Softwaretechnik

Prof. Dr. Rainer Koschke

Fachbereich Mathematik und Informatik
Arbeitsgruppe *Softwaretechnik*
Universität Bremen

Sommersemester 2006

Überblick I

Kosten- und Aufwandsschätzung

Kostenschätzung

Wichtige Fragen vor einer Software-Entwicklung:

- Wie hoch wird der Aufwand sein?
- Wie lange wird die Entwicklung dauern?
- Wie viele Leute werden benötigt?

Frühzeitige Beantwortung wichtig für:

- Kalkulation und Angebot
- Personalplanung
- Entscheidung „make or buy“
- Nachkalkulation

Kostenschätzung

Ansätze:

- Expertenschätzung
- Berechnung aus früh bekannten Größen (algorithmische Schätzung)
  - COCOMO: Anzahl Codezeilen
  - Function Points: Ein- und Ausgaben
- Analogiemethode
- Top-Down-Schätzung: Ableitung aus globalen Größen
  - z.B. Aufwand steht fest, daraus Umfang ableiten
- Bottom-Up-Schätzung: Dekomposition und Schätzung der einzelnen Komponenten sowie deren Integrationsaufwand
- Daumen-Regeln
  - Gesamtaufwand: 1 DLOC/h (Delivered Line of Code)
  - Gesamtkosten: 50 Euro/DLOC
- Pricing-to-Win
- Parkinsons Gesetz

Kostenschätzung

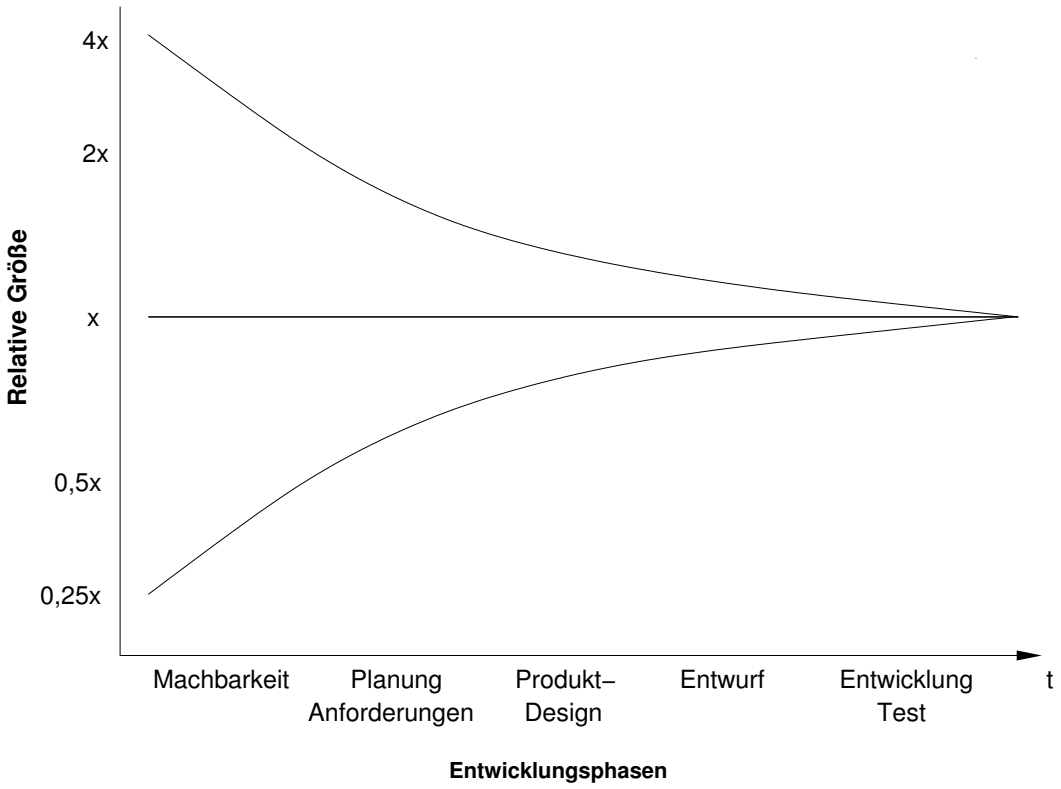
Ansätze:
- Expertenschätzung
- Berechnung aus früh bekannten Größen (algorithmische Schätzung)
  - COCOMO: Anzahl Codezeilen
  - Function Points: Ein- und Ausgaben
- Analogiemethode
- Top-Down-Schätzung: Ableitung aus globalen Größen
  - z.B. Aufwand steht fest, daraus Umfang ableiten
- Bottom-Up-Schätzung: Dekomposition und Schätzung der einzelnen Komponenten sowie deren Integrationsaufwand
- Daumen-Regeln
  - Gesamtaufwand: 1 DLOC/h (Delivered Line of Code)
  - Gesamtkosten: 50 Euro/DLOC
- Pricing-to-Win
- Parkinsons Gesetz

- mehrere Experten fragen; Abweichungen diskutieren, bis Konsens erreicht
- statistisches Kostenmodell aus Vergangenheitsdaten wird erstellt; Modell wird zur Vorhersage benutzt
- Bezug auf historische Daten eines ähnlichen Projekts
- ...
- Pricing-To-Win: Preis wird vereinbart; im Zuge des Projekts einigt man sich auf Funktionsumfang
- Parkinsons Gesetz: wenn X Zeit zur Verfügung steht, wird X Zeit benötigt

Interessanterweise sind Manager gar nicht so schlecht im Schätzen von Aufwänden; schlecht sind sie jedoch in der Schätzung von LOC.

Kostenschätzung

Boehm (1981)



Function-Point Methode

Zweck:

- Messen des Umfangs einer zu erstellenden Software aus Benutzersicht
- Umrechnung des Umfangs in personellen Aufwand
- Eingabe: Lastenheft

Entwicklung:

- Autor: Alan J. Albrecht (1979) (IBM)
- Heute zahlreiche Varianten
- IFPUG Int'l Function Point User Group

Function-Point Methode

Zweck:
- Messen des Umfangs einer zu erstellenden Software aus Benutzersicht
- Umrechnung des Umfangs in personellen Aufwand
- Eingabe: Lastenheft

Entwicklung:
- Autor: Alan J. Albrecht (1979) (IBM)
- Heute zahlreiche Varianten
- IFPUG Int'l Function Point User Group

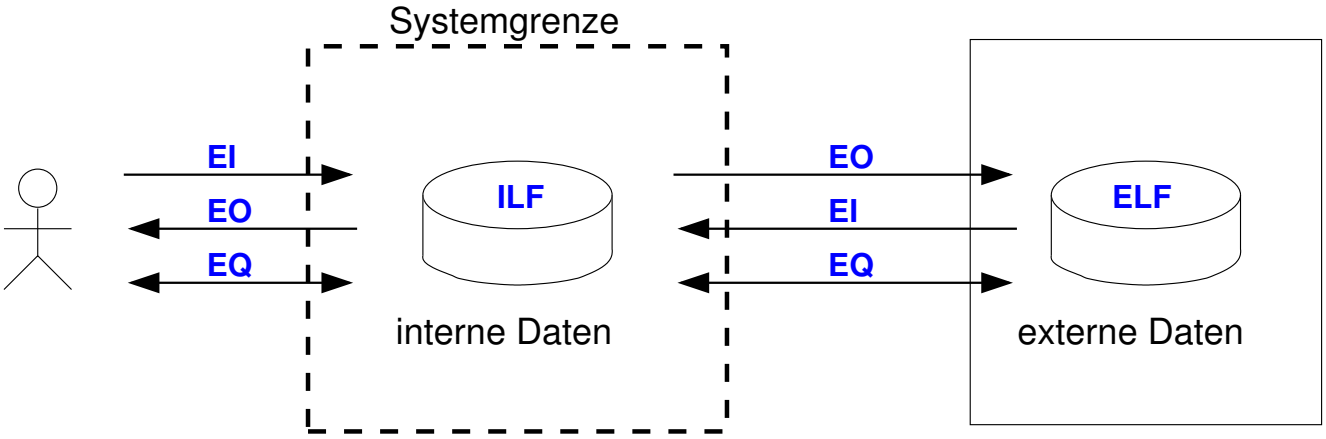
`http://www.ifpug.com/fpafund.htm`

FAQ: `http://ourworld.compuserve.com/homepages/softcomp/fpfaq.htm`

Function-Point Methode – Vorgehen

- Zähltyp festlegen: Neu-/Weiterentwicklung, bestehendes System
- Systemgrenzen festlegen
- Identifizieren der Funktionstypen
- Bewerten der Komplexität der Funktionstypen
- Ermittlung der gewichteten Function Points
- Ermittlung des Aufwands

FP – Identifizieren von Funktionstypen



Transaktions-Funktionstypen:

- Eingaben: EI (External Input; Eingabe für ILF)
- Ausgaben: EO (External Output; Ausgabe abgeleiteter Daten)
- Abfragen: EQ (External Inquiry; Eingabe: Anfrage, Ausgabe: Daten)

Daten-Funktionstypen:

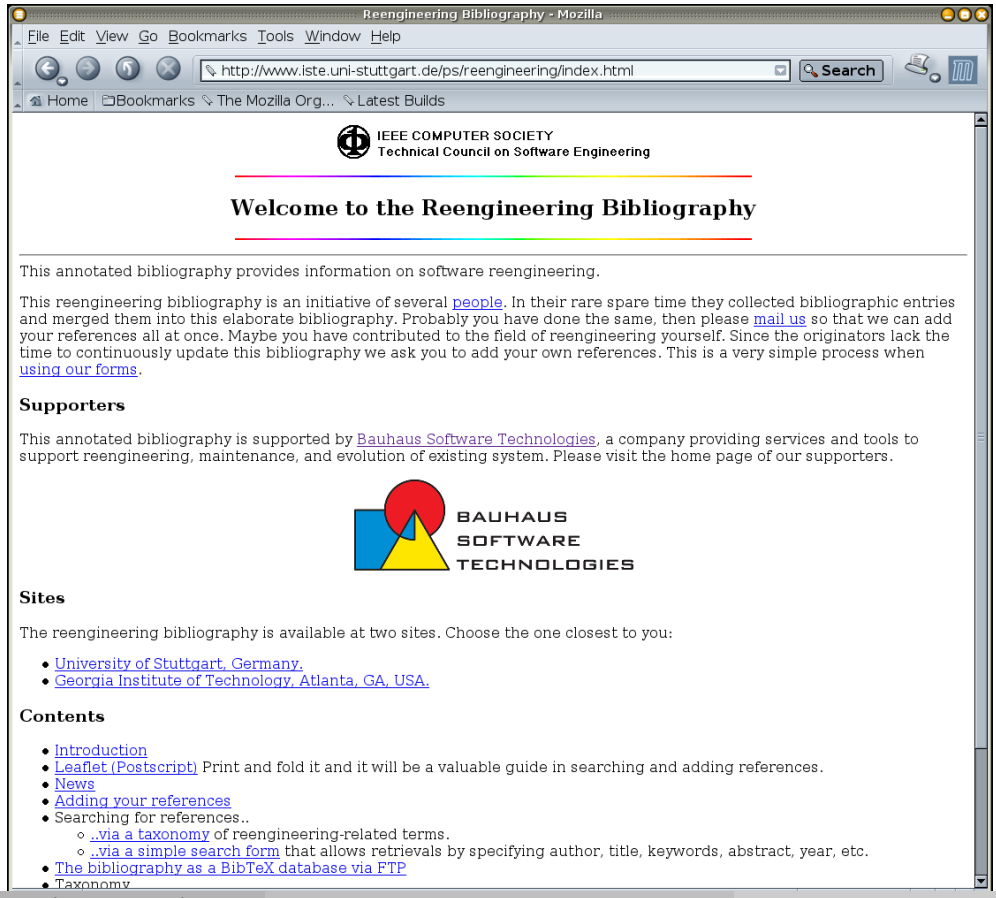
- Interner Datenbestand: ILF (Internal Logical File)
- Externer Datenbestand: ELF (External Logical File)

FP – Identifizieren von Funktionstypen

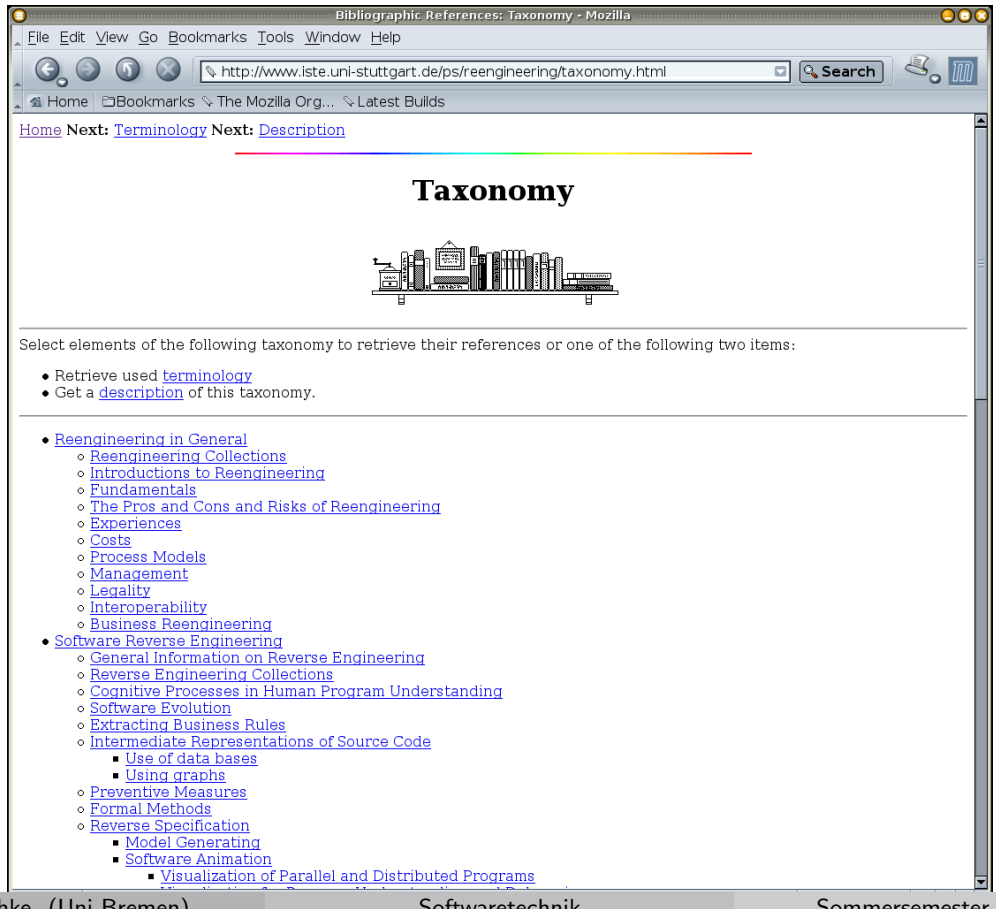
Schlüsselwörter geben Hinweise:

- EI: ablegen/speichern, de-/aktivieren, abbrechen, ändern/editieren/modifizieren/ersetzen, einfügen/hinzufügen, entfernen/löschen, erstellen, konvertieren, update, übertragen
- EO: anzeigen, ausgeben, ansehen, abfragen, suchen/durchsuchen, darstellen, drucken, selektieren, Anfrage, Abfrage, Report
- EQ: abfragen, anzeigen, auswählen, drucken, suchen/durchsuchen, darstellen/zeigen, drop down, extrahieren, finden, holen, selektieren, Ausgabe, Liste, Report

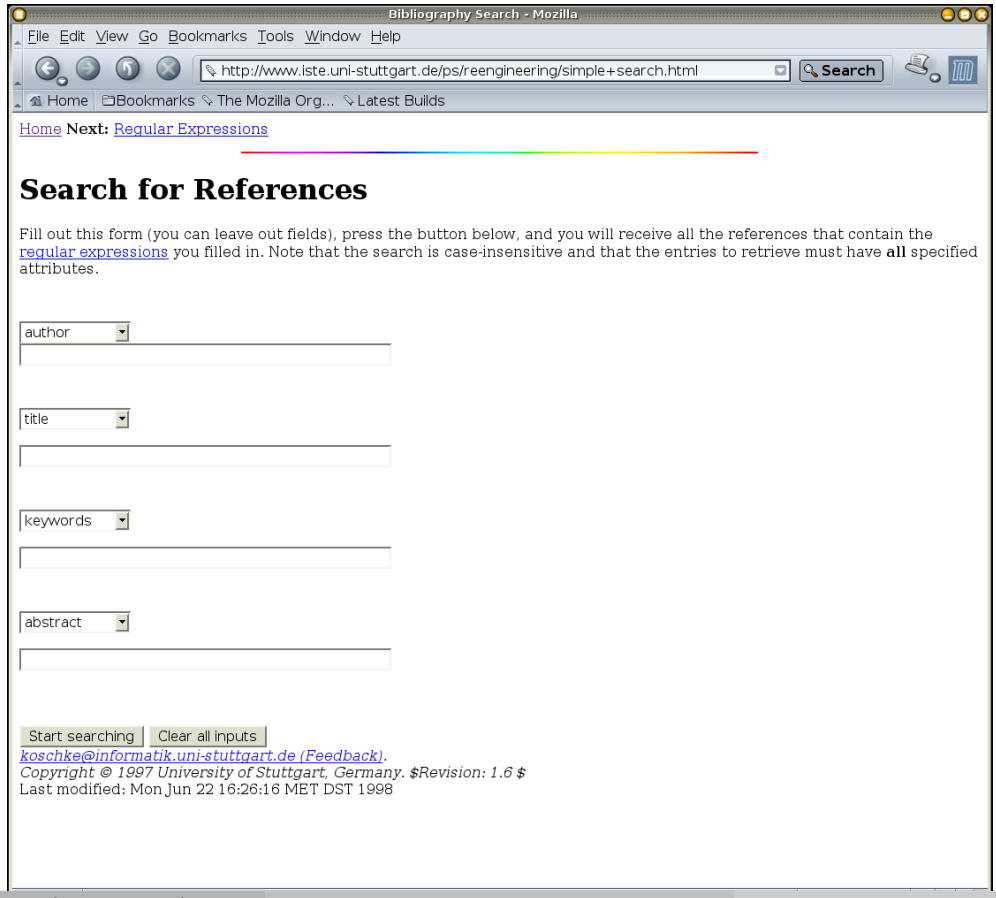
Online-Bibliographie: Startseite



Online-Bibliographie: Taxonomiesuche



Online-Bibliographie: Suche nach Eigenschaften



Beispiel: Online-Bibliographie

Systemgrenzen der Online-Bibliographie

- EI_1: BibTeX-Datei importieren
- EO_1: Abfrage über Taxonomie anzeigen lassen
- EQ_1: Artikel über Suchmaske abfragen
- ILF_1: Referenzen-Datenbank
- ELF_1: externe BibTeX-Datei

Function Points zusammenfassen

Parameter	Zähler	Gewicht	Wert
EI	c_1	w_1	$v_1 = c_1 \times w_1$
EO	c_2	w_2	$v_2 = c_2 \times w_2$
EQ	c_3	w_3	$v_3 = c_3 \times w_3$
ILF	c_4	w_4	$v_4 = c_4 \times w_4$
ELF	c_5	w_5	$v_5 = c_5 \times w_5$
Unadjusted Function Points (UFP)			$\sum v_i$

- zu schätzen: c_i und w_i
- feinere Aufteilung ist möglich
  - z.B. 3 EI mit Gewicht 6 und 4 EI mit Gewicht 3

Umfang $\sim$ Summe FPs; „ungewichtete Funktionspunkte“ (UFP)

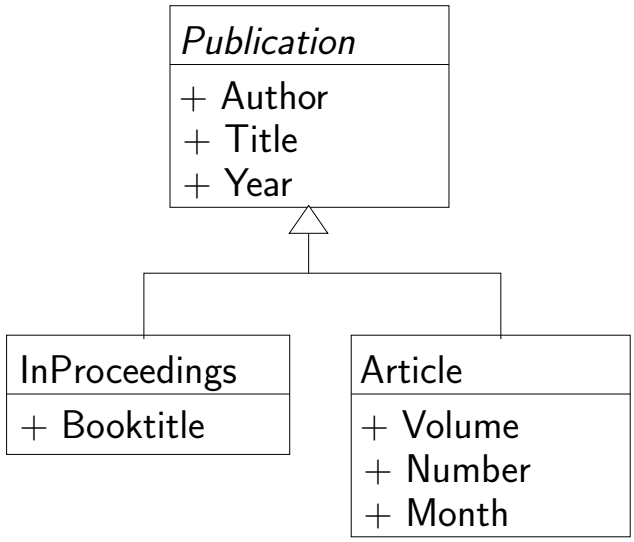
Beispiel: Komplexitätsgewichte w_i für ELF und ILF

Definition

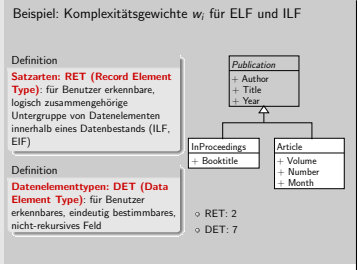
Satzarten: RET (Record Element Type): für Benutzer erkennbare, logisch zusammengehörige Untergruppe von Datenelementen innerhalb eines Datenbestands (ILF, EIF)

Definition

Datenelementtypen: DET (Data Element Type): für Benutzer erkennbares, eindeutig bestimmbares, nicht-rekursives Feld



- RET: 2
- DET: 7



Hier werden die Daten abgeschätzt.

Function Points zusammenfassen

Parameter	Zähler	RET	DET	Gewicht	Wert
ILF_1	1	2	7		
ELF_1	1	2	7		
Parameter	Zähler	FTR	DET	Gewicht	Wert
EI_1					
EO_1					
EQ_1					
Unadjusted Function Points (UFP)					

Komplexitätsgewichte w_i für EI, EO, EQ

Definition

Referenzierte Datenbestände: FTR (File Type Referenced): von Transaktion verwendeter Datenbestand (ILF, ELF)

Beispiel: Datenbank oder Textdatei, die bei der Ausgabe von Kundendaten herangezogen wird

Beispiele für DETs im Kontext Transaktionen:
Eingabe-/Ausgabefelder (GUI), Spalten u.Ä. bei Berichten

Komplexitätsgewichte w_i für EI, EO, EQ
Definition
Referenzierte Datenbestände: FTR (File Type Referenced): von Transaktion verwendeter Datenbestand (ILF, ELF)
Beispiel: Datenbank oder Textdatei, die bei der Ausgabe von Kundendaten herangezogen wird
Beispiele für DETs im Kontext Transaktionen:
Eingabe-/Ausgabefelder (GUI), Spalten u.Ä. bei Berichten

Hier werden die Funktionen abgeschätzt.

Function Points zusammenfassen

- EI_1: BibTeX-Datei importieren
- EO_1: Abfrage über Taxonomie anzeigen lassen
- EQ_1: Artikel über Suchmaske abfragen
- ILF_1: Referenzen-Datenbank
- ELF_1: externe BibTeX-Datei

Parameter	Zähler	RET	DET	Gewicht	Wert
ILF_1	1	2	7		
ELF_1	1	2	7		
Parameter	**Zähler**	**FTR**	**DET**	**Gewicht**	**Wert**
EI_1	1	2	7		
EO_1	1	1	7		
EQ_1	1	1	7*		
Unadjusted Function Points (UFP)					

*) jedes DET wird nur einmal gezählt

Function Points zusammenfassen
- EI_1: BibTeX-Datei importieren
- EO_1: Abfrage über Taxonomie anzeigen lassen
- EQ_1: Artikel über Suchmaske abfragen
- ILF_1: Referenzen-Datenbank
- ELF_1: externe BibTeX-Datei

Parameter	Zähler	RET	DET	Gewicht	Wert
ILF_1	1	2	7		
ELF_1	1	2	7		
Parameter	Zähler	FTR	DET	Gewicht	Wert
EI_1	1	2	7		
EO_1	1	1	7		
EQ_1	1	1	7*		
Unadjusted Function Points (UFP)					

*) jedes DET wird nur einmal gezählt

Bei EI_1 werden 7 DETs von ELF_1 gelesen und 7 DETs von ILF_1 geschrieben. Die geschriebenen DETs kreuzen aber nicht die Systemgrenze, so dass insgesamt nur 7 DETs und nicht 14 betrachtet werden.
Bei EQ_1 werden 4 DETs der Suchmaske gelesen (Author, Title, Keywords und Abstract) und 7 DETs von ILF_1 ausgegegen. Aus logischer Sicht werden aber die gleichen DETs (Author, Title, Keywords und Abstract) gelesen und angezeigt, so dass insgesamt nur 7 DETs und nicht 11 betrachtet werden.

FP – Bestimmung der Komplexitätsgewichte w_i

Komplexitätsmatrizen:

- (Funktionstyp, #FTRs/RETs, #DETs) $\rightarrow$ FPs
- Zählen mittels Zählregeln pro Funktionstyp

FTRs/RETs \ DETs	**Funktionstyp**	1 bis a	$a+1$ bis b	$> b$
	1 bis x	gering	gering	mittel
	$x+1$ bis y	gering	mittel	hoch
	$> y$	mittel	hoch	hoch

FP – Bestimmung der Komplexitätsgewichte w_i

Komplexitätsmatrizen:
- (Funktionstyp, #FTRs/RETs, #DETs) → FPs
- Zählen mittels Zählregeln pro Funktionstyp

DETs

FTRs/RETs	Funktionstyp	1 bis a	$a+1$ bis b	$> b$
	1 bis x	gering	gering	mittel
	$x+1$ bis y	gering	mittel	hoch
	$> y$	mittel	hoch	hoch

FTR: number of files updated or referenced. A record element type is a user recognizable subgroup of data elements within an ILF or EIF. A data element type is a unique user recognizable, nonrecursive, field.

Zählen von Datenelementtypen (DET)

Ein Datenelementtyp (DET) ist aus der Benutzersicht eindeutig bestimmbares, nicht rekursives Feld, das von der zu bewertenden externen Eingabe (EI) in einem internen Datenbestand (ILF) gepflegt wird.

Zählen Sie 1 DET für jedes aus Benutzersicht nicht rekursive Feld, das die Systemgrenze kreuzt und gebraucht wird, um den Elementarprozess abzuschließen.

Beispiel: Ein Texteingabefeld, in dem der Nachname eines neuen Kunden eingegeben wird, wird als 1 DET gezählt.

Gegenbeispiel: Eine Dateiauswahlliste, in der beliebig viele Dateien von der Festplatte des Benutzers ausgewählt werden können, ist rekursiv, und muss somit gesondert gezählt werden.

Zählen Sie keine Felder, die durch das System gesucht und/oder in einem ILF gespeichert werden, wenn die Daten nicht die Systemgrenze überqueren.

Zählen Sie nur 1 DET für die Fähigkeit, eine Systemantwort als Meldung für einen aufgetretenen Fehler aus der Anwendung heraus zu senden bzw. für die Bestätigung, dass die Verarbeitung beendet oder fortgesetzt werden kann Beispiel: Bei Eingabe eines Datums soll z.B. das Format TT/MM/JJJJ eingehalten werden. Gibt der Bearbeiter z.B. '12.03.1997' ein und bestätigt seine Eingabe, so erhält er die Meldung 'neuer Datensatz gespeichert'. Gibt er hingegen '12.3.97' ein (Jahreszahl nicht vierstellig) so erhält er die Fehlermeldung 'Fehler: Bitte Datum korrigieren'. Nur ein DET wird für diese Fähigkeit des Systems gezählt.

Zählen Sie nur 1 DET für die Möglichkeit, eine Aktion durchzuführen, auch wenn es viele Methoden gibt, die denselben logischen Prozess anstoßen. Beispiel: In einem Eingabeformular gibt es einen OK-Button zum Absenden der Daten. Die Tastaturkombination STRG-S führt ebenfalls zum Senden der Daten. Somit wird nur ein DET gezählt.

FP – Bestimmung der Komplexitätsgewichte w_i

Komplexitätsmatrizen:
- (Funktionstyp, #FTRs/RETs, #DETs) → FPs
- Zählen mittels Zählregeln pro Funktionstyp

FTRs/RETs	DETs			
Funktionstyp	1 bis a	$a+1$ bis b	$> b$
1 bis x	gering	gering	mittel
$x+1$ bis y	gering	mittel	hoch
$> y$	mittel	hoch	hoch

(Fortsetzung)

Zählen von referenzierte Datenbeständen (FTR)

Ein referenzierter Datenbestand (FTR) ist eine vom Benutzer definierte Gruppierung zusammengehöriger Daten oder Steuerungsinformationen in einem internen Datenbestand (ILF), die bei der Bearbeitung der externen Eingabe gelesen oder gepflegt wird.

Zählen Sie 1 FTR für jeden referenzierten Datenbestand, der während der Verarbeitung der externen Eingabe gelesen wird. Beispiel: Es werden durch eine externe Eingabe Produktdaten in einer Datenbank gespeichert. Dazu werden die Produktbezeichnungen aus einer weiteren Datenbank ausgelesen, die damit zusätzlich zu der zu aktualisierenden Produktdatenbank einen weiteren Datenbestand darstellt, der jedoch nur gelesen wird.

Zählen Sie 1 FTR für jede ILF, die während der Verarbeitung der externen Eingabe gepflegt wird. Beispiel: Es wird zusätzlich zu den Aktionen des vorigen Beispiels eine Textdatei aktualisiert, in der die Anzahl der Zugriffe auf die Datenbank verzeichnet wird.

Zählen Sie nur 1 FTR für jede ILF, die während der Verabeitung der externen Eingabe gelesen und gepflegt wird. Beispiel: Würden die Informationen der Textdatei ebenfalls in der Datenbank gespeichert werden, so wird diese nur als 1 FTR gezählt, obwohl die Datenbank zur Ein- und Ausgabe von Daten verwendet wird.

2006-05-17
Softwaretechnik
Kosten- und Aufwandsschätzung
Function-Points
FP – Bestimmung der Komplexitätsgewichte w_i

FP – Bestimmung der Komplexitätsgewichte w_i

Komplexitätsmatrizen:
- (Funktionstyp, #FTRs/RETs, #DETs) → FPs
- Zählen mittels Zählregeln pro Funktionstyp

FTRs/RETs	DETs			
Funktionstyp	1 bis a	$a+1$ bis b	$> b$
1 bis x	gering	gering	mittel
$x+1$ bis y	gering	mittel	hoch
$> y$	mittel	hoch	hoch

(Fortsetzung)

Besonderheiten bei grafischen Benutzungsoberflächen

Besonderheiten bei grafischen Benutzungsoberflächen Optionsfelder (Radiobuttons) stellen Datenelemente dar. Es wird pro Gruppe von Optionsfeldern 1 DET gezählt, da innerhalb einer Gruppe nur ein Optionsfeld ausgewählt werden kann. Beispiel: Eine Gruppe von 12 Radiobuttons, in der ein PKW-Typ ausgewählt werden kann, wird als 1 DET gezählt.

Kontrollkästchen (Checkboxen) stellen Datenelemente dar. Im Gegensatz zu Optionsfeldern können aus einer Gruppe von Checkboxen mehrere Elemente gleichzeitig ausgewählt werden. Somit wird jedes Element als 1 DET gezählt. Beispiel: Eine Gruppe von 12 Checkboxen, mit der ein Pizza-Belag zusammengestellt werden kann, wird als 12 DETs gezählt.

Eingabe- und Ausgabefelder stellen Datenelemente dar. Beispiel: In einer Bildschirmmaske werden Vorname, Nachname, Straße, Hausnummer, PLZ und Ort in Eingabefeldern erfasst. Somit werden 6 DET gezählt.

Literale stellen keine Datenelemente dar. Beispiel: Vor einem Feld ist die Textzeile 'monatliches Gehalt' und dahinter 'in Euro/Monat' angegeben. Beide Textzeilen sind Literale und werden nicht gezählt

Enter-, OK-Button und Programmfunktionstaste werden insgesamt als 1 DET gezählt, da jeweils die gleiche Funktion ausgeführt wird. Beispiel: Die Daten eines Dialogs werden nach Betätigen der Enter-Taste oder nach Betätigen der Schaltfläche 'übernehmen' (gleiche Funktion wie OK-Button) in einer Datenbank gespeichert. Es wird 1 DET gezählt.

Berichte/Reports können verschiedene Ausgabeformen haben. So kann die gleiche Datenbasis zur Darstellung als Tortendiagramm, Tabelle, textuell, als druckbares Format oder als Exportdatei dargestellt werden. Jedes Format stellt dabei eine externe Ausgabe (EO) dar.

FP – Werte der Komplexitätsgewichte w_i

FTRs \ DETs	**EI**	1-4	5-15	16+
	0-1	3	3	4
	2	3	4	6
	3+	4	6	6

FTRs \ DETs	**EO**	1-5	6-19	20+
	0-1	4	4	5
	2-3	4	5	7
	4+	5	7	7

FTRs \ DETs	**EQ**	1-5	6-19	20+
	0-1	3	3	4
	2-3	3	4	6
	4+	4	6	6

RETs \ DETs	**ILF**	1-19	20-50	51+
	1	7	7	10
	2-5	7	10	15
	6+	10	15	15

RETs \ DETs	**ELF**	1-19	20-50	51+
	1	5	5	7
	2-5	5	7	10
	6+	7	10	10

Beispiel: Function Points zusammenfassen

- EI_1: BibTeX-Datei importieren
- EO_1: Abfrage über Taxonomie anzeigen lassen
- EQ_1: Artikel über Suchmaske abfragen
- ILF_1: Referenzen-Datenbank
- ELF_1: externe BibTeX-Datei

Parameter	Zähler	RET	DET	Gewicht	Wert
ILF_1	1	2	7	7	7
ELF_1	1	2	7	5	5
Parameter	Zähler	FTR	DET	Gewicht	Wert
EI_1	1	2	7	4	4
EO_1	1	1	7	4	4
EQ_1	1	1	7	3	3
Unadjusted Function Points (UFP)					23

FP – Gewichtete Function-Points

Systemmerkmale:

- Datenkommunikation
- Verteilte Verarbeitung
- Leistungsfähigkeit
- Begrenzte Kapazität
- Transaktionsrate
- Interaktive Dateneingabe
- Benutzerfreundlichkeit
- Interaktive Änderung
- Komplexe Verarbeitung
- Wiederverwendbarkeit
- Installationshilfen
- Betriebshilfen
- Mehrfachinstallation
- Änderungsfreundlichkeit

Bewertung: 0 = kein Einfluss, 5 = starker Einfluss

Konkrete Fragen I

- Does the system require reliable backup and recovery? **3**
- Are data communications required? **2**
- Are there distributed processing functions? **0**
- Is performance critical? **1**
- Will the system run in an existing, heavily utilized operational environment? **1**
- Does the system require on-line data entry? **4**
- Does the online data entry require the input transaction to be built over multiple screens or operations? **3**

Konkrete Fragen II

- Are the master files updated on-line? **5**
- Are the inputs, outputs, files, or inquiries complex? **1**
- Is the internal processing complex? **1**
- Is the code designed to be reusable? **1**
- Are conversion and installation included in the design? **2**
- Is the system designed for multiple installations in different organizations? **2**
- Is the application designed to facilitate change and ease of use by the user? **3**

Konkrete Fragen II
- Are the master files updated on-line? 5
- Are the inputs, outputs, files, or inquiries complex? 1
- Is the internal processing complex? 1
- Is the code designed to be reusable? 1
- Are conversion and installation included in the design? 2
- Is the system designed for multiple installations in different organizations? 2
- Is the application designed to facilitate change and ease of use by the user? 3

Die Zahlen beziehen sich auf unser laufendes Beispiel.

FP – Gewichtete Function-Points

- TDI (Total Degree of Influence) = Summe der Bewertungen
- VAF (Value Adjustment Factor) = TDI/100 + 0,65
  → Gesamteinflussfaktor: 65% - 135%
- AFP (Adjusted Function Points) = UFP · VAF

Beispiel:

- TDI = 29
- VAF = 29/100 + 0,65 = 0,94
- AFP = 23 · 0,94 = 21,62

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs → Aufwand

- Erstellung einer neuen Erfahrungskurve
  (Zählen abgeschlossener Projekte, Regressionsanalyse)
- Datenbank, z.B. ISBSG[1]:
  - 3GL-Projekte: $PM = 0,971 \cdot AFP^{0,351}$
  - 4GL-Projekte: $PM = 0,622 \cdot AFP^{0,405}$
  - basierend auf 662 Projekten: $PM = 0,38 \cdot AFP^{0,37}$
- grobe Schätzung mit Faustregeln Jones (1996):
  - Entwicklungsdauer (Monate) = $AFP^{0.4}$
  - Personen = AFP / 150 (aufgerundet)
  - Aufwand (Personenmonate) = Personen · Entwicklungsdauer

Beispiel (mit Jones-Schätzung):

- Entwicklungsdauer (Monate) = $21.62^{0.4} = 3.42$
- Personen = 21.62/150 → 1
- Aufwand (Personenmonate) = 1 · 3.42 = 3.42

[1] International Software Benchmarking Standards Group

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs → Aufwand

- Erstellung einer neuen Erfahrungskurve (Zählen abgeschlossener Projekte, Regressionsanalyse)
- Datenbank, z.B. ISBSG[1]:
  - 3GL-Projekte: $PM = 0,971 \cdot AFP^{0,351}$
  - 4GL-Projekte: $PM = 0,622 \cdot AFP^{0,465}$
  - basierend auf 662 Projekten: $PM = 0,38 \cdot AFP^{0,37}$
- grobe Schätzung mit Faustregeln Jones (1996):
  - Entwicklungsdauer (Monate) = $AFP^{0,4}$
  - Personen = AFP / 150 (aufgerundet)
  - Aufwand (Personenmonate) = Personen · Entwicklungsdauer

Beispiel (mit Jones-Schätzung):

- Entwicklungsdauer (Monate) = $21.62^{0,4} = 3.42$
- Personen = $21.62/150 \rightarrow 1$
- Aufwand (Personenmonate) = $1 \cdot 3.42 = 3.42$

[1] International Software Benchmarking Standards Group

http://www.isbsg.org/ http://www.isbsg.nsf/weben/Project%20Duration

FP – Umrechnung in LOC

Mittlere Anzahl Codezeilen pro FP (Jones 1995):

Sprache	ØLOC
Assembler	320
C	128
FORTRAN	107
COBOL (ANSI 85)	91
Pascal	91
C++	53
Java	53
Ada 95	49
Smalltalk	21
SQL	12

Bewertung der Function-Point-Methode

+ Wird als beste verfügbare Methode zur Schätzung kommerzieller Anwendungssysteme angesehen (Balzert 1997)
+ Sinnvoll einsetzbar, wenn Erfahrungswerte von vergleichbaren Projekten vorliegen (Kemerer 1987)
– Bewertung der Systemmerkmale subjektiv (Symons 1988)
+ Studie: mittlere FP-Abweichung zwischen 2 „Zählern“ nur 12% (Kemerer und Porter 1992)
– Zählen der FPs relativ aufwendig

Object Points

- Für 4GLs (Query Languages, Report Writers, . . .)
- Haben nicht unbedingt mit OOP-Objekten zu tun
- Gewichtete Schätzung von
  - Anzahl verschiedener „Screens“
  - Anzahl erstellter „Reports“
  - Anzahl zu entwickelnder 3GL-Module
- Vorteil: Einfacher und weniger zu schätzen:
  vergleichbare Präzision wie Function-Point-Schätzung (Banker u. a. 1991)
  47% des Aufwands für Function-Point-Schätzung (Kauffman und Kumar 1993)

Object Points

Screens

# views contained	# data tables		
	< 4	< 8	8+
< 3	1	1	2
3-7	1	2	3
> 8	2	3	3

Reports

# sections contained	# data tables		
	< 4	< 8	8+
0-1	2	2	5
2-3	2	5	8
4+	5	8	8

Views: Menge logisch zusammengehöriger Daten (z.B. Kundenstammdaten)

Data Tables = # Server data tables + # Client data tables (Tabellen, die abgefragt werden müssen, um Daten zu bestimmen)

Jede 3GL-Komponente: 10 object points

COCOMO

COCOMO = Constructive Cost Model (Boehm 1981)

- Basiert auf Auswertung sehr vieler Projekte
- Eingaben: Projektkomplexität (3 Stufen), Systemgröße
- Ausgaben: Realisierungsaufwand, Entwicklungszeit
- Drei Genauigkeitsstufen (steigender Aufwand):
  - *Basic:* Aufwand $= a \cdot \text{KLOC}^b$, Dauer $= c \cdot \text{Aufwand}^d$
  - *Intermediate:* Dekomposition, 15 Einflussfaktoren (Kategorien: Produkt, Projekt, Computer, Personal)
  - *Advanced:* Einflussfaktoren pro Phase

COCOMO

a, b konstant, abhängig von Projektkomplexität:

- *Organic:* $\text{PM} = 2,4 \cdot \text{KDSI}^{1,05} \cdot M$
  - wohl verstandene Anwendungsdomäne mit kleinen Teams
- *Semidetached:* $\text{PM} = 3,0 \cdot \text{KDSI}^{1,12} \cdot M$
  - komplexere Projekte, bei dem Teams nur begrenzte Erfahrungen haben
- *Embedded:* $\text{PM} = 3,6 \cdot \text{KDSI}^{1,20} \cdot M$
  - Projekte, eingebettet in komplexe Systeme aus Hardware, Software, Vorschriften und betriebliche Abläufe

KDSI = Kilo Delivered Source Instructions

M ergibt sich aus Einflussfaktoren

COCOMO

Damals:

- Wasserfallprozess
- nur Neuentwicklung
- Mainframes

Heute:

- Inkrementelle Entwicklung
- Wiederverwendung, COTS-Komponenten
- PCs
- Reengineering
- Code-Generierung

→ COCOMO II (Boehm u. a. 1995)

COCOMO II

Unterscheidung nach Phasen (Boehm u. a. 2000):

- Frühe Prototypenstufe
- Frühe Entwurfsstufe
- Stufe nach Architekturentwurf

Spätere Schätzung → höhere Genauigkeit

COCOMO II – Early prototyping level

Eingaben:

- Object Points (OP)
- Produktivität (PROD):

Erfahrung/Fähigkeiten der Entwickler	- -	-	o	+	++
Reife/Fähigkeiten der CASE-Tools	- -	-	o	+	++
PROD (NOP/Monat)	4	7	13	25	50

- Wiederverwendungsanteil *%reuse* in Prozent

Abgeleitete Größen:

- New Object Points (NOP): berücksichtigen Wiederverwendung
  $NOP = \text{OP} \cdot (100 - \%\text{reuse})/100$
- Aufwand in Personenmonaten $\text{PM} = \text{NOP}/\text{PROD}$

Unterstützt Prototypen, Wiederverwendung

COCOMO II – Early design level

- Schätzung basiert auf Function Points (LOCs werden daraus abgeleitet)
- Personenmonate $PM_{NS} = A \cdot \text{KLOC}^E \cdot EM + \text{PM}_\text{m}$ bei nominalem Zeitplan
- $A = 2,94$ in initialer Kalibrierung
- Exponent E:
  - 5 Faktoren w_i für Exponent E (5 = sehr klein, 0 = sehr groß): Erfahrung mit Domäne, Flexibilität des Entwicklungsprozesses, Risikomanagement, Teamzusammenhalt, Prozessreife
  - $E = B + \sum w_i/100$ mit $B = 1.01$
- Effort Multiplier EM:
  - 7 lineare Einflussfaktoren (6 Stufen, Standard: 1.00, in Tabelle nachschlagen): Produktgüte und -komplexität, Plattformkomplexität, Fähigkeiten des Personals, Erfahrung des Personals, Zeitplan, Infrastruktur
  - $EM = \prod \textit{Effort-Multiplier}_\text{i}$
- Korrekturfaktor PM_m bei viel generiertem Code (höhere Produktivität; nicht weiter diskutiert hier)

Faktoren für Exponent E I

- Erfahrung mit Anwendungsbereich (PREC)
  - → Erfahrung mit vorliegendem Projekttyp
  - 5 keine Erfahrung
  - 0 vollständige Vertrautheit
- Entwicklungsflexibilität (FLEX)
  - → Grad der Flexibilität im Entwicklungsprozess
  - 5 Prozess vom Kunden fest vorgegeben
  - 0 Kunde legt nur Ziele fest
- Risikomanagement (RESL)
  - → Umfang der durchgeführten Risikoanalyse
  - 5 keine Risikoanalyse
  - 0 vollständige und genaue Risikoanalyse

Faktoren für Exponent E II

- Teamzusammenhalt (TEAM)
  - → Vertrautheit und Eingespieltheit des Entwicklungsteams
  - 5 schwierige Interaktionen
  - 0 integriertes und effektives Team ohne Kommunikationsprobleme
- Prozessreife (EPML)
  - → Reife des Entwicklungsprozesses (z.B. CMM);
  - → beabsichtigt: gewichteter Anteil der mit „ja“ beantworteten Fragen im CMM-Fragebogen
  - → pragmatisch: CMM-Level
  - 5 niedrigster CMM-Level
  - 0 höchster CMM-Level

Effort Multiplier RCPX: Product Reliability and Complexity

RELY	Required reliability
DOCU	Documentation match to life-cycle needs
CPLX	Product Complexity
DATA	Data base size

Grad:	very low	low	nominal	high	very high	extra high
Punkte:	1	2	3	4	5	6
RCPX						
RELY	very little	little	some	basic	strong	
DOCU	very little	little	some	basic	strong	
CPLX	very little	little	some	basic	strong	very strong
DATA		small	moderate	large	very large	

$\sum$ Punkte:	5,6	7,8	9–11	12	13–15	16–18	19–21
EM_{RPCX}	0.49	0.60	0.83	1.00	1.33	1.91	2.72

Effort Multiplier PDIF: Platform Difficulty

TIME	Execution time constraints
STOR	Main storage constraints
PVOL	Platform volatility

Grad:	low	nominal	high	very high	extra high
Punkte:	2	3	4	5	6
PDIF					
TIME		≤50%	≤65%	≤80%	≤90%
STORE		≤50%	≤65%	≤80%	≤90%
PVOL	very stable	stable	somewhat volatile	volatile	

$\sum$ Punkte:	8	9	10–12	13–15	16,17
EM_{PDIF}	0.87	1.00	1.29	1.81	2.61

Effort Multiplier PERS: Personnel Capability

ACAP	Analyst capability (gemessen als Perzentil)
PCAP	Programmer capability (gemessen als Perzentil)
PCON	Personnel continuity (gemessen durch Personalfluktuation)

Grad:	very low	low	nominal	high	very high
Punkte:	1	2	3	4	5
PERS					
ACAP	15%	35%	55%	75%	90%
PCAP	15%	35%	55%	75%	90%
PCON	48%	24%	12%	6%	3%

$\sum$ Punkte:	3,4	5,6	7,8	9	10,11	12,13	14,15
EM_{PERS}	2.12	1.62	1.26	1.00	0.83	0.63	0.50

Effort Multiplier PERS: Personnel Capability

ACAP Analyst capability (gemessen als Perzentil)
PCAP Programmer capability (gemessen als Perzentil)
PCON Personnel continuity (gemessen durch Personalfluktuation)

Grad:	very low	low	nominal	high	very high		
Punkte:	1	2	3	4	5		
PERS							
ACAP	15%	35%	55%	75%	90%		
PCAP	15%	35%	55%	75%	90%		
PCON	48%	24%	12%	6%	3%		
$\sum$ Punkte:	3,4	5,6	7,8	9	10,11	12,13	14,15
EM_{PERS}	2.12	1.62	1.26	1.00	0.83	0.63	0.50

A percentile rank is the proportion of scores in a distribution that a specific score is greater than or equal to. For instance, if you received a score of 95 on a math test and this score was greater than or equal to the scores of 88% of the students taking the test, then your percentile rank would be 88. You would be in the 88th percentile.

Effort Multiplier PREX: Personnel Experience

AEXP Applications experience
PLEX Platform experience
LTEX Language/tool experience

Grad:	very low	low	nominal	high	very high
Punkte:	1	2	3	4	5
PREX					
AEXP	$\leq$2 Mo.	6 Mo.	1 J.	3 J.	6 J.
PLEX	$\leq$2 Mo.	6 Mo.	1 J.	3 J.	6 J.
LTEX	$\leq$2 Mo.	6 Mo.	1 J.	3 J.	6 J.

$\sum$ Punkte:	3,4	5,6	7,8	9	10,11	12,13	14,15
EM_{PREX}	1.59	1.33	1.22	1.00	0.87	0.74	0.62

Effort Multiplier FCIL: Facilities

TOOL Use of Software Tools
SITE Multisite Development

Grad:	very low	low	nominal	high	very high		
Punkte:	1	2	3	4	5	6	
FCIL							
TOOL	(1)	(2)	(3)	(4)	(5)		→ nächste Folie
SITE	(1)	(2)	(3)	(4)	(5)	(6)	→ nächste Folie

$\sum$ Punkte:	2	3	4,5	6	7,8	9,10	11
EM_{FCIL}	1.43	1.30	1.10	1.00	0.87	0.73	0.62

TOOL und SITE

TOOL:

1. Editor, Compiler, Debugger
2. einfaches CASE-Werkzeug, schlechte Integration
3. Basis-Life-Cycle-Tools moderat integriert
4. weitergehende, reife Life-Cycle-Tools moderat integriert
5. weitergehende, proaktive Life-Cycle-Tools gut integriert mit Prozessen, Methoden und Wiederverwendung

SITE:

1. Telefon prinzipiell vorhanden und Post
2. individuelles Telefon und Fax
3. E-Mail (niedrige Bandbreite)
4. elektronische Kommunikation mit großer Bandbreite
5. elektronische Kommunikation mit großer Bandbreite, gelegentliche Videokonferenzen
6. Interaktive Multimedia

Effort Multiplier SCED: Schedule

Es besteht Notwendigkeit, den Zeitplan zu straffen bzw. es wird mehr Zeit als notwendig eingeräumt.

SCED = Verkürzung bzw. Verlängerung des nominalen Zeitplans.

	75%	85%	100%	130%	160%
EM_{SCED}	1.43	1.14	1.00	1.00	1.00

Effort Multiplier SCED: Schedule

Es besteht Notwendigkeit, den Zeitplan zu straffen bzw. es wird mehr Zeit als notwendig eingeräumt.

SCED = Verkürzung bzw. Verlängerung des nominalen Zeitplans.

	75%	85%	100%	130%	160%
EM_{SCED}	1.43	1.14	1.00	1.00	1.00

This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in the later phases of development because more issues are left to be determined due to lack of time to resolve them earlier. A schedule compress of 74% is rated very low. A stretch-out of a schedule produces more effort in the earlier phases of development where there is more time for thorough planning, specification and validation. A stretch-out of 160% is rated very high.
Stretch-outs (i.e., SCED $>$ 100) do not add or decrease effort. Their savings bacause of smaller team size are generally balanced by the need to carry project administrative functions over a longer period of time.

Rechenbeispiel

Nominaler Aufwand

Personenmonate $PM_{NS} = A \cdot \text{KLOC}^E \cdot EM$ mit $EM_{SCED} = 1,0$

mit $A = 2,94$ und $E = B + \sum w_i/100$ mit $B = 1,01$.

Annahme: es herrschen einfache Verhältnisse:

$\rightarrow \forall i : w_i = 0 \Rightarrow E = 1,01$ (bester Fall)

$\rightarrow$ nominale Effort-Multiplier = 1,00 (Normalfall) $\Rightarrow EM = 1,00$

Geschätzte Lines-of-Code = 100.000

$\rightarrow PM_{NS} = 2,94 \times 100^{1,01} \times 1,0 = 307,86$ Monate $\approx 25\frac{1}{2}$ Jahre

Entwicklungsdauer

Nominale Entwicklungsdauer (Kalenderzeit in Monaten)

$TDEV_{NS} = C \times PM_{NS}^{D+0,2\times(E-B)}$

mit $C = 3,67$ und $D = 0,28$.

Beispiel: $TDEV_{NS} = 3,67 \times 307,86^{0,28+0,2\times 0} = 18,26$

Anzahl Entwickler

$N = PM_{NS}/TDEV_{NS}$

Beispiel: $N = 307,86/18,26 \approx 17$

Verkürzte Entwicklungsdauer

Chef: „Wieso 18 Monate? Geht das nicht schneller?“

PM_{NS} geht von $SCED = 1,0$ aus.

Abweichung von der nominalen Entwicklungdauer

$TDEV = TDEV_{NS} \times SCED/100$

Wir verkürzen auf 75%:

$TDEV = 18,26 \times 75/100 = 10,3$ Monate

Chef: „Super!“

Wir setzen $SCED = 75$ in PM-Formel ein.

$PM = 2,94 \times 100^{1,01} \times 1,0 \times 1,43 = 440,23$

Erhöhung des Aufwands: $440,23/307,86 = 1,43$

Chef: „43% mehr Kosten? Seid Ihr wahnsinnig?“

COCOMO II – Post-architecture level

Berücksichtigt:

- Auswirkungen erwarteter Änderungen von Anforderungen
- Ausmaß/Aufwand der möglichen Wiederverwendung
  - Aufwand für Entscheidung, ob Wiederverwendung
  - Aufwand für das Verstehen existierenden Codes
  - Aufwand für Anpassungen
- 17 verfeinerte lineare Einflussfaktoren
- Schätzung basiert auf LOC

COCOMO II – Post-architecture level

Produktgüte/-kompl $\rightarrow$ Verlässlichkeit, Datenbasisgröße, Komplexität, Dokumentation

Plattformkomplexität $\rightarrow$ Laufzeit-, Speicherbeschränkungen, Plattformdynamik

Fähigkeiten Personal $\rightarrow$ Fähigkeiten der Analysten/Entwickler, Kontinuität des Personals

Erfahrung Personal $\rightarrow$ Domänenerfahrung der Analysten/Entwickler, Erfahrung mit Sprache und Werkzeugen

Infrastruktur $\rightarrow$ Tools, verteilte Entwicklung+Kommunikation

Einflussfaktoren (Cost Drivers) für Cocomo-2

	- -	-	o	+	++	+++
Product Attributes						
RELY – Required reliability	0.82	0.92	1.00	1.10	1.26	
DATA – Data base size		0.90	1.00	1.14	1.28	
CPLX – Product Complexity	0.73	0.87	1.00	1.17	1.34	1.74
RUSE – Required Reusability		0.95	1.00	1.07	1.15	1.24
DOCU – Doc. match to life-cycle needs	0.81	0.91	1.00	1.11	1.23	
Computer Attributes						
TIME – Execution time constr.			1.00	1.11	1.29	1.63
STOR – Main storage constr.			1.00	1.05	1.17	1.46
PVOL – Platform volatility		0.87	1.00	1.15	1.30	

Einflussfaktoren (Cost Drivers) für Cocomo-2

	- -	-	o	+	++	+++
Personell attributes						
ACAP – Analyst capability	1.42	1.19	1.00	0.85	0.71	
PCAP – Programmer capability	1.34	1.15	1.00	0.88	0.76	
AEXP – Applications experience	1.22	1.10	1.00	0.88	0.81	
PLEX – Platform experience	1.19	1.09	1.00	0.91	0.85	
LTEX – Language/tool exp.	1.20	1.09	1.00	0.91	0.84	
Project attributes						
TOOL – Use of software tools	1.17	1.09	1.00	0.90	0.78	
SITE – Multisite development	1.22	1.09	1.00	0.93	0.86	0.80
SCED – Required dev. schedule	1.43	1.14	1.00	1.00	1.00	

Zusammenfassung

- alle Schätzungen basieren auf Erfahrung
- kontinuierlich schätzen
- verschiedene Techniken anwenden

Wiederholungs- und Vertiefungsfragen

- Welche Möglichkeiten zur Schätzung von Aufwand und Kosten für die Software-Entwicklung gibt es?
- Wann wird geschätzt?
- Erläutern Sie die Function-Point-Methode.
- Was sind Adjusted Function Points im Unterschied zu Unadjusted Function Points?
- Wie errechnet sich der Aufwand aus den Function Points?
- Erläutern Sie die CoCoMo-Methode (zweite Version).
- Geben Sie die Formel für den Aufwand wieder. Was bedeuten die verschiedenen Parameter?
- Wozu die Unterscheidung in die verschiedene Stufen (Level)?
- Wie errechnet sich die Entwicklungsdauer aus dem Aufwand?
- Wie wird eine Verkürzung der nominalen Entwicklungsdauer behandelt?
- Vergleichen Sie die Function-Point-Methode mit CoCoMo.

1 Albrecht 1979 Albrecht, Alan: Measuring application development productivity. Monterey, CA, USA, 1979

2 Balzert 1997 Balzert, Helmut: *Lehrbuch der Software-Technik*. Spektrum Akademischer Verlag, 1997. – ISBN 3827400651

3 Banker u. a. 1991 Banker, R. ; Kauffmann, R. ; Kumar, R.: An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment. In: *Journal of Management Information Systems* 8 (1991), Nr. 3, S. 127–150

4 Boehm 1981 Boehm, B.: *Software Engineering Economics*. Prentice Hall, 1981

5 Boehm u. a. 1995 Boehm, Barry ; Clark, Bradford ; Horowitz, Ellis ; Madachy, Ray ; Shelby, Richard ; Westland, Chris: Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. In: *Annals of Software Engineering* (1995)

6 Boehm u. a. 2000 Boehm, Barry W. ; Abts, Chris ; Brown, A. W. ; Chulani, Sunita ; Clark, Bradford K. ; Horowitz, Ellis ; Madachy, Ray ; Reifer, Donald ; Steece, Bert: *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000

7 Jones 1995 Jones, Capers: Backfiring: Converting Lines of Code to Function Points. In: *IEEE Computer* 28 (1995), November, Nr. 11, S. 87–88

8 Jones 1996 Jones, Capers: Software Estimating Rules of Thumb. In: *IEEE Computer* 29 (1996), March, Nr. 3, S. 116–118

9 Kauffman und Kumar 1993 Kauffman, R. ; Kumar, R.: Modeling Estimation Expertise in Object Based ICASE Environments / Stern School of Business, New York University. Januar 1993. – Report

0 Kemerer 1987 Kemerer, Chris F.: An Empirical Validation of Software Cost Estimation Models. In: *Comm. ACM* 30 (1987), May, Nr. 5

1 Kemerer und Porter 1992 Kemerer, Chris F. ; Porter, Benjamin S.: Improving the Reliability of Function Point Measurement: An Empirical Study. In: *TSE* 18 (1992), Nov., Nr. 11

2 Symons 1988 Symons, C. R.: Function Point Analysis: Difficulties and Improvements. In: *TSE* 14 (1988), Jan., Nr. 1, S. 2–11