Software Engineering

Informatik II.

6. Software-Entwicklung

– Aufwandsabschätzung –

Dipl.-Inform. Hartmut Petters

Vorwort – was ich noch zu sagen hätte ...

Basis dieser Vorlesung sind vor allem die folgenden Ausarbeitungen

- Vorlesungsskript „Software Engineering“
  von Prof. Dr. Martin Glinz Universität Zürich
  http://www.ifi.unizh.ch/groups/req/courses/ses/
- Vorlesungsskript „Informatik II – Software Engineering“
  von Frau Prof. Dr. Kühn FH Karlsruhe FB W
  http://www.home.fh-karlsruhe.de/~kuin0001/inhalt.htm
- Das Buch „Software Engineering“ 6. Auflage/2001
  von Prof. Ian Sommerville University of Lancaster (UK)
  Addison Wesley ISBN 3-8273-7001-9

Konkret entnommene Beiträge sind i.d.R. mit einem Quellen-Verweis gekennzeichnet – sollte dieser fehlen bitte ich um Nachsicht.

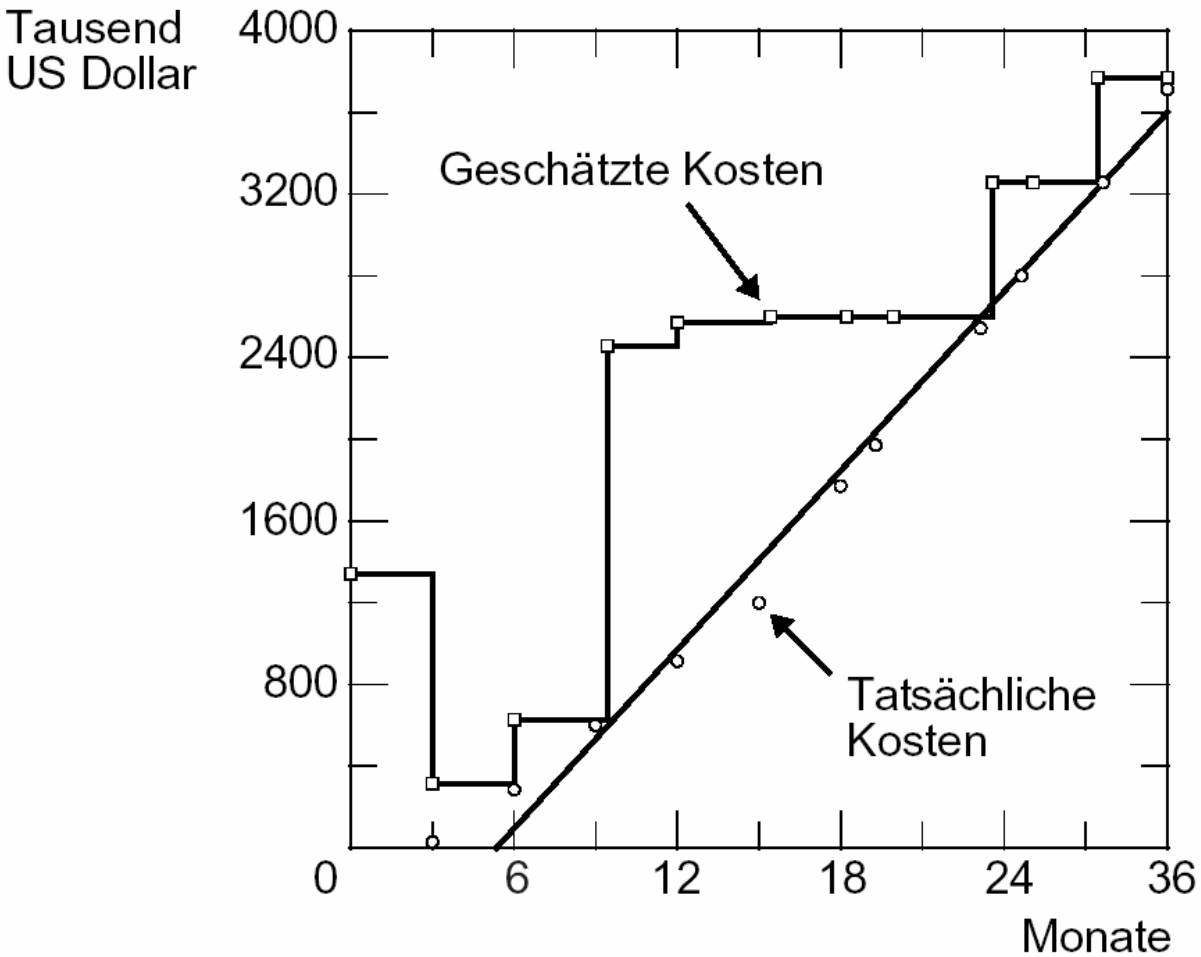
Den „roten Faden“ durch die Vorlesung habe ich dem Skript der Vorlesung von Prof. Dr. Martin Glinz entnommen und um eigene Beiträge erweitert bzw. aus den beiden anderen Quellen ergänzt.

Für die Möglichkeit der Verwendung der wesentlichen Inhalte möchte ich mich an dieser Stelle bei den Autoren herzlich bedanken.

Aufwandsabschätzung

- Voraussetzung zur Aufwandsabschätzung
  - Genaue Kenntnis der Anforderungen
    - ↳ *Was soll gemacht werden?*
  - Klare Abgrenzung der Funktionalitäten
    - ↳ *Was wird nicht gemacht + was gehört nicht dazu?*
  - Genaue Kenntnis der Anwendungsumgebung
    - ↳ *Welche Integrationen sind erforderlich?*
  - Zusammenfassung der konzeptionellen Ideen
    - ↳ *Wie könnte das Zielsystem aussehen?*
  - Dokumentation der zugrundeliegenden Architektur
    - ↳ *Wie sieht die Architektur aus +*
      warum wurde diese ausgewählt –
      Pro´s + Con´s?

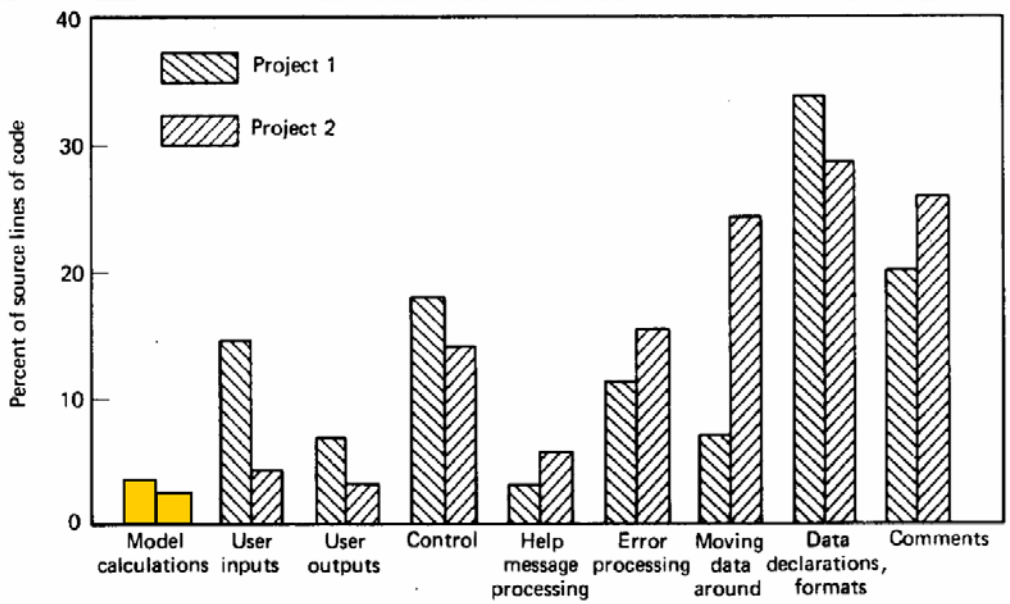
Wie laufen Kostenschätzungen ab



nach Boehm (1981)

Probleme bei der Aufwandsabschätzung

- Kreativität ist nicht kontrollierbar
- Software-Entwicklung ist Kopfarbeit
- Kernfunktionen werden mit dem Produkt verwechselt
- Erfahrungen aus Kleinprojekten werden linear extrapoliert
- Programmierer programmieren nicht zu 100%



Aufwandsabschätzung

- Projektmanager müssen die Fragen klären
  - Wieviel Aufwand ist für die Erledigung einer definierten Aufgabe erforderlich?
  - Wieviel Zeit ist für die Erledigung einer definierten Aufgabe erforderlich?
  - Wie hoch sind die anfallenden Gesamtkosten für eine definierte Aufgabe
- Projektaufwandsschätzung + Zeitplanung erfolgen gemeinsam im Team
- Kontinuierliche Aktualisierung der Aufwands-Abschätzungen, sobald mehr Infos vorliegen

Einflussfaktoren

- 3 Faktoren sind von Bedeutung
  - Hardware- und Software-Kosten (einschl. Wartung)
  - Reise- und Schulungskosten
  - Personalkosten
- Kostenfaktoren beim Personal
  - Büro-Umgebung (Räume, Heizung, Beleuchtung, ...)
  - Umlage für sonstiges Personal (Buchhaltung, ...)
  - Kosten für Netz + Kommunikation
  - Sonstige Umlagen für Einrichtungen (Bibliotheken, ...)
  - Kosten für Sozialversicherungen (KV, RV, Prämien, ...)

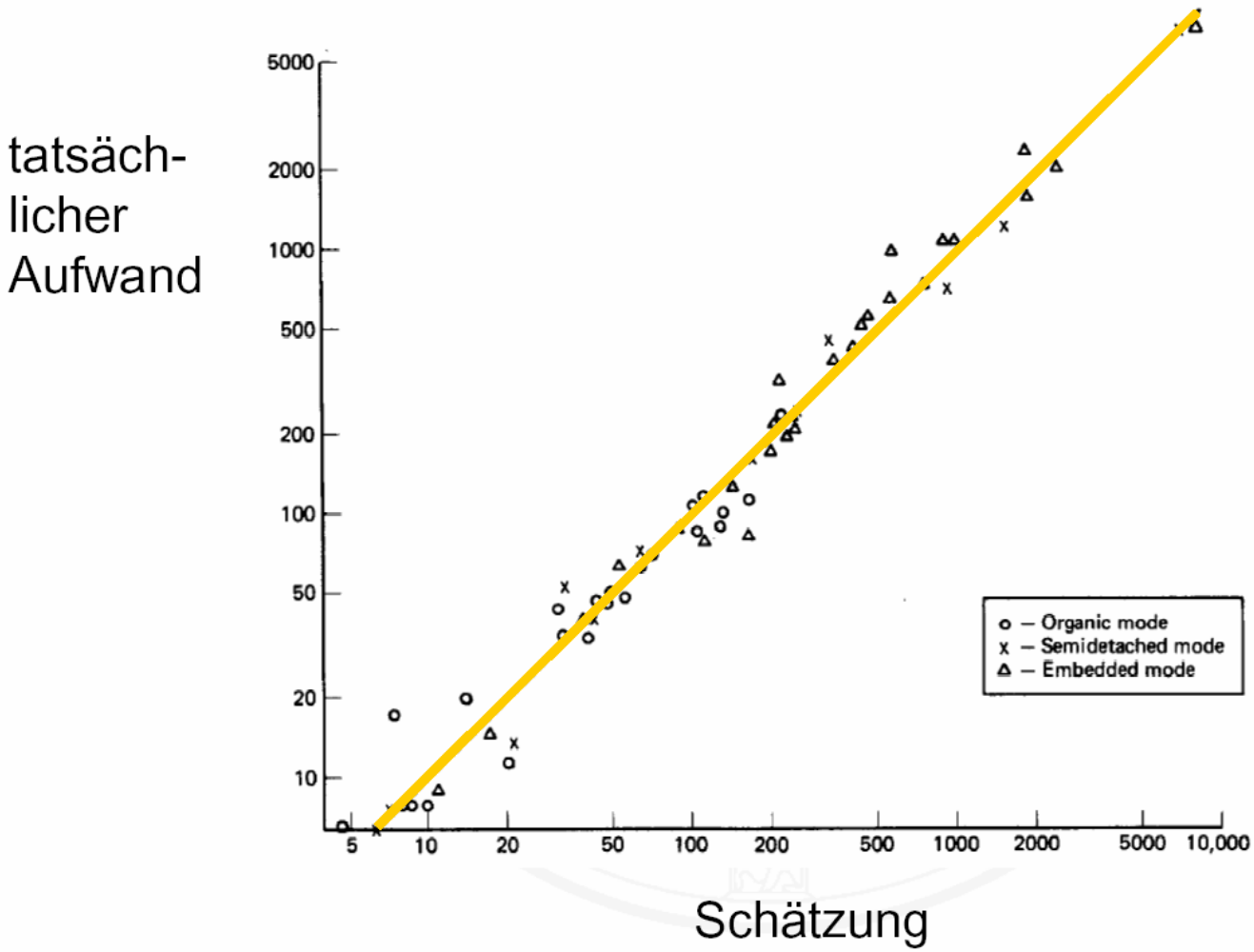
Empirische Schätzverfahren

- Analogieschlüsse basierend auf *Erfahrungen* der Schätzer
- *Experten-Beurteilung*
  - Brauchbar, wenn Erfahrungen mit gleichartigen Projekten vorhanden sind
  - Vorteil: Einfach + billig
  - Nachteil: Krasse Fehler sind möglich
- *Delphi-Methode* (wie Experten-Beurteilung nur iterativ)
  - Schätzung durch mehrere unabhängige Experten
  - Mehrere Runden zur Schätzung
  - Vorteil: Konvergiert (hoffentlich) – eliminiert Ausreißer
  - Nachteil: sehr aufwendig

Algorithmische Schätzverfahren

- Berechnung von *Kosten*- und *Durchlaufzeit-Funktionen*
- *Eingangsvariablen* müssen zutreffend geschätzt werden
- Modell muss *„kalibriert"* werden
- Liefert bei richtiger Kalibrierung die *besten Prognosen*
- Ohne *Maßzahlen* über *abgewickelte Projekte* keine zuverlässigen *Prognosen*!
- Zwei bekannte *Verfahren*
  - *COCOMO*
  - *Function Point*

Was leisten algorithmische Verfahren



Algorithmische Verfahren

- Stark abhängig von
  - Eingangsgrößen der Kostenfunktion
  - Modell muß kalibriert werden
    - ↳ Dann werden die besten Ergebnisse geliefert
- Probleme
  - Nur einsetzbar nach entsprechender Vorarbeit (Kalibrierung)
  - Stark abhängig von der Genauigkeit der Eingangsgrößen

COCOMO (Constructive Cost Model)

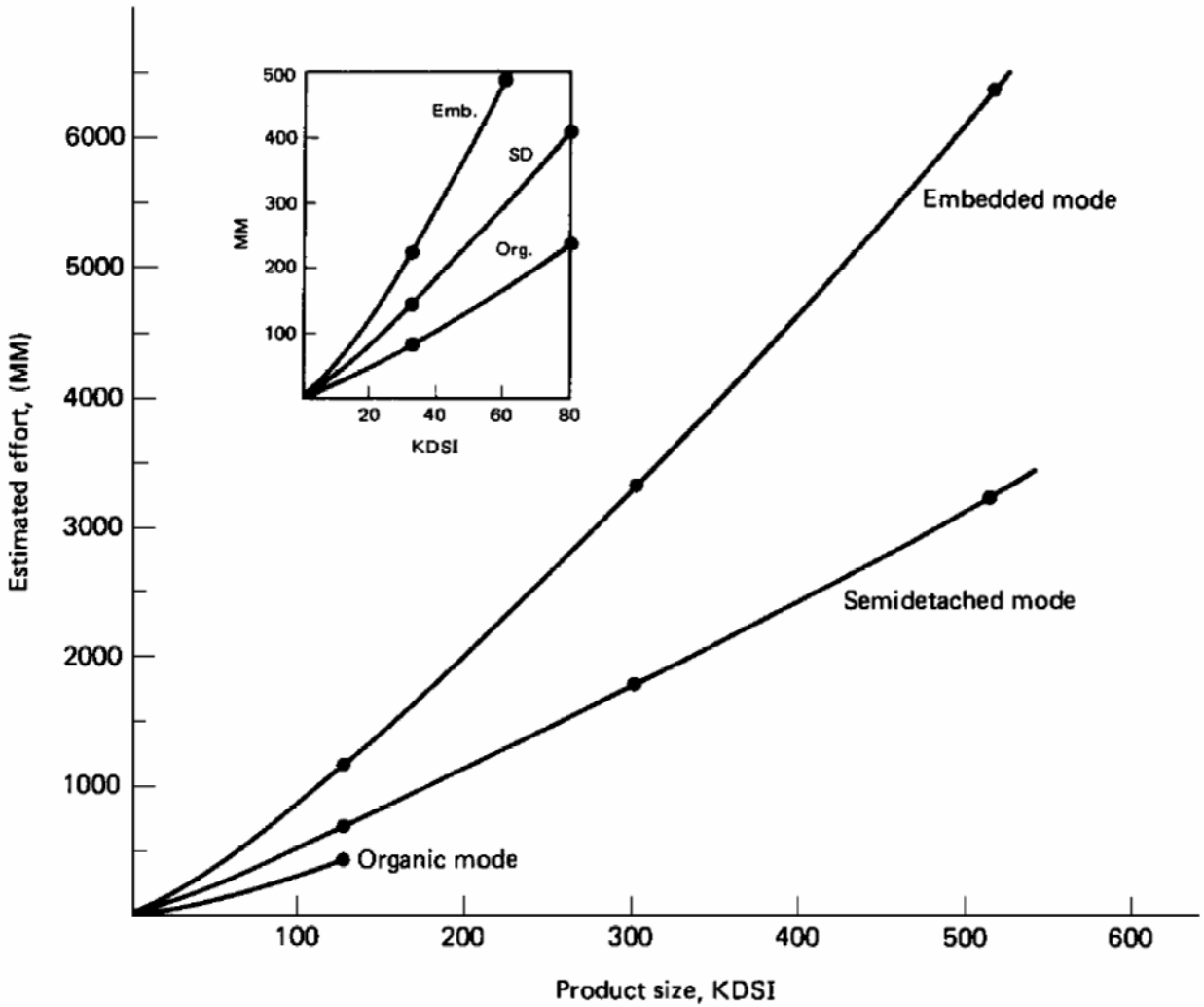
- Bekanntes algorithmisches Schätzverfahren
- Geht von der Schätzung der Produktgröße aus
- Grundgleichungen:

$$MM = 2{,}4\ KDSI^{1{,}05}$$
$$TDEV = 2{,}5\ MM^{0{,}38}$$

MM: Man Month (Personen-Monate)
KDSI: Kilo delivered source instructions
(Anzahl der ausgelieferten Codezeilen in 1.000)
TDEV: Time to Develop (Entwicklungszeit)

- Gilt für *einfache Anwendungs-Software* (organic mode) ...
- ... in einer *stabilen Umgebung*

COCOMO – Größe vs. Aufwand



COCOMO - Berechnungsgrundlagen

- Randbedingungen
  - Schätzungen schließen den Aufwand für die Anforderungsdefinition nicht mit ein
  - Gleichungen müssen *unternehmensspezifisch kalibriert* werden

- Grundgleichungen für andere Software
  - *Programmsysteme* (semi-detached mode)
    $MM = 3{,}0\ KDSI^{1{,}12}$ $TDEV = 2{,}5\ MM^{0{,}35}$
  - *Eingebettete Systeme* (embedded mode)
    $MM = 3{,}6\ KDSI^{1{,}2}$ $TDEV = 2{,}5\ MM^{0{,}32}$

COCOMO – Präzisierung der Schätzung

- Durch Berücksichtigung unternehmensspezifischer und Projektspezifischer Kostenfaktoren (cost drivers)
  (1) Bestimmung des Nominalwerts für den Aufwand
  (2) Bestimmung der Kostenfaktoren
  (3) Multiplikation des Nominalwerts mit dem Produkt der Kostenfaktoren:

$$MM_{Korr} = \text{Produkt der Kostenfaktoren} \times MM_{nominal}$$

COCOMO - Kostenfaktoren

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
RELY Required software reliability	.75	.88	1.00	1.15	1.40	
DATA Data base size		.94	1.00	1.08	1.16	
CPLX Product complexity	.70	.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME Execution time constraint			1.00	1.11	1.30	1.66
STOR Main storage constraint			1.00	1.06	1.21	1.56
VIRT Virtual machine volatility[a]		.87	1.00	1.15	1.30	
TURN Computer turnaround time		.87	1.00	1.07	1.15	
Personnel Attributes						
ACAP Analyst capability	1.46	1.19	1.00	.86	.71	
AEXP Applications experience	1.29	1.13	1.00	.91	.82	
PCAP Programmer capability	1.42	1.17	1.00	.86	.70	
VEXP Virtual machine experience[a]	1.21	1.10	1.00	.90		
LEXP Programming language experience	1.14	1.07	1.00	.95		
Project Attributes						
MODP Use of modern programming practices	1.24	1.10	1.00	.91	.82	
TOOL Use of software tools	1.24	1.10	1.00	.91	.83	
SCED Required development schedule	1.23	1.08	1.00	1.04	1.10	

Nähere Informationen zu COCOMO sind in Boehm (1981): „Software Engineering Economics" nachzulesen

Das „Function-Point-Verfahren"

- *Relatives Maß* zur Bewertung der Funktionalität
- Von A. Albrecht 1979 bei IBM entwickelt
- Falls *Erfahrungszahlen* (Kosten pro „Function Point") vorliegen
  - ↳ Kostenschätzung möglich
- Anwendung primär für *Informationssysteme*
- Idee
  - ↳ In geeigneter Weise zählen der
    - *Dateneingaben* (External Input)
    - *Datenausgaben* (External Output)
    - *Anfragen* (External Inquiry)
    - *Schnittstellen* zu *externen Datenbeständen* (External Interface File)
    - *Interne Datenbestände* (Logical Internal File)

Zählung + Gewichtung der „Function Points“

- *Eingaben, Ausgaben, Anfragen*
  - Zählen anhand der *logischen Transaktionen* des Systems
  - Gleiche Daten in verschiednen Transaktionen werden mehrmals gezählt
- *Externe / interne Datenbestände*,
  d.h. *logische Dateien* bzw. *Datengruppen in Datenbanken*
  (Gegenstandsgruppen oder Relationen) zählen
- Zählung ist in der *Anforderungsspezifikation* möglich
- Werte werden *gewichtet*
  - Schwierigkeitsgrad pro Element
    - ↳ einfach – mittel – komplex
  - Gewichtung der Summe mit dem Wertkorrekturfaktor VAF
- Es gibt unterschiedliche *Zählverfahren* für „Function Points“
  - ↳ Hier: Verfahren der
    „International Function Point Users Group (IFPUG)“

Beispiel – Gewichtung + Zählschema

Anzahl bearbeiteter Datenbestände	Anzahl unterscheidbarer Datenelemente in der Eingabe		
	1–4	5–15	>15
0–1	einfach	einfach	mittel
2	einfach	mittel	komplex
>2	mittel	komplex	komplex

Element	Schwierigkeitsgrad			
	einfach	mittel	komplex	Summe
Dateneingaben	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Datenausgaben	_____ x 4 = _____	_____ x 5 = _____	_____ x 7 = _____	_____
Anfragen	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Ext. Schnittstellen	_____ x 5 = _____	_____ x 7 = _____	_____ x10 = _____	_____
Int. Datenbestände	_____ x 7 = _____	_____ x10 = _____	_____ x15 = _____	_____
			Function Point Rohwert (UFP)	

gemäß IFPUG (1994)

Anpassung – Der Gesamt-Einflussfaktor TDI

Nr.	Faktor	Wert	Einzusetzen sind Werte zwischen 0 und 5
1	Datenkommunikation		
2	Verteilte Funktionen		0 nicht vorhanden, kein Einfluss
3	Leistungsanforderungen		1 unbedeutender Einfluss
4	Belastung der Hardware		2 mäßiger Einfluss
5	Verlangte Transaktionsrate		3 durchschnittlicher Einfluss
6	Online-Dateneingabe		4 erheblicher Einfluss
7	Effiziente Benutzerschnittstelle		5 starker Einfluss
8	Online-Datenänderungen		
9	Komplexe Verarbeitungen		
10	Wiederverwendbarkeit		
11	Einfache Installation		
12	Einfache Benutzbarkeit		
13	Installation an mehreren Orten		
14	Änder- und Erweiterbarkeit		
	Summe der Faktoren (TDI)		

Anpassung – Berechnung des Korrekturfaktors

(1) Berechnung des Gesamt-Einflussfaktors TDI

(2) VAF = 0,65 + 0,01 x TDI

(3) FP = UFP x VAF

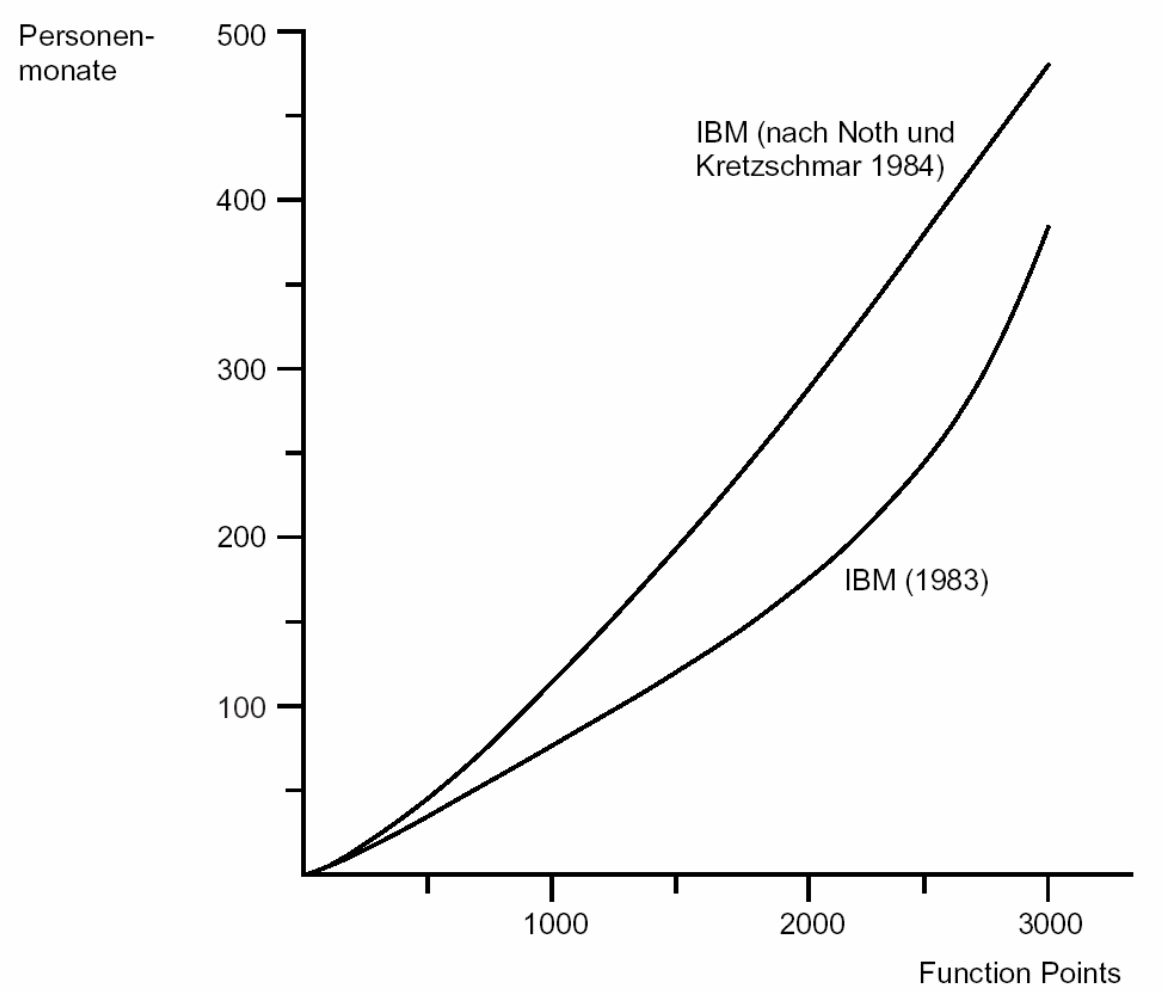
DI	Degree of Influence
TDI	Total Degree of Influence
UFP	Unadjusted Function Points
VAF	Value Adjustment Factor

„Function Points“ zur Aufwandschätzung

- *Mittlere Aufwand pro „Function Point“* muss bekannt sein
- *Umrechnungsfaktoren* müssen unternehmensspezifisch *kalibriert* und projektspezifisch angepasst werden
- *Umrechnungstabellen* und *Faustregeln* sind nur mit Vorsicht anzuwenden

- Faustregel von Jones
  - *Durchlaufzeit* [in Monaten] = $FP^{0,4}$
  - Anzahl *Mitarbeiter* = FP / 150
  - *Aufwand* = Durchlaufzeit x Anzahl Mitarbeiter = $FP^{0,4}$ x FP / 150

Aufwandsbestimmung aus „Function Points“



„Function Point“ vs. „Codezeilen“

+ *Eingangsgrößen* für „Function Points“ *genauer bestimmbar* als Programmgröße in Codezeilen abschätzbar
– „Function Points“ sind *nicht additiv*

Umrechnung abhängig von der Programmiersprache:

Sprache	Mittlere Anzahl Codezeilen pro „Function Point“
Assembler	320
C	128
FORTRAN	107
COBOL	197
Pascal	91
C++	53
Ada95	49
Smalltalk	21
SQL	12

Weitere Methoden zur Aufwandsabschätzung

- „Koste es was es wolle“-Schätzung
- Schmerzschwellen-Schätzung
- Schätzung nach dem „Parkinson´schen Gesetz

Schätzung der Wartungskosten

- Kostenverhältnis: *Entwicklung* zu *Pflege* etwa 40:60 bis (bestenfalls) 50:50
- Faustregel für die *Kostenverteilung* von Pflegekosten
  - *60 % Verbesserungen*
  - *20 % Anpassungen*
  - *20 % Fehlerbehebung*
- Faustregel von C. Jones für den Pflegeaufwand
  - Benötigtes Pflegepersonal = *FP / 500*
    oder oder = *KDSI / 50*
- KDSI / Personen-Rate aus verschiedenen Projekten sehr unterschiedlich

Pflegekosten – Einige Zahlen

- Zeile / Person-Rate in der Software-Pflege

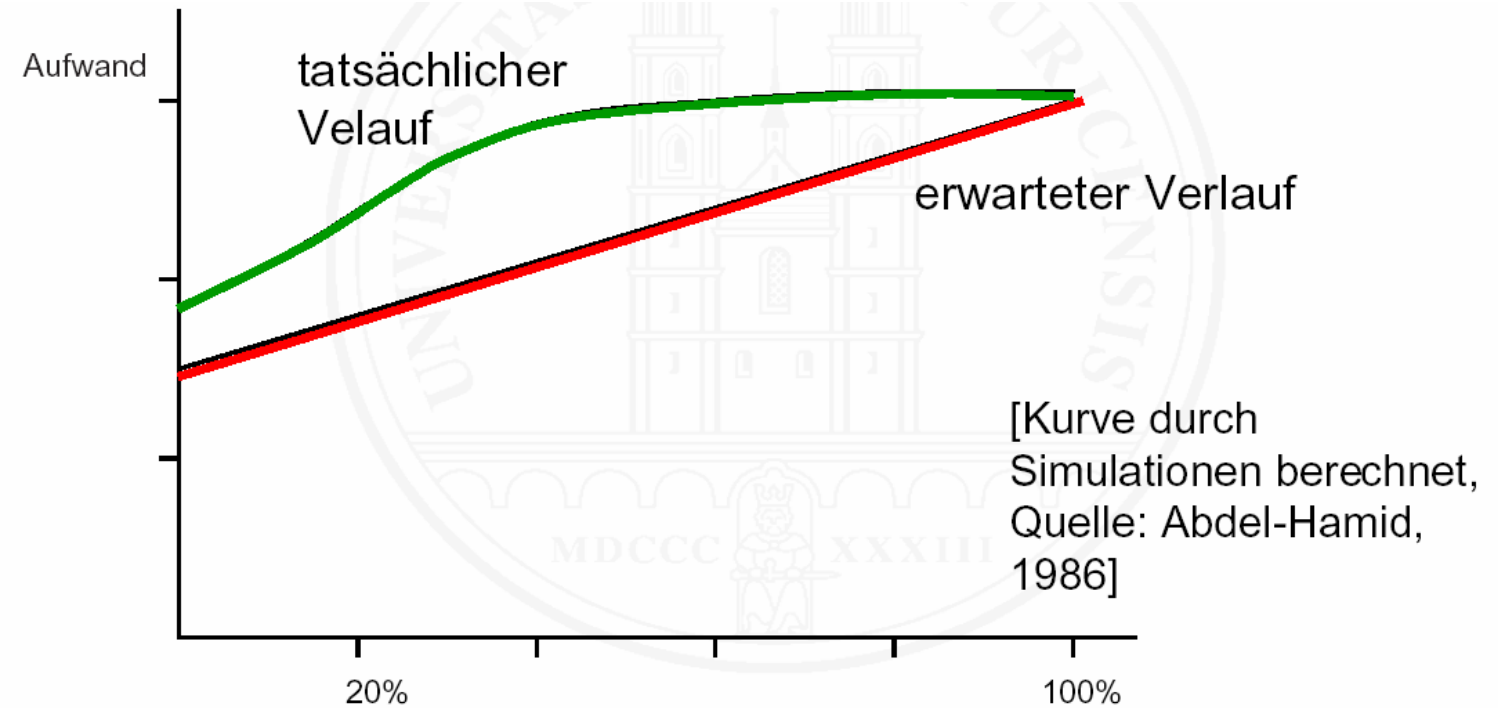
Source	Application Type	$(KDSI/FSP)_M$
COCOMO, lowest		3
[Wolverton, 1980]	Aerospace	8
[Ferens–Harris, 1979]	Aerospace	10
COCOMO, 25th percentile		10
[Daly, 1977]	Real-time	10–30
[Griffin, 1980]	Real-time	12
[Elliott, 1977]	Business	20
[Graver and others, 1977]	Business, HOL	20
[Graver and others, 1977]	Business, MOL	22
COCOMO, median		25
[Lientz–Swanson, 1980]	Business, 487 installations	32
COCOMO, 75th percentile		36
[Daly, 1977]	Support software	30–120
COCOMO, highest		132

Quelle: Boehm (1981)

- Medianwert liegt bei etwa 20 KDSI / FSP M
  - ↳ Pro 20 KDSI wird eine Person (Vollzeit) für Wartungsarbeiten benötigt

Einfluss der Schätzung auf den Aufwand

- Schätzung + tatsächlicher Aufwand sind nicht voneinander unabhängig
- Parkinson-Effekt
  - Korrelation von geschätztem + effektivem Aufwand



Zusammenfassung - Aufwandsabschätzung

- Aufwandsabschätzung ist sehr *schwer* + *heuristisch*
- Stark von der *Erfahrung* des Schätzers abhängig
- Konsens *mehrerer Experten verbessert Ergebnis*
- *Algorithmische Verfahren*
  - können nur *anwendungs- + unternehmensspezifisch* angewendet werden
  - erforderliche **Eckdaten** schwer ermittelbar
  - wenn Eckdaten vorhanden, dann *gute Ergebnisse*
- *„Function Point"-Verfahren*
  - kann auf *Spezifikation* angewendet werden
  - *normierte Zählverfahren* verwenden (Nachvollziehbarkeit)
- *Schätzung + Aufwand* sind *nicht unabhängig* voneinander

Literatur – Software Engineering

- Skript Informatik II — Prof. Dr. Kühn / Fb W FH Karlsruhe
  http://www.home.fh-karlsruhe.de/~kuin0001/inhalt.htm
- Skript Software Engineering — Prof. Dr. Martin Glinz Universität Zürich
  http://www.ifi.unizh.ch/groups/req/courses/se_I/
- Skript Software Engineering II — Bernd Kahlbrandt FH Hamburg
  http://www.informatik.fh-hamburg.de/~khb/st4se2/
- Software Engineering
  Ian Sommerville — (ISBN3-82737-001-9)
- Software Engineering
  - Grundkurs für Praktiker –
  Roger S. Pressman — (ISBN 3-89028-163-X)
- Software Entwurf
  - Methoden und Werkzeuge –
  A. Schulz — (ISBN 3-486-21608-2)
- Software Engineering und Prototyping
  Thorsten Spitta — (ISBN 3-540-17542-3)
- CASE
  Helmut Balzert — (ISBN 3-411-03224-3)
- Software-Qualitätssicherung
  Ernest Wallmüller — (ISBN 3-446-15846-4)

Software Engineering

Informatik II.

7. Software-Entwicklung

– Realisierung –