Vorlesung "Software-Engineering"

Rainer Marrone, TUHH, Arbeitsbereich STS

- Vorige Vorlesung
  - Projektphasen und Vorgehensmodelle
- Heute:
  - Lastenheft
  - Verfahren zur Aufwandsschätzung

Produktplanung (1)

- **Produktauswahl**
  Trendstudien, Marktanalysen, Forschungsergebnisse, Kundenanfragen, Vorentwicklungen, ...

- **Voruntersuchung des Produkts**
  - u.U. gezielte Ist-Aufnahme, wenn bereits Vorgängerprodukt vorhanden;
    anschließend Ist-Analyse
  - Festlegen der Hauptanforderungen
    - Festlegen der Hauptfunktionen
    - Festlegen der Hauptdaten
    - Festlegen der Hauptleistungen
    - Festlegen der wichtigsten Aspekte der Benutzungsschnittstelle
    - Festlegen der wichtigsten Qualitätsmerkmale.

Produktplanung (2)

make or buy?

- **Durchführbarkeitsuntersuchung**
  - Prüfen der fachlichen Durchführbarkeit (softwaretechnische Realisierbarkeit, Verfügbarkeit von Entwicklungs- und Zielmaschinen, ...)
  - Prüfen alternativer Lösungsvorschläge (Beispiel: Kauf und Anpassung von Standardsoftware vs. Individualentwicklung)
  - Prüfen der personellen Durchführbarkeit:
    Verfügbarkeit qualifizierter Fachkräfte für die Entwicklung
  - Prüfen der Risiken

- **Prüfen der ökonomischen Durchführbarkeit**
  - Aufwands- und Terminschätzung
  - Wirtschaftlichkeitsrechnung

Problemanalyse und Planung

- Analyse des Ist-Zustandes (Aufgabenbereiche)
- Systemabgrenzung
  - Festlegung, welche Teile zum System gehören und damit Gegenstand der weiteren Untersuchung sind
  - Ermittlung der Umgebungsbedingungen des Systems (Schnittstellen)
- Systemerhebung
  - Sammeln und Strukturieren von Informationen über das System und seine Eigenschaften (insbes. Anforderungen u. Änderungswünsche)

Lastenheft: Produktanforderungen

Pflichtenheft: Systemanforderungen

- **Aufgabe**: Zusammenfassung aller fachlichen Basisanforderungen aus Sicht des Auftraggebers
- **Adressat**: Auftraggeber sowie Auftragnehmer (Projektleiter, Marketing, ...)
- **Inhalt**: Basisanforderungen („*Was?*“, nicht „*Wie?*“)
- **Form**: standardisiertes, nummeriertes Gliederungsschema (s. Beispiel)
- **Sprache**: verbale Beschreibung
- **Umfang**: wenige Seiten

Beispiel für ein Lastenheft: Seminarorganisation (1)

Version	Autor	QS	Datum	Status	Kommentar
2.1	Schmidt	Hupe	2/03	akzeptiert	
2.2	Schmidt	Hupe	3/03	akzeptiert	/LF40/ gelöscht

Versions-historie

- **1 Zielbestimmung**
  - Die Firma *Teachware* soll durch das Produkt in die Lage versetzt werden, die von ihr veranstalteten Seminare rechnerunterstützt zu verwalten.

informell

- **2 Produkteinsatz**
  - Das Produkt dient zur Kunden- und Seminarverwaltung der Firma *Teachware*. Außerdem sollen verschiedene Anfragen beantwortet werden können.
  - Zielgruppe: die Mitarbeiter der Firma *Teachware*.

informell

Beispiel für ein Lastenheft: Seminarorganisation (2)

- **3 Produktfunktionen**
  - **/LF10/**
    Ersterfassung, Änderung und Löschung von Kunden (Teilnehmer, Interessenten)
  - **/LF20/**
    Benachrichtigung der Kunden (Anmeldebestätigung, Abmeldebestätigung, Änderungsmitteilungen, Rechnung, Werbung)
  - **/LF30/**
    Ersterfassung, Änderung und Löschung von Seminarveranstaltungen und Seminartypen

  ...

  - **/LF70/**
    Erstellung verschiedener Listen (Teilnehmerliste, Umsatzliste, Teilnehmerbescheinigungen)
  - **/LF80/**
    Anfragen der folgenden Art sollen möglich sein: Wann findet das nächste Seminar X statt? Welche Mitarbeiter der Firma Y haben das Seminar X besucht?

Label **/LF.../** zur Referenzierung von **Funktionen**

Beispiel für ein Lastenheft: Seminarorganisation (3)

Label **/LD.../** zur Referenzierung von **Daten**

- **4 Produktdaten**
  - **/LD10/**
    Es sind relevante Daten über die Kunden zu speichern.
  - **/LD20/**
    Falls ein Kunde zu einer Firma gehört, dann sind relevante Daten über die Firma zu speichern.
  - **/LD30/**
    Es sind relevante Daten über Seminarveranstaltungen, Seminartypen und Dozenten zu speichern.
  - **/LD40/**
    Bucht ein Kunde eine Seminarveranstaltung, dann sind entsprechende Buchungsdaten zu speichern.

Beispiel für ein Lastenheft: Seminarorganisation (4)

- **5 Produktleistungen (nicht-funktionale Anforderungen)**
  - **/LL10/**
    Die Funktion /LF80/ darf nicht länger als 15 Sekunden Interaktionszeit benötigen, alle anderen Reaktionszeiten müssen unter 2 Sekunden liegen.
  - **/LL20/**
    Es müssen maximal 50.000 Teilnehmer und maximal 10.000 Seminare verwaltet werden können.

Label **/LL.../** zur Referenzierung von **Leistungen**

Beispiel für ein Lastenheft: Seminarorganisation (5)

- **6 Qualitätsanforderungen**

Produktqualität	sehr gut	gut	normal	irrelevant
Funktionalität		x		
Zuverlässigkeit			x	
Benutzbarkeit		x		
Effizienz		x		
Änderbarkeit			x	
Übertragbarkeit			x	

- **7 Ergänzungen**

  [keine]

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**).
  - Bei höheren Sprachen werden z.B. alle Vereinbarungs- und Anweisungszeilen geschätzt.
  - 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, auch MJ)** oder **Personenmonaten (PM, auch MM)**
  - 1 PJ = 9 PM oder 10 PM (Urlaub, Krankheit, Schulung, ...)
  - Der so ermittelte Aufwand wird durch die nach der Terminvorgabe zur Verfügung stehende Entwicklungszeit geteilt.
  - Ergebnis: Anzahl der einzusetzenden, parallel arbeitenden Mitarbeiter.

Einfluß der Arbeitsplatzsituation

Arbeitsplatzfaktoren	bestes Viertel der Teilnehmer	schlechtestes Viertel der Teilnehmer
1 Wieviel Arbeitsplatz steht Ihnen zur Verfügung?	7m²	4.1m²
2 Ist es annehmbar ruhig?	57% ja	29% ja
3 Ist Ihre Privatsphäre gewahrt?	62% ja	19% ja
4 Können Sie Ihr Telefon abstellen?	52% ja	10% ja
5 Können Sie Ihr Telefon umleiten?	76% ja	19% ja
6 Werden Sie oft von anderen Personen grundlos gestört?	38% ja	76% ja

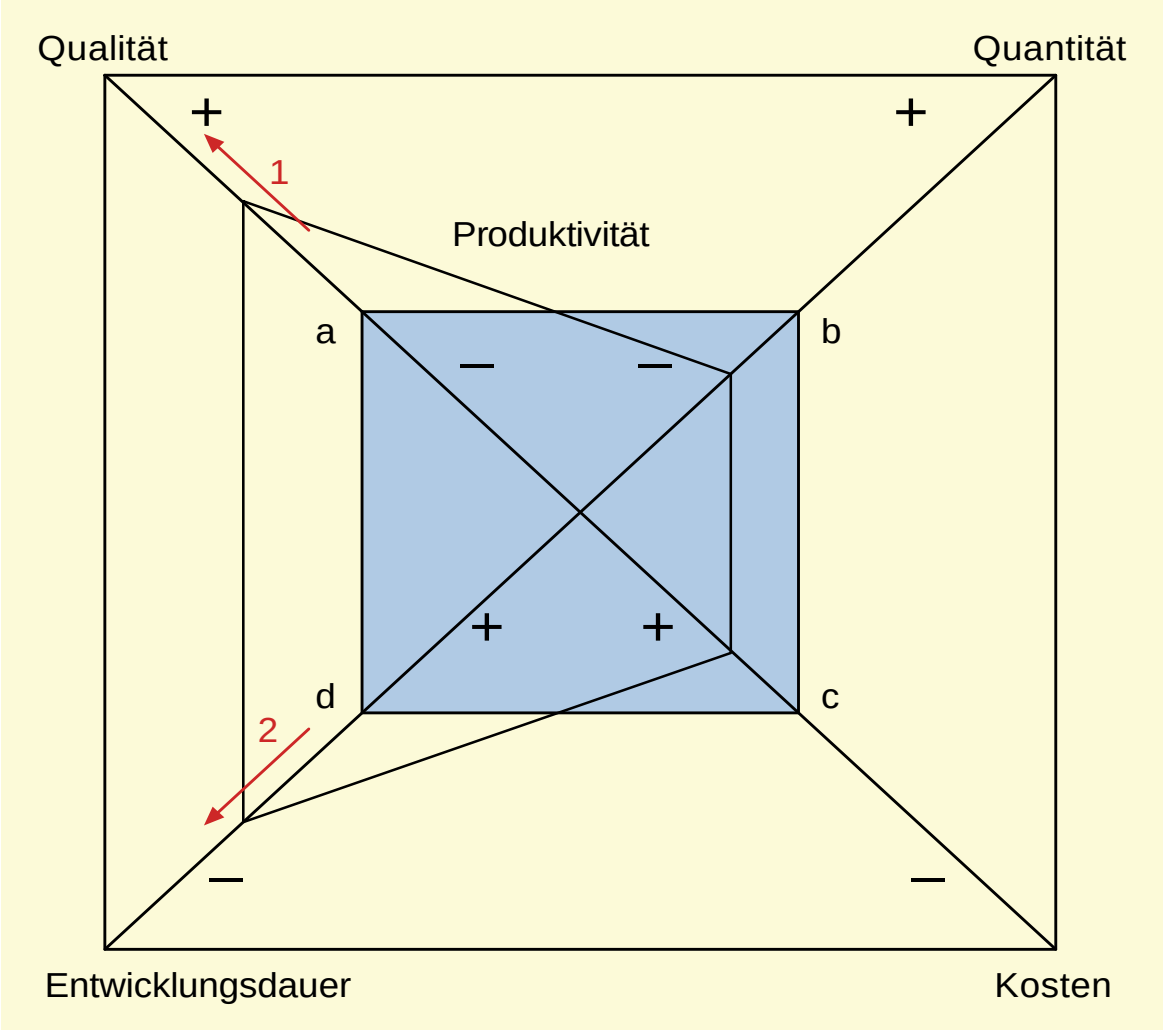
Tab. 1.4-1: Der Arbeitsplatz der Besten und der Schlechtesten /DeMarco, Lister 91, S. 57/

Einflussfaktoren der Aufwandsschätzung (1)

- Quantität
- Qualität
- Entwicklungsdauer
- Kosten

bedingen einander

è **Teufelsquadrat**



Einflussfaktoren der Aufwandsschätzung (2)

- **Quantität**
  - **Größe** des Programmtextes
    Maß „Anzahl Programmzeilen“ (LOC)
    lineare oder überproportionale Beziehung zwischen LOC und dem Aufwand
  - **Funktions- und Datenumfang**
    Maß unabhängig von einer Programmiersprache
  - evtl. zusätzliche Gewichtung mit **Komplexität**
    qualitative Maße, z.B. „leicht“, „mittel“ und „schwer“
    Abbildung auf Zahlenreihe. Beispiel: Noten zwischen 1 und 6.

in Planungsphase unbekannt

früh bekannt

- **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.

Methoden zur Kosten- und Terminschätzung

- Beispiel:
  - Es soll ein Software-Produkt mit geschätzten 21.000 LOC realisiert werden
  - Durchschnittliche Produktivität pro Mitarbeiter: 3.500 LOC/Jahr [HP, Grady 92]

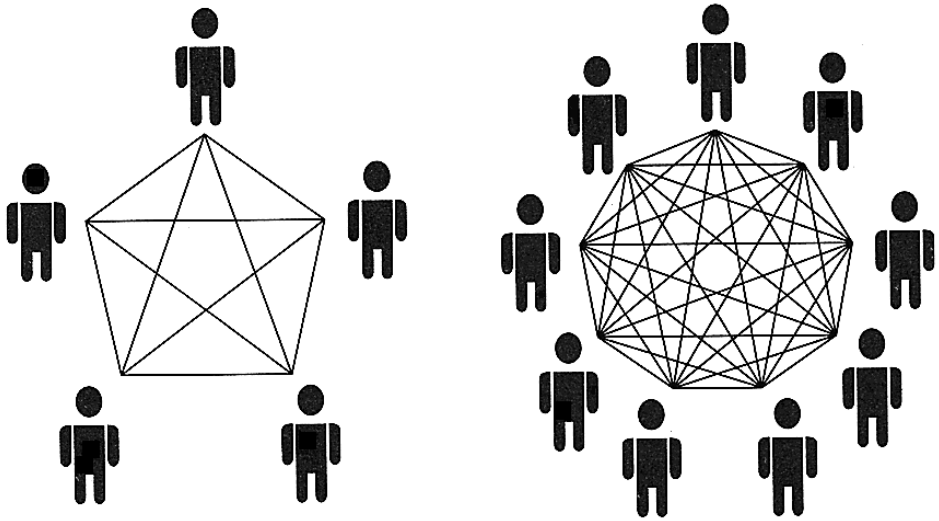
  è 6 Mitarbeiterjahre werden benötigt

  è Arbeiten 3 Mitarbeiter im Team zusammen, so werden 2 Jahre bis zur Fertigstellung benötigt.

- **Faustregeln**
  - Eine durchschnittliche Software-Entwicklung liefert ungefähr 350 Quellcodezeilen (ohne Kommentare) pro Personenmonat.
  - Dabei umfasst die Personen-Zeit alle Phasen von der Definition bis zur Implementierung.

Einflussfaktoren der Aufwandsschätzung (3)

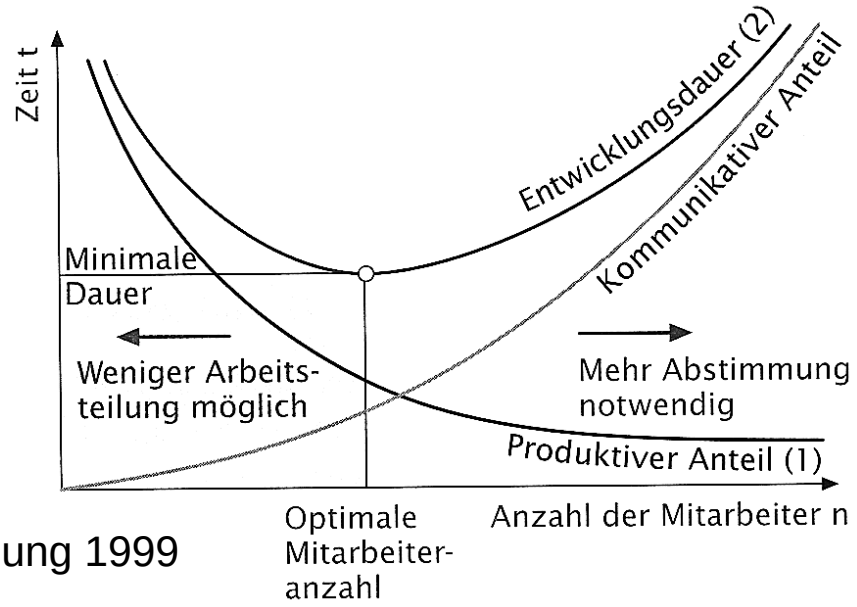
- Zusätzlicher Faktor: **Produktivität**
  - Wird von vielen verschiedenen Faktoren beeinflusst
  - Die **Lernfähigkeit** und **Motivation** der Mitarbeiter ist entscheidend.



- **Brooksches Gesetz:** „*Adding manpower to a late software project makes it later*“

Einflussfaktoren der Aufwandsschätzung (4)

- **Entwicklungsdauer**
  - Soll die Zeit verkürzt werden, dann werden mehr Mitarbeiter benötigt.
  - Mehr Mitarbeiter erhöhen den Kommunikationsaufwand im Entwicklungsteam.
  - Der höhere Kommunikationsanteil (z.B. durch Unterweisung in der Einarbeitungszeit) reduziert die Produktivität.
  - Kann die Entwicklungsdauer verlängert werden, so werden weniger Mitarbeiter benötigt.



Graphik: Steinweg, C.
Projektkompass Softwareentwicklung 1999

Aufwandsschätzung: Analogiemethode (1)

- Bei hoher Übereinstimmung kann der bekannte Aufwand unverändert übernommen werden
- **Faustregel**:
  - Software-Entwicklung mit weitgehender **Übernahme von existierender Software** benötigen ungefähr **1/4** der Zeit von Neuentwicklungen
- Nachteile der Analogiemethode
  - intuitive, sehr globale Schätzung auf der Basis individueller Erfahrungen
  - keine Nachvollziehbarkeit der Schätzergebnisse

Analogiemethode (2)

- Beispiel: Analogiemethode
  - abgeschlossenes Produkt: Pascal-Compiler: 20 MM
  - zu entwickelndes Produkt: Modula-2-Compiler?
    - 20% neue Konstrukte
    - 50% des Codes wiederverwendbar
    - 50% müssen überarbeitet werden
  - Schätzung
    - 50% leicht modifiziert: 1/4 von 10 MM = 2,5 MM
    - 50% völlige Überarbeitung: 10 MM
    - 20% zusätzliche Neuentwicklung hoher Komplexität:
      4 MM * 1,5 (Komplexitätszuschlag) = 6 MM
    - Schätzung Modula-2: 18,5 MM.

Aufwandsschätzung: Relationsmethode (1)

- Das zu schätzende Produkt wird direkt mit ähnlichen Entwicklungen verglichen.
- Aufwandsanpassung erfolgt im Rahmen einer formalisierten Vorgehensweise.
- Für die Aufwandsanpassung stehen *Faktorenlisten* und *Richtlinien* zur Verfügung.
- Beispiel:

Programmiersprache
PL/1 = 100
COBOL = 120
ASSEMBLER = 140

Programmiererfahrung
5 Jahre = 80
3 Jahre = 100
1 Jahr = 140

Dateiorganisation
sequentiell = 80
indexsequentiell = 120

- Werte geben an, in welcher Richtung und wie stark die einzelnen Faktoren den Aufwand beeinflussen.

Relationsmethode (2)

- Beispiel (Fortsetzung):
  - Ein neues Produkt soll in PL/1 realisiert werden
    - Das Entwicklungsteam hat im Durchschnitt 3 Jahre Programmiererfahrung
    - Es ist eine indexsequentielle Dateiorganisation zu verwenden.
  - Zum Vergleich: Entwicklung...
    - die im Assembler programmiert wurde
    - eine sequentielle Dateiorganisation verwendete
    - von einem Team mit 5 Jahren Programmiererfahrung erstellt wurde
  - Geht man davon aus, dass alle 3 Faktoren den Aufwand gleichgewichtig beeinflussen, dann ergibt sich folgende Kalkulation:
    - Assembler zu PL/1: 140 zu 100 = 40 Punkte Einsparung
    - 5 Jahre zu 3 Jahre: 80 zu 100 = 20 Punkte Mehraufwand
    - sequentiell zu indexsequentiell: 80 zu 120 = 40 Punkte Mehraufwand
  - Es ergibt sich ein Mehraufwand von 20 Punkten.

Aufwandsschätzung: Multiplikatormethode (1)

- 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.
- 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.

Aufwandsschätzung: Multiplikatormethode (4)

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

Kategorie	**Teil-produkte**	**Summe LOC**	**Aufwands-faktor**	**LOC bewertet**
Steuerprogramm	1*500 LOC	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**

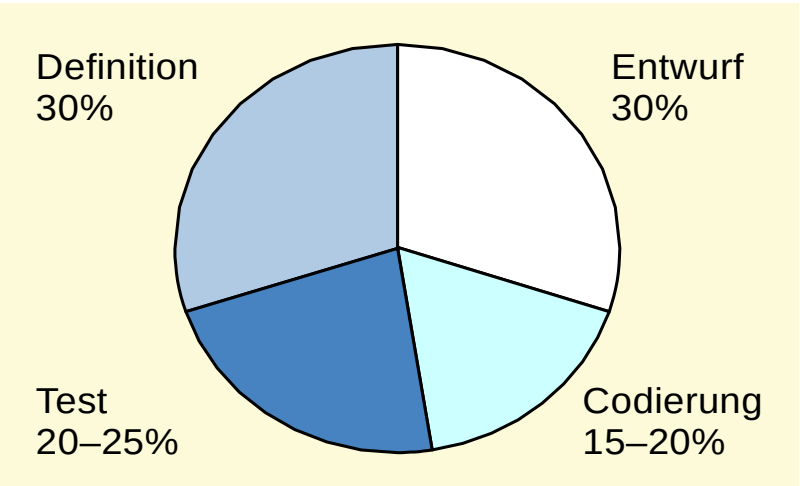
Aufwandsschätzung: Multiplikatormethode (5)

- **Bewertung**
  - Es ist eine umfangreiche, empirische Datensammlung und -auswertung erforderlich, um die zu berücksichtigenden Faktoren unternehmensspezifisch zu bewerten.
  - Die Koeffizienten müssen permanent überprüft werden, um den technischen Fortschritt zu berücksichtigen.

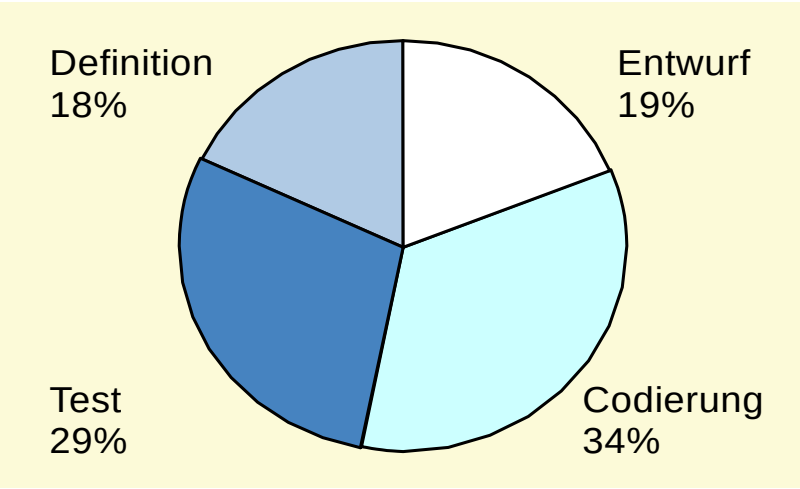
Aufwandsschätzung: Prozentsatzmethode

- 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 Soll-Aufwand für die restlichen Phasen,
- oder man führt eine detaillierte Schätzung einer Phase durch und schließt hieraus dann auf den Gesamtaufwand.
- Kann bereits frühzeitig eingesetzt werden, wenn der Aufwand für mindestens eine Phase durch den Einsatz einer anderen Methode bestimmt wurde.

Bertelsmann



Hewlett-Packard



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 müssen die
  - Analogie-
  - Relations- und
  - Prozentsatzmethode eingesetzt werden.
- Sind die Einflussfaktoren während der Entwicklung dann genauer bekannt, dann sollten die genaueren Methoden, wie die
  - Multiplikatormethode Verwendung finden.

Aufwandsschätzung: COCOMO-Methode (1)

- COnstructive COst MOdel von B. Boehm entwickelte Sammlung von Schätzmodellen
- Modell 1 (Basisversion):
- Modell 2 (Zwischenversion):
  - zusätzliche Berücksichtigung von kostenbeeinflussenden Faktoren
- Modell 3 (Detailversion):
  - zusätzliche Berücksichtigung der unterschiedlichen Projektphasen, Verteilung des Aufwandes auf die Projektphasen

Aufwandsschätzung: COCOMO-Methode (2)

- Unterscheidung von 3 Projektkategorien zur Anpassung der Kennzahlen:
  - Einfache Projekte

    (vertraute Systemumgebung, große Erfahrung, wenige Schnittstellen)
  - Mittelschwere Projekte
  - Komplexe Projekte

    (enge Verzahnung mit Systemumgebung, zahlreiche Schnittstellen, geringe Erfahrung)

COCOMO-Methode: Basisversion (1)

- Berechnung des Aufwandes (LOC -> MM):

$$A = C * KLOC^{B}$$

mit
A: Entwicklungsaufwand in MM
B,C: Konstanten gemäß Tabelle

Projekt	C	B
Einfach	2,4	1,05
Mittel	3,0	1,12
Komplex	3,6	1,20

COCOMO-Methode: Basisversion (2)

- Berechnung der optimalen Entwicklungszeit:

$$T = D * A^E$$

mit
T: Entwicklungszeit
D,E : Konstanten gemäß Tabelle

Projekt	D	E
Einfach	2,5	0,32
Mittel	2,5	0,35
Komplex	2,5	0,38

Aufwandsabschätzung - Aufgabe

Aufgabe:
Es soll eine einfache Online-Anwendung erstellt werden.
Für die abgeschlossene Definitions- und Entwurfsphase wurden
15 MM benötigt.

Erfahrungswerte:

Definitionsphase	18%
Entwurfsphase	19%
Realisierung	53%
Einführungsphase	10%

- **Fragen:**
- Aufwand der gesamten Entwicklung in MM?
  -
- Restaufwand /-dauer?
  -

- Benötigte Teamgröße ?

Aufwandsabschätzung - Lösung

Aufgabe:
Es soll eine einfache Online-Anwendung erstellt werden.
Für die abgeschlossene Definitions- und Entwurfsphase wurden
15 MM benötigt.

Erfahrungswerte:

Definitionsphase	18%
Entwurfsphase	19%
Realisierung	53%
Einführungsphase	10%

- **Fragen:**
- Aufwand der gesamten Entwicklung in MM?
  - (15 MM * 100) /37 = 40,54 MM
- Restaufwand /-dauer?
  - Gesamtaufwand – Aufwand für Planung und Entwurf
    (40,54 MM – 15 MM) = 25,54 MM
    opt. Dauer = 2,5 * $25.54^{0,32}$ gleich ca. 7 Monate
- Benötigte Teamgröße ?
  - 25,54 / 7 = 3,6 gleich ca. 3-4 Mitarbeiter

CoCoMo II

- **COCOMO II** (Boehm et al. 2000) ist eine Weiterentwicklung von COCOMO zu einem 3-Stufen Modell, das bei Entwicklungsfortschritt immer genauere Schätzungen ermöglicht.

COCOMO II unterscheidet:

- **Frühe Entwicklungsphasen (Early prototyping level)**
  - Schätzung für Projekte mit Prototyperstellung und hoher Wiederverwendung basierend auf Anwendungspunkten (application points) und einfacher Formel für die Aufwandschätzung
- **Frühe Entwurfsphase (Early design level)**
  - Schätzung nach **abgeschlossenen Festlegung der Systemanforderungen und einem ersten Entwurf** basierend auf Funktionspunkten, die in LOC übersetzt werden.
- **Nach-Architektur-Phase (Post-architecture level)**
  - Schätzung nach **Erstelleung der Architektur** basierend auf LOC

Nach-Architektur-Phase(Post-architecture level)

- Wir betrachten hier **nur die Post-Architektur-Phase**

- Neue universelle Grundgleichungen
  - **Aufwand** $PM = 2{,}45 * KSLOC^{B} * M$
  - **Entwicklungszeit** $T = 2{,}66 * PM^{\,0{,}33 + 0{,}2 \cdot (B - 1{,}01)}$

  wobei
  - **KSLOC: Kilo Source Lines of Code**
  - **Wachstumsfaktor** $B = 1{,}01 + 0{,}01 \, \Sigma \, SF_i$
  - SF_i **sind insgesamt fünf Skalierungsfaktoren (scaling factors)**
  - **M ist das Produkt der insgesamt 17 Kostenfaktoren (effort multipliers)**

COCOMO II: Skalierungsfaktoren (scalingfactors)

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

- **Erfahrung:** Vertrautheit des Entwicklungsteams mit dem Produkt
- **Flexibilität** bezüglich der Einhaltung von Anforderungen / Vorgaben
- **Risiko-Umgang:** Qualität des Risikomanagements
- Güte der **Zusammenarbeit** zwischen allen Projektbeteiligten
- **Reife** des Entwicklungsprozesses

Beispiel

- **Eine Firma will ein Projekt in einem neuen Gebiet durchführen. Der Kunde hat keinen speziellen Entwicklungsprozess gefordert und Zeit für die Risikoanalyse eingeplant. Die Firma besitzt eine Prozessreife der Stufe 1**

- **Skalierungsfaktoren:**

Erfahrung:	neues Projekt	S. gering (4,05)
Prozessflexibilität	Kunde lässt Freiheit	S. hoch (1,21)
Architecture/risk resolution	Keine Risikoanalyse	S. gering (4,22)
Teamzusammenhalt	Neues Team	Nominal (2,97)
Prozessreife	CMM Level 1	Gering (3,64)
Summe		**16,19**

- **Der Skalierungsfaktor ist also 1,01 + 0,16 = 1.17**

COCOMO II: Kostenfaktoren (effort multipliers)

Factor	Very low	Low	Nominal	High	Very High	Extra high
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	

Beispiel: 2,45 * KSLOC$^{1.17}$ * M

Exponent value	1.17
System size (including factors for reuse and requirements volatility)	128, 000 SLOC
Initial COCOMO estimate without cost drivers	**730 person-months**
Reliability	Very high, multiplier = 1.39
Complexity	Very high, multiplier = 1.3
Memory constraint	High, multiplier = 1.21
Tool use	Low, multiplier = 1.12
Schedule	Accelerated, multiplier = 1.29
Adjusted COCOMO estimate	**2306 person-months**
Reliability	Very low, multiplier = 0.75
Complexity	Very low, multiplier = 0.75
Memory constraint	None, multiplier = 1
Tool use	Very high, multiplier = 0.72
Schedule	Normal, multiplier = 1
Adjusted COCOMO estimate	**295 person-months**

Aufwandsschätzung: Function Point-Methode (1)

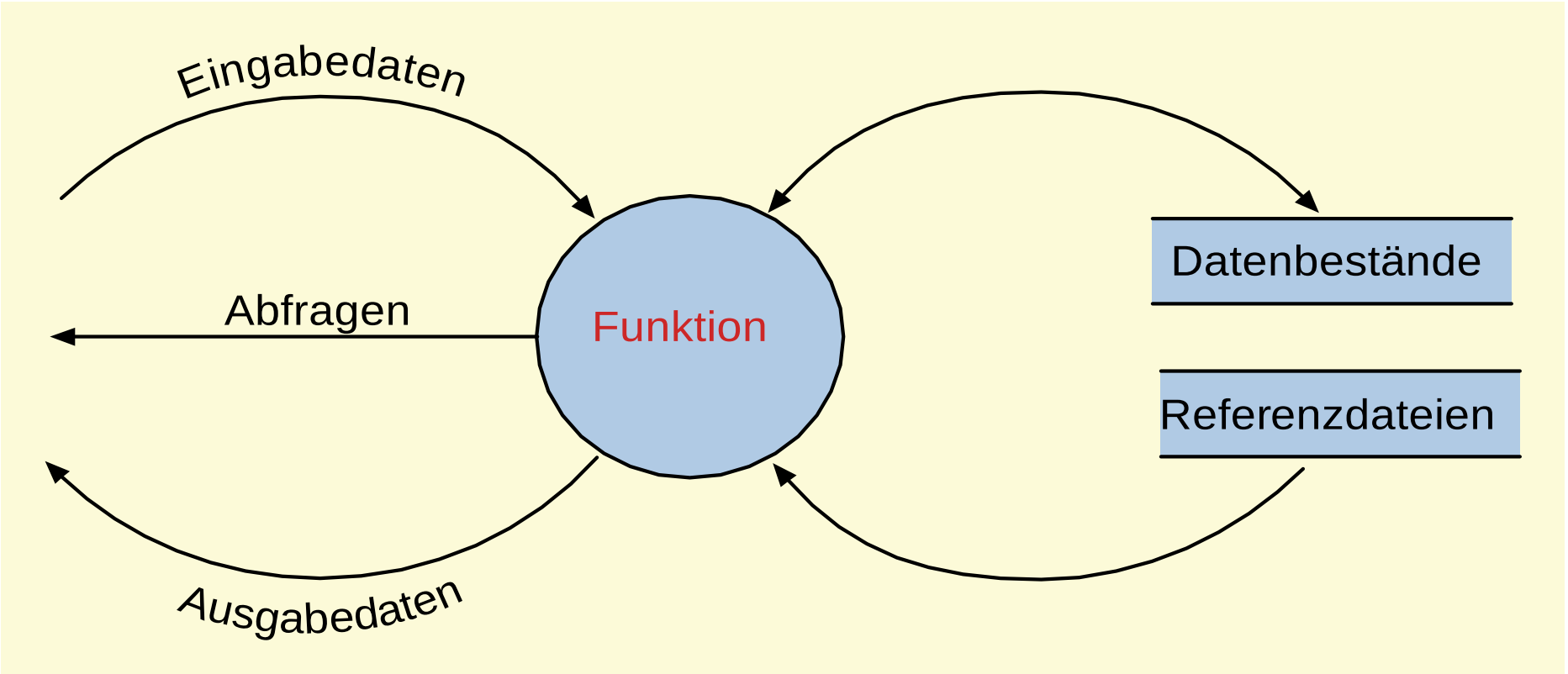
- Allen J. Albrecht (Ausarbeitung nach Balzert, 00)
- Ausgangspunkte:
  - Aufwand hängt vom Umfang und vom Schwierigkeitsgrad des neuen Produktes ab
  - Umfang wird nicht in Lines of Code (LOC) ausgedrückt, sondern direkt aus den Anforderungen abgeleitet
- Vorgehen
  - Zuordnung jeder Produktanforderung in eine von fünf Kategorien

Die 7 Schritte der *Function-Point* Methode

1. Kategorisierung jeder Produktanforderung
   Eingabedaten, Abfragen, Ausgabedaten, Datenbestände, Referenzdaten
2. Klassifizierung jeder Produktanforderung
   einfach, mittel, komplex
3. Eintrag in Berechnungsformular
4. Bewertung der Einflussfaktoren
5. Berechnung der bewerteten ***Function Points*** (**FP**)
6. Ermitteln des Personalaufwands aus einer **FP-PM** (Personenmonaten)-Kurve oder Tabelle
7. Aktualisierung der empirischen Daten als Schätzgrundlage für Folgeprojekt

1. Kategorisierung der Produktanforderungen (1)

- **Kategorien**



- Ermittelt pro Funktion im Lastenheft
- Erfordert Identifikation und Bewertung der Einzelfunktionen

1. Kategorisierung der Produktanforderungen (2)

- Beispiel:

- **/LF 20/:**
  „Benachrichtigung der Kunden (Anmeldebestätigung, Abmeldebestätigung, Änderungsmitteilungen, Rechnung, Werbung)“
  - Diese Anforderung ist der Kategorie „Ausgabedaten“ zuzuordnen
  - Da es sich um 5 verschiedene Ausgaben handelt, wird im folgenden von 5 Ausgaben ausgegangen.

2. Klassifizierung der Produktanforderungen

- Einordnung der Anforderungen in eine der Klassen „einfach“, „mittel“ oder „komplex“

Hauptschwierigkeit!

- Beispiel: Klassifizierung der Datenbestände einer Funktion

Kriterium	**einfach**	**mittel**	**komplex**
Anzahl unterschiedl. Datenelemente	1-5	6-10	>10
Eingabeprüfung	formal	formal logisch	formal logisch DB Zugriff
Ansprüche an die Bedienerführung	gering	normal	hoch

3. Eintrag in Berechnungsformular

Kategorie	Anzahl	Klassifizierung	Gewichtung	Zeilensumme
Eingabedaten		einfach	x 3	=
		mittel	x 4	=
		komplex	x 6	=
Abfragen		einfach	x 3	=
		mittel	x 4	=
		komplex	x 6	=
Ausgaben		einfach	x 4	=
		mittel	x 5	=
		komplex	x 7	=
Datenbestände		einfach	x 7	=
		mittel	x 10	=
		komplex	x 15	=
Referenzdaten		einfach	x 5	=
		mittel	x 7	=
		komplex	x 10	=
Summe			E1	=
Einflußfaktoren (ändern den Function Point-Wert um ± 30%)		1 Verflechtung mit anderen Anwendungssystemen (0–5)		=
		2 Dezentrale Daten, dezentrale Verarbeitung (0–5)		=
		3 Transaktionsrate (0–5)		=
		4 Verarbeitungslogik a Rechenoperationen (0–10) b Kontrollverfahren (0–5) c Ausnahmeregelungen (0–10) d Logik (0–5)		 = = = =
		5 Wiederverwendbarkeit (0–5)		=
		6 Datenbestands-Konvertierungen (0–5)		=
		7 Anpaßbarkeit (0–5) =		
Summe der 7 Einflüsse		E2		=
Faktor Einflußbewertung = E2 100 + 0,7 E3		=		
Bewertete Function Points: E1 * E3				=

Quelle: IBM 85, S. 12

IBM-Werte

- 3. Schritt
- 4. Schritt
- Berechnung der Gewichtung

E3 = E2 / 100 + 0.7

4. Bewertung der Einflussfaktoren

- Die Einflussfaktoren beziehen sich auf die Anwendung als Ganzes und nicht auf einzelne Funktionen oder Funktionspunkte.
- Es wird nicht nur das bloße Vorliegen eines Zustandes bewertet, sondern sein Einfluss auf die Entwicklung.

- **Alternative Ansätze für Bewertung der Einflussfaktoren**:

Ansatz	Anzahl der Faktoren	Bewertungs-spanne	Faktor Einflussbewertung
nach Albrecht	14 Faktoren	(0 ... 5)	(0 ... 70) / 1,64
nach IBM	7 Faktoren	(0 ... 5 / 0 ... 10)	(0 ... 60) / 100 + 0,7
nach IFPUG	14 Faktoren	(0 ... 5)	(0 ... 70) / 100 +0,65
nach Hürten	7 Faktoren	(-2,5% ... 2,5%) (-5% ... 5%)	-30% ... 30% (*)

(*) Auswirkung der Einflussfaktoren in Prozent der *Function Points*

5. Berechnung der bewerteten *Function Points*

- E1 = Summe der Kategorien
- E2 = Summe der 7 Einflüsse

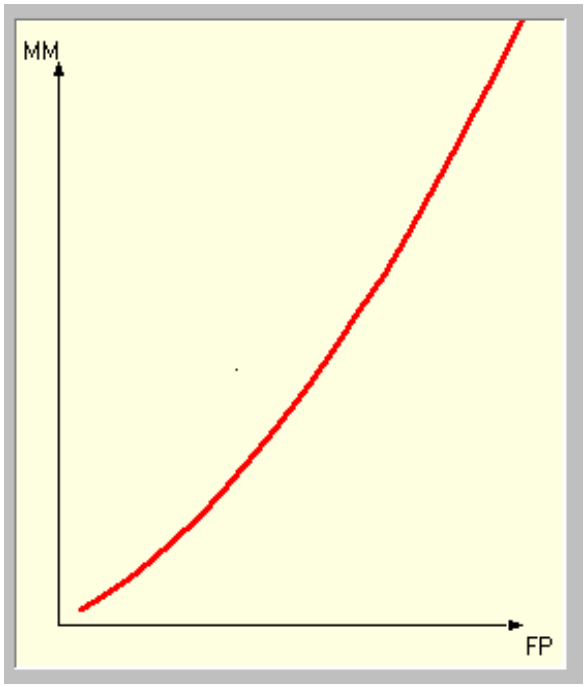
Bewertete *Function Points* = E1 * (E2 / 100 + 0,7)

- Die Summe der Einflussfaktoren (ein Wert zwischen 0 und 60) ändert den *Function Point*-Wert um +/- 30 %

6. Ablesen des Aufwands in MM

- Produktivität nimmt bei großen Projekten ab è nicht-linearer Zusammenhang

Aufwand



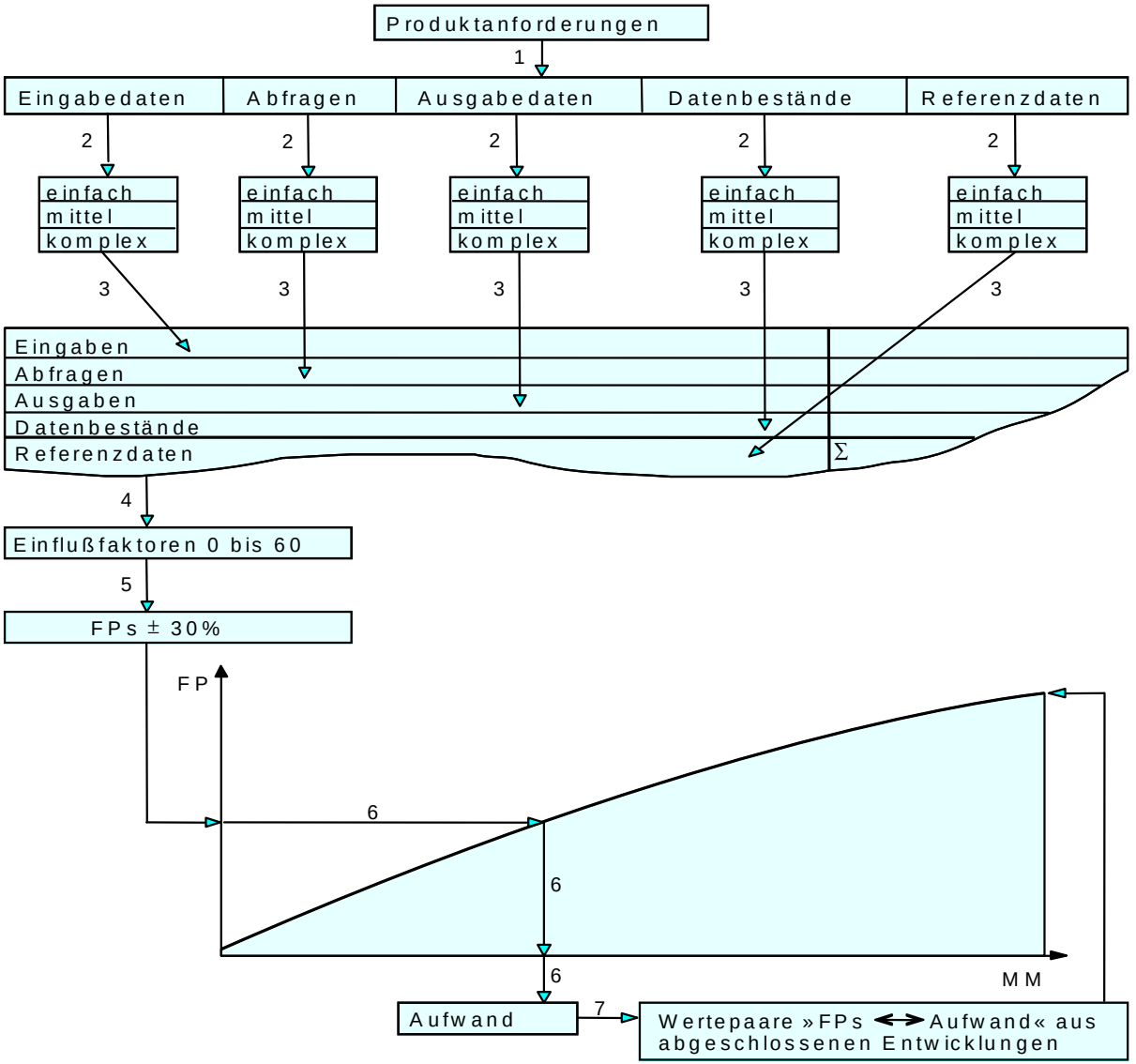
FP

Wertepaare (**FP**, **PM**) aus abgeschlossenen Entwicklungen des eigenen Unternehmens

7. Aktualisierung der empirischen Daten

- Nach der Beendigung einer mit der *Function Point*-Methode geschätzten Entwicklung ist das neue Wertepaar **FP** è tatsächliche **PM** zu verwenden, um die bestehende Kurve zu aktualisieren.

Zusammenfassung



1. Schritt: Kategorisierung für jede Anforderung

2. Schritt: Klassifizierung jeder Anforderung

3. Schritt: Eintrag der jeweiligen Anzahl in das Berechnungs-formular und Ermittlung der unbewerteten FPs

4. Schritt: Bewertung der Einflußfaktoren

5. Schritt: Berechnung der bewerteten FPs

6. Schritt: Ablesen des Aufwandes

7. Schritt: Aktualisierung der Wertepaare, Neu-berechnung der Aufwandskurve

Die *Function Point*-Methode: Voraussetzungen

- Eine Bewertung kann erst dann durchgeführt werden, wenn die Projektanforderungen bekannt sind.
- Eine Bewertung sollte von Mitarbeitern durchgeführt werden, die ein ausreichendes Wissen über die Anforderungen haben.
- Bei der Bewertung muss die gesamte Anwendung / Projektanforderung betrachtet werden.
- Beim Einsatz dieser Methode ist sehr streng darauf zu achten, dass das Anwendungsprojekt aus Sicht der Benutzung betrachtet wird.
- Unternehmensspezifische Schulung und Vorgabe von Beispielen, damit die Wirkung individueller Schätzungen (Klassifizierung und Bewertung der Einflussfaktoren) minimiert werden.
- Der Ist-Aufwand muss für die Nachkalkulation ermittelbar sein.

Die *Function Point*-Methode: Vorteile (1)

- Produktanforderungen, nicht LOC als Ausgangspunkt
- Anpassbarkeit an verschiedene Anwendungsbereiche (Änderung der Kategorien)
- Anpassbarkeit an neue Techniken (Änderung der Einflussfaktoren und der Einflussbewertung)
- Anpassbarkeit an unternehmensspezifische Verhältnisse (Änderung der Einflussfaktoren, der Einflussbewertung und der Klassenfaktoren pro Klasse)

Die *Function Point*-Methode: Vorteile (2)

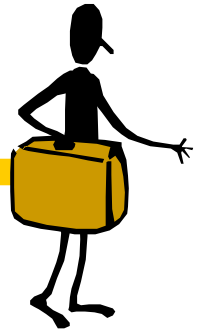
- Verfeinerung der Schätzung entsprechend dem Entwicklungsfortschritt (iterative Methode)
  - Beispiel
    1. Schätzung auf der Grundlage des Lastenheftes
    2. Schätzung auf der Grundlage des Pflichtenheftes
    3. Schätzung nach Erstellung des formalen Modells
- Erste Schätzung bereits zu einem sehr frühen Zeitpunkt möglich (Planungsphase)
- Festgelegte methodische Schritte
- Leicht erlernbar
- Benötigt nur einen geringen Zeitaufwand
- Gute Transparenz
- Gute Schätzgenauigkeit
- Werkzeugunterstützungen verfügbar

Die *Function Point*-Methode: Nachteile

- Es kann nur der Gesamtaufwand geschätzt werden. Eine Umrechnung auf einzelne Phasen muss mit der Prozentsatzmethode erfolgen.
- In der Originalform von Albrecht personalintensiv und nicht automatisierbar
- Zu stark funktionsbezogen
- Qualitätsanforderungen werden nur pauschal berücksichtigt

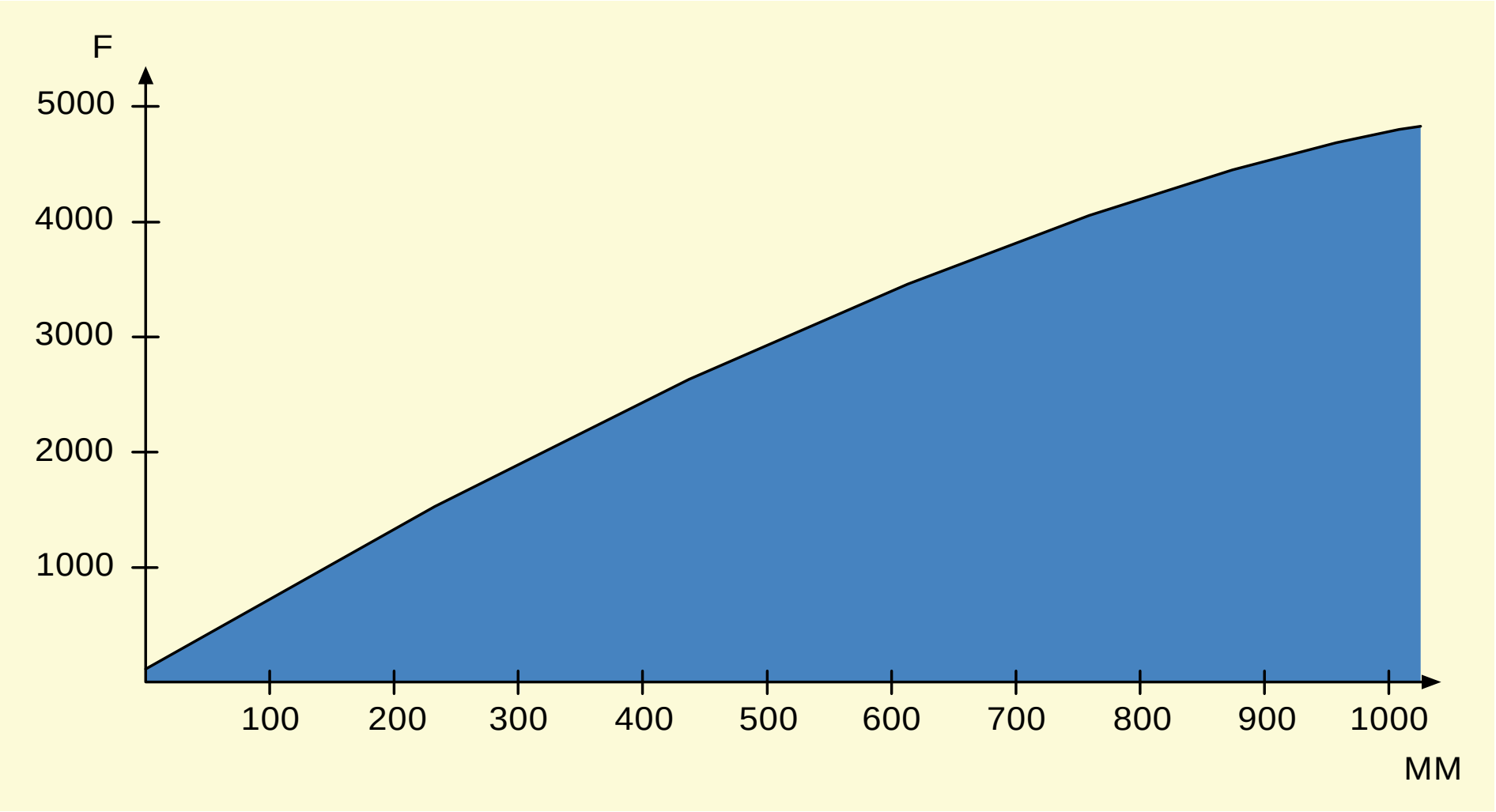
- Hausarbeit als Überblick:
  http://danae.uni-muenster.de/~lux/seminar/ss01/Michaelsen.pdf

Zusammenfassung, Kernpunkte



- Techniken zur Aufwandsabschätzung
- Einfache Methoden
- Kombination von einfachen Methoden
  - COCOMO
  - Function Point-Methode

Anhang: Die IBM-Kurve: **FP** -> **MM**



Schätzmethode	Erscheinungsjahr	Zugrundeliegende Basismethode	Einsatzzeitpunkt	Berücksichtigte Faktorengruppe
SDC	67	Ps	P	1
IBM-Handbuch	68	G, P	D	1, 4
SURBOCK	78	G, P	P, D	1, 2, 4
ARON	69	M, P	P	1, 2, 3, 4
T.O.P	71	A, G	P, D, E	1, 2, 3
Shell	72	G, P, Ps	D	1, 4
Wolverton	74	M	P	1
SLIM	74	PS	P, D, E	1, 4
FUTH	75	G, P	D	1, 4
Software-Part	76	A	P	1, 2, 4
EGW	77	G	P, E	1, 4
Boeing	77	G, Ps, M	P, E	1, 4
IFA-PASS	77	A, P	P	1, 2, 4
DOTY	77	G, Ps	P	1, 2, 4
GRIFFIN	77	G, P	P	1, 4
Schneider	78	Ps	P	1
INVAS	80	R, G	I	1, 2, 3, 4
ZKP	80	G, P	D	1, 4
COCOMO	81	G, Ps	P	1, 2, 3, 4
Function Point	81	A, G	I	1, 2, 3, 4

Tab. 1.4-1: Übersicht über Aufwandsschätzmethoden nach /Noth, Kretschmar 86/

Legende:

Zugrundeliegende Basismethode:
- A = Analogiemethode
- M = Multiplikatormethode
- R = Relationsmethode
- G = Gewichtungsmethode
- Ps = Parametrische Schätzgleichung
- P = Prozentsatzmethode

Einsatzzeitpunkt:
- P = Planungphase
- D = Definitionsphase
- E = Entwurfsphase
- I = Iteratives Verfahren

Berücksichtigte Faktorengruppe:
- 1 = Quantität
- 2 = Qualität
- 3 = Entwicklungsdauer
- 4 = Produktivität