

Teil III:

Begleitende Aktivitäten

Projektorganisation: Teilaspekte

Dokumentation

Qualitätssicherung

9. Projektorganisation: **Teilaspekte**

Ziele:

Teilaspekt Management, insb. Teamorganisationsformen

Teilaspekt Kosten-Schätzung

PO allgemein → Spezialvorlesung

Allgemeines

- eingebettet in Projektorganisation:
 - Projektplanung
 - Projektdurchführung
 - Projektüberwachung
 - Projektadministration (Betr.-wirt., Personalw.)

} Projektmanagement

andere Einteilung (s. o.) Prozesse, Produkte, Ressourcen

s. a. Folie (s. o.) verschiedenen Bedeutungen von Management

- Projektorganisation wird erleichtert durch das bisherige
 - a) Untergliederung des Herstellungsprozesses in Phasen/Tätigkeiten und Dokumenteerstellung
 - Phasenmodelle, Modell f. Dokumentation, Qualitätssicherung (insb. Integr.), Projektplanung, Erstellung Anforderungsdef., Entwurf., etc.

b) Arbeitsteilung

- c) Regelung der Dokumentation, QS
 - Dokumentation
 - QS
 - techn. Tätigk. RE, PiG, PiK

} vgl. Kap. 1-6, 10 ff

Projektmanagement und Betriebsorganisation

Allgemeines

Projekt:

- zeitlich begrenzt
- genau definierte Aufgabenstellung
- Zusammensetzung / Arbeitsstil / Organisationsform nicht notwendigerweise an Unternehmenshierarchie gebunden

Matrix-Organisation

- Überlagerung von Projekt und Abteilungsstruktur
 - jedes Projekt hat Mitarbeiter aus i. a. versch. Abteilungen
 - jede Abteilung hat Mitarbeiter, die keinem, einem oder mehreren Projekten angehören

	Material- wirtschaft	Gehalts- stelle	Program- mierung	System- analyse
	Aquin		hegel	
Projekt Lohnabrechnung		Engels fichte	hegel isidor Jaspers	Ortega Platon
	Bloch cicero	fichte	Kant	
Projekt Lagerverwaltung	cicero		isidor Lenin Marx	Rousseau Schelling
	Descartes	Gramsci	Nietzsche	

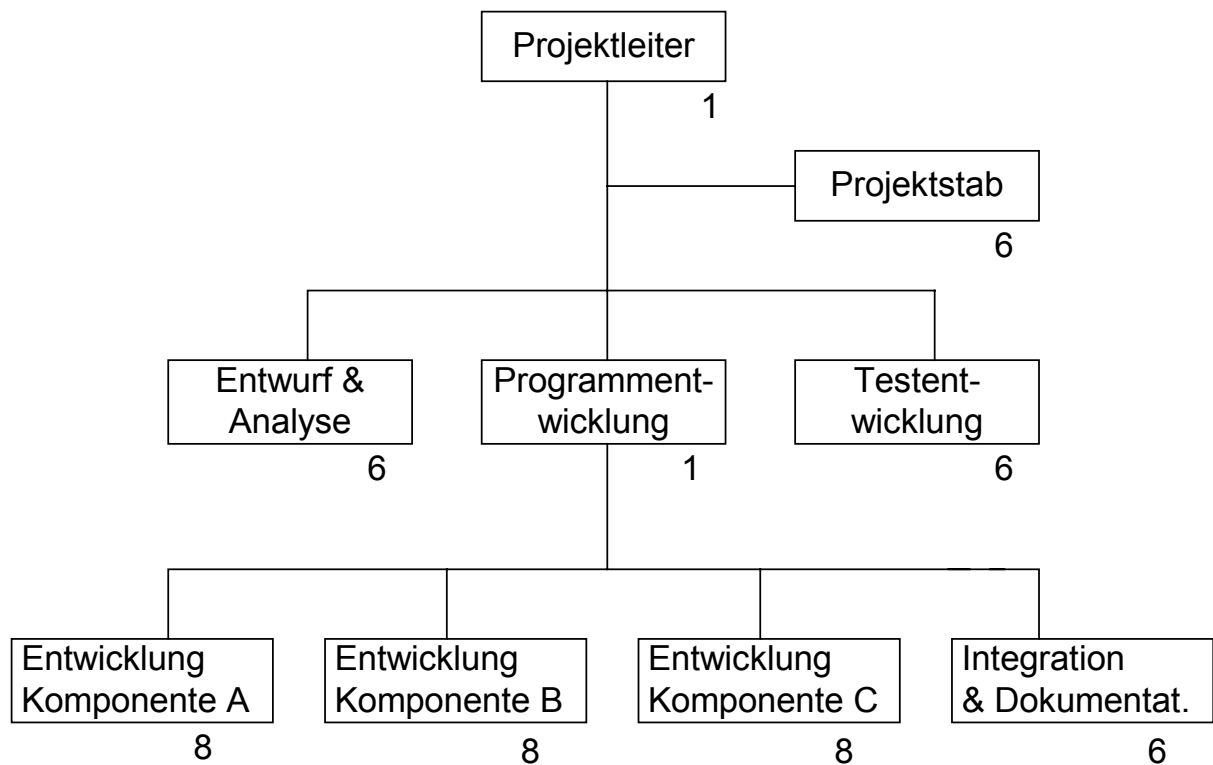
Matrix-Organisation (Groß-/Kleinschreibung bedeutet
Vollzeit-/Teilzeitarbeit)

- Charakterisierung
 - Projektmitarbeiter bleiben Abteilungen zugeordnet, bleiben in gewohnter Arbeitsumgebung
 - führen ggfs. noch andere Projekte oder andere Aufgaben durch
 - Projektleiter ist keiner Abteilung zugeordnet, hat keine Weisungsbefugnis gegenüber Projektmitarbeitern
- Probleme:
 - Konflikte Projektleiter- Abteilungsleiter
 - Mitarbeiter stets im Abteilungskontext: vermindert Engagement (wenig Gruppeneffekte)
 - Kommunikationsproblem (insb. falls Projektmitarbeiter räumlich in ihren Abteilungen verbleiben)
 - kaum Kontrolle des Projektfortschritts
 - Abteilungen sind dann wenig projektstrukturiert wegen der Flexibilität des Matrixmodells: "Arbeitsumgebung" für Mitarbeiter wechselt von Projekt zu Projekt
- behandeln hier nur Softwareabteilung (im eigenen Haus, in Softwarehaus)
 - Projekt eine betriebswirtschaftliche Einheit: Etat, wirtsch. Erfolg/Mißerfolg
Projektleiter: Etatplanung, Etatüberw., Personalverantwortung
 - betriebsw. Einheiten auch innerhalb des Softwareprojekts: jedes Teilprojekt hat Konto, finanziellen Spielraum (bei einigen Planungsmodellen kommt die Ermittlung des Kostenrahmens ganz oder teilweise aus der jew. Software-Fachabteilung)

Klassische Projektmodelle

hierarchische, pyramidenförmige Projektorganisation

Beispiel mit 50 Mitarbeitern nach /Endres 75/:




Organisation eines Programmierprojektes [Endres75, S. 5-6]

- Aufgaben:

Projektleiter:

- Verbindung zu Auftraggeber
- Verbindung im eigenen Unternehmen
- Arbeitsteilung im Großen
- empfängt Fortschrittsberichte, ergreift Maßnahmen bei Verzögerungen, neu auftauchenden Problemen
- hat Personalführung: Einst., Bewertung, Karriereförd.

Projektstab:

- Termin- und Kostenkontrolle
 - Organisation von Schulungen
 - Anfertigen von Richtlinien
- 
- Querschnittsaufgaben

Entwurf u. Analyse:

- für Entwicklung und ges. Dokumentation zuständig
- kontrolliert Änderungen an Anf. def., Ent. spez.
- analysiert Leistung der hergestellten Software

Programmentwicklung:

- Implementierung der einzelnen Module
- Funktions-, Leistungsüberprüfung, Integration
- Dokumentation auf techn. Ebene

Testentwicklung:

- bereitet Integrationstest und Installationstest vor

- Probleme:
 - Funktionstest (Testentwicklung) und Leistungstest (Entwurf und Analyse) getrennt
 - Projektleiter ist durch unter-, neben- oder übergeordn. Leiter zu unterstützen, der sich um den wirtschaftlichen Teil des Projekts kümmert
 - mehrstufige Hierarchie verhindert wirksame Projektfortschrittskontrolle: ist auf Aussagen anderer angewiesen (Aussagen über mehrere Stufen)
 - ist zu weit weg von Einzeltätigkeiten: wie soll er dabei Arbeit verteilen, diese koordinieren
 - finanzielle und soziale Lage aus der Höhe der Stufe in der Hierarchie: sobald Erfahrung vorliegt, ist Tätigkeit nur noch Management (Peter-Prinzip)

Chief Programmer Team

- Charakterisierung:

- Verzicht auf reine Manager
 - Spezialisierung der Tätigkeiten
- } → hohe Arbeitsproduktivität
Modell für kleine Teams
≤ 10 Mitarbeiter
- Projektleiter ist nicht nur verantwortlich, sondern auch produktiv tätig: alle anderen helfen ihm
 - Kern des Teams:
 - Chef-Programmierer: entwirft Gesamtsystem, implementiert wesentliche Teile, kontrolliert andere Implementierungen
 - Projektassistent: assistiert bei Entwurf, ist Ersatzmann für Chef-Programmierer, plant z. B. Systemtests
 - Projektsekretär: entlastet Team von bürokratischer Arbeit: verwaltet Projektergebnisse, gibt Auskunft über Stand des Projekts
 - Erweiterung der Mannschaft (vorgenommen von Chef-Programmierer):
 - Spezialisten für Implementierungssprache
 - Spezialisten für Basismaschine
 - Programmierer für Implementierung von Modulen mit vorgegebener Spezifikation
- } etc.

- Probleme des CPT:
 - Projektsekretär besser Projektverwalter,
durch Projektdatenbank und zug. Werkzeuge zu entlasten
 - sehr hohe Anforderungen an Chef-Programmierer: in technischer, organisatorischer und menschlicher Hinsicht
 - Teammitglieder unterscheiden sich weniger durch Ausbildung als durch ihre Erfahrung → Kritik an Entscheidungen des Chef-Programmierers
 - Beschränkung auf ca. 10 Teammitglieder:
größere Projekte mit hierarchischem Chief-Programmer-Team-Modell:
Mitglieder eines Teams sind Chef-Programmierer auf der nächsten Ebene
⇒ Vorteil der direkten Kommunikation geht bei Hierarchisierung verloren!

Generelle Probleme von Team-Organisationsformen

- bisher organisierte Festlegung von Kompetenzen
- bisher organisierte Festlegung von Kommunikation
- informelle Kommunikation unvermeidlich: darf aber die organisierte Kommunikation nicht ersetzen
- Team ist soziales Gebilde
- Zufriedenheit beeinflusst Projekterfolg:
 - allgemein: Urlaub, Arbeitszeit, Lohn, Aufgabenstellung des Gesamtprojekts
 - speziell für das Projekt:
 - Jeder hat gew. Überblick über Projekt
 - Zufriedenheit mit der Aufgabenverteilung:
Interesse an Teilaufgaben, gerechte Verteilung der undankbaren Aufgaben
 - Führungsstil und Kompetenz des Leiters
 - Verhältnis zu Arbeitskollegen:
kollegial und solidarisch
sachliche Auseinandersetzungen bei Überprüfung von Softwaredokumenten, Festlegung von Konventionen
 - Organisationsform des Teams:
solidarisches Handeln versus Ehrgeiz und Aufstieg

Projektplanung und -schätzung

Übersicht

- Aufgaben (zeitliche Einordnung: RE):
 - Gesamtkosten
 - Zeitplan
 - Personaleinsatz
 - Ermittlung der sonstigen Ressourcen u. Kosten

Je gründlicher, desto mehr Entwurf oder Realisierung

- Zusammenhang Kosten und Umfang

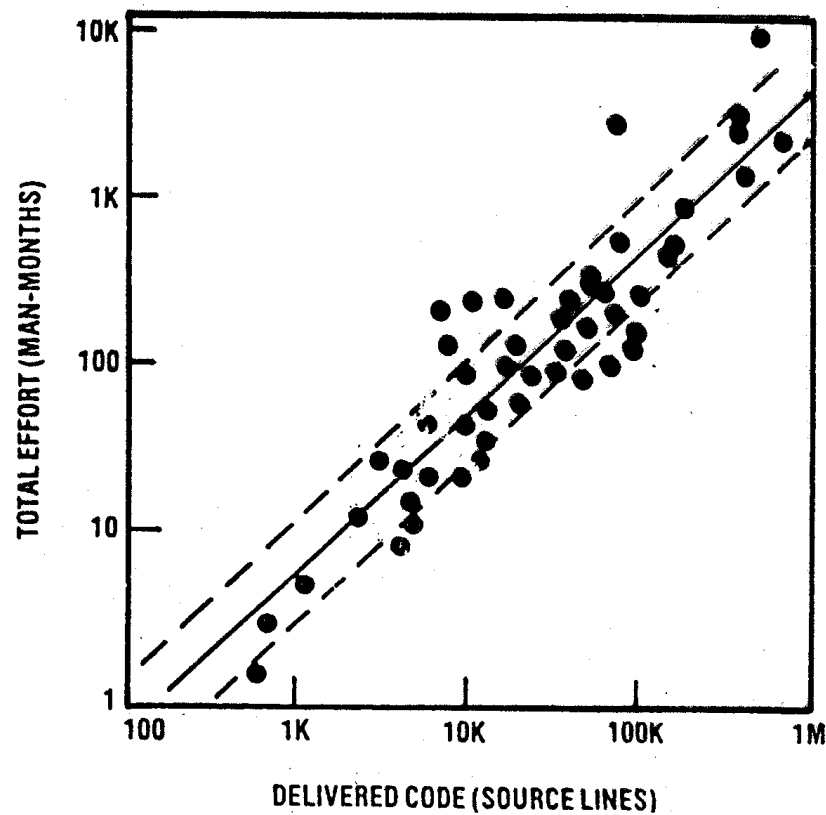


Figure 2. Empirical relationship between size and total effort.

Einschub Kosten

- eigene Vermarktung:
$$\text{Gewinn/Verlust} = \text{Preis} * \text{Menge} -$$
$$(\text{Entwicklungs-} + \text{Wartungs-} + \text{Vertriebs-} + \text{Einführungsk.})$$
- Projekte im Auftrag:
$$G = \text{Preis} - \text{Ermittlung der Kosten für Entwicklung/Wartung}$$
- bei Käufer:
Ermittlung der Wirtschaftlichkeit des Produkteinsatzes
- Kosten:
 - Personalkosten (wesentlich)
 - Sachmittel
 - Lizenzen
 - sonstige Kosten:...
- Kosten aus Umfang u. best. Faktoren

Schätzstrategien, Allgemeines

- top-down-Strategie, besser Analogie-Schätzstrategie, Black-Box-Strategie:
 - Orientierung an bekannten Kosten ähnl. Projekte oder Projektteile
 - neue Schätzung: alte Daten plus (begr. Annahmen, Intuition)

Probleme:

- bei neuer Aufgabenstellung (keine Erfahrung)
 - Übersehen spezieller Schwierigkeiten
 - große Projekte
 - Erfahrung des Schätzers
 - Projektdatenbank erforderlich
- bottom-up-Strategie (White-Box-Strategie):
 - Aufgabe in Teile zerlegt bis auf schätzbar. Einheiten
 - Schätzung der Kosten durch diejenigen, die Einheiten realisieren

Probleme:

- Entwurfsphase müßte beendet sein: Kosten hoch
- Gesamtkostenermittlung aus Einzelteilen?
(zur Kontrolle manchmal top-down-Schätzung)
- Wie kommt man zur fundierter Schätzung der Teile?

- Ähnlichkeits- und Differenzenschätzung (Mischverf.)
 - Zergliederung, bis Ähnlichkeiten oder Differenzen zu bekannten Projekten festgest. werden können
 - nichtvergleichbare Einheiten: andere Schätzung
- Verhältnis- oder Faktorenschätzung (ratio estimating), algorithmische Schätzung
 - basiert auf relevanten Koeffizienten, die ungefähr projektinvariant sind
 - Modul (Typ, Anz. Befehle, rel. Komplexität)
 - Gesamtsystem/Teilsysteme (Größe, Typ, Dateihandhabung, verw. Software etc.)
 - Daten vergangener Projekte in Kostenmatrix

Problem:

- validierte Kostenbasis für viele Schätzsituationen

Umfangsschätzung

Probleme:

- Maß für die Schätzung?
 - üblich: Code-Zeilen (LOC) oder Anzahl Instruktionen ist vage:
 - Niveau der Programmiersprache (Ada oder Masch.-Code)
 - Konstrukte auf mehrere Zeilen verteilt
mehrere Konstrukte in einer Zeile
 - mit oder ohne Kommentare
 - auch Zwischenversionen
 - mit JCL-Anweisungen (JCL=Job-Control, Großrechner)
 - wie zählt benutzte Software
 - wie zählen instantiierte, generische Teile, generierte Teile
 - nur Quellcode oder Länge aller Dokumente
 - mit oder ohne Wartung?
- Subjektivität des Schätzers
 - Ehrgeiz
 - Pessimismus
 - Erfahrung

Gewichtung der Quantität/ Berücksichtigung des Typs von Software

- Art des Systems und Produktivität

Art des Programms	Instruktionen/Pers.-Jahr
Kontrollprogramm	500 - 700
Compiler	1000 - 1700
Anwendungsprogramm	2000 - 4000

[Endres75, S. 3-23f]

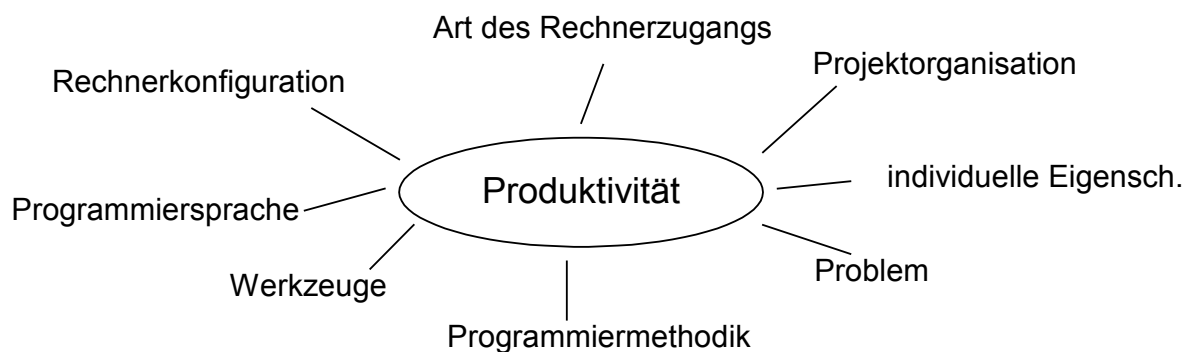
- Produktivität und Erfahrung

<div style="text-align: center;">Projektdauer Schwierigkeit</div>	Instruktionen/Pers. -monat		
	6-12 Mon.	12-24 Mon.	> 24 Mon.
wenig Wechselwirkung	400	500	800
einige Wechselwirkung	200	250	400
starke Wechselwirkung	100	125	125

[Endres75, S. 3-24]

Produktivität des Softwareentwicklers

- Durchschnittszahlen: s. o.
- Maß = Codelänge / Personenmonate
Codelänge Probleme: s. o.
Personenmonat Probleme: Gesamtprojekt oder Teil des Projekts; Krankheit/Urlaub/Wechsel/Einarbeitung
⇒ Gefahr: unsolide Software
- Einflüsse auf Produktivität



- Leistungsanstieg bei Wiederholung

Compiler	Pers.-Monate
1	72
2	36
3	14

Leistungsanstieg bei Wiederholung [McClure68]

- Leistungsunterschiede von Systementwicklern

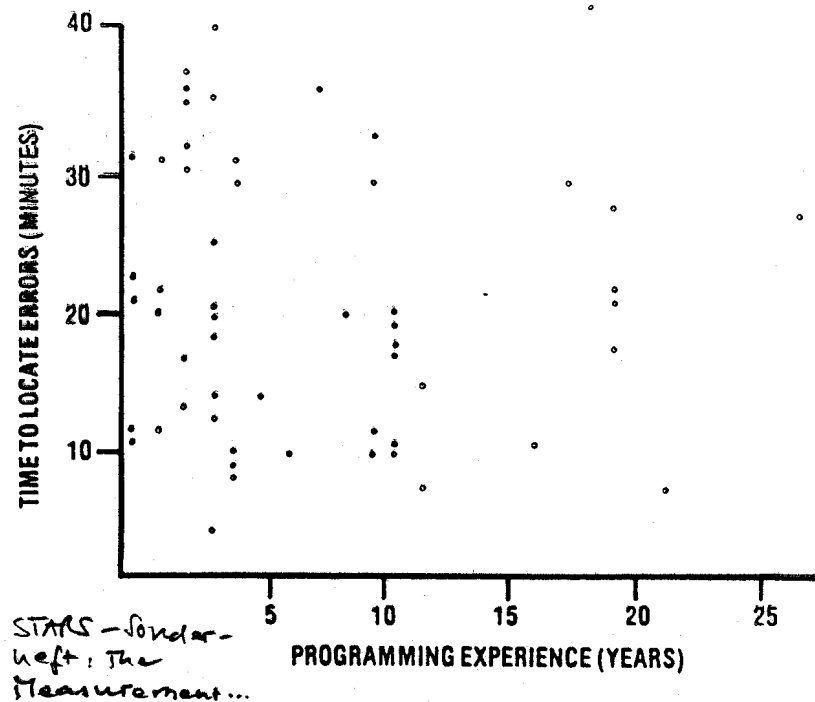


Figure 4. Scatterplot of years of experience versus debugging time.

Leistung bei	schlechtester/bester
Testzeit	26 / 1
Rechenzeit	11 / 1
Codierzeit	25 / 1
Codelänge	5 / 1
Laufzeit	13 / 1

Leistungsunterschiede bei Programmieren [David68]

- private / persönliche Situation
- soziales Umfeld in Entwicklermannschaft
- Zufriedenheit mit Aufgabe

Ein Ansatz zur Kostenschätzung: COCOMO /Boehm 81/

Übersicht

basic intermediate detailed model

enthält Reihe von Konstanten, die an spezielle Situationen anpaßbar sind

- einfaches Modell:
 - Ziel: Größenordnungsangabe d. Kosten
 - Unterscheidung von 3 Projektklassen
 - organic mode projects:
kleine Projekte, vertraute Entw.-"Umgebung" vertraut mit Anwendung, Kommunikationsaufw. klein, bald zur Sache
 - semi detached mode projects:
Zwischenform, erfahrene u. unerfahrene Projektmitglieder, begrenzte Erfahrung mit Problem, keine Kenntnis u. Erfahrung in best. Teilen des Projekts
 - embedded mode projects:
Projekt hat scharfe Einschränkungen, Hardware, Software, Regeln, Organisationsstruktur, Anforderungsspezifikation zur Erzielung einer "runden" Problemlösung nicht möglich, Überprüfungskosten sind hoch, wegen Schwierigkeiten der Aufgabe liegt selten große Erfahrung bzgl. der gesamten Aufgabenstellung vor

- Aufwand in p. m.:

OM: $PM = 2.4 (KDSI)^{1.05}$

SDM: $PM = 3.0 (KDSI)^{1.12}$

EM: $PM = 3.6 (KDSI)^{1.2}$

PM: Anzahl p. m. à 152 Std. (Urlaub, Training, Krankheit etc.)

KDSI: Anzahl abgeg. Quelltextzeilen in Tausend
Quelltextzeilen wörtlich, ohne Kommentar, ohne unterstützende Software

- Entwicklungszeit in m:

OM: $TDEV = 2.5 (PM)^{0.38}$

SDM: $TDEV = 2.5 (PM)^{0.35}$

EM: $TDEV = 2,5 (PM)^{0.32}$

Annahme: genügend Entwicklungskapazität

Zeit kann durch bel. Aufwand nicht bel. reduziert werden

- Beispiel:

OM-Projekt mit 32 KDSI (= 640 Seiten Quelltext)

$PM = 2.4(32)^{1.05} = 91$ p. m.

$TDEV = 2.5(91)^{0.38} = 14$ m

Anzahl der Personen bei min. Bearbeitungszeit:

$PM/TDEV=91/14=6.5$ Pers.

- Annahmen des Grundmodells

- 1) eingebaute Produktivitätsannahme aus Projekterfahrungen abgeleitet (abh. von Größe, hier 10 KDSI),
z. B. für OM

$$\text{KDSI} \approx 370 \text{ DSI} / \text{p. m.}$$

$$\approx 18 \text{ DSI} / \text{p. d.}$$

für EM

$$\text{KDSI} \approx 8 \text{ DSI} / \text{p. d.}$$

} Produktivität fällt mit zunehmender Größe des Systems, Spreizung OM und EM nimmt zu

- 2) Zeit ist minimale Zeit, d. h. Dreisatz gilt nicht
insb. kann verzögertes Projekt durch zus. Aufwand nicht unbedingt wieder in den Zeitplan gebracht werden
- 3) COCOMO sagt nichts darüber aus, wenn man bzgl. Personalmittel eingeschränkt ist, aber Zeit hat
z. B. 14 m für 6.5 Personen \cong 30 m für 3 Personen?

Hier nur Größe und Typ.

Gibt es weitere Faktoren?

Das mittlere (verfeinerte) Kostenmodell

- weitere Faktoren

Zuverlässigkeit

Größe der Datenbasis des Programms

Laufzeit- / Speicherplatzvorgabe

Eigenschaften der beteiligten Personen

Software-Werkzeuge

insg. 15 Faktoren

in 6 Abstufungen

VL	L	N	H	VH	EH
very low	low	nominal	high	very high	extr. high

- Produktfaktoren:

- RELY erforderliche Software-Zuverlässigkeit

VL Softwarefehler erzeugen "leichte Unbequemlichkeiten"

N mittelschwerer Verlust, der repariert werden kann

VH Risiko für menschl. Existenz/Gesundheit

- DATA Größe der Datenbasis

L Datenbasis kleiner DSIs*10

N Faktor zwischen 10 und 100

VH Faktor größer gleich 100

- CPLX Produktkomplexität
 - L einfache EA, einf. Datenstr., "straightforward" Programm
 - N einige EA, auf mehreren Dateien, Bibliotheks-module, Datenaust. zwischen Modulen
 - VH, EH reentrant, rekursiver Code, nebenl. Problem, komplexe Datenstrukturen
- Rechnerattribute
 - TIME Laufzeiteinschränkungen
 - N ca. 50% der CPU-Zeit genutzt
 - ...
 - EH ca. 95% der CPU-Zeit genutzt
 - STOR Speicherplatzeinschränkung
 - N ca. 50% des Hauptspeichers genutzt
 - ...
 - EH ca. 95% des Hauptspeichers genutzt
 - VIRT Veränderbarkeit der Basismaschine (Hard- u. Software)
 - L selten geändert (einmal(!) pro Jahr)
 - N alle 6 Monate
 - VH alle 2 Wochen
 - TURN turnaround time
 - L interaktive Systementwicklung
 - VH mehr als 12 Stunden Wartezeit

- Persönliche Faktoren

- | | | | | |
|--------|--|---|----|----------------------------|
| - ACAP | Fähigk. in Problemanalyse u. Entwurf | } | VL | keine, geringe Erfahrungen |
| - AEXP | Anwendungserfahrung | | N | einjährige Erfahrungen |
| - VEXP | Vertrautheit mit der Basismaschine | | VH | mehr als drei Jahr Erf. |
| - PCAP | Programmierkenntnisse | | | |
| - LEXP | Erf. in zugrundel. Programmiersprache(n) | | | |

- Projektattribute

- MODP Erf. in modernen Programmierpraktiken, Progr.-Methodiken (top-down-Entw., Code Review, strukt. Progr., Progr. aus Fertigteilen etc.)
 - VL keine Benutzung bisher
 - N teilweise Benutzung
 - VH Benutzung ist Standard, es liegt erh. Erfahrung vor
- TOOL Entwicklungswerkzeuge
 - VL nur Assembler
 - N übliches Programmiersystem
 - VH Entwicklungsumg. (à la UNIX ?)
- SCED geforderter Zeitplan in Bezug auf minimale Entwicklungszeit s. o.
 - VL engerer Zeitplan
 - VH größere Zeitspanne

- Gewichtungsfaktoren für Attribute

Table 11.1 Project attribute multipliers

<i>Cost driver</i>	<i>Ratings</i>					
	<i>VL</i>	<i>L</i>	<i>N</i>	<i>H</i>	<i>VH</i>	<i>EH</i>
RELY	0.75	0.88	1.00	1.15	1.40	—
DATA	—	0.94	1.00	1.08	1.16	—
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME	—	—	1.00	1.11	1.30	1.66
STOR	—	—	1.00	1.06	1.21	1.56
VIRT	—	0.87	1.00	1.15	1.30	—
TURN	—	0.87	1.00	1.07	1.15	—
ACAP	1.46	1.19	1.00	0.86	0.71	—
AEXP	1.29	1.13	1.00	0.91	0.82	—
PCAP	1.42	1.17	1.00	0.86	0.70	—
VEXP	1.21	1.10	1.00	0.90	—	—
LEXP	1.14	1.07	1.00	0.95	—	—
MODP	1.24	1.10	1.00	0.91	0.82	—
TOOL	1.24	1.10	1.00	0.91	0.83	—
SCED	1.23	1.08	1.00	1.04	1.10	—

Tables 11.1 and 11.2 are reproduced from *Software Engineering Economics*, B.W. Boehm (1981), by permission of Prentice-Hall Inc.

Berechnung des Aufwands

$A = \text{Aufwand in p. m. (nach Grundmodell)} * \prod F_i$

- Beispiel:

eingebettetes Softwaresystem für Mikrocomputer (16Bit) in
64K Speicher

einfaches Modell liefere 45 p. m.

RELY = H: 1.15

STOR = VH: 1.21

TIME = H: 1.11

TOOL = L: 1.10

→ $45 \text{ p. m.} * 1.15 * 1.21 * 1.11 * 1.10 = 76 \text{ p. m.}$

Gesamtkosten bei 6000\$ monatl. Kosten f.

Softwareentwickler

$K = 76 * 6000\$ = 456\ 000\$$

- Abschätzung von Alternativen:

z. B. Investition in ein UNIX-System

VEXP wird von 1.0 auf 1.10 erhöht (Einarbeitungsaufw.)

TOOL wird von 1.1 auf 0.91 erniedrigt

TURN wird von 1.0 auf 0.87 erniedrigt

TIME, STOR werden auf 1.0 erniedrigt

→ $K = 45 * 1.15 * 0.91 * 1.1 * 0.87 * 6000\$ = 270\ 000\$$

d. h. mit Investition billiger als vorher

Probleme/Vorteile

- Probleme

- basiert auf Produktgröße in KDSI: schwer vorherzusagen
 - Faktoren schwer einzustufen
 - Dokumentationsanforderungen
 - "Kontinuität" des Personals
 - "Güte" der Softwarearchitektur
- } gehen nicht ein

→ Redundanz: Schätzung zusätzl. nach anderer Vorgehensweise
z. B. bottom-up-Schätzung: Aussage der Entwickler

- Vorteile

- kann angepaßt werden an spez. organisatorische Umgebung: Kalibrierung
- neue Faktoren können hinzugenommen werden
- Faktoren können verschwinden, da z. B. in Kontext konstant:
 - z. B. Programmiersprache
 - z. B. Softwareentwicklungswerkzeuge
 - z. B. Schulung in bes. "Methoden" stets erfüllt
- in gew. Grenzen können Alternativen diskutiert und Entscheidungen begründet werden
- indirekte Faktoren

- insgesamt bei Kostenschätzung: Größenordnungsangabe ist besser als gar nichts

Abschätzung des Wartungsaufwands

Abschätzung vorab: z. B. 1.3-mal Erstellungsaufwand in einer nicht festgelegten Zeit

im folgenden Abschätzung nach Erstellung: gehört eigentlich nicht hierher

Grundmodell

$$\text{AME} = 1.0 * \text{ACT} * \text{SDT}$$

Annual Maintenance Effort

Annual Change Traffic
Anteil des Quelltexts des Produkts, der pro Jahr durch Wartung anfällt

Entwicklungsaufw.
Bei Grundmodell oder verf. Modell

Beispiel Entwicklungsaufwand 236 p.m.

$$\text{ACT} = 15 \%$$

$$\text{AME} = 1.0 * 0.15 * 236 \text{ p.m.} = 35.4 \text{ p.m.}$$

Wartungskosten: verfeinertes Modell

- Faktoren und Abstufungen wie bei Entwicklungsaufwands-schätzung bis auf

SCED Entwicklungszeitplan irreversibel, deshalb 1.0

MODP noch bedeutsamer für die Wartung

deshalb andere Faktorenwerte

ferner Faktorenwerte unterschiedlich je nach Produktgröße

Table 11.2 MODP attribute values

<i>Product size</i> <i>KDSI</i>	<i>MODP rating</i>				
	<i>VL</i>	<i>L</i>	<i>N</i>	<i>H</i>	<i>VH</i>
2	1.25	1.12	1.00	0.90	0.81
8	1.30	1.14	1.00	0.88	0.77
32	1.35	1.16	1.00	0.86	0.74
128	1.40	1.18	1.00	0.85	0.72
512	1.45	1.20	1.00	0.84	0.70

- RELY modifiziert, auch höher, wenn Zuverlässigkeitsanf. f. Entw. niedrig war

Maintenance effort multipliers

	VL	L	N	H	VH
RELY	1.35	1.15	1	0.98	1.10

- Beispiele verfeinertes Modell:

RELY VH

AEXP H

LEXP H

MODP VH

$$\text{AME} = 35.4 * 1.1 * 0.91 * 0.95 * 0.72 = 24.2 \text{ p.m.}$$

RELY VH

AEXP H

LEXP H

MODP VL

$$\text{AME} = 35.4 * 1.1 * 0.91 * 0.95 * 1.4 = 47.1 \text{ p.m.}$$

- Probleme Wartungskostenschätzung

- Zerlegung des Systems nicht berücksichtigt
Änderungen in verschiedenen Teilen kann sehr verschieden sein
- es wird nicht berücksichtigt, daß Systemstruktur durch Wartung zerstört wird, d. h. Wartungsaufwand müßte progressiv steigen
- wenn Qualität der Entwickler von der der Pfleger abweicht:
keine vernünftigen Aussagen

Zusammenfassung: Faktoren für Kostenschätzungen

- Folgende Faktoren und Attributgruppen werden häufig in Kostenmodellen verwendet:

Folgende Faktoren- und Attributgruppen werden häufig in Kostenmodellen verwendet:

- (1) Systemumfang
 - Anzahl geschätzter LOCs im fertigen System (Maschinenbefehle oder Assemblerbefehle oder Quellprogramm-Anweisungen in höheren Sprachen).
- (2) Systemkomplexität
 - Schwierigkeitsgrad des Systems (leicht, mittel, schwer).
 - Komplexität der Schnittstellen.
 - Innovationsgrad des Systems (Vertrautheit mit dem Problem, der Hardware, der Software).
 - Hardware-Software-Schnittstellen.
 - Anzahl benötigter Dateien und Anwendungsprogramme.
 - Programmstruktur.
 - Personalaufwand pro Phase.
 - Zeitaufwand pro Phase.
- (3) Programmtyp
 - Anwendungstyp (kaufmännisch/nicht-kaufmännisch).
 - Programm-Kategorie.
 - Echtzeit/nicht Echtzeit.
- (4) Dokumentation
 - Umfang der Dokumentation in Seiten.
 - Anzahl unterschiedlicher Dokumentarten.
- (5) Environment
 - Entwicklungs-Environment.
 - Neuer oder alter Computer.
 - Anzahl der Terminal-Arbeitsplätze.
 - Benutzte Programmiersprache.
 - Speicherbeschränkungen.
 - Geschwindigkeit und Speicherkapazität des Computers.
 - Dialog- oder Stapelverarbeitung.
 - Vertrautheit der Mitarbeiter mit der Sprache, dem Compiler usw.
 - Programmiererfahrung der Mitarbeiter in Jahren.
 - Kontinuität des Personals.
 - Anzahl der räumlich verteilten Entwicklungsorte.
 - Produktivität der Mitarbeiter (LOC/Zeiteinheit).

- Vergleich von 12 Kostenmodellen mit 36 000 LOC u. weit. Annahmen nach /Mohanty 81/

$$\frac{\text{niedrigste Kosten}}{\text{höchste Kosten}} = \frac{1}{7.6}$$

Gründe:
s. Faktorenliste

$$\frac{\text{niedrigste Entwicklungsz.}}{\text{höchste Entwicklungsz}} = \frac{1}{1.9}$$

Zusammenfassung

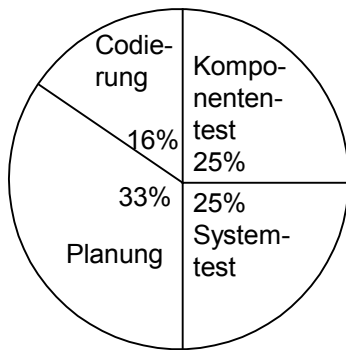
Zusammenfassend lassen sich folgende Feststellungen treffen:

- Es gibt heute ungefähr 20 verschiedene Kostenmodelle, um Kosten, Zeit und Personal eines Software-Projektes zu schätzen (siehe auch /Monanty 81/).
- Jedes Modell benutzt unterschiedliche Faktoren zur Berechnung.
- Kein Modell schätzt die Kosten firmenunabhängig mit ausreichender Genauigkeit.
- Die Kalkulationsergebnisse variieren vor allem wegen der unterschiedlichen Entwicklungsumgebungen und der nicht ausreichenden quantitativen Berücksichtigung von Qualitäts-Merkmalen.
- Die Mitarbeiterproduktivität ist nicht konstant, sondern schwankt statistisch in Abhängigkeit von einer Vielzahl von Faktoren.
- Obwohl die Kalkulationsergebnisse noch ungenau sind, ist eine unsichere Schätzung immer noch besser als gar keine.
- Jede Kalkulation sollte zu einer optimistischen, einer wahrscheinlichen und einer pessimistischen Schätzung führen, damit die Risikobreite deutlich sichtbar wird.

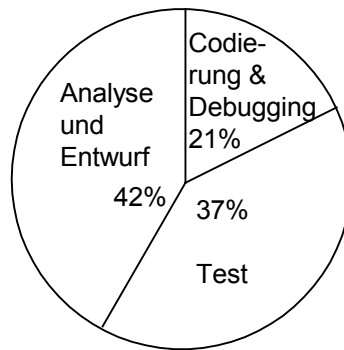
Zeitplan- und Personaleinteilung

Allgemeines/Erfahrungen

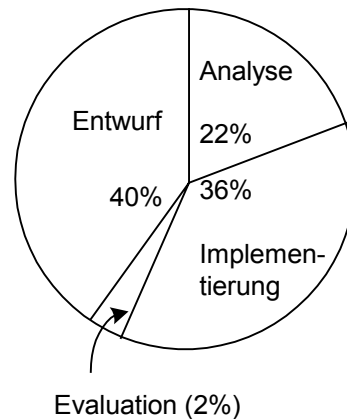
- Größe der Software, Schwierigkeit, Personalbedarf in p. m. stehen nun fest
noch offen: Planung des Personaleinsatzes
Aufteilung auf Phasen
- firmeninterne Erfahrungen



[Brooks75, S. 20]



[Wolverton74]



[Mobil]

Table 1.1 Relative costs of software systems

System type	Phase costs(%)		
	Requirements/design	Implementation	Testing
Command/control systems	46	20	34
Spaceborne systems	34	20	46
Operating systems	33	17	50
Scientific systems	44	26	30
Business systems	44	28	28

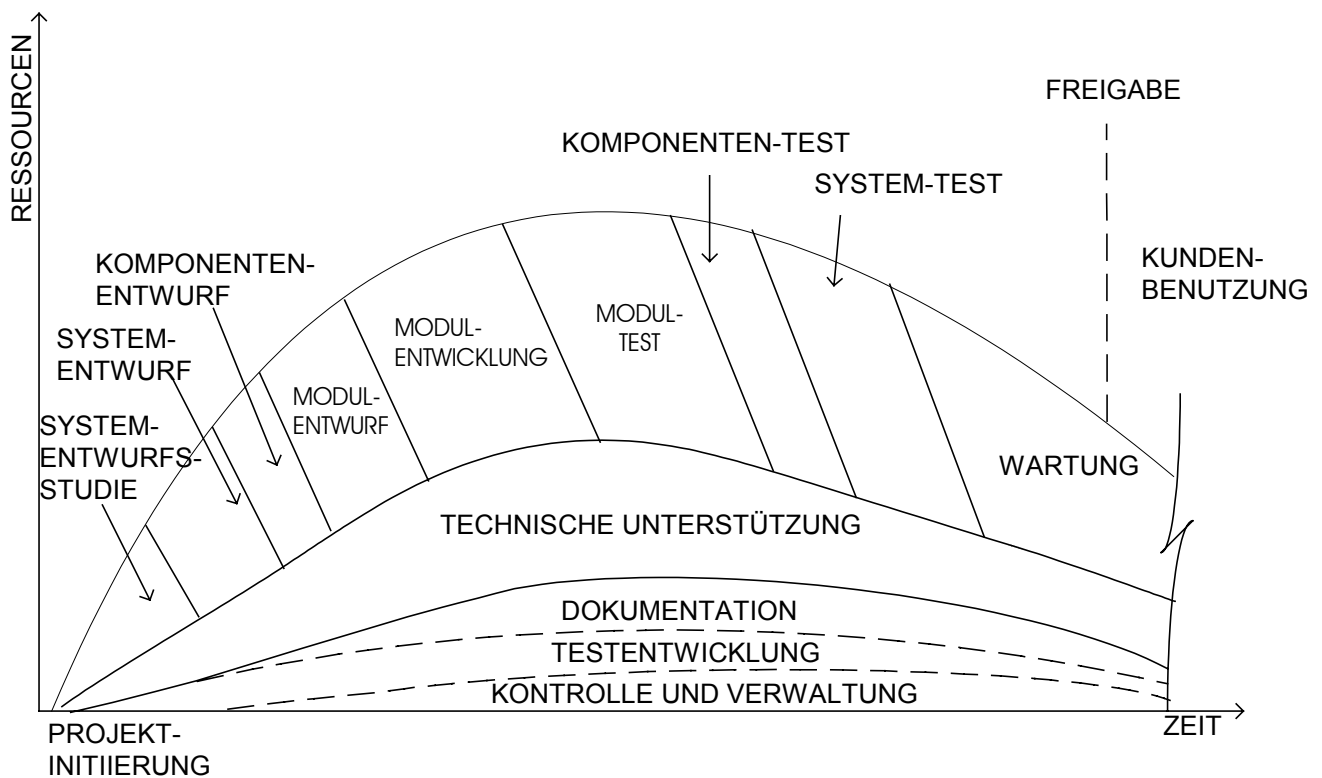
- Meilensteine festlegen und insb. Zeitpunkt der Fertigstellung aus Softwarelebenszyklus
oder feiner: z. B. Testumgebung für Modul XYZ fertig
Modul XYZ getestet
- Ziel Meilensteinplan:
 - Erkennung von Verzögerungen
 - Ergreifung von Maßnahmen bei Verzögerungen
 - Unterbrechung/Beendigung des Projekts an vordef. Punkten
- Darstellung durch Balkendiagramme, Netzpläne

Personaleinteilungsabschätzung

- Faktoren

- Qualifikation d. Programmierer (ggf. Schulung einplanen)
- "konstante" Größe der gesamten Firma, Abteilung etc.
- Mitarbeit an mehreren Projekten
- Überlappung von Tätigkeiten

- Zeitverlauf Ressourcenverbrauch



Die Systementwicklung. Phasen und Funktionen [Nash68]

Weitere Kosten

Weitere Personalkosten

nicht nur Kosten für Programmierer

- sondern anderes Personal

nach /Endres 75/ kommen auf 100 "Programmierer":

20 technische Autoren

10 Manager

10 Sekret. / Verwaltung

8 Programmierer f. Systemverwaltung / Leistungsmessung

2 Instruktoren

6 Operateure

2 Datentypisten

2 Wartungsingenieure

+ Anteil allg. Personal

Post

Küche

Hausmeister

Technik

Schließdienst

etc.

Sachmittelkosten

- zusätzliche Sachmittel
für Rechnerbenutzung, Rechnerkauf,
für Mieten, Reisen, Büromaterial,
für Schulungsmaßnahmen,
umgelegte Allgemeinkosten (Telefon, Kopien, Schreibmaterial etc.)
- Rechnerkosten bei 4 Projekten

Dauer in Monaten	CPU-Std./Pers.-Monat	Terminalstd./Pers.-Monat
36	2.20	7.60
35	1.27	4.40
33	1.30	4.50
33	2.02	7.00

Rechenzeit und Terminalbenutzung [Wolverton74]

Literatur zur Kap. 9:

- /Ba 72/ I. T. Baber: Chief Programmer Team Management of production programming, IBM SJ 1, 56—73 (1972).
- /Bo 73/ B. W. Boehm: Software and its Impact, A quantitative Assessment, Datamation 19, 5, 48—59 (1973).
- /Bo 81/ B. W. Boehm: Software Eng. Economics, Englewood Cliffs: Prentice Hall (1981)
- /Br 75/ I. H. Brooks: The mythical Man-Month, Essays in Software Engineering Reading: Addison Wesley (1975).
- /En 75/ A. Endres: Planung und Durchführung von Programmierungsprojekten, Vorlesungsmanuskript Universität Stuttgart, 1975
- /Mc 68/ R. M. McClure: Projection versus Performance in Software Production, in: /Na 68b/, 73—75 u. a.
- /Da 68/ E. E. Jr. David: Some Thoughts about the Production of Large Software, in: /Na 68b/, 83 u. a.
- /Mo 81/ S. N. Mohanty: Software Cost Estimation: Present and Future, SP&E 11, 103—121 (1981).
- /Mobil/ Mobil System Development Methodology Reference Card, o. O., o. J.
- /Na 68a/ J. Nash, Some Problems in the Production of Large-Scale Software Systems, in /Na 68b/, 20 u. a.
- /Na 68b/ P. Naur, B. Randell (eds.): Software Engineering, Report on a Conference, Garmisch, (1968), Brüssel: NATO Scientific Affairs Division (1969)
- /Wo 74/ R. W. Wolverton: The cost of Developing Large-Scale Software, Transact. on Comp. C-23,6, 615—636 (1974).