Software Engineering I

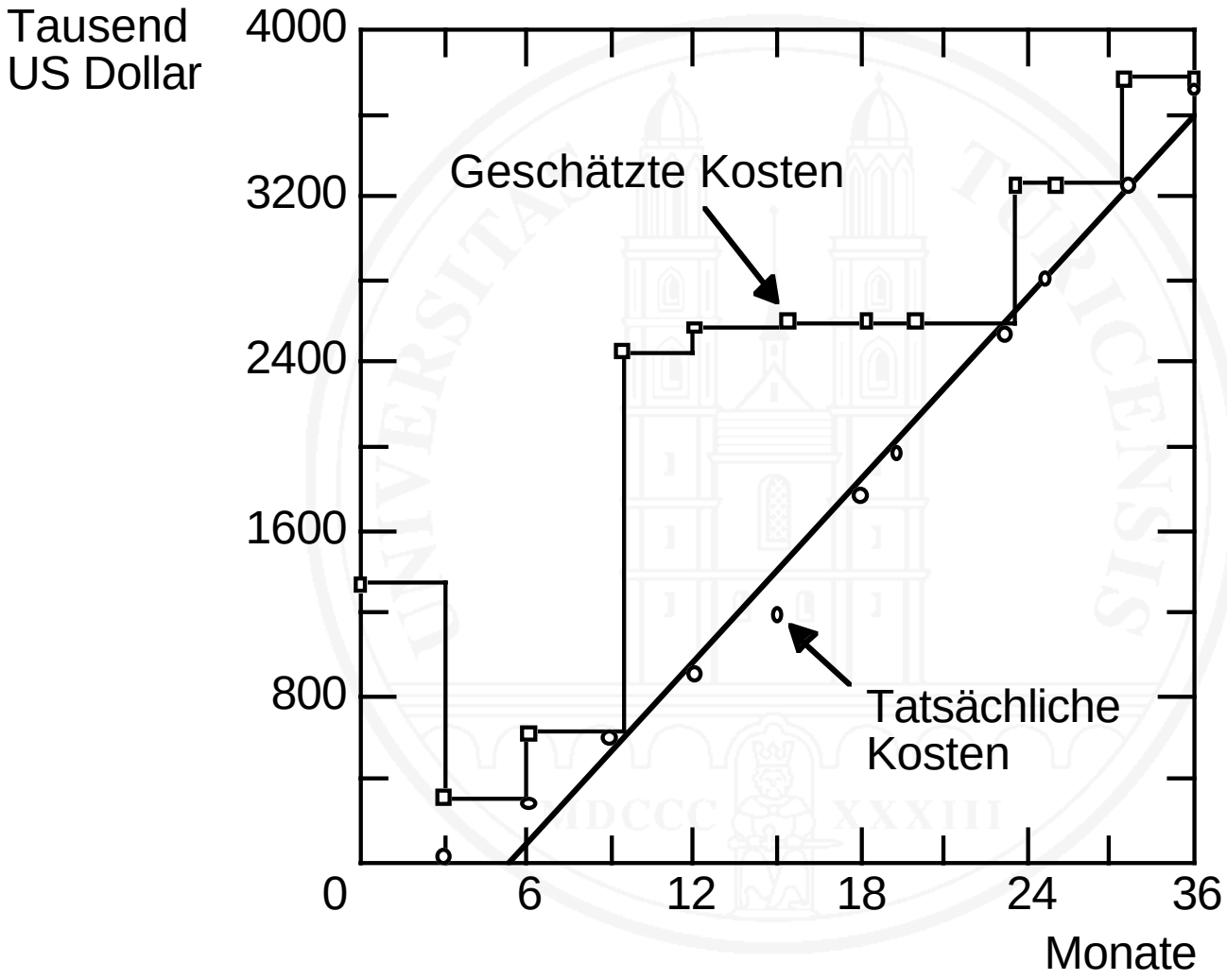
Prof. Dr. Martin Glinz

Kapitel 5

Software-Aufwandschätzung

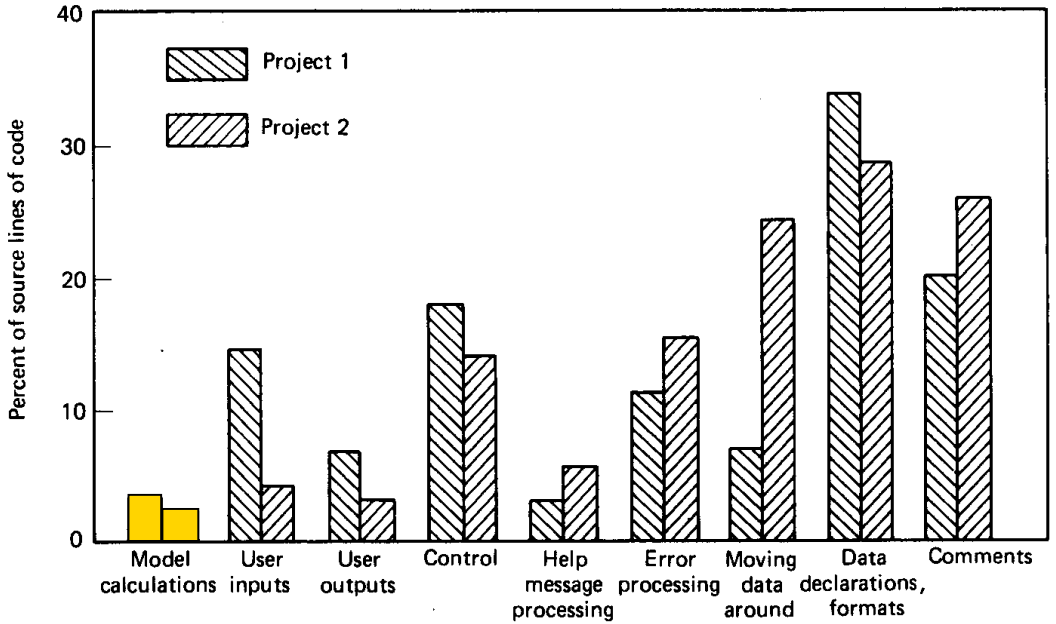
Universität Zürich
Institut für Informatik

Wie man es nicht machen sollte



Probleme der Aufwandschätzung

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



Empirische Schätzverfahren

- Schätzung basiert auf
  - Erfahrung der Schätzenden
  - Vergleich mit abgewickelten Projekten
- Expertenschätzung
  - Gut bei genug Erfahrung mit gleichartigen Projekten
  - Krasse Fehler möglich
  - Einfach und billig
- Delphi-Methode
  - Schätzung mit mehreren unabhängigen Experten
  - Mehrere Runden
  - Konvergiert (hoffentlich); eliminiert Ausreißer
  - Nachteil: Aufwand

Techniken für die Expertenschätzung

- Direkte Analogieschätzung
  - Vergleich mit möglichst ähnlichem abgewickeltem Projekt
  - Abschätzung der Mehr- bzw. Minderaufwendungen
- Schätzung durch Zerlegung
  - Zerlegung in Teilaufgaben oder Teilschritte
  - Schätzung der Teile
  - Achtung: Einzelaufwendungen nicht einfach addierbar!
  - Vorteil: Ableitung von Meilensteinen möglich
- Zu beachten:
  - Einfluss der Produktivität der Projektbeteiligten
  - Geschätzte Werte sind faktisch statistische Verteilungen

Algorithmische Schätzverfahren

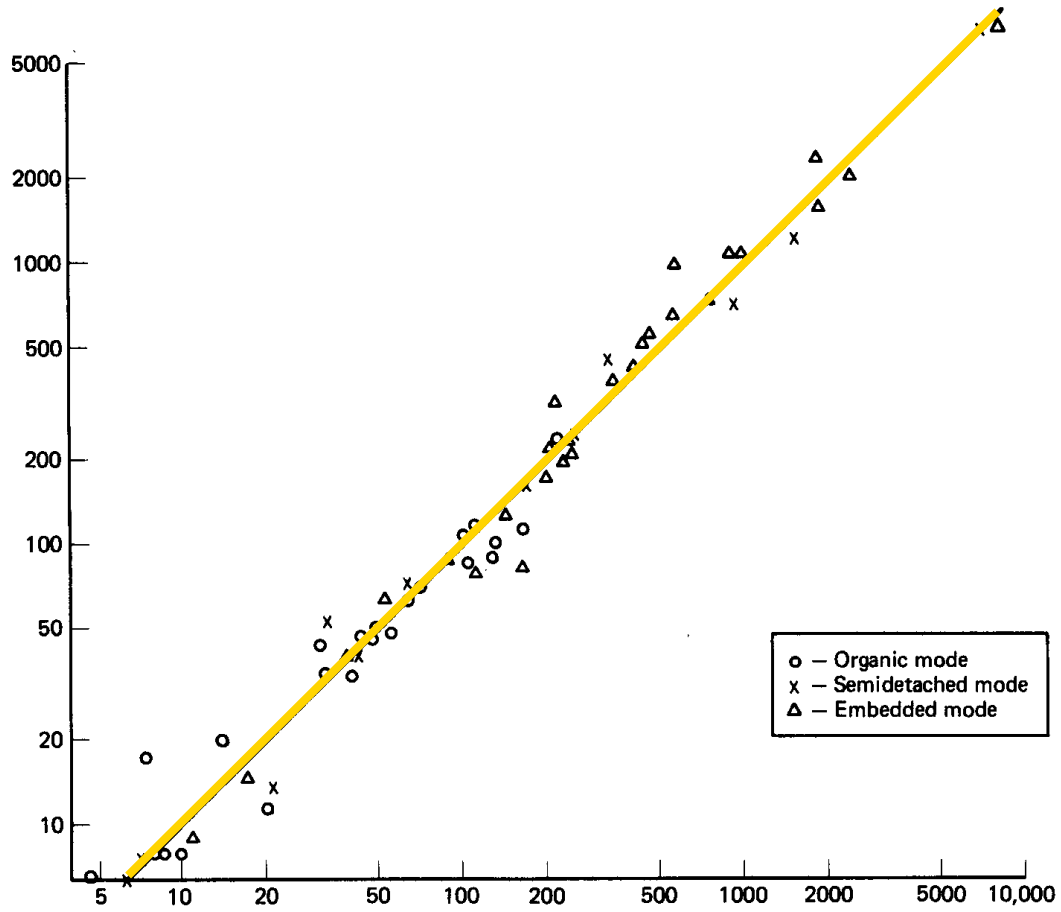
Berechnung von Kosten- und Durchlaufzeit-Funktionen

- 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 mit algorithmishen Verfahren möglich ist

tatsäch-
licher
Aufwand



Schätzung

COCOMO (*Co*nstructive *Co*st *Mo*del)

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

Grundgleichungen: $MM = 2.4\ KDSI^{1.05}$

$TDEV = 2.5\ MM^{0.38}$

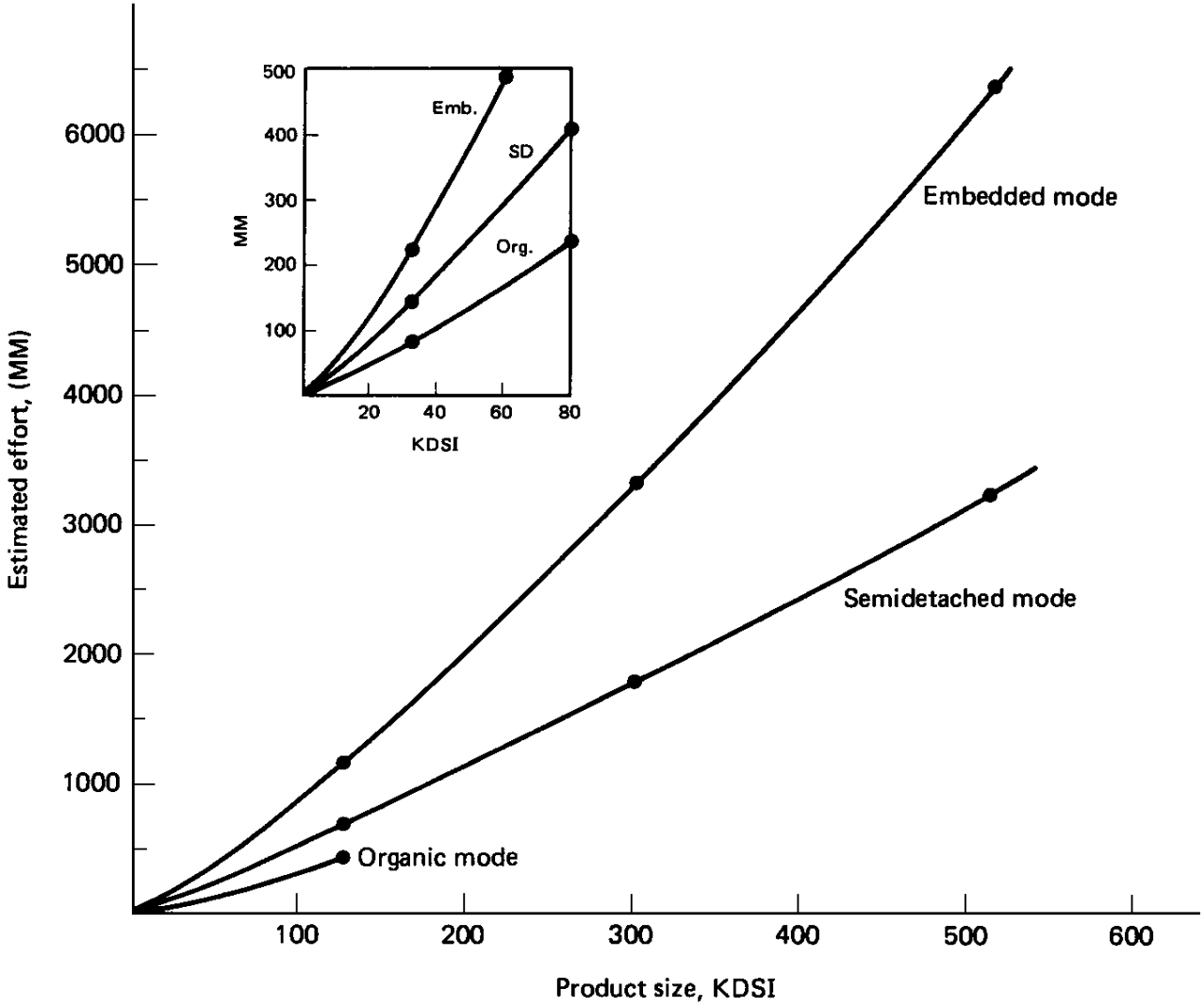
KDSI: Kilo delivered source instructions (Anzahl ausgelieferte Codezeilen in 1000)

MM: Man month (Personenmonat)

TDEV: Time to develop (Durchlaufzeit)

- Gilt für einfache Anwendungssoftware (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	

COCOMO II

- Weiterentwicklung von COCOMO
- Publiziert in Boehm et al. (2000)
- Wird im Rahmen dieser Vorlesung nicht behandelt

Das Function-Point-Verfahren

- Relatives Maß zur Bewertung der Funktionalität
- Von A. Albrecht 1979 bei IBM entwickelt
- Falls Erfahrungszahlen (Kosten pro Function Point) vorhanden
  ➪ Kostenschätzung möglich
- Anwendung primär für Informationssysteme

- Idee: Zähle in geeigneter Weise
  - Dateneingaben (External input)
  - Datenausgaben (External output)
  - Anfragen (External inquiry)
  - Schnittstellen zu externen Datenbeständen (External interface file)
  - Interne Datenbestände (Logical internal file)

Zählung und Gewichtung der Function Points

- Eingaben, Ausgaben, Anfragen:
  - Zählen anhand der logischen Transaktionen des Systems
  - Gleiche Daten in verschiedenen Transaktionen werden mehrmals gezählt
- Externe / interne Datenbestände: logische Dateien bzw. Datengruppen in Datenbanken (Gegenstandstypen 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 für Dateneingaben

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

gemäß IFPUG (1994)

Zählschema für Function Point

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 – 1: 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 – 2: 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

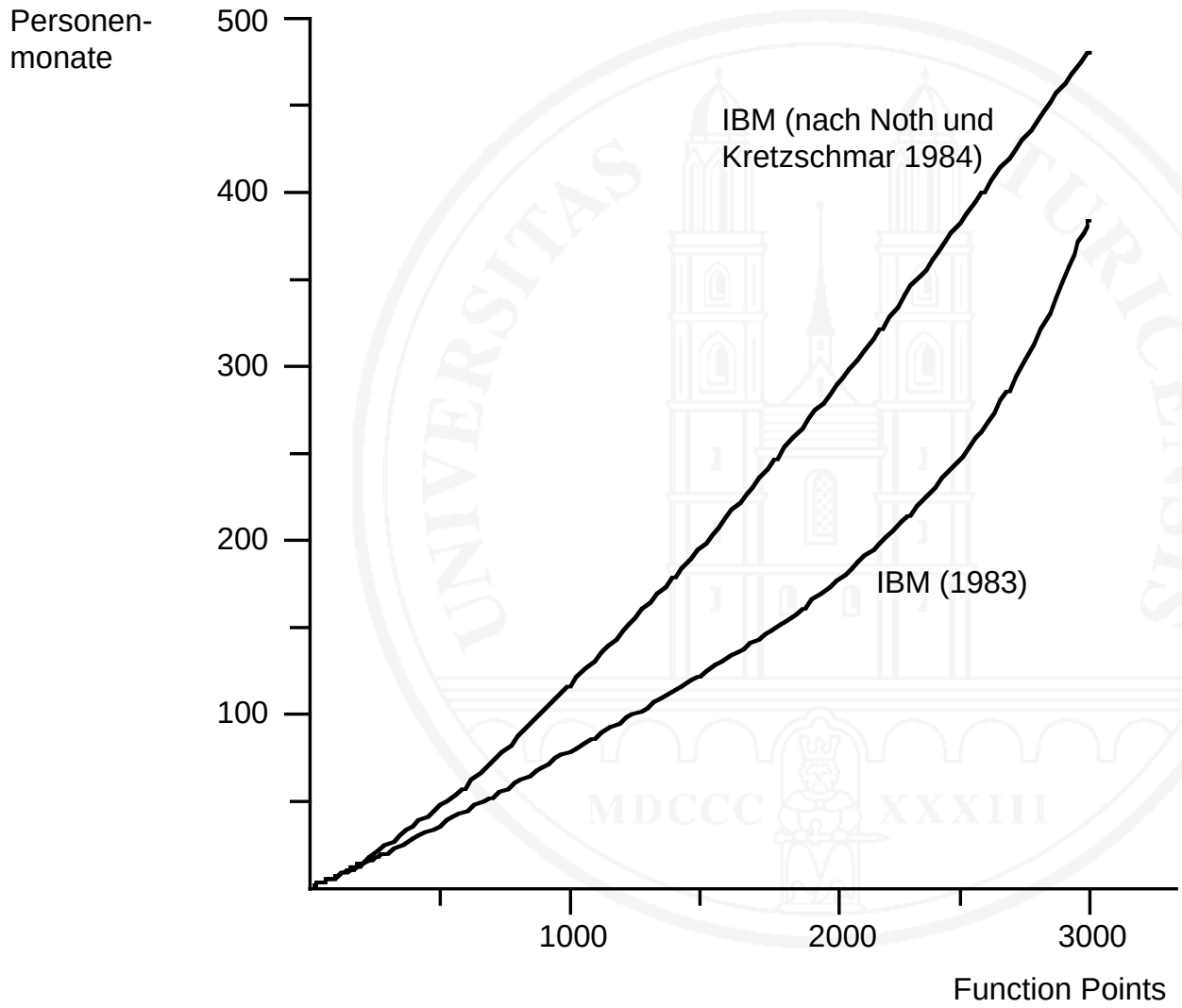
Function Points zur Aufwandschätzung

- Mittlerer Aufwand pro Function Point muss bekannt sein
- Umrechnungsfaktoren müssen unternehmensspezifisch kalibriert und projektspezifisch angepasst werden
- Umrechnungstabellen und Faustregeln sind mit Vorsicht anzuwenden

Faustregeln von Jones:

- Durchlaufzeit [in Monaten] = $FP^{0.4}$
- Anzahl Mitarbeiter = FP / 150
- Aufwand = Durchlaufzeit x Anzahl Mitarbeiter = $FP^{0.4}$ x FP / 150

Bestimmung des Aufwand aus den Function Points



Function Point vs. Codezeilen

+ Eingangsgrößen für Function Points genauer bestimmbar als Programmgröße in Codezeilen schätzbar
– Function Points sind nicht additiv

Umrechnung abhängig von 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

Sonstige Methoden zur Aufwandschätzung

- «Koste es, was es wolle»-Schätzung
- Schmerzschwellen-Schätzung
- Schätzung nach dem Parkinson'schen Gesetz

Schätzung von Pflegekosten

- 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 = KDSI / 50

- KDSI/Person-Raten aus verschiedenen Projekten sehr unterschiedlich

Pflegekosten: Einige Zahlen

Zeile/Person-Raten 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 die Pflege benötigt

Einfluss der Schätzung auf den Aufwand

- Schätzung und tatsächlicher Aufwand nicht voneinander unabhängig
- Parkinson-Effekt: Korrelation von geschätztem und effektivem Aufwand: