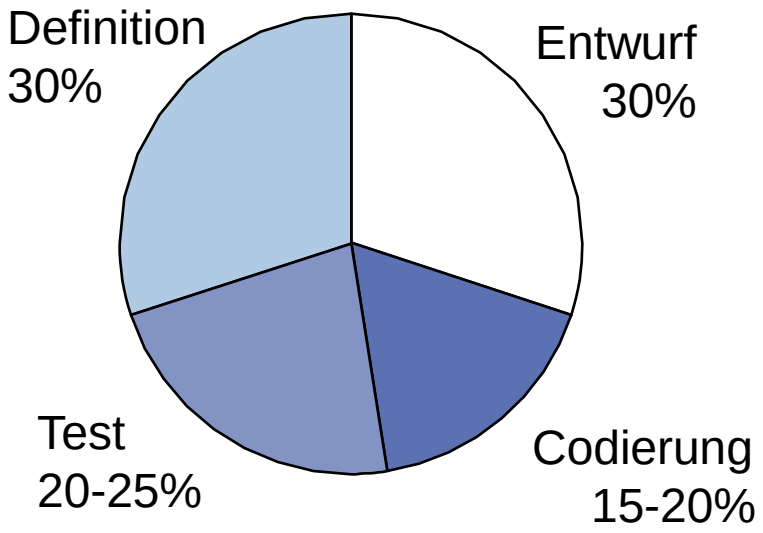
- Es ist eine umfangreiche, empirische Datensammlung und -auswertung erforderlich, um die zu berücksichtigenden Faktoren unternehmensspezifisch zu bestimmen.
- Außerdem ist eine Umrechnung der bewertete LOC in PM erforderlich.
- Die Faktoren und die Aufteilung in Kategorien müssen permanent überprüft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



Basismethoden: Phasenaufteilung

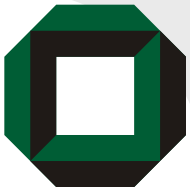
- Aus abgeschlossenen Entwicklungen wird ermittelt, wie der Aufwand sich auf die einzelnen Entwicklungsphasen verteilt hat.
- Bei neuen Entwicklungen schließt man entweder eine Phase zunächst vollständig ab und ermittelt aus dem Ist-Aufwand dann anhand der Aufwandsverteilung den Aufwand für die restlichen Phasen.
- Oder man führt eine detaillierte Schätzung einer Phase durch und schließt hieraus dann auf den Gesamtaufwand.

Basismethoden: Phasenaufteilung



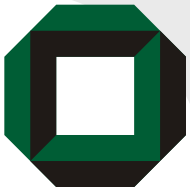
Wenn hier die Definitionsphase z.B. 6 Monate erforderte, wird bei dieser Verteilung ein Gesamtaufwand von 20 Monaten angenommen.

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



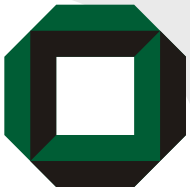
Basismethoden: Bewertung

- 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…
  - Analogieschätzung,
  - Relationsmethode.



Basismethoden: Bewertung

- Sind die Einflussfaktoren während der Entwicklung dann genauer bekannt, dann sollten die genaueren Methoden, benutze die …
  - Gewichtungsmethode oder
  - Multiplikatormethode.



Die Schätzmethode COCOMO II

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

Schätzformel von COCOMO II

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

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

COCOMO II Skalierungsfaktoren

Scale Factors (SF_i)	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	throughly 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
- Platformfaktoren
  - 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.

USC Expert COCOMO II - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html

USC Expert COCOMO II

COCOMO II with Heuristic Risk Assessment

Model: Post-architecture
Calibration: COCOMOII.2000
Current rule base implementation

Size

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New							
Reused		0	0				
Modified							

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	

Fertig

de-DE

COCOMO II – Quelle und Anwendung

- Webseite: http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html
- COCOMO II Rechner: 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