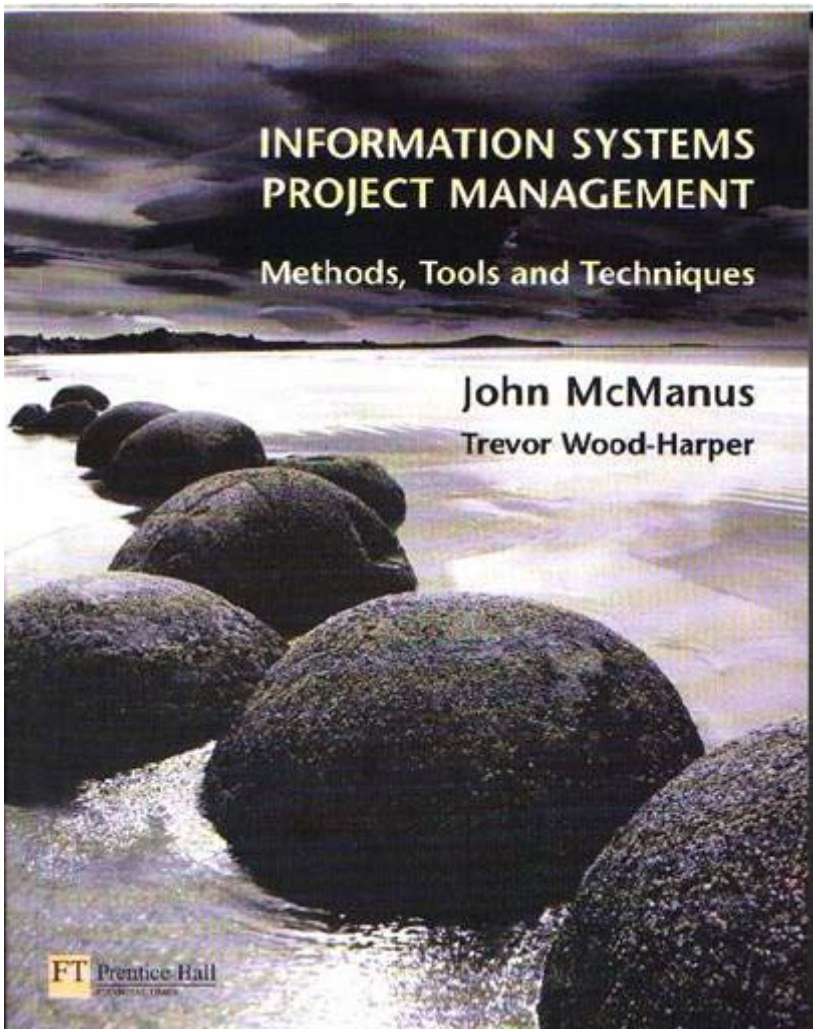
Management großer Softwareprojekte

Prof. Dr. Holger Schlingloff

Humboldt-Universität zu Berlin,
Institut für Informatik

Fraunhofer Institut für Rechnerarchitektur
und Softwaretechnik FIRST

Neuer Literaturhinweis



- gerade erschienen („2003"), Pearson
- ca. 47,27 €
- Managementaufgaben - Lebenszyklus – Kostenschätzung - Qualitätsmanagement

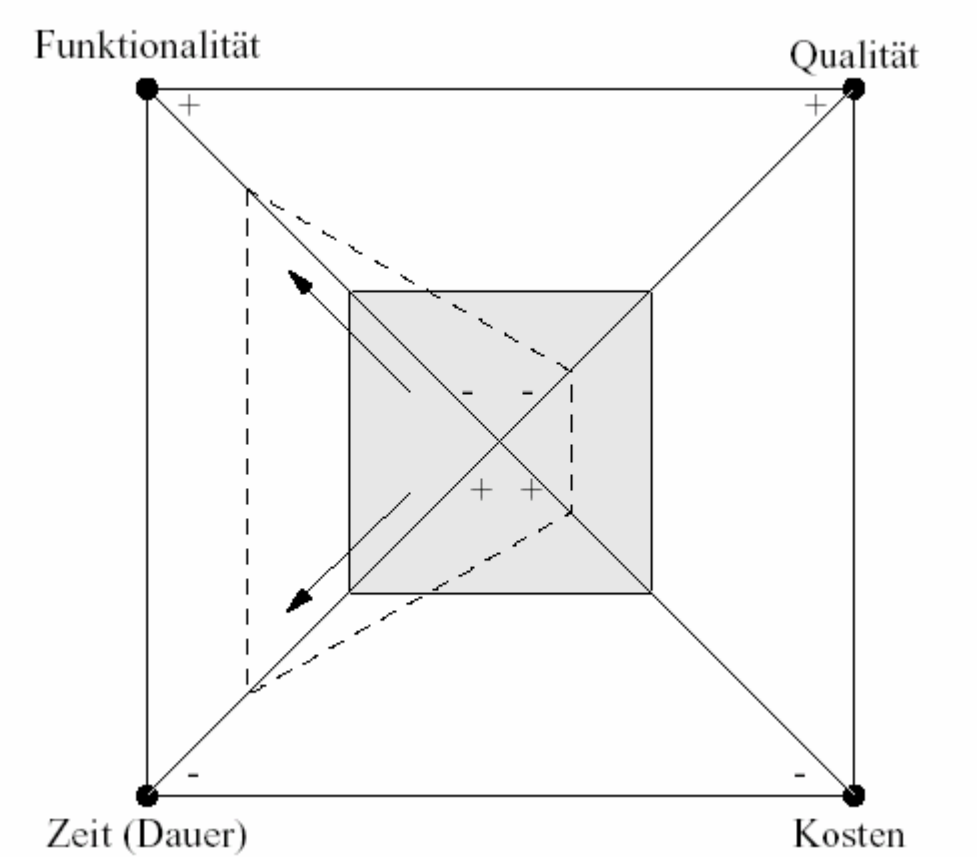
4.1 Aufwandsschätzung - allgemein

- Schätzen ist kein Raten!
- Vorhersage auf der Basis früher gesammelter Informationen
- Auswahl, Gewichtung, Auswertung und Interpretation von Vergleichsdaten

Grundregeln beim Schätzen

- 1. Grundregel: Müll rein – Müll raus
  ➔ präzise Anforderungen!
- 2. Grundregel: Je ferner die Zukunft, desto schwieriger sind die Schätzungen
  ➔ kontinuierliche Neuschätzungen!
- 3. Grundregel: große Blöcke sind schwieriger zu schätzen als kleine, abstrakte schwieriger als konkrete
  ➔ Granularität, Konkretisierung
- 4. Grundregel: Schätzungen sind keine Weissagungen, d. h. keine *verbindlichen* Voraussagen (selbsterfüllende Prophezeiung)
  ➔ Toleranzspielraum!

Sneed's Teufelsquadrat



Problematik des Schätzens

„Prognosen sind besonders dann schwierig, wenn sie sich auf die Zukunft beziehen“

- eigentliche Funktionalität nur kleiner Teil des Aufwands (Verwaltung, Fehlerbehandlung, Benutzungsschnittstelle, …)
- Übernahme früherer Erfahrungen nicht immer möglich (Einmaligkeit der Projektbedingungen)
- Software-Erstellung weitgehend personenbezogen; Produktivität kann um mehr als Faktor 10 schwanken
- „Mythos Personenmonat“ (lesen: Brooks)
- Programmierer programmieren nicht 100% ihrer Zeit.

Schätzen im Projektablauf

- Projekte scheitern nicht wegen fehlerhafter Schätzung, sondern wegen anderer Ursachen:
  - Motivation, Identifikation der Mitarbeiter
  - Zusagen auf Grundlage falscher Daten
  - fehlendes Anforderungs-/ Änderungsmanagement
  - mangelnde Kontrolle / Management

- Für das Schätzen gibt es feste Techniken

Planspiel: Pötzseil

Granularität

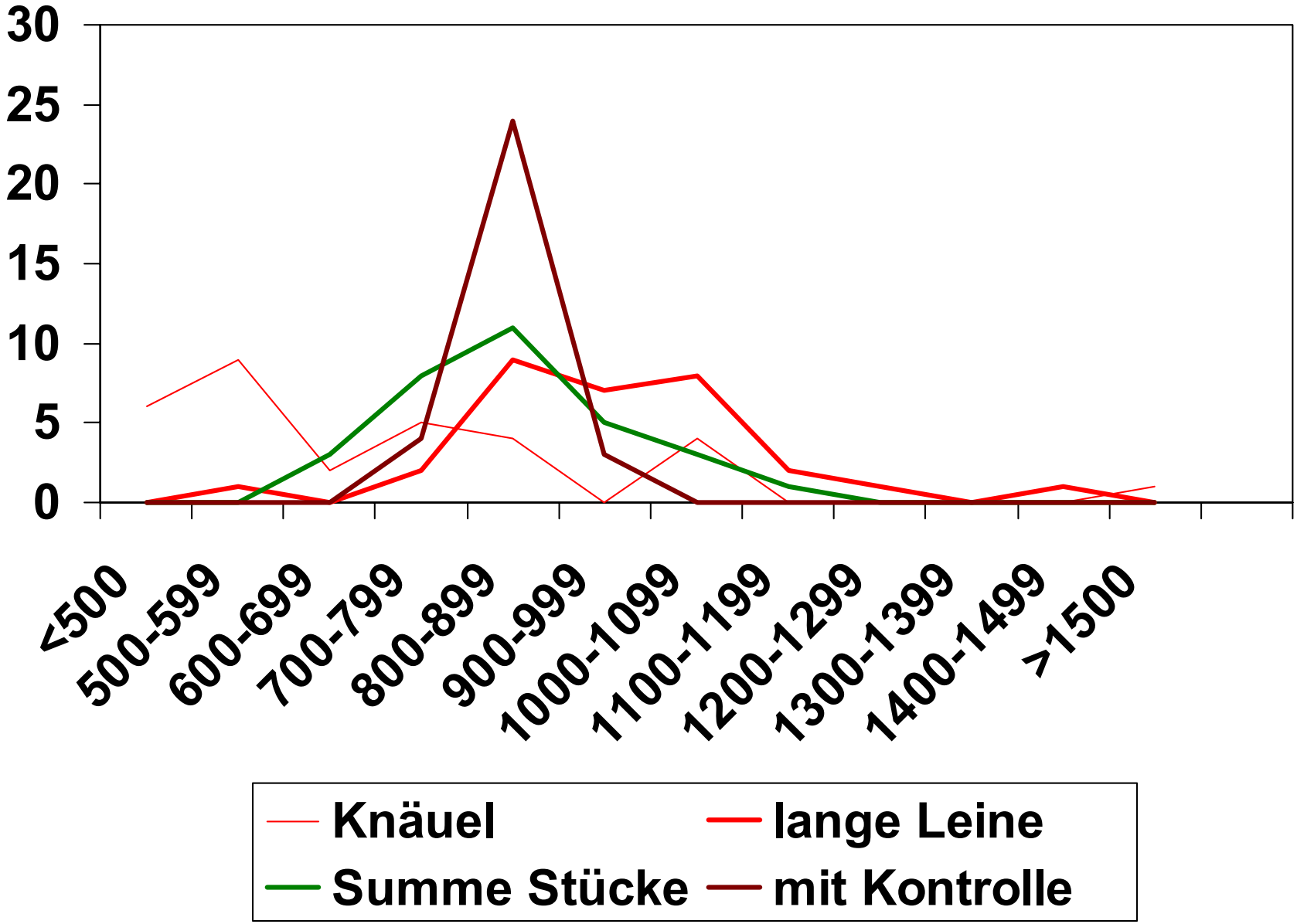
- Faustregel: je feingranularer die Aufgabe, desto präziser das Resultat

  es ist einfacher, den Aufwand für eine Funktion mit 20 Zeilen zu schätzen als für ein Programm von 20000 Zeilen

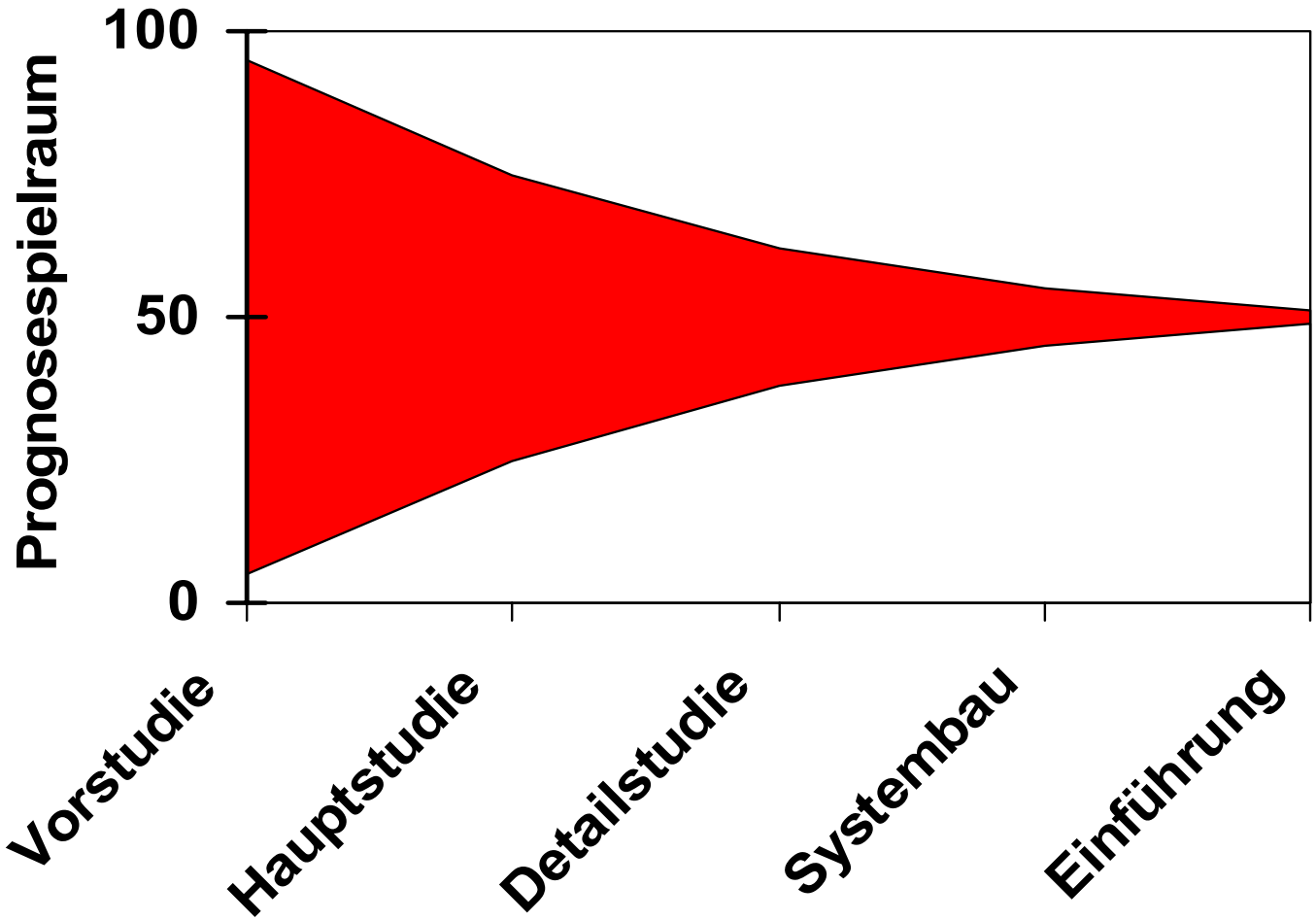
- Methode: Aufteilung der Schätzung des gesamten Entwicklungsaufwandes
  - gemäß Funktionalität oder
  - gemäß Phasenablauf

Top-down versus bottom-up

- *Top-down Schätzung:*
  Schätzung auf Grund der allgemeinen Funktionalität und deren Aufteilung auf Teilfunktionen.
  Basis: logische Funktionen statt Komponenten, welche die Funktionalität implementieren
- *Bottom-up Schätzung:*
  Bestimmung des Aufwandes für jede einzelne Komponente; Gesamtaufwand ist Summe aller Teilaufwände


30
25
20
15
10
5
0
<500
500-599
600-699
700-799
800-899
900-999
1000-1099
1100-1199
1200-1299
1300-1399
1400-1499
>1500
Knäuel
lange Leine
Summe Stücke
mit Kontrolle

Zeitabhängigkeit der Schätzung



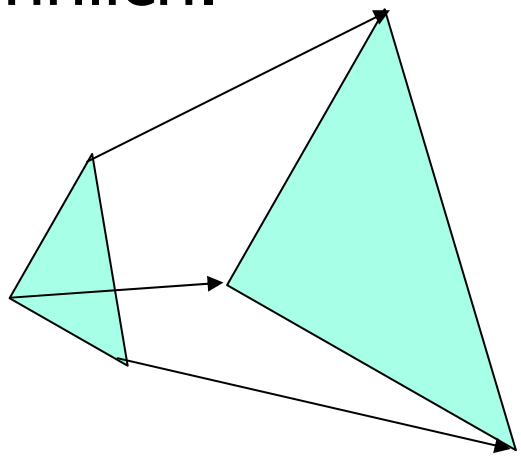
Stetigkeit des Projektverlaufs

- Faustregel: aus der unmittelbaren Vergangenheit lässt sich auf die nahe Zukunft schließen
- Methode: Abweichungskontrolle
  - Für jede Phase wird zusätzlich mit dem geschätzten Aufwand eine maximale Abweichung festgelegt
  - Bei Über- oder Unterschreitung erfolgen Korrekturmaßnahmen

8,70

Analogieschlüsse

- Faustregel: Unter ähnlichen Voraussetzungen verhalten sich Menschen ähnlich.
- Methode: Analogieschluss



Ähnlichkeitskriterien:

- Anwendungsgebiet
- Produktumfang
- Komplexitätsgrad
- Werkzeuge, Sprachen
- Personal
- Qualitätsanforderungen
- ...

Beispiel Analogiemethode

- Modula2- aus Pascal-Compiler (20 PM)
  - 50% Wiederverwendung vorhandenen Codes
  - 50% Überarbeitung vorhandenen Codes
  - 20% Neue Features (schwierig)
- Faktoren für den Analogieschluss
  - Wiederverwendung: 0.25
  - Überarbeitung: 0.8
  - Neue Features: 1.5
- Rechnung
  - 10 PM * 0.25 + 10 PM * 0.8 + 4 PM * 1.5 = 16,5 PM

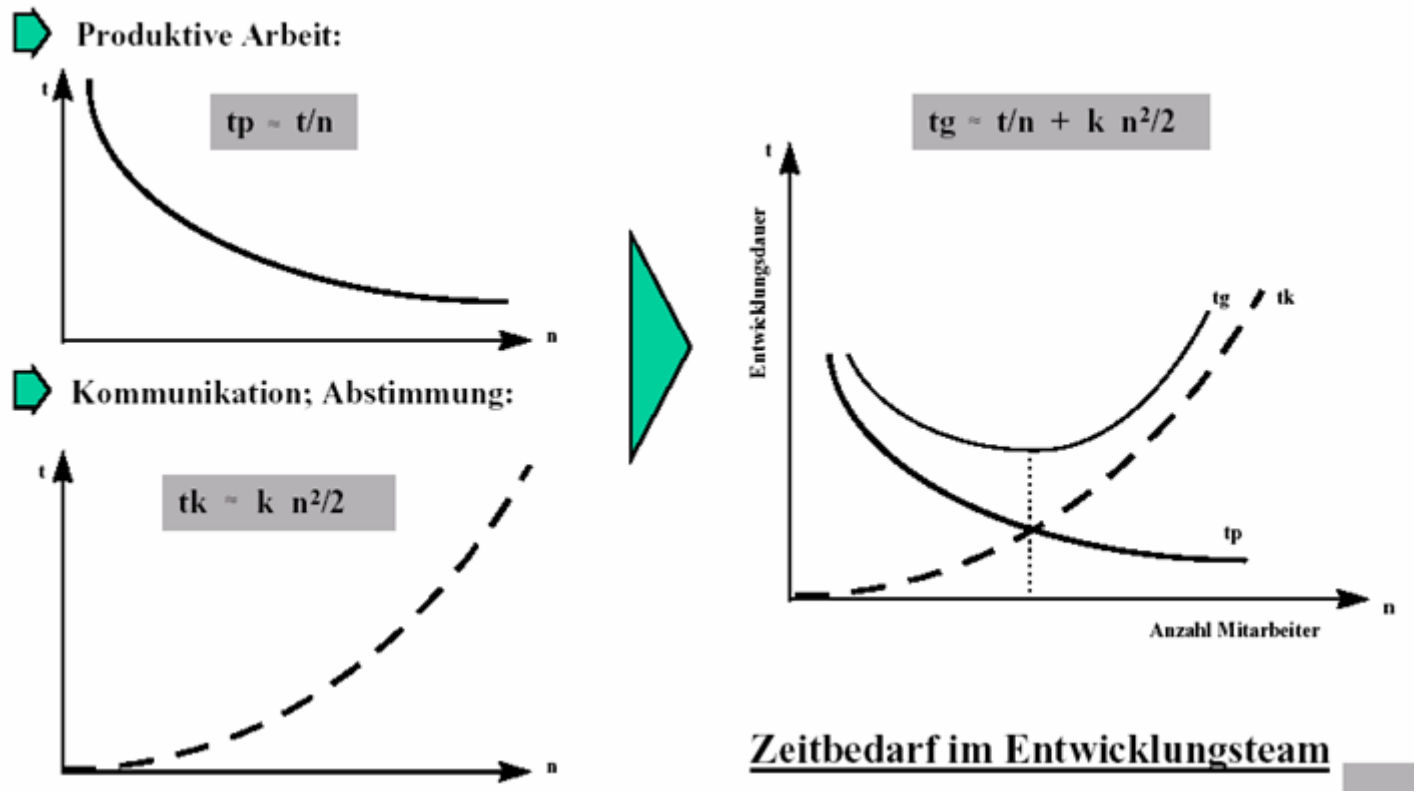
Einfluss von Vergleichsdaten

- Hausbau: Kosten für ein Einfamilienhaus können mit einer Genauigkeit von weniger als 1000 € (< 1 %) angegeben werden
- Umzug nach Berlin: Abweichungen von mehr als 10% sind möglich
- Typische Softwareprojekte: Abweichungen von 30% sind üblich
- Flug zum Mars: Gesamtkosten können erst sehr spät abgeschätzt werden

„Mythos Personenmonat“

Brooks‘s Law: Adding manpower to a late project makes it later

- Parallelitätsfaktor linear, Kommunikations-Overhead quadratisch



Entwicklungszeit

- Entwicklungsdauer $t \sim \frac{1}{n} + k \cdot \frac{n \cdot (n-1)}{2} \sim \frac{1}{n} + k \cdot \frac{n^2}{2}$

  ➔ Optimum $n \sim \sqrt[3]{\frac{1}{k}}$ (nahe bei 1)

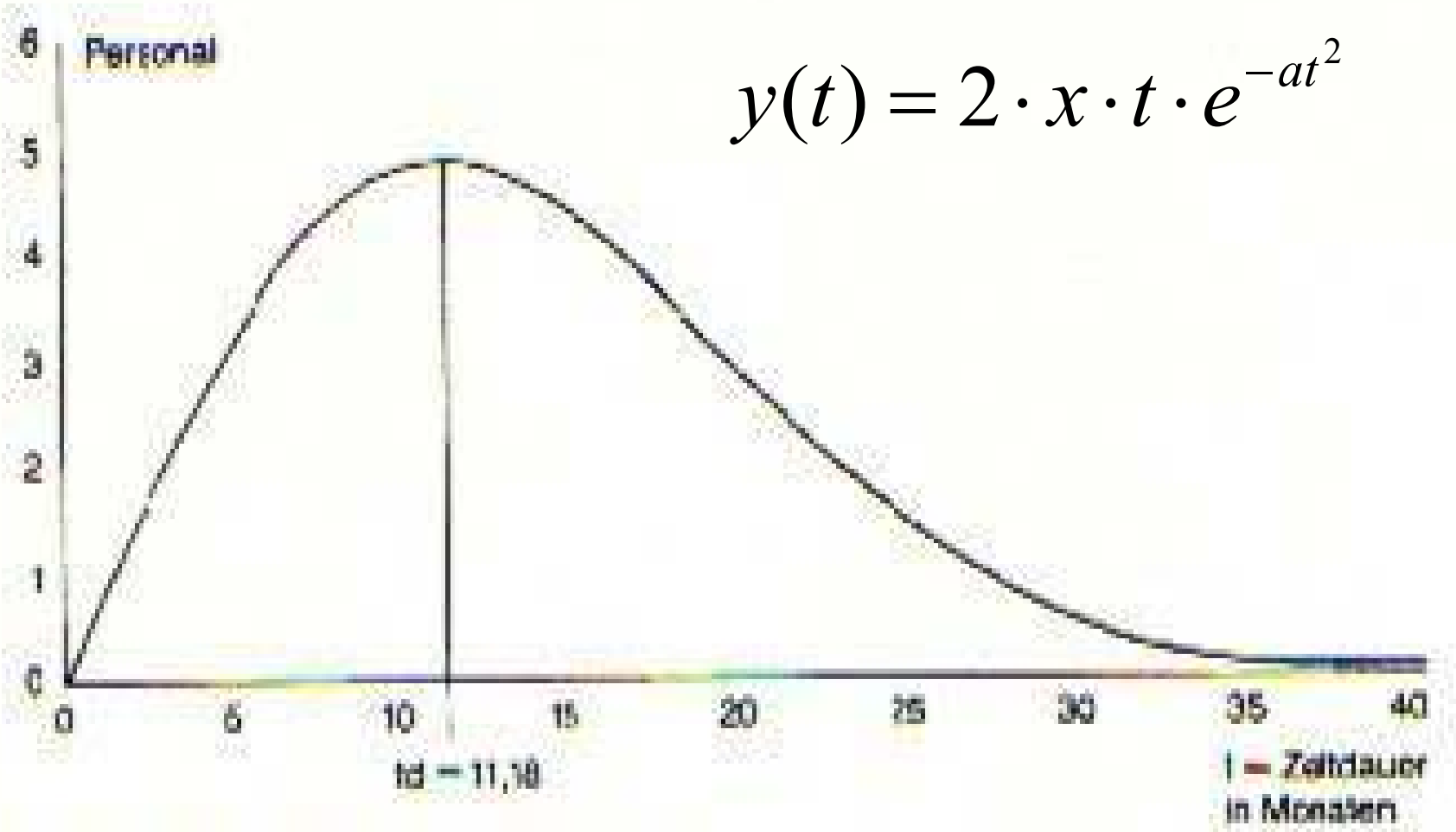
- Gesamtaufwand x [PM]
- optimale Entwicklungszeit: $t \sim 2.5 \cdot \sqrt[3]{x}$

  (Parameter adjustierbar)

  z.B. x = 100 PM ➔ t = 12 M ➔ n = 8,33

  plus Kommunikationsoverhead = 10 MA

Personaleinsatzplanung



$$y(t) = 2 \cdot x \cdot t \cdot e^{-at^2}$$

Was kostet ein Projekttag?

- Ein Projektjahr (PJ) hat 9-10 Projektmonate
  (Urlaub, Krankheit, Feiertage, …)
- Ein PM hat 16 PT (nicht 20)
  (Fortbildung, Administration, …)
- Ein PT hat 6 PS (nicht 8)
  (Mail, Besprechungen, Berichte, Kaffee, …)
- Ein PJ besteht aus 1000 PS effektiver Arbeit („offiziell" 1688 = 225 d * 7,5 h/d)
- Ein PJ kostet 55/100/150 K€ (Uni/FhG/Firma)

Produktivität

- LOC/PT ? NCSS/PT ? FP ?
- Beispiel: 200 LOC, Produktivität 25 LOC/PT
- Dauer 5 PT ➔ 1,6 MA nötig
- Brooks‘s Law hier außer Betracht

Def.:

- LOC = Lines of Code
- NCSS = Non-Commented Source Statements
- FP = Function Points

Wie groß ist die Produktivität?

- Faustregel: 350 LOC / PM über alles

  (HP 1992, Durchschnitt über 135 Projekte)

- Codierung = 20-30% des Gesamtaufwandes

  ➔ 70 - 100 LOC / PT in der Implementierungsphase

- Beispiel:
  - geschätzte Größe = 10 KLOC
  - ➔ geschätzter Aufwand 30 PM

ACHTUNG ACHTUNG: sehr grobe Schätzung !!!

Parkinsons Gesetz

Jede Arbeit verteilt sich auf die dafür vorgesehene Zeit

- 2 Richtungen
  - zu viel Zeit ➔ Arbeit wird gestreckt
  - zu wenig Zeit ➔ Arbeit wird komprimiert
- Gilt natürlich nur in gewissen Grenzen
- Extrem gefährliches Steuerungsmittel!
- Sättigungseffekt