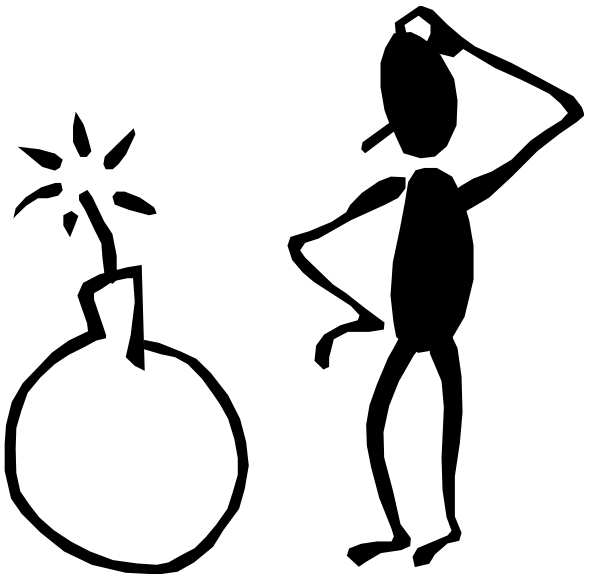
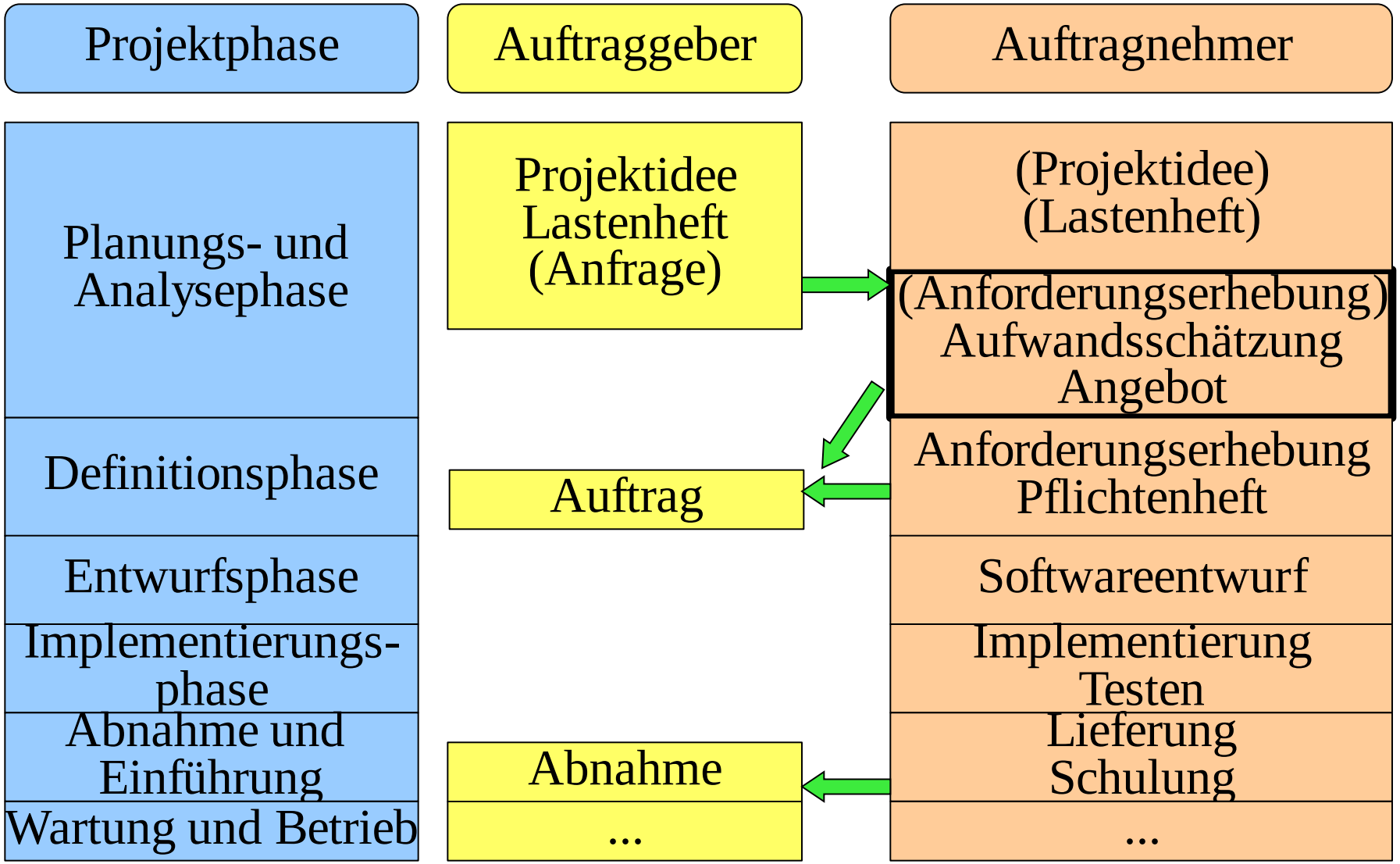
Themencluster: Softwareentwicklung

Thema: Machbarkeitsstudie (feasibility study)
Aufwandschätzung

Dr. Walter Rafeiner-Magor

Weshalb benötige ich eine Machbarkeitsstudie?




Projektphase
Auftraggeber
Auftragnehmer
Planungs- und Analysephase
Definitionsphase
Entwurfsphase
Implementierungs-phase
Abnahme und Einführung
Wartung und Betrieb
Projektidee Lastenheft (Anfrage)
Auftrag
Abnahme
...
(Projektidee) (Lastenheft)
(Anforderungserhebung) Aufwandsschätzung Angebot
Anforderungserhebung Pflichtenheft
Softwareentwurf
Implementierung Testen
Lieferung Schulung
...

Aufgaben der Planungsphase

- Erstellen eines Lastenhefts
- **Machbarkeitsstudie**
- Aufwands- (Zeit- und Kosten-)schätzung
  - Angebotskalkulation
- Risikoanalyse
- Erstellen eines (vorläufigen) Angebots

Definition Machbarkeitsstudie

Feasibility Study (engl) Die Projektstudie nach DIN 69905 umfasst die Untersuchung von Lösungsmöglichkeiten zur Erreichung des benannten Projektziels und die Überprüfung ihrer jeweiligen Machbarkeit.

Im allgemeinen Sprachgebrauch werden meist die Begriffe "Machbarkeitsprüfung" bzw. "Machbarkeitsstudie" verwendet.

In der DIN wird explizit darauf hingewiesen, dass das Ergebnis der Projektstudie eine Neuformulierung des Projektzieles sein kann.

Aufgaben der Machbarkeitsstudie

- Hauptfunktionen
  - Realisierungsfähigkeit
  - Variantenbildung
  - Risikoanalyse
- Entscheidung über weitere Vorgangsweise
  - Go/nogo

Achten Sie bei Software-Produktion:
- konkreten Auftraggeber
- Anonymen Markt

Realisierbarkeit

- 3 Richtungen der Prüfung
  - **ökonomische** Durchführbarkeit
  - **fachliche** Durchführbarkeit
  - **personelle** Durchführbarkeit

Softwareproduktion für den anonymen Markt:
- Bedarf und Rentabilität muss geklärt werden

Allgemeine Studie

Varianten

Ökonomische Durchführbarkeit

Fachliche Durchführbarkeit

Personelle Durchführbarkeit

Risiko-analyse

Aufwandsschätzung
Terminschätzung
Wirtschaftlichkeit

Technische Realisierbarkeit (Aufgabenstellung, Entwicklungs- und Produktionssystem)

Skills der Fachkräfte (Menge, Qualität und Zeit)

Alternative Lösungsmöglichkeiten
Technische Varianten
Wirtschaftliche Varianten (zb. Applikationsentwicklung versus Customizing)

Rentabilitätsanalyse

Erweiterte Studie

Varianten

Ökonomische Durchführbarkeit

Fachliche Durchführbarkeit

Personelle Durchführbarkeit

Risiko-analyse

Markt-analyse

(Standort-analyse)

Analyse Nutzungs-konzept

Wettbewerbs-analyse

Rentabilitätsanalyse

Planungsgrundlagen der Machbarkeitsstudie

- Trendstudien
- Marktanalysen
- Kundenanfragen
- Forschungsergebnisse
- Vorentwicklungen

Quelle:H. Balzert: Lehrbuch der Softwaretechnik, 2001

Beispiel für eine Machbarkeitsstudie

- Problem (Lockheed, Kalifornien 1981):
  - täglich ein Dutzend Zeichnungen über 25 Meilen zu übertragen
  - Kurier braucht eine Stunde (einfach)
  - kostet ca. $100 am Tag
- Aufgabe
  - alternative Lösungen der DFÜ vorschlagen
  - Kosten abschätzen

Quelle:J Bentley Programming Pearls.

Beispiel für eine Machbarkeitsstudie

Mögliche Lösungen

1. existierende Funkstrecke benutzen
   a. zweiter Plotter erforderlich
   b. teuer und langsam
2. Mikroverfilmung der Zeichnungen und verschicken mit Brieftauben
   a. im Vergleich zum Kurier Hälfte der Zeit und 1% der Kosten
   b. in 16 Monaten mehrere hundert Zeichnungen bei nur zwei Verlusten

Quelle:J Bentley Programming Pearls.

Aufwandsschätzung

- Wirtschaftlichkeit des Produkts
- Kosten der Entwicklung/Betrieb
- Entwicklungsdauer

Verfahren der Aufwandsschätzung

- Analogiemethode
- Relationenmethode
- Function-Point-Methode
- Usecase-Point-Methode
- …

Der Second-System-Effekt

- Das erste System
  - ist klein und fein
  - man weiß, dass man nichts weiß
  - Ideen werden für Folgesysteme aufgehoben
- Ergebnis
  - **wird** irgendwann fertig
  - man kann´s

Der Second-System-Effekt

- Das zweite System
  - ist das gefährlichste, das man jemals baut
  - man **weiß**, wie es geht
  - alle **guten Ideen** endlich umsetzen
- Ergebnis
  - total over-designed/over-engineered
  - veränderte Benutzeranforderungen ignoriert

Der Second-System-Effekt

- Das dritte und alle weiteren Systeme
  - sind (hoffentlich) **wieder ok**
- Was tun?
  - Zurückhaltung üben (k.i.s.s)
  - jemanden einstellen, der schon mehr als zwei Projekte abgewickelt hat

Produktivitätszahlen
Größe

Größe	**Programmierer**	**Dauer**	**LOC** [1]
trivial	1	1–4 W	< 500
klein	1	1–12 M	500–5k
mittel	2–5	1–2 J	5k–50k
groß	5–100	2–3 J	50k–100k
sehr groß	100–1.000	3–5 J	100k–1M
richtig groß	1.000–5.000	5–10 J	> 1M

[1] LOC: Lines of Code

Produktivitätszahlen

Arten von Software

- **Application Software:**
  Software designed to fulfill specific needs of a user; for example, software for navigation, payroll, or process control.
- **Support Software:**
  Software that aids in the development or maintenance of other software, for example compilers, loaders, and other utilities.
- **System Software:**
  Software designed to facilitate the operation and maintenance of a computer system and its associated programs; for example, operating systems, assemblers, utilities.

Quelle: IEEE Standard Glossary of Software Engineering Terminology (1990).

Produktivitätszahlen

Produktivitätsunterschiede

- nach Produktart
  - Anwendungsprogramme 25–100 LOC/d
  - Dienstprogramme 5–10 LOC/d
  - Systemprogramme < 1 LOC/d
- individuell
  - 10:1 vom Besten zum Schlechtesten
  - 2,5:1 vom Besten zum Mittleren
  - 2:1 Anzahl bessere zu schlechtere Mitarbeiter

Quelle: Tom DeMarco, Timothy Lister Peopleware – Productive Projects and Teams (Dorset House Publishers1987).

Produktivitätszahlen
weiter Faktoren

- **Firmenkultur**
  - Gute Software-Ingenieure scharen sich in bestimmten Firmen zusammen,
  - schlechte in anderen.
  - Gute Software-Ingenieure werden von schlechten Firmen nicht angezogen oder können nicht gehalten werden
  - Verhältnis Produktivität beste zu schlechteste Firma: 11:1
- **Faktoren mit wenig Einfluss**
  - Programmiersprache
  - Berufserfahrung
  - Anzahl der Fehler
  - Gehalt

Quelle:H. Balzert: Lehrbuch der Softwaretechnik, 2001

Produktivitätszahlen

Zeitaufteilung von Programmierern

- Post 5%
- Ausbildung 6%
- Urlaub/Krankheit 13%
- Programmieren 13%
- Verschiedenes 15%
- Lesen 16%
- Besprechungen 32%

Programmierer verbringen 90% ihrer Zeit mit den ersten 80% des Projekts, und weitere 90% mit den verbleibenden 20%.

Wirtschaftlichkeit des Produkts

Gewinn (Verlust)=

Deckungsbeitrag * geschätzte Menge

- einmalige Entwicklungskosten

Deckungsbeitrag=

Preis – (laufende) variable Kosten

Quelle:H. Balzert: Lehrbuch der Softwaretechnik, 2001

Kosten eines Software-Systems

- Entwicklungskosten
  - Hauptanteil der Entwicklungskosten:
    - Personalkosten
    - Lizenzkosten
  - anteilige Umlegung der CASE-Umgebungskosten (einschl. Hardware und Systemsoftware) für die Produktentwicklung
  - Kosten für andere Dienstleistungen, Büromaterial, Druckkosten, Dokumentation, Reisekosten usw. im Verhältnis zu den Personalkosten bedeutungslos

Methoden zur Kosten- und Terminschätzung

- basieren zumeist auf dem geschätzten Umfang des zu erstellenden Software-Produktes in Lines of Code (LOC)
  - bei höheren Sprachen z. B. alle Vereinbarungs- und Anweisungszeilen geschätzt
  - geschätzter Umfang wird durch einen Erfahrungswert für die Programmierproduktivität (in LOC) eines Mitarbeiters pro Jahr oder Monat geteilt
  - Ergebnis: geschätzter Aufwand in MJ oder MM (1 MJ = 9 MM oder 10 MM)

Beispiel

- Software-Produkt mit geschätzten 21.000 LOC soll realisiert werden
- durchschnittliche Produktivität pro Mitarbeiter: 3.500 LOC/Jahr
  - 6 Mitarbeiterjahre werden benötigt
  - arbeiten 3 Mitarbeiter im Team zusammen, dann werden 2 Jahre bis zur Fertigstellung benötigt!

Gefährlicher Trugschluß

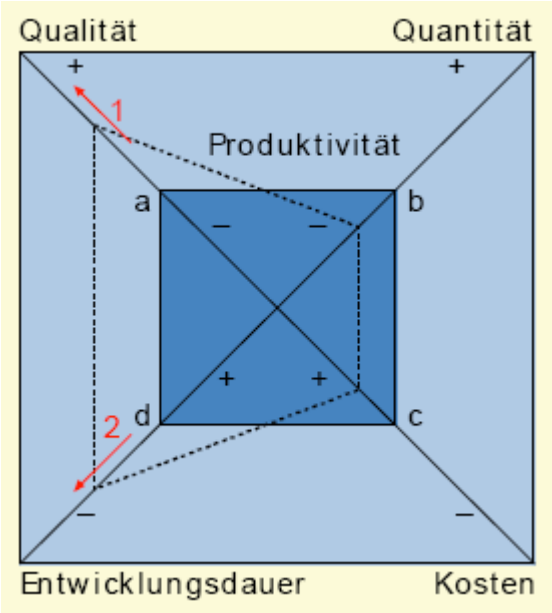
Dauer ≠ Aufwand / Mitarbeiter

Neun Frauen können ein Baby nicht in einem Monat bekommen.

Teufelsquadrat nach Sneed

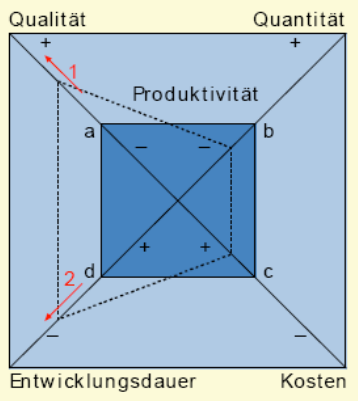
Fläche ist konstant!

- Einflussfaktoren
  - Quantität
  - Qualität
  - Entwicklungsdauer
  - Kosten
- Produktivität
  (=Fläche des Quadrats)



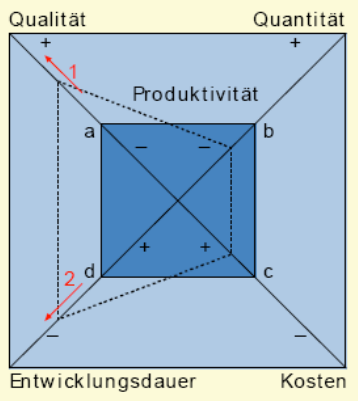
Quelle:H. Balzert: Lehrbuch der Softwaretechnik, 2001

Quantität



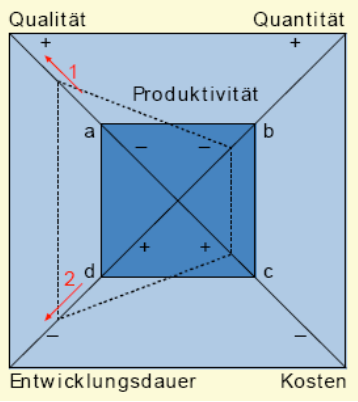
- Größe
  - Maß „Lines of Code“ (LOC) oder „Function Points“ (FP)
  - lineare oder überproportionale Beziehung zwischen LOC und dem tatsächlichen Aufwand
- Umfang
  - Problem, den Funktions- und Datenumfang exakt zu definieren
  - Maß unabhängig von einer Programmiersprache
- Komplexität
  - qualitative Maße »leicht«, »mittel« und »schwer«
  - Abbildung auf Zahlenreihe (z. B. Noten zwischen 1 und 6)

Qualität



- je höher die Qualitätsanforderungen, desto größer der Aufwand
- Es gibt nicht die Qualität, sondern es gibt verschiedene Qualitätsmerkmale.
- Jedem Qualitätsmerkmal lassen sich Kennzahlen zuordnen.
- Beispiele: **Fehlerfreiheit, Robustheit, Zuverlässigkeit**

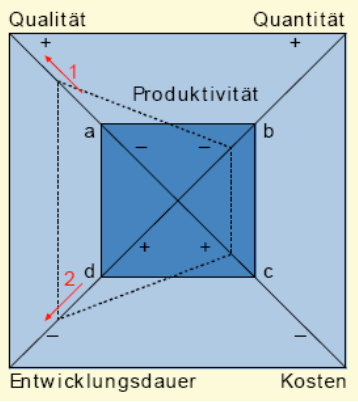
Entwicklungsdauer



- Soll die Zeit verkürzt werden, dann werden mehr Mitarbeiter benötigt.
- Mehr Mitarbeiter erhöhen den Kommunikationsaufwand innerhalb des Entwicklungsteams.
- Der höhere Kommunikationsanteil reduziert die Produktivität.
- Kann die Entwicklungsdauer verlängert werden, dann werden weniger Mitarbeiter benötigt:
  - der Kommunikationsanteil sinkt und
  - die Produktivität jedes Mitarbeiters steigt.

Adding more people to a late software project makes it later.

„Optimale Entwicklungsdauer“



optimale Entwicklungsdauer =

$2{,}5 * (\text{Aufwand in MM})^{s}$ [Monate]

mit $s = 0{,}38$ für Stapel-Systeme

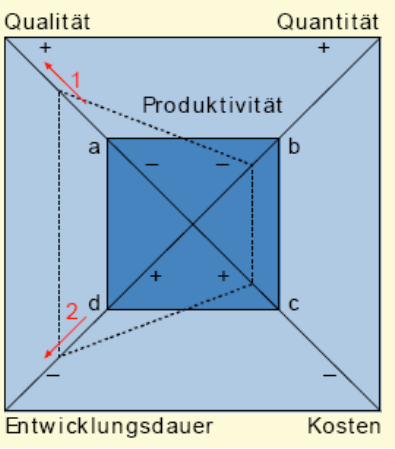
$s = 0{,}35$ für Dialog-Systeme

$s = 0{,}32$ für Echtzeit-Systeme

- Beispiel
- geschätzter Entwicklungsaufwand Dialog-System: 9 MM
  - optimale Entwicklungsdauer: 2,5 * 90,35 [Monate] = 5,3 Monate
  - durchschnittliche Teamgröße: 9 MM / 5,3 Monate = 1,7.

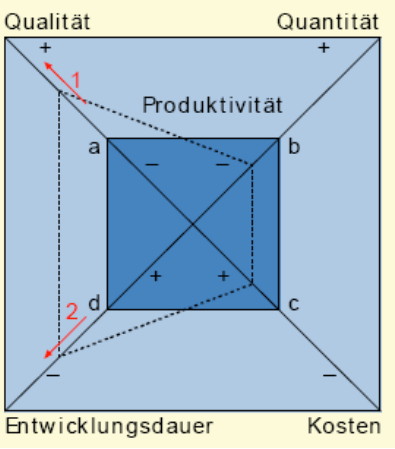
Quelle:H. Balzert: Lehrbuch der Softwaretechnik, 2001

Kosten



- bei Eigenentwicklung proportional zu Aufwand in Mitarbeitermonaten (unter Vernachlässigung unterschiedlicher Bezahlung)
- im Teufelsquadrat Maß für die Teamgröße (da Entwicklungsdauer eigenständiger Faktor)
- ggf. auch Zukauf von Lösungen

Produktivität



- wird von vielen Faktoren beeinflusst (nicht alle veränderbar)
  - Lernfähigkeit und Motivation der Mitarbeiter entscheidend
  - Einsatz geeigneter Werkzeuge ebenso
  - aber: versprochene Produktivitätssteigerungen durch neue Technologien bleiben häufig aus

Projektplanung

Good judgement comes from experience, and experience comes from bad judgement.

Das einzige, das ich nicht genau vorhersagen kann, ist die Zukunft.
Woody Allen

Planung ersetzt den Zufall durch Irrtum.
Albert Einstein

Der Gesundheitsminister warnt: Projektaufwandsschätzung schadet Ihrer Karriere.
Phillip Armour

Vielen Dank!