Softwaresystemmaße am Beispiel von CoCoMo II

Dr. Bernhard Schätz
Fakultät für Informatik
TU München

Messung von Softwaresystemen

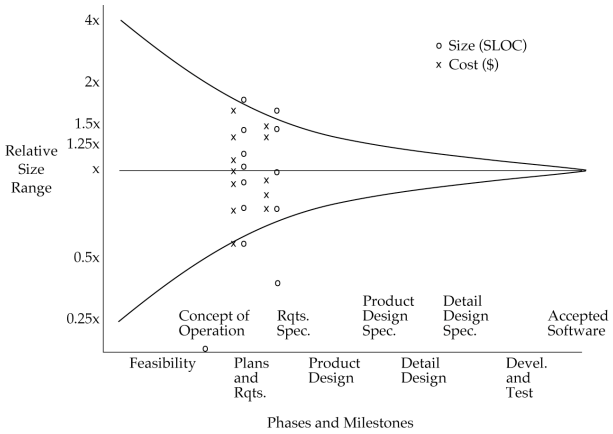
„Was man nicht misst,
das kann man nicht steuern.“

Tom de Marco

Schätzen von Software

- Ziel des Schätzen:
  - Bestimmung „Systemumfang"
    - Systemkomplexität
    - Realisierungsaufwand
    - Realisierungszeit
  - Möglichst vor Systemrealisierung
- Prinzipielle Strategie: per Analogie
- Schätzen:
  - Black-Box/White-Box
  - Mischformen
  - Modellbasiert

Güte der Schätzung

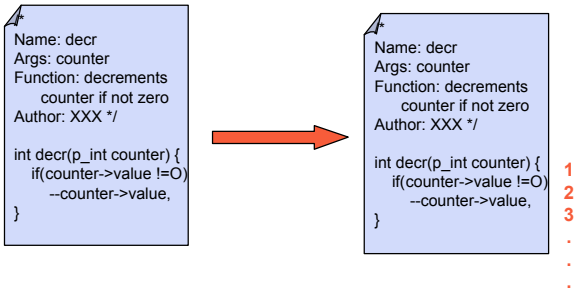


- Genauigkeit Schätzung ≤ Genauigkeit Produkt/Prozessbeschreibung

Übersicht

- Grundlagen der Softwareschätzung
  - Schätzung von Systemkomplexität
  - Komplexität, Aufwand, Laufzeit
  - Modellparameter: Projekt vs. Prozess

Softwarekomplexität



- Software = Programmcode
- Softwareumfang = Anzahl Programmzeilen

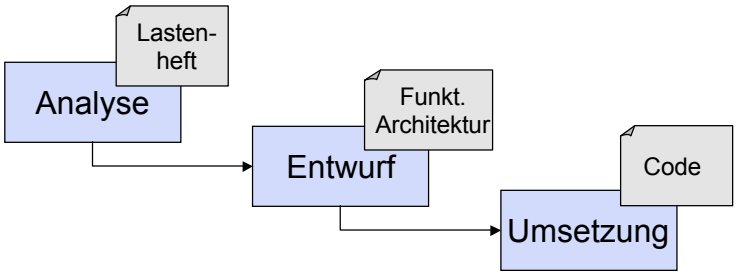
Code als Komplexitätsmaß

```
/*
Name: decr
Args: counter
Function: decrements
      counter if not zero
Author: XXX */

int decr(p_int counter) {
    if(counter->value !=O)
        --counter->value,
}
```

- Komplexität
  - Größe
  - Struktur
- Verbesserung: Standardisierte Code-Zeilen
  - Größe: Zählregeln
  - Struktur: Sprachebene (Assembler, Prozedural, Objektorientiert)

Komplexität und Prozess

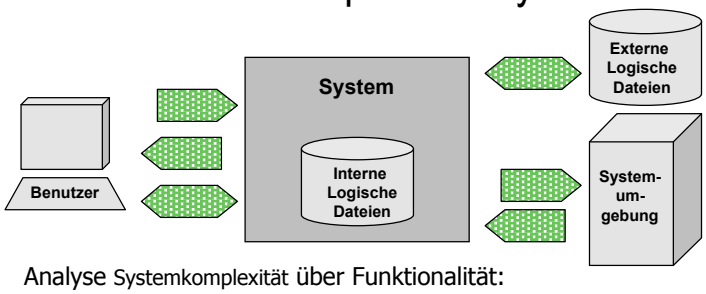


- Verschiedene Maße in verschiedenen Phasen
  - Implementierung: Codezeilen
  - Post-Architektur (Feindesign): Abstrakter Code
  - Prä-Architektur: Funktionspunkte

Funktionspunktanalyse

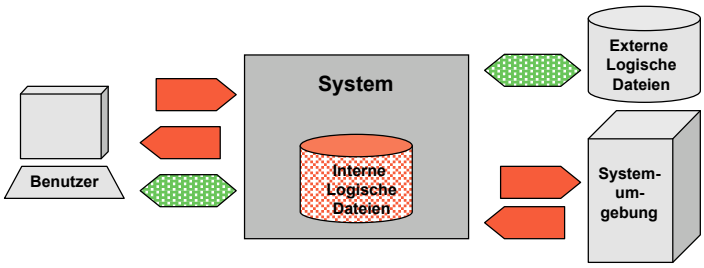
- Verfahren:
  - Analyse aus Sicht der Systemumgebung
  - Messung über funktionale Schnittstellen:
    - Externe Eingabe
    - Externe Ausgabe
    - Externe Anfrage
    - Interne Logische Daten
    - Externe Logische Daten

Funktionspunktanalyse



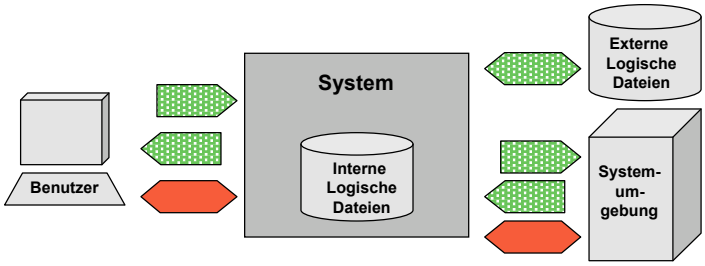
- Analyse Systemkomplexität über Funktionalität:
  - Analyse aus Sicht der Systemumgebung
  - Messung über funktionale Schnittstellen:
    - Externe Eingabe
    - Externe Ausgabe/Externe Anfrage
    - Interne Logische Daten/Externe Logische Daten
- Vorteile:
  - Auf hohem Niveau (Funktionspunkte)
  - Frühzeitig (Systemspezifikation)

FPA: Externe Ein-/Ausgabe



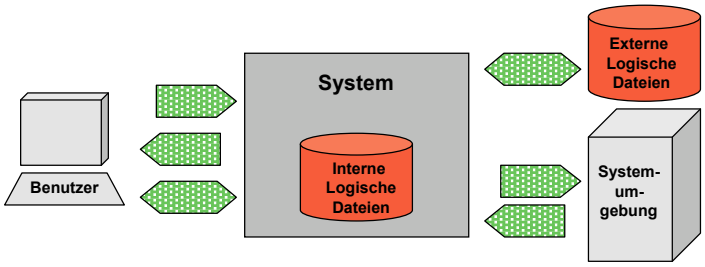
- Ein- bzw. Ausgabe von Umgebung an System
  - Logisch zusammenhängend
  - Benutzer- oder Kontrollein- bzw. Ausgabe
  - Eingabe: Verändert /erweitert internen Datenzustand

FPA: Externe Anfrage



- Ein-/Ausgabekombination
  - Logisch zusammenhängend
  - Von Benutzer oder Systemumgebung
  - Eingabe löst sofortige Reaktion aus

FPA: Interne Logische Dateien



- Gruppe von intern bzw. extern verarbeiteter Daten
  - Logisch Zusammenhängend
  - Benutzer- oder Kontrolldaten
  - Interne: Erzeugt, verwaltet oder bearbeitet
  - Externe: Von Außen zur Verfügung gestellt

Berechnung Funktionspunkte

For ILF and EIF				For EO and EQ				For EI			
Record Elements	Data Elements			File Types	Data Elements			File Types	Data Elements		
	1 - 19	20 - 50	51+		1 - 5	6 - 19	20+		1 - 4	5 - 15	16+
1	Low	Low	Avg	0 or 1	Low	Low	Avg	0 or 1	Low	Low	Avg
2 - 5	Low	Avg	High	2 - 3	Low	Avg	High	2 - 3	Low	Avg	High
6+	Avg	High	High	4+	Avg	High	High	3+	Avg	High	High

- Berechnung:
  - Ermittlung Liste aller Funktionsarten nach Typ
    - Beispiel: EI: Eingabe Kundendaten mit 5 Feldern
  - Bestimmung Komplexität pro Funktionsart
    - Beispiel: 1 ILF, 5 Elemente = LOW
  - Berechnung der Funktionspunkte pro Funktionsart

Berechnung Funktionspunkte (2)

Function Type	Complexity-Weight		
	Low	Average	High
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

- Berechnung:
  - Bestimmung Anzahl/Art Funktionspunkte:
    - Z.B. 3 EI Low + 2 EI Average = 3*3 + 2*4 = 17
  - Bestimmung Unangepasste Funktionspunkte:
    - Summe aller gewichteten Funktionstypen
  - Umrechnung in KLOC:
    - 1 FP = ca. 320 Ass.-LOC, 130 C-LOC, 90 Pascal-LOC

CoCoMo

- Verfahren: Constructive Cost Modeling
- Entwickler: Barry Boehm
- Anwendungsgebiete:
  - Informationssysteme (Honeywell, IBM,US Army)
  - Eingebettete System (AT&T, Boeing, Motorola)
- Ansatz:
  - Eingabe:
    - Abschätzung der Systemkomplexität
    - Abschätzung von Prozessparametern
  - Ausgabe:
    - Abschätzung des Realisierungsaufwands
    - Abschätzung der minimalen Realisierungszeit

Modellbasiertes Schätzen

- Schematisiertes Analogieverfahren
- Ansatz: Aus Projektdaten math. Modell
  - Definition geeigneter Modellparameter
    - Projekt (Größe, Komplexität, Randbedingungen)
    - Projektumfeld (Team, Infrastruktur)
  - Bestimmung von Modellgleichungen
    - Berechnung Projektaufwand aus Parameter
    - Berechnung Projektlaufzeit aus Parameter
  - Überprüfung Modell
  - Anwendung
    - Bestimmung Parameter
    - Berechnung Aufwand/Laufzeit

Einfaches CoCoMo-Modell

- Eingaben:
  - Ermittlung Systemgröße (KLOC)
  - Systemklasse:
    - Organic: Vertrautes Problem, erfahrenes Team
    - Semi-Detached: Zwischenform
    - Embedded: Schwere Randbedingungen, neues Problem, wenig Erfahrung
- Ausgaben:
  - Realisierungsaufwand (PM)
  - Entwicklungszeit (TDEV)

CoCoMo-Gleichungen

- Aufwand:
  - OM: PM = 2,4 * (KLOC)^1,05
  - SDM: PM = 3,0 * (KLOC)^1,12
  - EM: PM = 3,6 * (KLOC) ^1,2
- Entwicklungszeit:
  - OM: TDEV = 2,5 * (PM)^0,38
  - SDM: TDEV = 2,5 (PM)^0,35
  - EM: TDEV = 2,5 (PM)^0,32

Modellparameter

- Gleichung:
  - Aufwand: PM = A * (KLOC)^B
  - Entwicklungszeit: TDEV = C * (PM)^D
- Parameterarten: F(P) = L * P^E
  - Linear: Unkritisch ("Balance of Scales")
  - Polynomial: Polynomial: Unkritisch (Eingabeparameter)
  - Exponential: Exponentiell: ("Diseconomy of Scales")

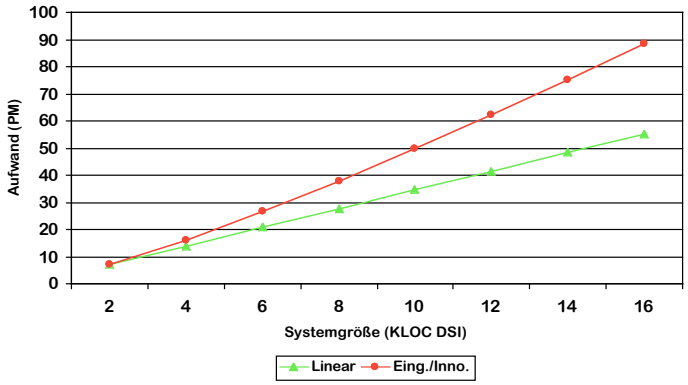
CoCoMo II

- Beobachtung:
  - Schätzung stark abhängig von
    - Eigenschaften zu erstellendes System
    - Eigenschaften erstellendes Unternehmen
  - Unterschiedliche Parameter:
    - Linear (Frühes Design): Projektorientiert
      - Mensch: Qualifikation Personal
      - Technik: Produktgüte, Legacy, Plattform, Entwicklungsumgebung, Zeitplan
    - Exponentiell: Prozessorientiert
      - Technik: Randbedingungen, Risikomanagement, Prozessreife
      - Mensch: Erfahrung, Kohäsion

CoCoMo II Modellgleichungen

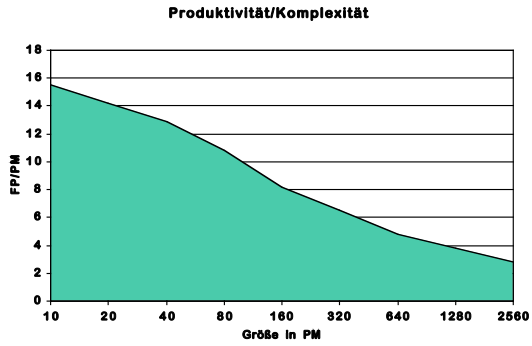
- Parameter:
  - Aufwandsfaktor EM aus Einzelfaktoren EM_i
  - Skalierfaktor B aus Einzelgewichten W_i
  - Grundfaktor A
  - Zeitplanfaktor SCED%
- Gleichungen
  - Skalierfaktor: B = 0,91 + 0,01 * Summe(W_i)
  - Grundaufwand: PM_{Grund} = A * Größe(FPA) ^ B
  - Aufwandsfaktor: EM = Produkt(EM_i)
  - Gesamtaufwand: PM_{Gesamt} = EM * PM_{Grund}
  - Laufzeit: TDEV = [3,67 * PM_{Gesamt} ^(0,28+0,2*(B-1,01))] * SCED%/100

CoCoMo II - Aufwand



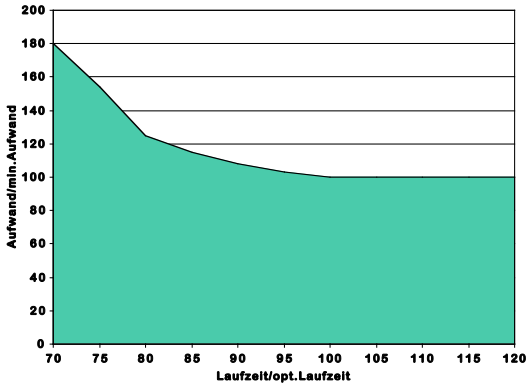
- Generell:
  - Auswirkung des exponentiellen Faktors Prozess
    - Faktor > 1
    - "Diseconomy of Scales"

Komplexität und Produktivität



- Produktivität beeinflußt:
  - Durch administrativen Aufwand
  - Durch personellen Aufwand
  - Durch Abstimmungsaufwand

Realisierungsaufwand und Laufzeit



- Laufzeit beeinflusst Aufwand:
  - Durchschnittliche Laufzeit minimiert Aufwand
  - Zusätzlicher Aufwand erlaubt Kürzung (70%)
  - Erhöhte Laufzeit verringert Aufwand nicht

Aufwandsfaktoren

- Projektparameter
  - Parameter wirken linear auf Aufwand
  - Arten von Parameter:
    - Produktfaktoren, z.B.:
      - Zuverlässigkeit
      - Anforderungen an Dokumentation
      - Vorbereitung Wiederverwendung
    - Personalfaktoren, z.B.:
      - Fluktuation
      - Anwendungserfahrung
    - Projektfaktoren:
      - Werkzeugeinsatz
      - Zeitplan

Skalierfaktoren

- Prozessparameter
  - Faktoren wirken exponentiell auf Aufwand/Laufzeit
  - Arten von Faktoren:
    - Portfolio: Projektdomäne vs. Unternehmensknowhow
    - Prozess: Vorgaben, Reife, Risikomanagement
    - Personal: Teamkohäsion

Zusammenfassung

- Drei Merksätze zur Schätzung
  - Funktionelle Abhängigkeit kann Code als Komplexitätsmaß ersetzen.
  - Entwicklungsaufwand wird von Projektfaktoren linear, von Prozessfaktoren exponentiell beeinflusst.
- Zwei Warnungen zur Schätzung:
  - Detaillierung bestimmt Genauigkeit (20-50%)
  - Modelle benötigen Kalibrierung