

Martin Glinz Harald Gall

# Software Engineering

Wintersemester 2005/06

Kapitel 16

## Software-Aufwandschätzung



Universität Zürich  
Institut für Informatik

# 16.1 Allgemeines

---

16.2 Empirische Schätzverfahren

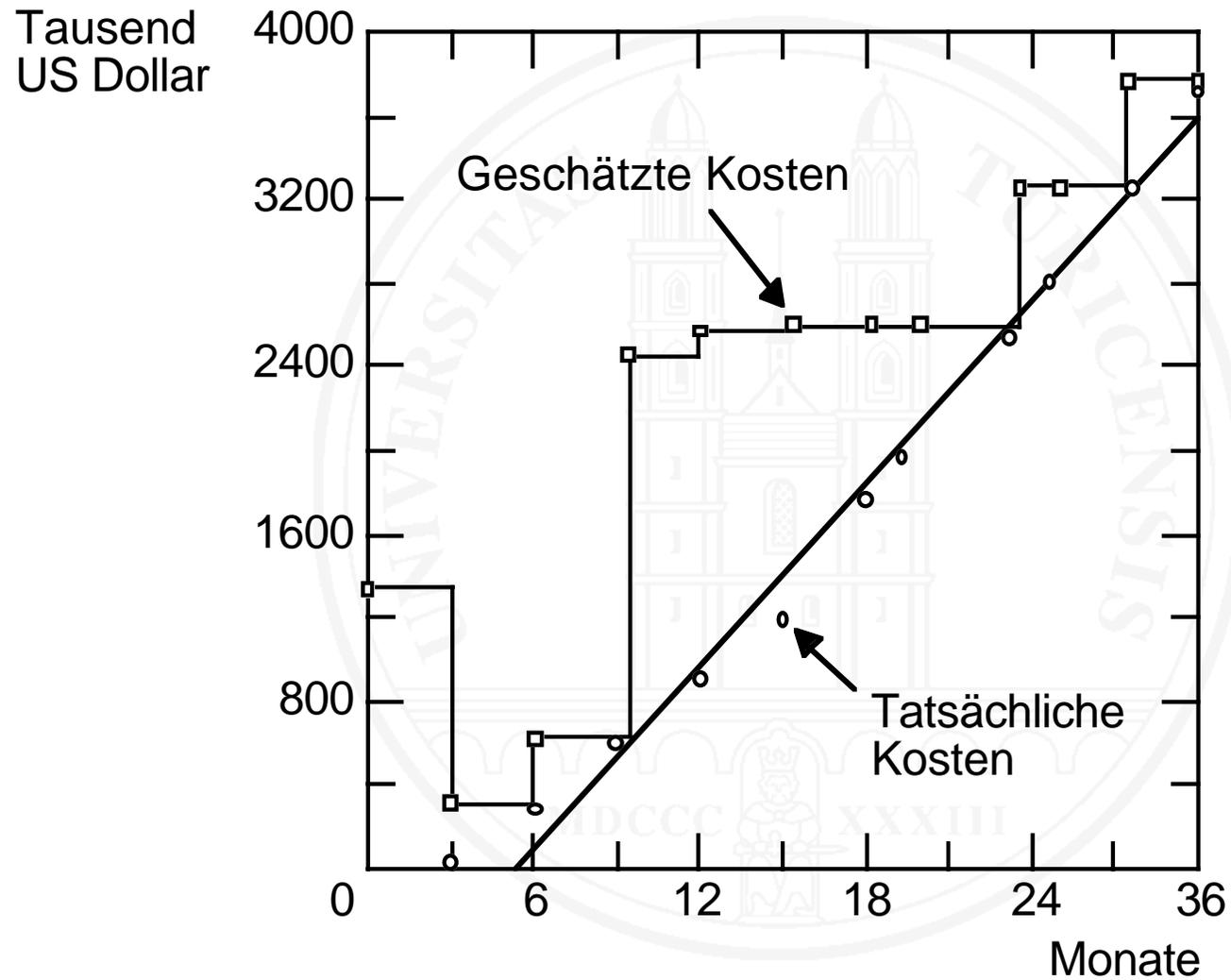
16.3 Algorithmische Schätzverfahren

16.4 Sonstige Verfahren

16.5 Schätzung von Pflegeaufwand

16.6 Einfluss der Schätzung auf den Aufwand

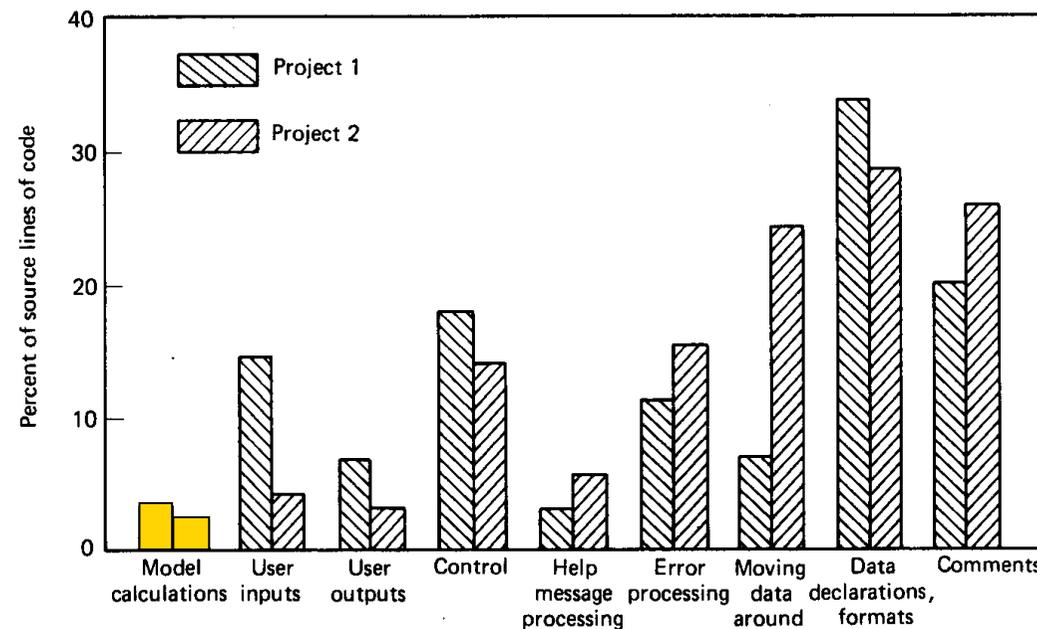
# Wie man es nicht machen sollte



# Probleme der Aufwandschätzung

---

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



16.1 Allgemeines

**16.2 Empirische Schätzverfahren**

---

16.3 Algorithmische Schätzverfahren

16.4 Sonstige Verfahren

16.5 Schätzung von Pflegeaufwand

16.6 Einfluss der Schätzung auf den Aufwand

# 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 abgewickelterm 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

16.1 Allgemeines

16.2 Empirische Schätzverfahren

**16.3 Algorithmische Schätzverfahren**

---

16.4 Sonstige Verfahren

16.5 Schätzung von Pflegeaufwand

16.6 Einfluss der Schätzung auf den Aufwand

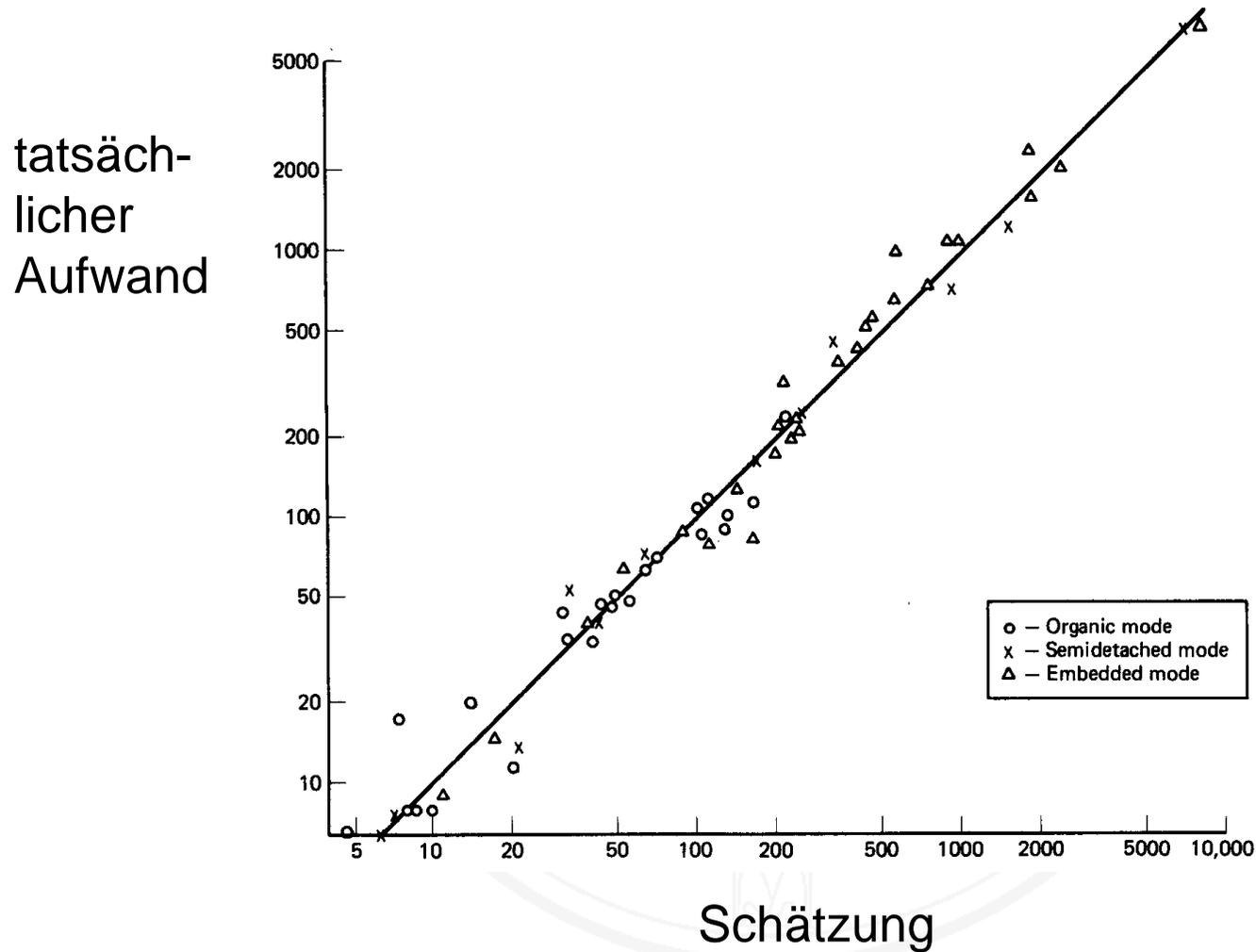
# Algorithmische Schätzverfahren

---

## 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 algorithmischen Verfahren möglich ist



# COCOMO (*Constructive Cost Model*)

---

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

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

$$TDEV = 2.5 MM^{0.38}$$

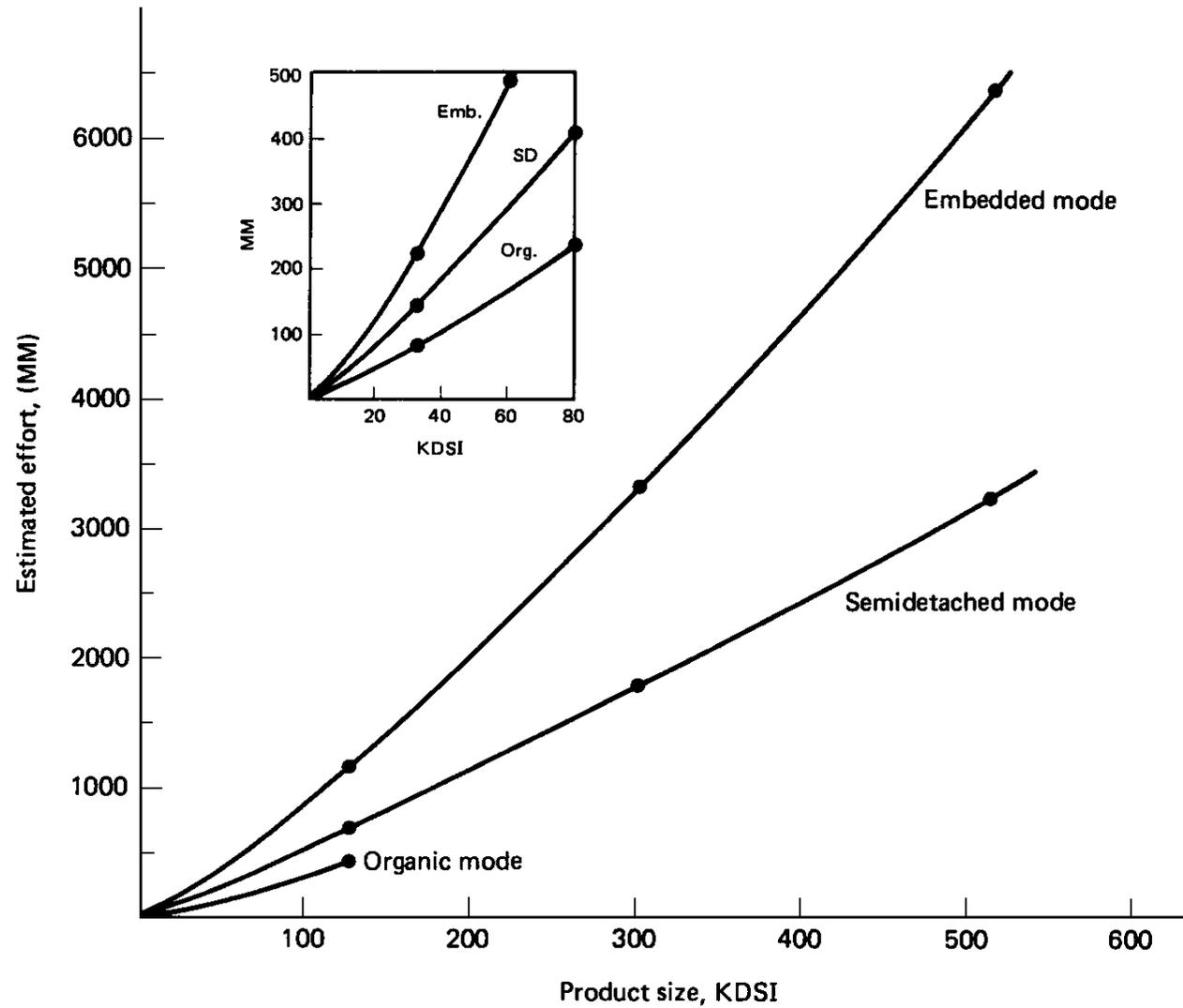
KDSI: Kilo delivered source instructions (Anzahl ausgelieferte Codezeilen in 1000)

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)

$$MM = 3.0 \text{ KDSI}^{1.12} \quad TDEV = 2.5 \text{ MM}^{0.35}$$

- **Eingebettete Systeme** (embedded mode)

$$MM = 3.6 \text{ KDSI}^{1.2} \quad TDEV = 2.5 \text{ MM}^{0.32}$$

# 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:

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

# COCOMO: Kostenfaktoren

| Cost Drivers                                 | Ratings  |      |         |      |           |            |
|--|----------|------|---------|------|-----------|------------|
|  | Very Low | Low  | Nominal | High | Very High | Extra High |
| <b>Product Attributes</b>                    |          |      |         |      |           |            |
| 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       |
| <b>Computer Attributes</b>                   |          |      |         |      |           |            |
| 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 <sup>a</sup> |          | .87  | 1.00    | 1.15 | 1.30      |            |
| TURN Computer turnaround time                |          | .87  | 1.00    | 1.07 | 1.15      |            |
| <b>Personnel Attributes</b>                  |          |      |         |      |           |            |
| 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 <sup>a</sup> | 1.21     | 1.10 | 1.00    | .90  |           |            |
| LEXP Programming language experience         | 1.14     | 1.07 | 1.00    | .95  |           |            |
| <b>Project Attributes</b>                    |          |      |         |      |           |            |
| 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 (Boehm et al. 2000)
- Neue universelle Grundgleichungen

$$\text{Aufwand} = 2,45 \cdot \text{KSLOC}^B \cdot \prod_{i=1}^{17} \text{EM}_i$$

$$\text{Durchlaufzeit} = 2,66 \cdot \text{Aufwand}^{0,33 + 0,2 \cdot (B - 1,01)}$$

KSLOC: Kilo Source Lines of Code

$$\text{Wachstumsfaktor } B = 1,01 + 0,01 \sum_{i=1}^5 \text{SF}_i$$

SF<sub>i</sub> sind insgesamt fünf Skalierungsfaktoren (scaling factors)

EM<sub>i</sub> sind insgesamt 17 Kostenfaktoren (effort multipliers)

## Mini-Übung 16.1 (Aufgabe 5.1 im Skript)

---

Sie sind Mitarbeiter(in) in der Informatikabteilung eines Unternehmens. Im Rahmen des unternehmensweiten Qualitätsmanagementsystems nehmen Sie in Ihrer Abteilung die Rolle des Qualitätsbeauftragten ein.

Ihr Chef hat auf einer Konferenz von COCOMO gehört. Da es mit der Aufwandschätzung in Ihrer Abteilung im Argen liegt, hat er beschlossen, das Gehörte in die Tat umzusetzen und ab sofort alle den Aufwand für alle neuen Projekte mit COCOMO schätzen zu lassen. Bevor er seinen Entschluss verkündet, fragt er Sie als Qualitätsbeauftragte(n) noch um Ihre Meinung. Was raten Sie ihm?

# COCOMO II: Skalierungsfaktoren (scaling factors)

---

| Faktor         | Sehr gering | Gering | Nominal | Hoch | Sehr hoch |
|----------------|-------------|--------|---------|------|-----------|
| Neuartigkeit   | 4,05        | 3,24   | 2,43    | 1,62 | 0,81      |
| Flexibilität   | 6,07        | 4,86   | 3,64    | 2,43 | 1,21      |
| Risiko         | 4,22        | 3,38   | 2,53    | 1,69 | 0,84      |
| Zusammenarbeit | 4,94        | 3,95   | 2,97    | 1,98 | 0,99      |
| Prozessreife   | 4,54        | 3,64   | 2,73    | 1,82 | 0,91      |

- **Neuartigkeit** des Produkts (als solches und für das Entwicklungsteam)
- **Flexibilität** bezüglich der Einhaltung von Anforderungen / Vorgaben
- **Risiko**: Projektrisiken gewichtet mit der Qualität des Risikomanagements
- Güte der **Zusammenarbeit** zwischen allen Projektbeteiligten
- **Reife** des Entwicklungsprozesses

# COCOMO II: Kostenfaktoren (effort multipliers)

---

| <b>Factor</b>                               | <b>Very low</b> | <b>Low</b> | <b>Nominal</b> | <b>High</b> | <b>Very High</b> | <b>Extra high</b> |
|---|-----------------|------------|----------------|-------------|------------------|-------------------|
| Reliability required                        | 0.75            | 0.88       | 1.00           | 1.15        | 1.39             |                   |
| Database size                               |                 | 0.93       | 1.00           | 1.09        | 1.19             |                   |
| Product complexity                          | 0.75            | 0.88       | 1.00           | 1.15        | 1.30             | 1.66              |
| Reuse required                              |                 | 0.91       | 1.00           | 1.14        | 1.29             | 1.49              |
| Documentation required                      | 0.89            | 0.95       | 1.00           | 1.06        | 1.13             |                   |
| Execution time constraint                   |                 |            | 1.00           | 1.11        | 1.31             | 1.67              |
| Storage constraint                          |                 |            | 1.00           | 1.06        | 1.21             | 1.57              |
| Platform volatility                         |                 | 0.87       | 1.00           | 1.15        | 1.30             |                   |
| Analyst capability                          | 1.50            | 1.22       | 1.00           | 0.83        | 0.67             |                   |
| Programmer capability                       | 1.37            | 1.16       | 1.00           | 0.87        | 0.74             |                   |
| Personnel continuity (turnover)             | 1.24            | 1.10       | 1.00           | 0.92        | 0.84             |                   |
| Application experience                      | 1.22            | 1.10       | 1.00           | 0.89        | 0.81             |                   |
| Platform experience                         | 1.25            | 1.12       | 1.00           | 0.88        | 0.81             |                   |
| Language and tool experience                | 1.22            | 1.10       | 1.00           | 0.91        | 0.84             |                   |
| Use of software tools                       | 1.24            | 1.12       | 1.00           | 0.86        | 0.72             |                   |
| Team co-location and communications support | 1.25            | 1.10       | 1.00           | 0.92        | 0.84             | 0.78              |
| Required development schedule               | 1.29            | 1.10       | 1.00           | 1.00        | 1.00             |                   |

# 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<br>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 |                   |                   | Summe |
|------------------------------|--------------------|-------------------|-------------------|-------|
|                              | einfach            | mittel            | komplex           |       |
| 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** =  $0,65 + 0,01 \times \text{TDI}$
- ③ **FP** =  $\text{UFP} \times \text{VAF}$

DI      Degree of influence

TDI     Total degree of influence

UFP     Unadjusted function points

VAF     Value adjustment factor

## Mini-Übung 5.2

Berechnen Sie die Function Points der folgenden Eingabemaske. Gehen Sie davon aus, dass beim Einloggen auf zwei als „einfach“ zu bewertende externe Datenbestände (Kunde und Kundenlog) zugegriffen wird.

Gehen Sie bei den Einflussfaktoren davon aus, dass die Faktoren 7, 10, 12 und 14 als sehr hoch bewertet werden und der Faktor 9 keinen Einfluss hat. Die übrigen Faktoren sollen durchschnittlichen Einfluss haben.

**AGP** Absolut Gute Produkte

**Willkommen in unserem Webshop!**

Ich bin ein neuer Kunde

Ich bin bereits Kunde

Name

Kundennummer

Passwort

Anwendung auswählen

# Function Points zur Aufwandschätzung

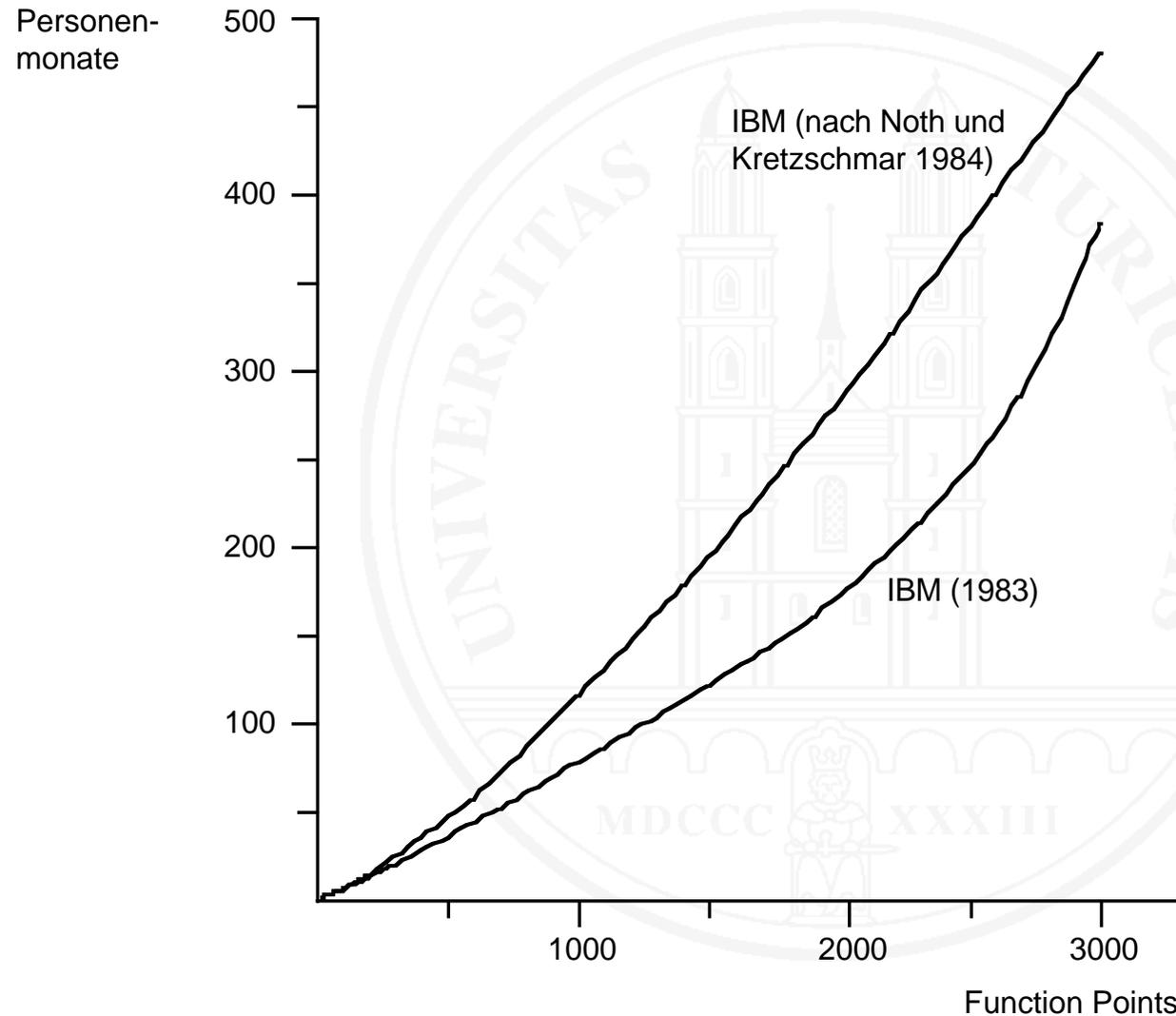
---

- 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** =  $Durchlaufzeit \times Anzahl\ Mitarbeiter = FP^{0.4} \times FP / 150$

# 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  |
| Java      | 35  |
| Smalltalk | 21  |
| SQL       | 12  |

16.1 Allgemeines

16.2 Empirische Schätzverfahren

16.3 Algorithmische Schätzverfahren

**16.4 Sonstige Verfahren**

---

16.5 Schätzung von Pflegeaufwand

16.6 Einfluss der Schätzung auf den Aufwand

# Sonstige Methoden zur Aufwandschätzung

---

- «Koste es, was es wolle»-Schätzung
  - So tief schätzen, dass man auf jeden Fall den Zuschlag für ein Projekt bekommt
- Schmerzschwellen-Schätzung
  - Abtasten, wie viel der Auftraggeber bereit ist zu zahlen und die Schätzung danach ausrichten
- Schätzung nach dem Parkinson'schen Gesetz
  - Der Aufwand passt sich der vorhandenen Kapazität an

16.1 Allgemeines

16.2 Empirische Schätzverfahren

16.3 Algorithmische Schätzverfahren

16.4 Sonstige Verfahren

**16.5 Schätzung von Pflegeaufwand**

---

16.6 Einfluss der Schätzung auf den Aufwand

# 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) <sub>M</sub> |
|---------------------------|-----------------------------|-------------------------|
| 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<sub>M</sub>
- ⇒ Pro 20 KDSI wird eine Person (Vollzeit) für die Pflege benötigt

16.1 Allgemeines

16.2 Empirische Schätzverfahren

16.3 Algorithmische Schätzverfahren

16.4 Sonstige Verfahren

16.5 Schätzung von Pflegeaufwand

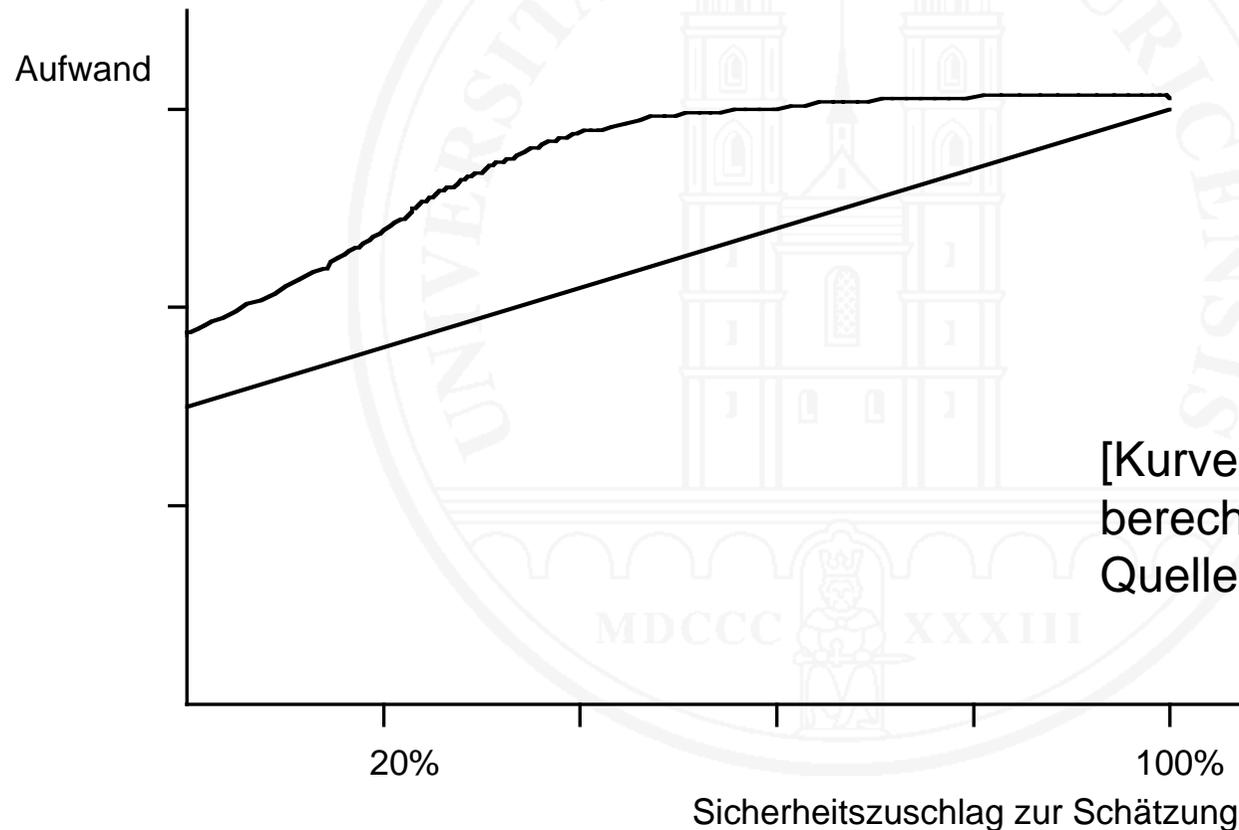
**16.6 Einfluss der Schätzung auf den Aufwand**

---

# Einfluss der Schätzung auf den Aufwand

---

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



[Kurve durch Simulationen  
berechnet;  
Quelle: Abdel-Hamid, 1986]

# Literatur

---

Siehe Literaturverweise im Kapitel 5 des Skripts.

Im Skript [M. Glinz (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich] lesen Sie bitte Kapitel 5.

Im Begleittext zur Vorlesung [S.L. Pfleeger, J. Atlee (2006). *Software Engineering: Theory and Practice*, 3rd edition. Upper Saddle River, N.J.: Pearson Education International] lesen Sie bitte Kapitel 3.3: Effort Estimation sowie die Seiten 514-518 im Kapitel 11 über Maintenance Cost.