

Arbeitsgruppe Software Engineering Prof. Elke Pulvermüller

Universität Osnabrück
Institut für Informatik, Fachbereich Mathematik / Informatik
Raum 31/318, Albrechtstr. 28, D-49069 Osnabrück

elke.pulvermueller@informatik.uni-osnabrueck.de

<http://www.inf.uos.de/se>

Sprechstunde: mittwochs 14 – 15 und n.V.



- 1 Software-Krise und Software Engineering**
- 2 Grundlagen des Software Engineering**
- 3 Projektmanagement**
- 4 Konfigurationsmanagement**
- 5 Software-Modelle**
- 6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle**
- 7 Qualität**
- 8 ... Fortgeschrittene Techniken**

3.1 Grundlagen

3.2 Projektaufbau und Rollen

3.3 Projektplanung und Darstellung

3.4 Projektkontrolle und –steuerung

3.5 Brook'sches Gesetz

3.6 Aufwandsschätzung

3.6 Aufwandsschätzung: Eingliederung

Eingliederung:

1. Machbarkeitsstudie

(Projekttyp, Wiederverwendung, funktionale + nichtfunktionale Anforderungen, Qualität, Risiken)

2. Aufwandsschätzung

3. Auftragsvergabe

(Pflichtenheft mit Ist-, Soll-Zustand und anstehenden Aufgaben, exakter Leistungsbeschreibung, Vereinbarungen und Zeitrahmen, erwartete Dokumentation als Anwenderhandbuch und technische Dokumentation, quantitative Abnahmefestlegung, Kundenpflichten, Gewährleistungsfestlegung, Wartung, Vertragsbeendigung, Kompetenzfestlegung, Vergütung)

4. Konkrete Projektplanung

3.6 Aufwandsschätzung: Nutzen und Anforderungen

Nutzen der Aufwandsschätzung:

- **Projektkalkulation, Beurteilung der Projektwürdigkeit, Angebotserstellung**
- **Entscheidungsgrundlage für eine Auswahl zwischen Alternativen**
(z.B. „make or buy“)
- **Fortlaufende Wirtschaftlichkeits(be)rechnungen, Nachkalkulation**
- **Zeitplanung, Kapazitätsplanung / Ressourcenplanung, Personalplanung**

Anforderungen an ein Verfahren:

**Genauigkeit, Frühzeitigkeit, Detaillierbarkeit, Eindeutigkeit,
Erfassbarkeit/geringe Kosten, Objektivität, Transparenz, Stabilität,
Flexibilität, Benutzerfreundlichkeit**

Überblick der Ansätze

- **Aus-dem-Bauch-Methode:**
formlos, sinnvoll bei genügend Erfahrung
- **Expertenbefragung bzw. Delphi-Methode**
Anonymität in der Delphi-Methode
- **Analogiemethode (Erweiterung: Relationsmethode)**
Gemeinsamkeiten und Parallelen von vorangegangenen Projekten zum aktuellen Vorhaben (Voraussetzung: Aktualität des Zahlenmaterials!);
Relationsmethode berücksichtigt unternehmensspezifische Faktoren
- **Multiplikationsmethode:**
Zerlegung des Projekts in „Bausteine“, Schätzung der einzelnen Teile,
Gesamtaufwand durch Multiplikation der einzelnen Bausteine
- **Gewichtungsmethode:**
Identifikation von „Aufwandstreibern“ (Funktionsmerkmale)
Aufwandberechnung mittels Formel

Überblick der Ansätze

- **Prozentsatzmethode:**

Hochrechnung des Gesamtaufwands aus der detaillierten Schätzung einer (Teil-)Phase (basiert auf Erfahrungswerten der prozentualen Aufwandsverteilung)

- **Damit zwei Hauptrichtungen:**

1. **Expertenschätzung**

2. **Algorithmische Schätzverfahren:**

- Kostenberechnung aus frühzeitig bekannten oder leichter schätzbaren Größen;
- setzt detailliertes Wissen über das Projekt voraus (nutzbar erst in späteren Phasen)

- **In der Praxis: Kombination der Verfahren**

3.6 Aufwandsschätzung: Ansatzarten

Überblick der Ansätze

- In der Praxis häufig stark formalisiert und Zusammensetzung von Elementen aus allen Schätzansätzen
- Erfahrung mit der Methode und den Parametern (Projekt, Betrieb, Mitarbeiter) notwendig
- „Nebentätigkeiten“ explizit oder implizit (z.B. innerhalb des Produktivitätsmaß) mitberücksichtigen (Projektmanagement, Versionsverwaltung, Qualitätssicherung):
Nettoaufwand \leftrightarrow Bruttoaufwand
- Nur Schätzung! Je früher die Schätzung desto mehr Unsicherheit
- Produktivitätsbestimmung nicht einfach:
 - vielfältige Einflussfaktoren
 - Produktivitätsvarianzen
 - Problematische Produktivitätsmessung

3.6 Aufwandsschätzung: Verfahren

Konkrete Schätzverfahren

- O-Notation
- LOC, KLOC, MLOC
- CoCoMo (Constructive Cost Model), CoCoMo II
- Function Points (FP)
- Use Case Points (UCP)
- Data Points, Object Points, Testfall-Methode, ...

Problem:

- breite Basis der Erfahrungswerte ist zentral
- Wandel der Projekttypen, Einsatzumgebungen, Szenarien der Softwareentwicklung
- konsistente Interpretation der Werte erforderlich

Werkzeugunterstützung (z.B. zur Parameterverwaltung für Einflussgrößen)

LOC, KLOC, MLOG

- **LOC = Lines of Code**
Üblich auch: SLOC (Source Lines of Code), KLOC, MLOC
- **Maßeinheiten zur Bestimmung der Software-“Größe“**
- **Zählung?: Kommentarzeilen, Leerzeilen, physikalische oder logische Zeilen, generierter Code?**
Alternative Zählung z.B.: NCSS = Non-Commentary Source Statements
- **relativ unabhängig von der Programmiersprache, Assembler oder Hochsprache (abhängig von der Sichtweise)**
- **Automatisch messbar**
- **Messung der Leistung eines Programmierers: LOC/Zeiteinheit**
(ein Programmierer schreibt etwa 40 - 100 LOC pro Woche)
- **ungeeignet für die Messung der Programmierer-Produktivität als Steuerinstrument**

COCOMO Aufwandsschätzverfahren (COCOMO 81 und COCOMO II), B. Boehm

- **COCOMO II (1999/2000) als Weiterentwicklung von COCOMO 81 zur Anpassung an geänderte Problemstellungen**
- **Basis: KLOC, Delivered Source Instructions**
- **Algorithmisch (auf Rechnung basierendes) Verfahren (wie Function-Points) \Rightarrow Bestimmung Aufwand + Entwicklungsdauer**
- **Je nach Schwierigkeitsgrad des Problems und gewünschtem Detaillierungsgrad der Schätzung werden unterschiedliche Gewichtungsfaktoren (in Wert und Anzahl) herangezogen**
- **Das Softwareproblem wird unterschieden in:**
 - **einfach (organic)**
 - **mittelschwer (teilintegriert, semidetached)**
 - **schwer (eingebettet, embedded)**

Modell 1:

- Formel (VD = voraussichtliche Dauer)

$$VD = a * (KLOC \uparrow b)$$

VD wächst in
Bezug auf LOC
exponentiell

Produktivität in LOC/PM
z.B. Produktivität von 400 LOC/PM
→ $a = 2,5$ ($1000/400$)

- Erfahrungswerte für a und b:

- | | | |
|-----------------------------------|-----------|------------|
| - einfaches Softwareproblem: | $a = 2,4$ | $b = 1,05$ |
| - mittelschweres Softwareproblem: | $a = 3,0$ | $b = 1,12$ |
| - schweres Softwareproblem: | $a = 3,6$ | $b = 1,2$ |

Beispiel für ein einfaches Problem mit geschätzt 2 KLOC:

$$VD \text{ (in Personenmonaten)} = 2,4 * 2^{1,05} = 4,97 \text{ PM}$$

3.6 Aufwandsschätzung: COCOMO

Modell 2:

- Formel (VD = voraussichtliche Dauer)

$$VD = a * (KLOC \uparrow b) * EF$$

- EF = Einflussfaktor

Ursprünglich 12 Einflussfaktoren, z.B.

1. Kompatibilität des Entwicklungsrechners mit dem Zielrechner
(Voll: 1,00, Teils: 1,15, Nichts: 1,30)
8. Projektbegleitende Qualitätssicherung
(Automatisiert: 0,90, Manuell: 1,05, Nichts: 1,20)
11. Sprachkenntnisse der Entwickler
(Viel: 0,80, Mittel: 1,00, Wenig: 1,20)

- Gesamter EF ermittelt sich durch Multiplikation der einzelnen Einflussfaktoren

Function Point Methode/Verfahren = Function Point Analysis (FPA):

- **Entwickler: Albrecht, IBM, 1979**
- **Heute: FPA User Groups, z.B. IFPUG (International Function Point Users' Group): <http://www.ifpug.org/>
IFPUG FSM Method (ISO/IEC 20926 SOFTWARE ENGINEERING - FUNCTION POINT COUNTING PRACTICES MANUAL)**
- **Messen des (erwarteten) Funktionalitätsumfangs eines Systems
Ausgangspunkt: Functional User Requirements (z.B. im Analysedokument)**
- **Unabhängigkeit von der Technologie (z.B. Sprache, Entwicklungsmethode, Plattform)**
- **Einsatz: Bestimmung von Erweiterungs-, Wartungskosten, Bestimmung der Projektproduktivität nach Projektabschluss, Kostenschätzung der Softwaregröße**

3.6 Aufwandsschätzung: Function Points

- **Schritt 1:** a) Kategorisierung jeder Anforderung/Funktionalität
b) Klassifizierung: Schätzung der Komplexität pro Funktionalität
c) Gewichtung entsprechend Tabelle

Element	niedrig	mittel	hoch
Eingabe	3	4	6
Ausgabe	4	5	7
Datenbestände	7	10	15
Abfrage	3	4	6
Referenzdaten	5	7	10

- d) Zählen der fachlichen Funktionalitäten in jeder Kategorie/Klasse
(Berechnung der Function Points durch Multiplikation/Addition)

- **Schritt 2: Einflussfaktoren einschätzen**

Neben der Komplexität des Produkts gibt es zahlreiche andere Produkteinflüsse, die den Aufwand nach oben oder nach unten verändern können (oft geforderte Qualitätsmerkmale oder erschwerende Rahmenbedingungen).

In der Function Point Methode gibt es 14 Einflüsse:

- | | |
|--|--|
| 1. Datenkommunikation | 7. End-user Effizienz |
| 2. verteilte Funktionen | 8. online updates |
| 3. Performanz Anforderungen | 9. komplexe Berechnungen |
| 4. Hardware Konfigurationen | 10. Wiederverwendbarkeit |
| 5. hohe Rate an Transaktionen | 11. Einfachheit der Installation |
| 6. Dateneingang über Online Funktionen | 12. Einfachheit des Betriebs des Systems |
| | 13. verschiedene technische Plattformen |
| | 14. Änderbarkeit des Codes |

- Schritt 2: Einflussfaktoren einschätzen

Einschätzung pro Einflussfaktor durch Zuordnung eines Werts (0 - 5):

- | | | |
|---|---|-------------------------------|
| 0 | - | kein Einflussfaktor |
| 1 | - | gelegentlicher Einflussfaktor |
| 2 | - | geringer Einfluss |
| 3 | - | mittlerer Einfluss |
| 4 | - | starker Einfluss |
| 5 | - | kritischer Einfluss |

Die Summe der Einflussfaktoren ergibt damit einen Wert zwischen 0 und 70.
Für die Function Points errechnet sich dann der Faktor zu

$$\left(\sum_{\text{alle Einflüsse}} g * 0.01 \right) + 0.65$$

So dass die Einflussfaktoren die Komplexität um 35% „verändern“ können.

- **Schritt 3: Berechnung der Function Points mit der Formel:**

$$FP = \sum_{gew\ Anf} A_g * ((\sum_{Einfl} g_i * 0.01) + 0.65)$$

- **Schritt 4: Errechnung des Aufwands in PM auf der Basis der errechneten Function Points mit Hilfe einer Erfahrungskurve (10 FP ergeben in etwa 10 – 15 PM)**
- **Schritt 5: Aktualisierung der Aufwandskurve (Rückkopplung der Erfahrungswerte nach Abschluss des Projekts)**

3.6 Aufwandsschätzung: Use Case Points

- Analog zum Function Point Verfahren, aber Schätzung der Use Cases statt der Anforderungen (Orientierung am OO Paradigma)
- Use Case: beschreibt das Verhalten und die Interaktion eines Systems als Reaktion auf die zielgerichtete Anfrage oder Aktion eines Akteurs (Anwendersicht)
- Problem: einheitliche Granularität der Use Cases

3 Projektmanagement

3.6 Aufwandsschätzung: Use Case Points

Management Factor (Einfluss der organisatorischen Komplexität und des Umfelds)

Technical Complexity Factor (Einfluss der technologischen Randbedingungen)

Aufwand über alle Phasen*

Produktivitätsfaktor

M-Faktor

Fragebogen mit
Kostenfaktoren

X

T-Faktor

Fragebogen mit
Kostenfaktoren

X

Use Case Points = Bereinigte Use Case Points

Σ Use-Case-Gewichte

+

Σ Aktoren-Gewichte

Use Case	Komplexität	Use-Case-Gewicht
Auftrag verwalten	hoch	15
Kunde verwalten	einfach	5
Produkt verwalten	mittel	10
...

Aktor	Typ	Aktoren-Gewicht
Stammdaten	Nachbarsystem (API)	5
Geschäftspartner	Nachbarsystem (Protokoll)	10
Händler	Benutzer-Interface	15
...

ABC Individuelle Analyse → Berechnung nach Standard-Metrik (einfach, mittel, komplex)
 ■ Berechnung nach firmeneigener Metrik

[InnoZent OWL: Arbeitskreissitzung für IT-Verantwortliche im Unternehmen, "Use Cases als Basis der Aufwandsschätzung von Software Projekten", Stephan Frohnhoff, sd&m software design & management AG, 4.6.2008]

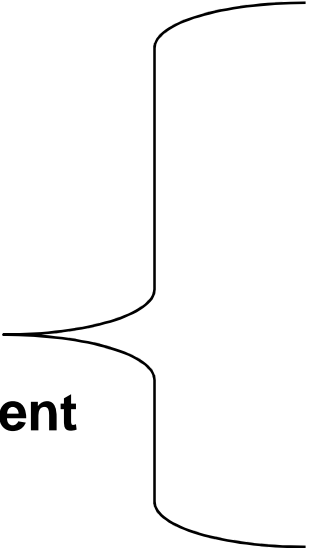
Schätzung des Gesamtaufwands an Hand der Testfälle, die benötigt werden, um alle Funktionen der Software auszutesten.

- **Testfälle = Indikator für die Komplexität eines Systems (z.B. Testaufwand)**
- **Schätzung des Aufwands pro Testfall und Addition**
- **Ggf. Adjustierung mit Einflussfaktoren**

Aufwandsplan: Gesamtaufwand → Wirtschaftlichkeitsrechnung:

- **Nettogesamtaufwand:** Summe der geschätzten Nettoaufwände pro Arbeitspaket (Hochrechnung über alle Phasen und ggf. zusätzlichen Einflussfaktoren)
- **Bruttogesamtaufwand:** zusätzliche, geschätzte Aufwände z.B. für Projektleitung, Qualitätssicherung, Infrastruktur/Werkzeuge, Mitarbeiterereinarbeitung
- **Wirtschaftlichkeitsrechnung:** Gegenüberstellung von geschätzten Kosten zu geschätztem Nutzen (Vorstufe zur Investitionsrechnung)

- 1 Software-Krise und Software Engineering
- 2 Grundlagen des Software Engineering
- 3 Projektmanagement
- 4 Konfigurationsmanagement
- 5 Software-Modelle
- 6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle
- 7 Qualität
- 8 ... Fortgeschrittene Techniken

- 
- 3.1 Grundlagen
 - 3.2 Projektaufbau und Rollen
 - 3.3 Projektplanung und Darstellung
 - 3.4 Projektkontrolle und -steuerung
 - 3.5 Brook'sches Gesetz
 - 3.6 Aufwandsschätzung

→ Softwareentwicklung in Projekten benötigt
Planung, Steuerung, Kontrolle (Projektmanagement)
UND Management der im Projekt anfallenden
Einheiten: Konfigurationsmanagement

