Softwaretechnik

Prof. Dr. Rainer Koschke

Fachbereich Mathematik und Informatik
Arbeitsgruppe *Softwaretechnik*
Universität Bremen

Sommersemester 2009

Überblick I

1. Kosten- und Aufwandsschätzung

Personalkosten

Neu- / Weiterentwicklung?

Technische Faktoren

- technisch
- Anwendungsdomäne

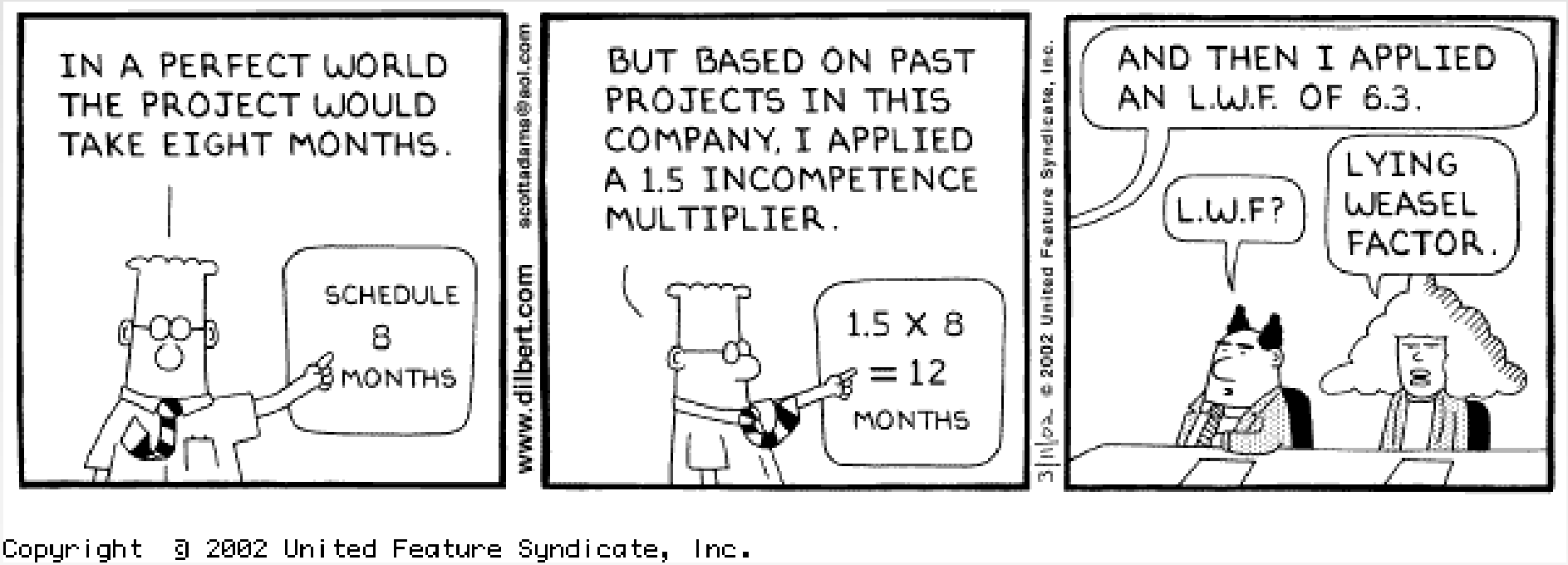
Erfahrung / Qualität Entwickler

Anforderungen

- Dokumente

Kosten- und Aufwandsschätzung

Kostenschätzung



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

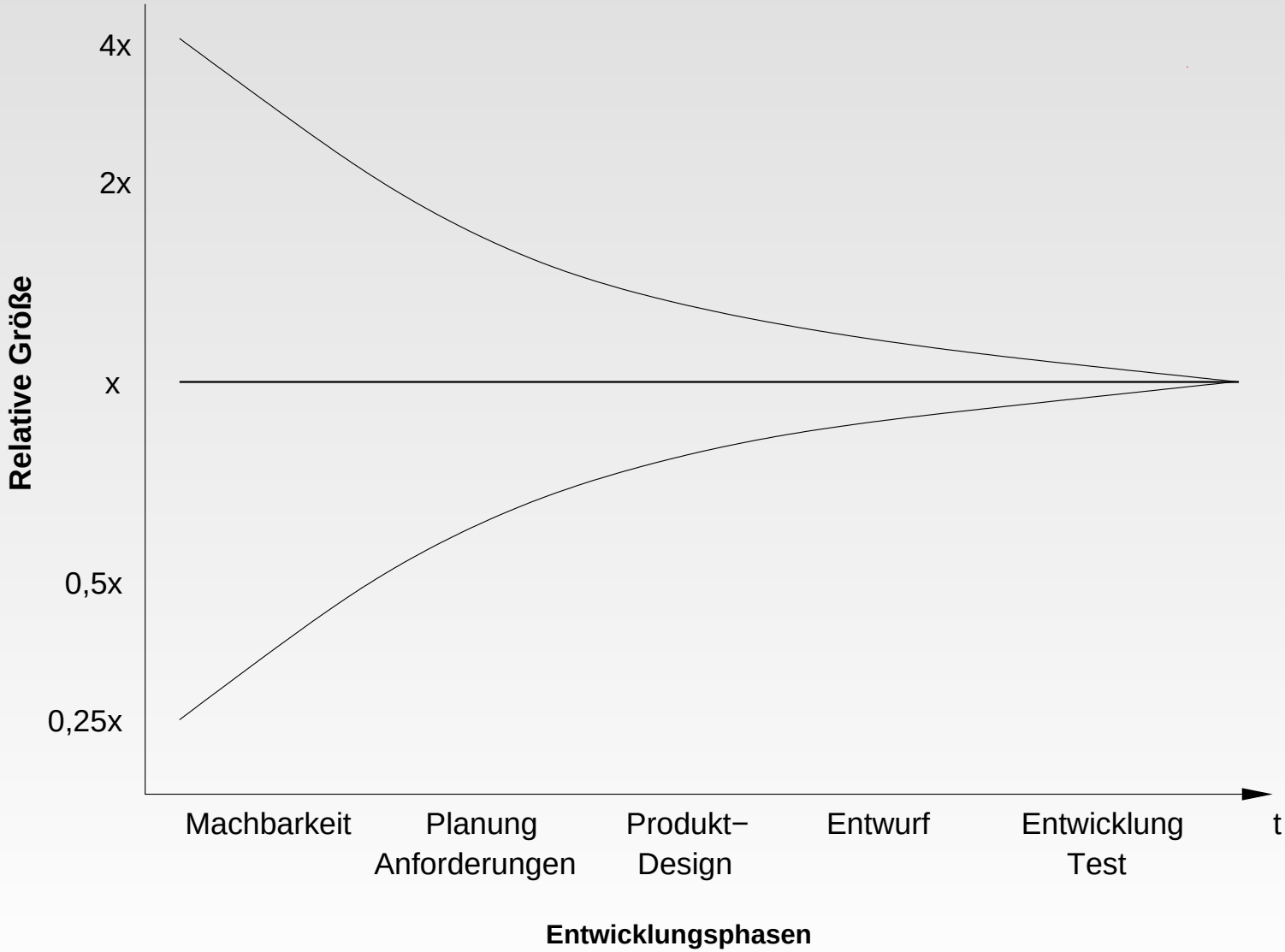
Kostenschätzung

Ansätze:

- Expertenschätzung — Delphi
- Berechnung aus früh bekannten Größen (algorithmische Schätzung)
  - COCOMO: Anzahl Codezeilen
  - Schätzung auf Basis von 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

Boehm (1981)



Gegeben ist ein Puzzle mit 500 Teilen. Entwickeln Sie eine Software, die dieses Puzzle löst. Wie lange werden Sie für die Entwicklung dieser Software brauchen?

Literatur zur Function-Point-Methode

Poensgen und Bock (2005)

Function-Point-Methode

Benötigt für frühe Kostenschätzung: Maß für den Umfang der Software.

Function-Points:

- Messen des **Umfangs**[1] einer zu erstellenden Software aus Benutzersicht
- Eingabe: Lastenheft

[1] nicht der Kosten der Erstellung

Function-Point-Methode

Benötigt für frühe Kostenschätzung: Maß für den Umfang der Software.

Function-Points:

- Messen des **Umfangs**[1] einer zu erstellenden Software aus Benutzersicht
- Eingabe: Lastenheft

Einsatz:

- Umrechnung des Umfangs in personellen Aufwand
- Vergleich und Bewertung von Systemen
- Messung von Produktivität, Benchmarking

[1] nicht der Kosten der Erstellung

Historische Entwicklung der Function-Point-Methode

- 1979: erste Veröffentlichung: Alan J. Albrecht (1979) (IBM)
- 1985: Veröffentlichung der „IBM-Kurve“ (IBM 1985): Zusammenhang von Aufwand und Function-Points für 54 Projekte
- 1990: Hype: fast alle großen Unternehmen probieren FPA aus
- 1995: Abkühlung des Interesses
  - Unerfahrenheit der Anwender
  - unrealistische Erwartungen
  - zu wenig Erfahrungsdaten vorhanden
- 1995-heute: wieder gesteigertes Interesse:
  - Benchmarking
  - Wirtschaftlichkeit (Function-Points versus Kosten)
  - Outsourcing
  - Offshoring
- Heute:
  - zahlreiche Varianten
  - IFPUG Int'l Function Point User Group

Function-Point-Methode – Vorgehen

1. Zähltyp festlegen: Neu-/Weiterentwicklung, bestehendes System
2. Systemgrenzen festlegen
3. Identifizieren der Elementarprozesse und deren Funktionstypen sowie der Datenbestände
4. Bewerten der Komplexität der Funktionstypen
5. Ermittlung der gewichteten Function Points
6. Verwendung; z.B. Ermittlung des Aufwands

Elementarprozess

Definition

Elementarprozess: atomare und einzigartige Aktivität des Systems aus Benutzersicht

Elementarprozess: Atomarizitätsprinzip

Definition

Atomarizitätsprinzip: Elementarprozess ist kleinste aus fachlicher Sicht sinnvolle, in sich abgeschlossene Aktivität

Bsp.: Erfassung einer Kundenadresse (auch über mehrere Bildschirmmasken)

Elementarprozess: Einmaligkeitsprinzip

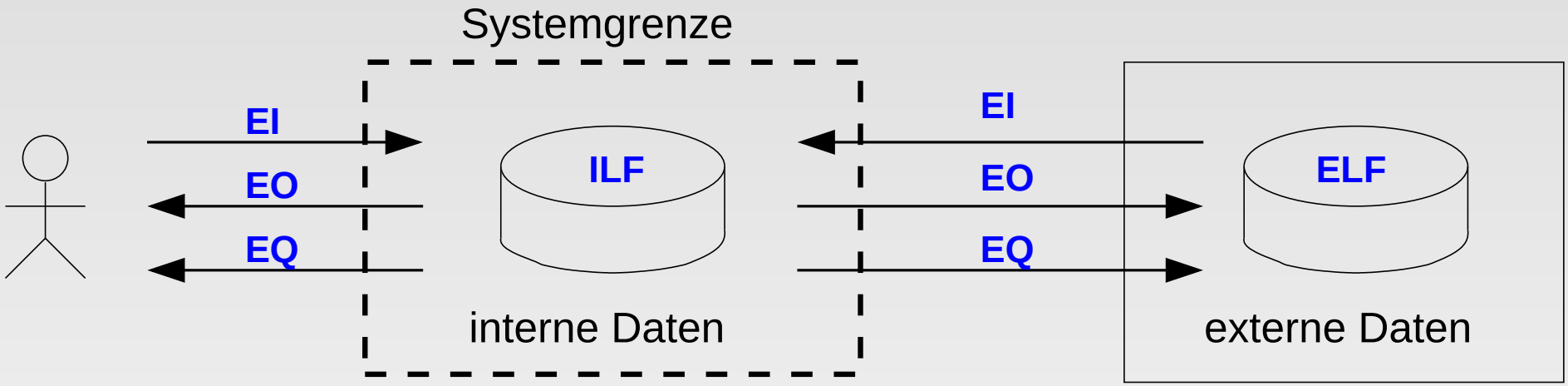
Definition

Einmaligkeitsprinzip: Elementarprozess gilt als einmalig (einzigartig), wenn er durch die ein- oder ausgegebenen Daten oder durch die Verarbeitungslogik unterscheidbar ist (aus Sicht des Anwenders); Unterscheidung durch

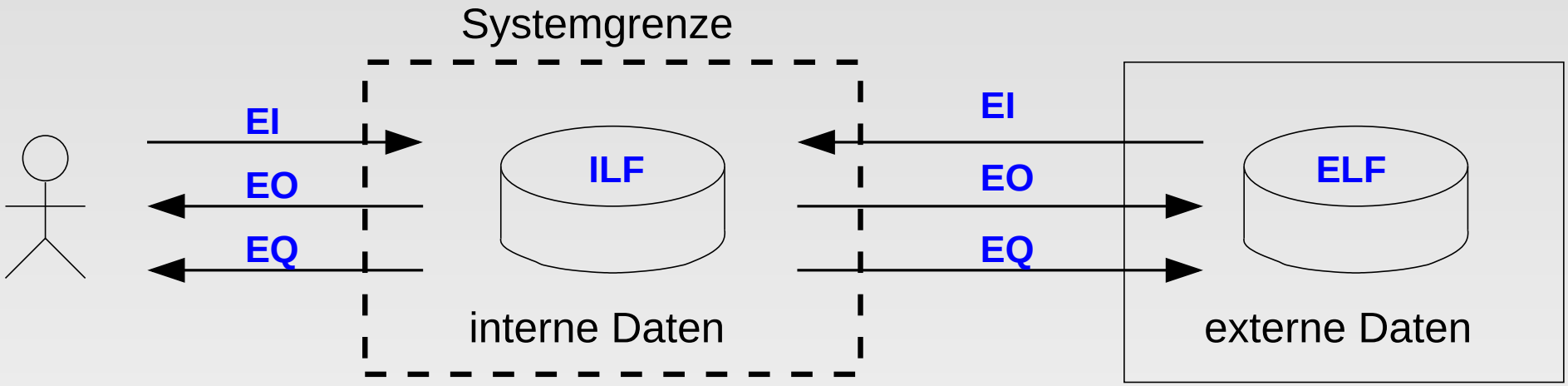
- besondere Verarbeitungslogik oder
- verarbeitete Felder der Datenbestände oder
- verarbeitete Datenbestände selbst

Bsp.: Berechnung des Krankenkassenbeitrags einer privaten Versicherung für Neu- bzw. Bestandskunden sind verschiedene Elementarprozesse

FP – Identifizieren der Funktionstypen



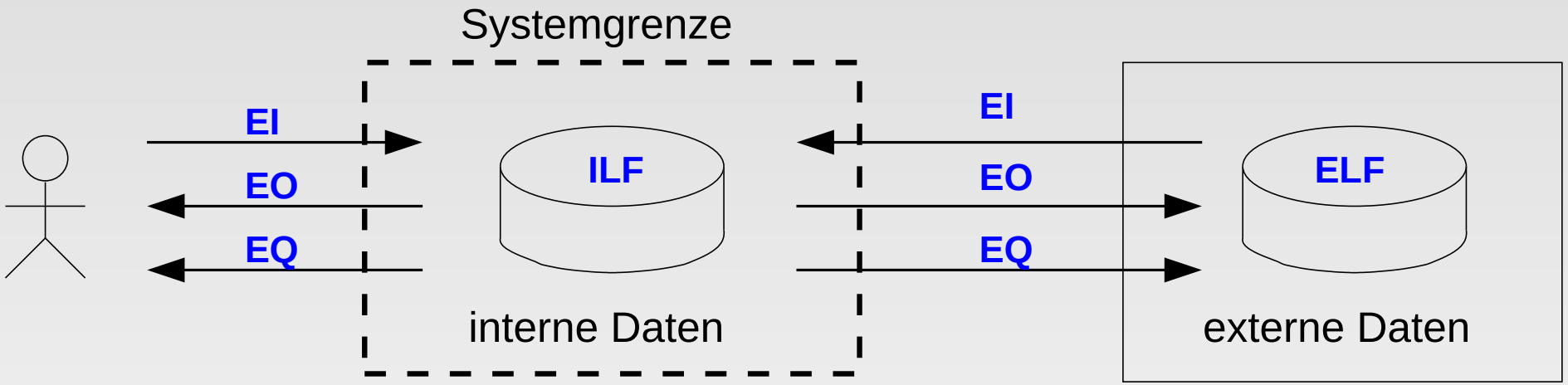
FP – Identifizieren der Funktionstypen



Funktionstypen von Elementarprozessen:

- Eingaben: EI (External Input; Eingabe für ILF)
- Ausgaben: EO (External Output; Ausgabe abgeleiteter Daten)
- Abfragen: EQ (External Inquiry; reine Abfrage von Daten ohne Berechnung/Ableitung)

FP – Identifizieren der Funktionstypen



Funktionstypen von Datenbeständen:

- Interner Datenbestand (ILF: Internal Logical File): fachliche Daten, die vom System selbst gepflegt werden (anlegen, ändern, löschen)
- Externer Datenbestand – auch: Referenzdatenbestand– (ELF: External Logical File): fachliche Daten, die vom System nur gelesen werden

Funktionstyp Eingabe

Unterscheidung der Funktionstypen auf Basis des Hauptzwecks eines Elementarprozesses

Definition

Eingabe: Hauptzweck: internen Datenbestand pflegen oder Systemverhalten ändern, wobei gilt:

- Daten oder Steuerinformationen kommen von außerhalb des Systems
- und mindestens ein ILF wird gepflegt, falls die Daten, die die Systemgrenze überqueren, keiner Steuerinformationen sind, die das Systemverhalten verändern
- und mindestens eine der folgenden Bedingungen ist erfüllt (Einzigartigkeit):
  - Verarbeitungslogik ist einzigartig gegenüber anderen Eingaben
  - andere Datenfelder als bei anderen Eingaben werden verwendet
  - andere Datenbestände als bei anderen Eingaben werden verwendet

Funktionstyp Ausgabe

Definition

Ausgabe:

- Hauptzweck: dem Anwender Informationen präsentieren
- Daten oder Steuerinformationen werden über Systemgrenze geschickt und
- Elementarprozess ist eindeutig (s.o.) und
- und mindestens eine der folgenden Bedingungen ist erfüllt (Abgrenzung zu Abfrage):
  - Verarbeitungslogik enthält mindestens eine mathematische Formel oder Berechnung
  - Verarbeitungslogik pflegt mindestens einen ILF
  - Verarbeitungslogik verändert das Systemverhalten

Funktionstyp Abfrage

Definition

Abfrage:

- Hauptzweck: dem Anwender Informationen präsentieren
- Daten oder Steuerinformationen werden über Systemgrenze geschickt und
- Elementarprozess ist eindeutig (s.o.) und
- und **alle** folgende Bedingungen sind erfüllt (Abgrenzung zu Ausgabe):
  - Elementarprozess bezieht Daten oder Steuerinformationen von einem ILF oder ELF
  - Verarbeitungslogik enthält keine mathematische Formel oder Berechnung
  - Verarbeitungslogik erzeugt keine abgeleiteten Daten
  - Verarbeitungslogik pflegt keinen ILF
  - Verarbeitungslogik verändert das Systemverhalten nicht

FP – Identifizieren von Funktionstypen

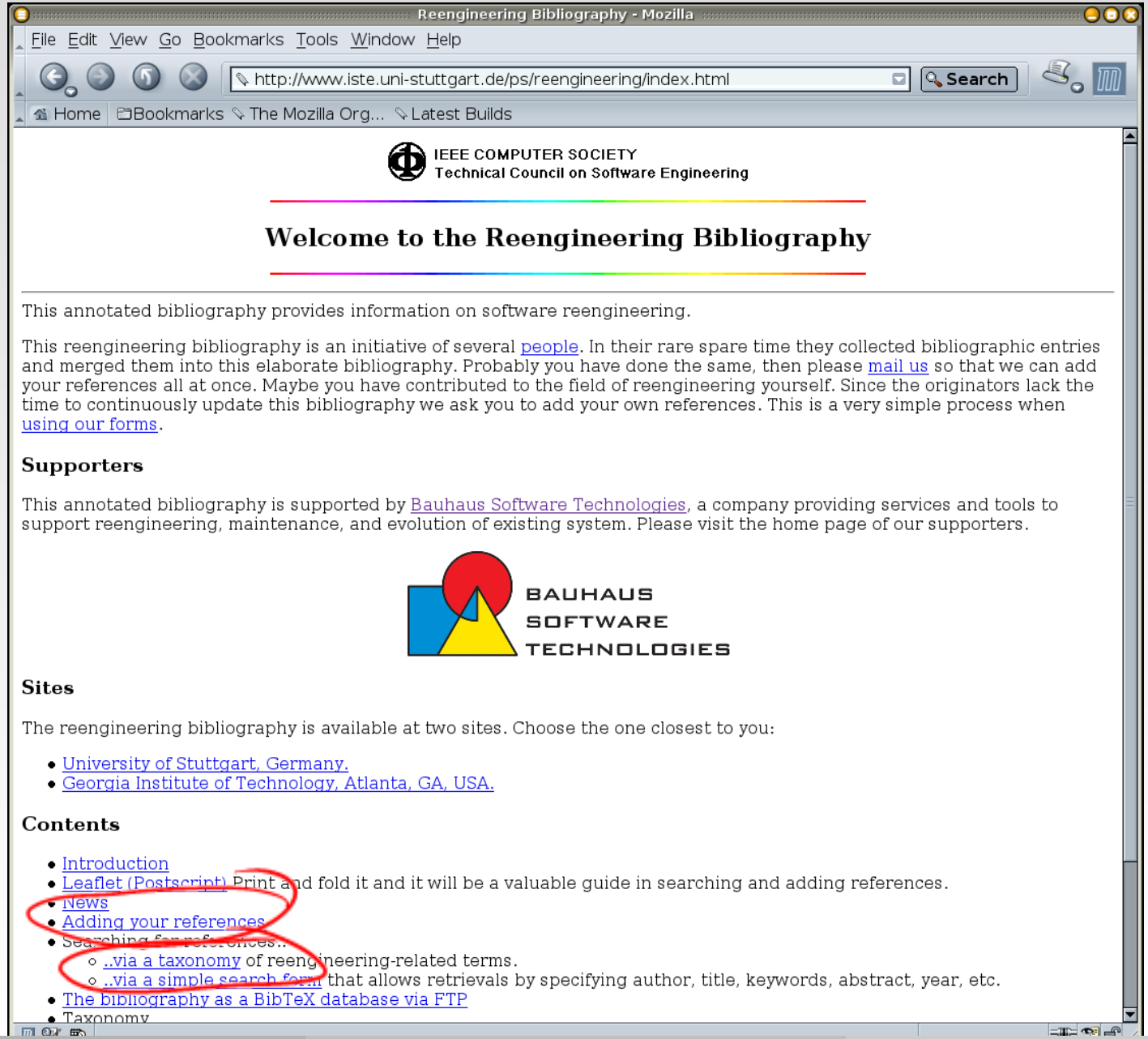
Schlüsselwörter geben Hinweise:

- 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

Gegeben ist ein Puzzle mit 500 Teilen. Entwickeln Sie eine Software, die dieses Puzzle löst. Wie lange werden Sie für die Entwicklung dieser Software brauchen?

- **das Puzzle ist dreidimensional**

Online-Bibliographie: Startseite



Beispielelementarprozesse und -datenbestände

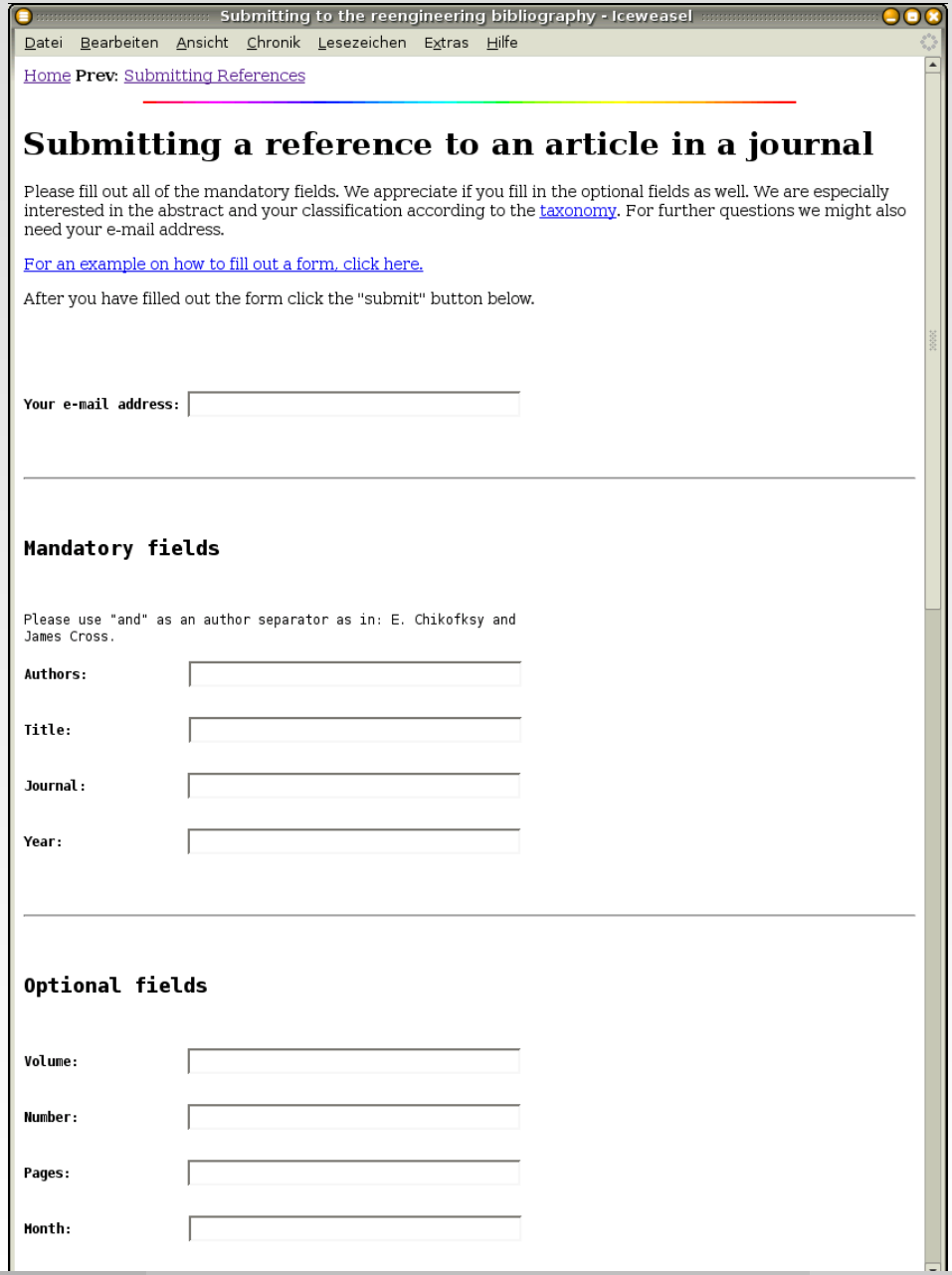
Elementarprozesse:

- Neuen Artikel einpflegen
- Suche über Taxonomie
- Artikel über Suchmaske abfragen

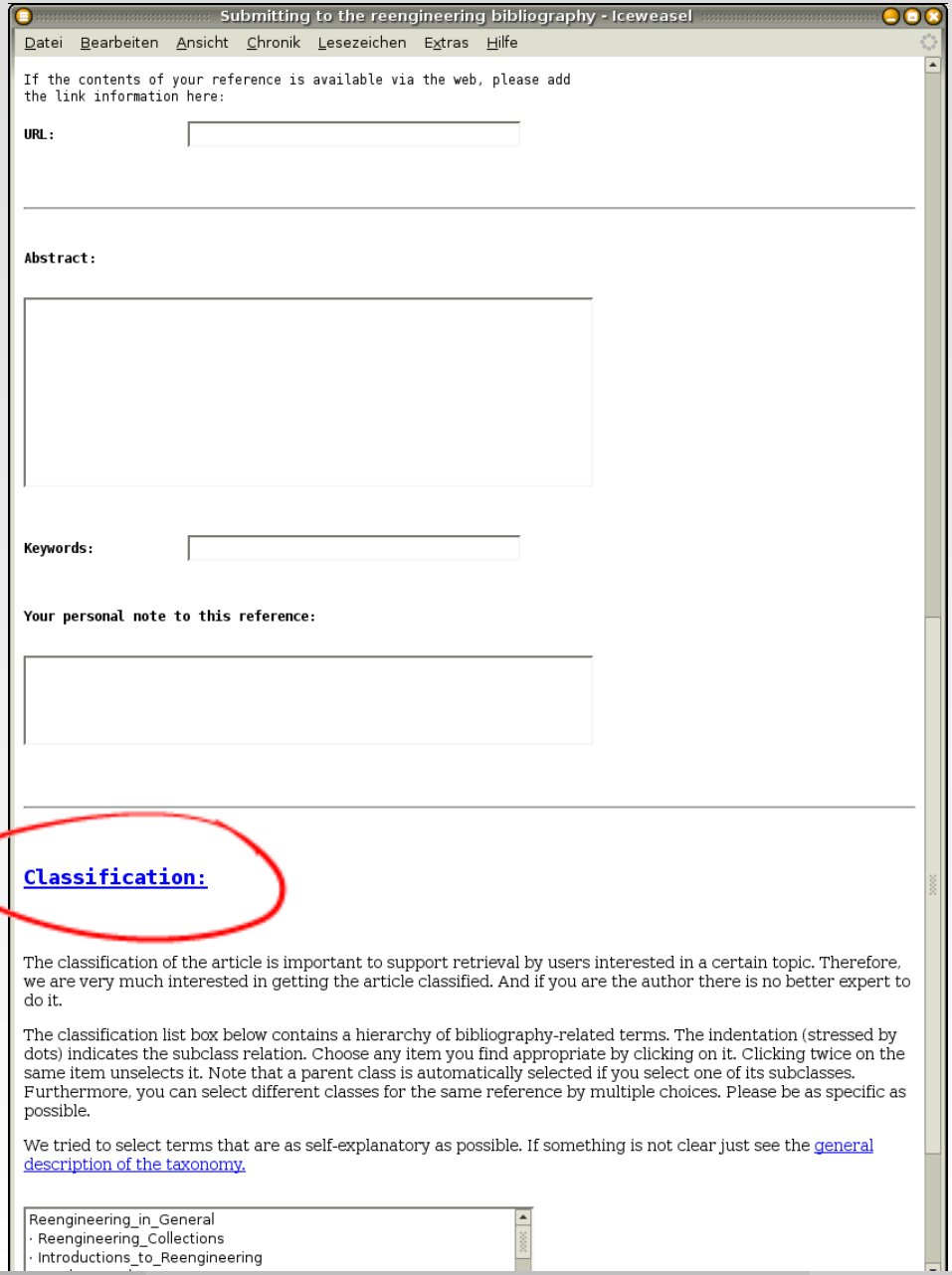
Datenbestände:

- Referenzen
- Taxonomie-Metadaten
- externer Datenbestand tritt hier nicht auf

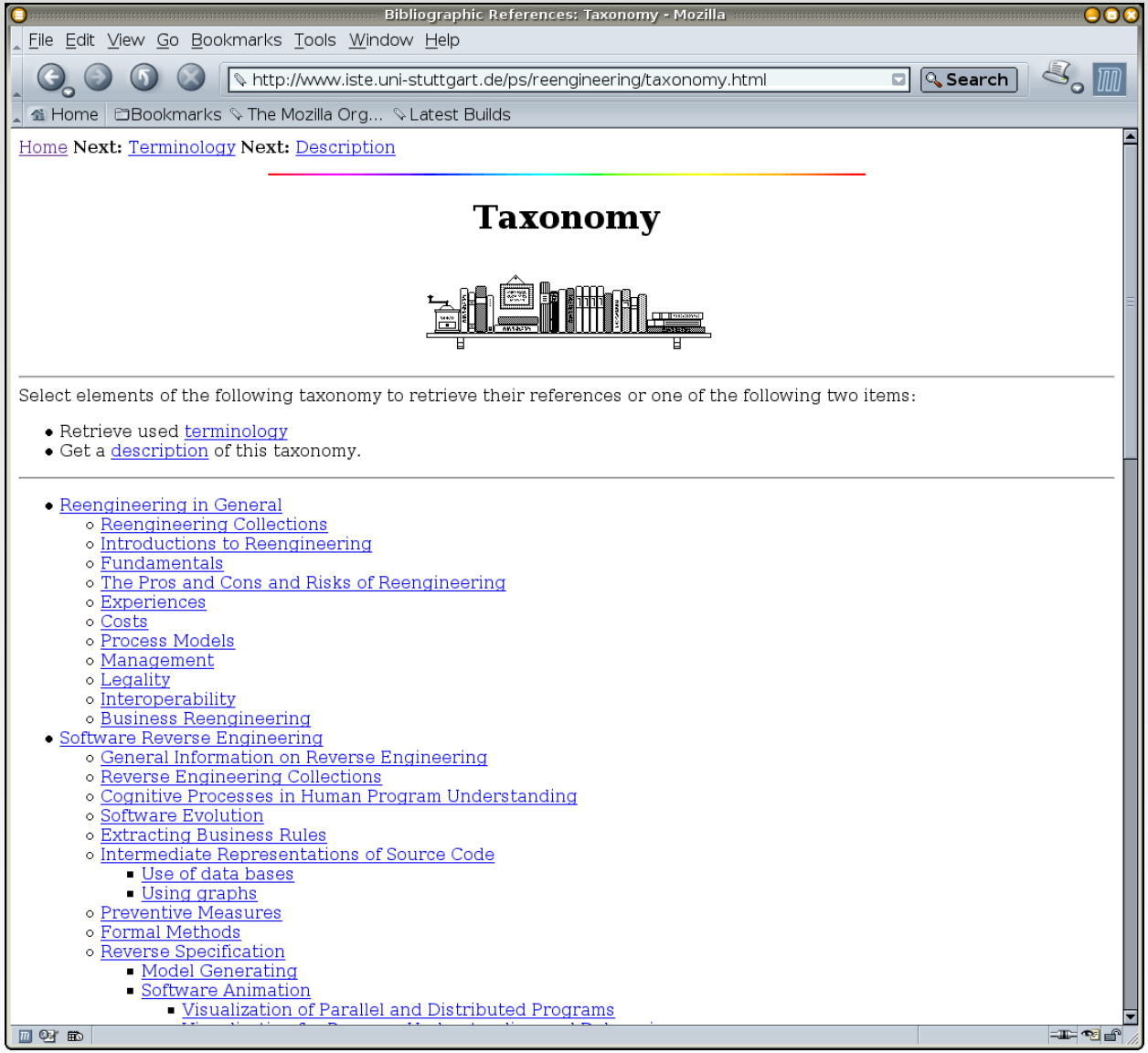
Online-Bibliographie: Neuen Artikel einpflegen



Online-Bibliographie: Neuen Artikel einpflegen

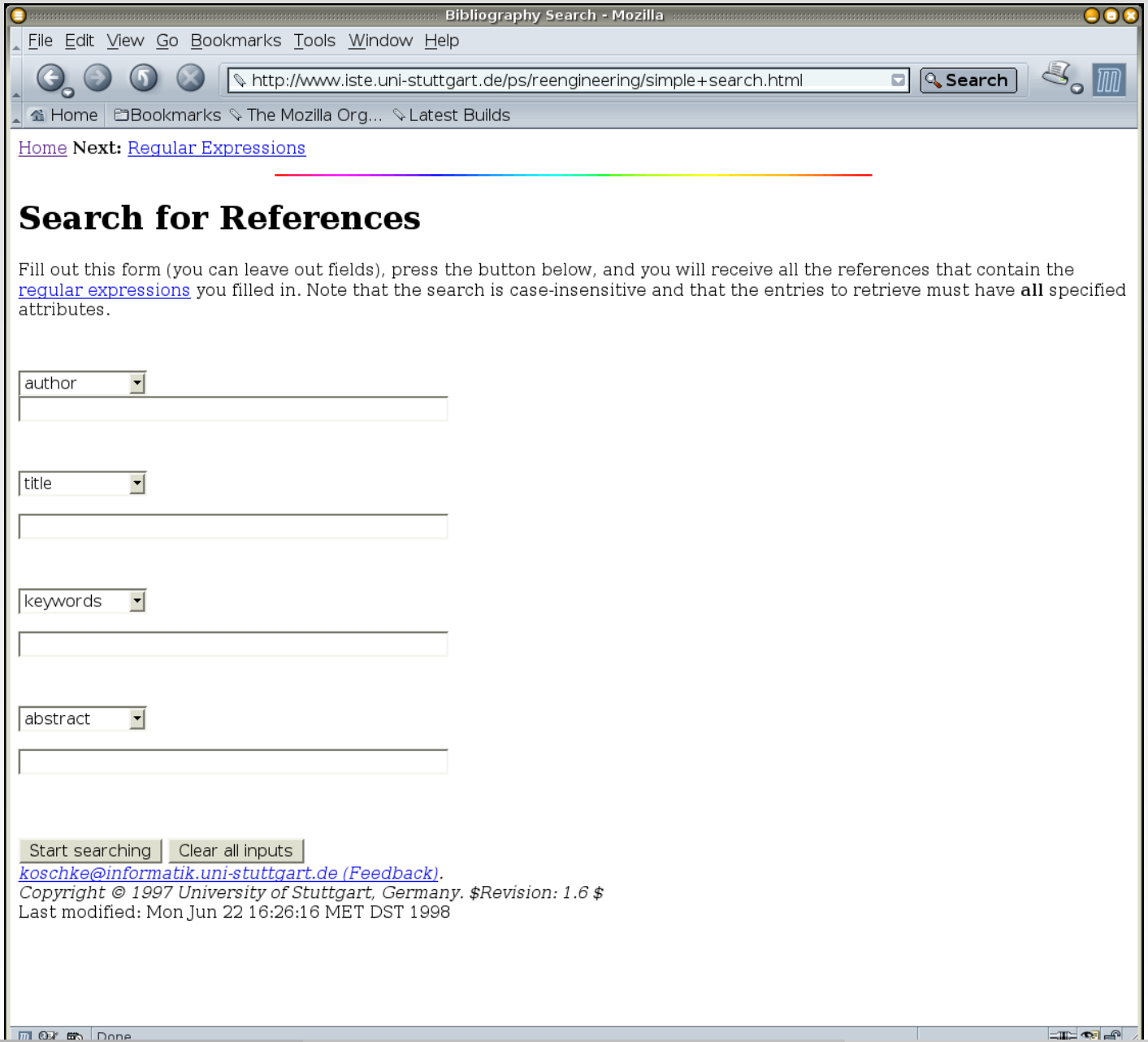


Online-Bibliographie: Taxonomiesuche



Annahme: Ein BibTeX-Eintrag habe max. 8 Einträge

Online-Bibliographie: Suche nach Eigenschaften



Beispiel: Online-Bibliographie

Elementarprozesse und Datenbestände der Online-Bibliographie:

- EI_1: Neuen Artikel einpflegen
- EQ_1: Beschreibung der Taxonomie
- EQ_2: Suche über Taxonomie
- EQ_3: Artikel über Suchmaske abfragen
- ILF_1: Referenzen
- ILF_2: Taxonomie-Metadaten

Function Points zusammenfassen

Unadjusted Function Points (UFP):

Parameter	Gewicht
EI	w_1
EO	w_2
EQ	w_3
ILF	w_4
ELF	w_5
UFP	$\sum w_i$

$\longrightarrow$ zu ermitteln: w_i

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

Gegeben ist ein Puzzle mit 500 Teilen. Entwickeln Sie eine Software, die dieses Puzzle löst. Wie lange werden Sie für die Entwicklung dieser Software brauchen?

- das Puzzle ist dreidimensional
- **die Puzzle-Teile haben alle dieselbe Farbe**

Beispiel: Komplexitätsgewichte w_i für ELF und ILF

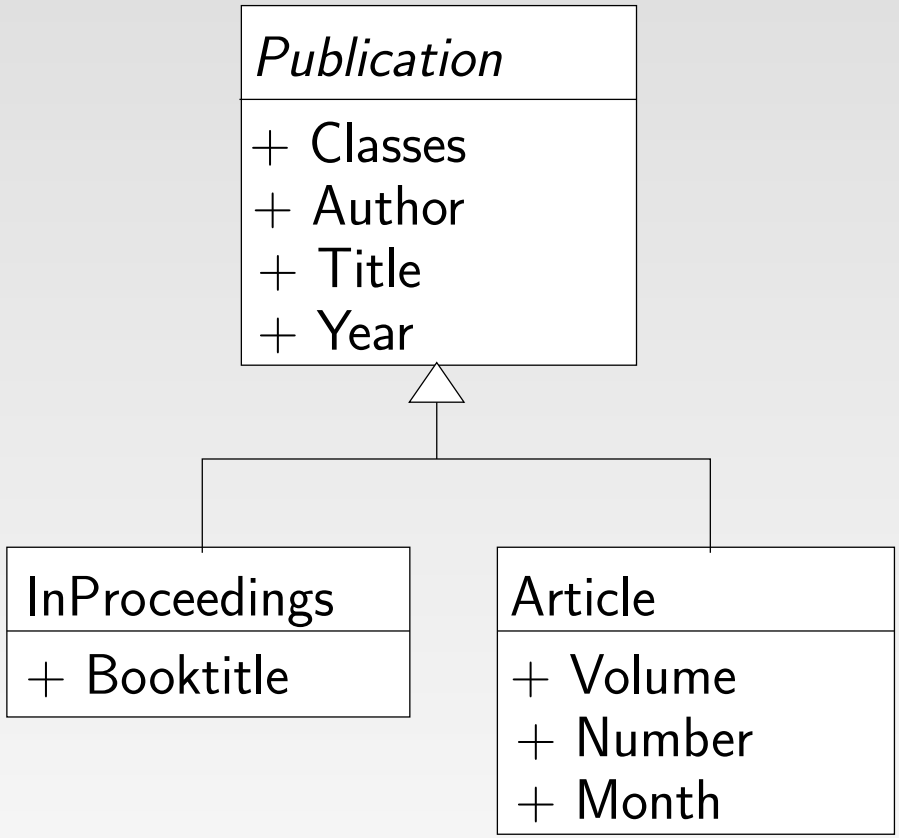
≈ Σ nicht abstrakte Klassen

Definition

Feldgruppe: 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-wiederholtes Feld



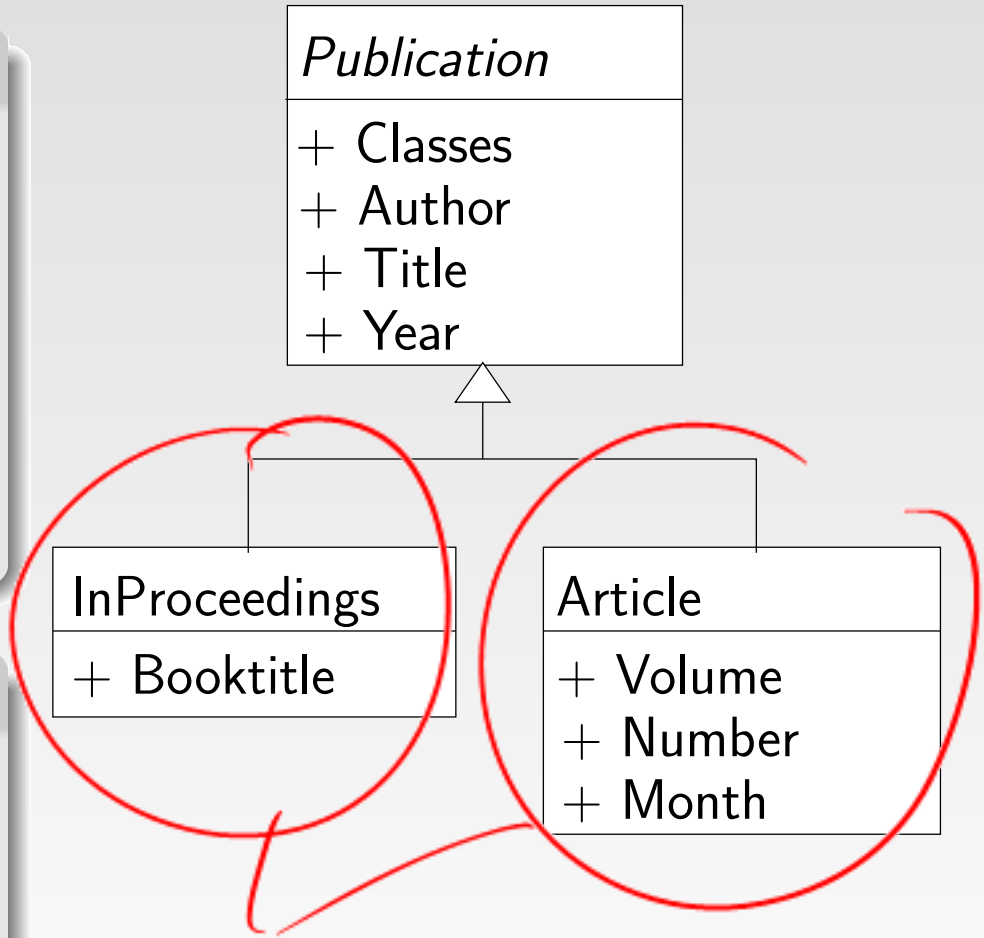
Beispiel: Komplexitätsgewichte w_i für ELF und ILF

Definition

Feldgruppe: 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-wiederholtes Feld



- RET: 2

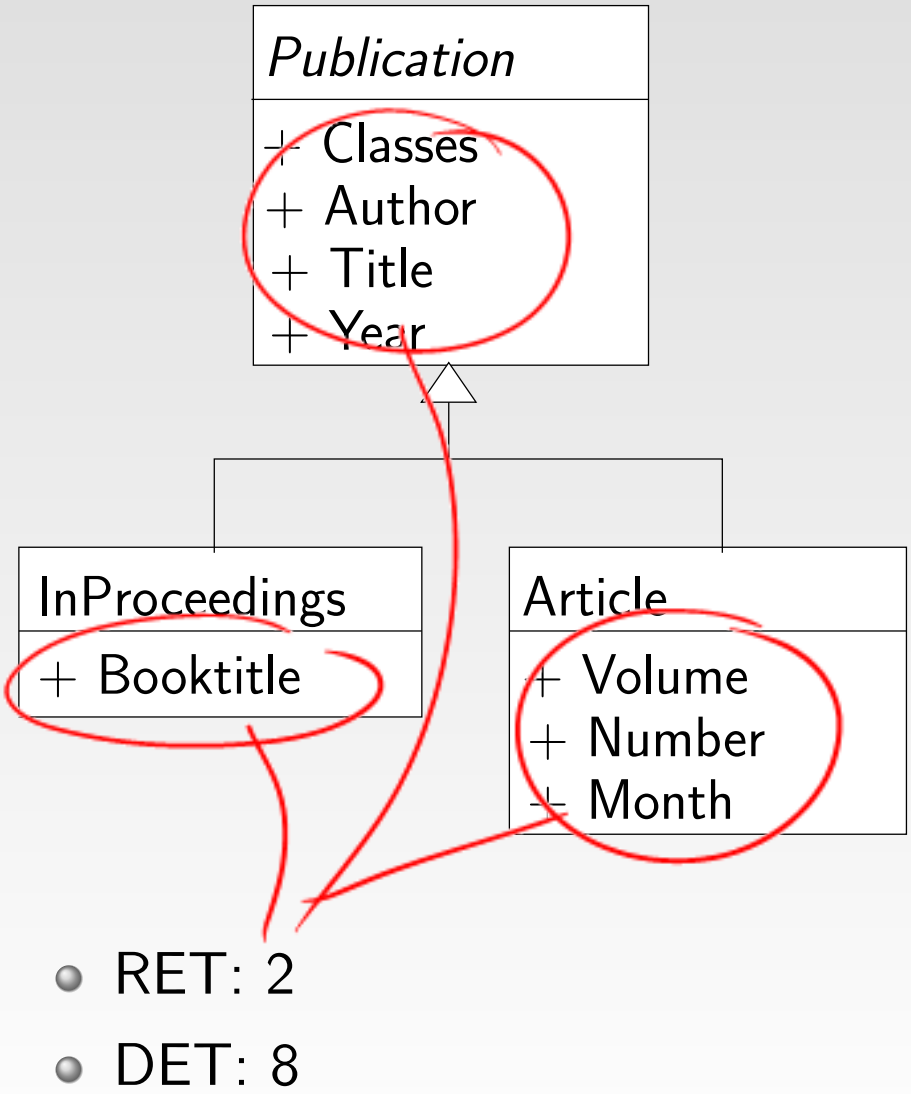
Beispiel: Komplexitätsgewichte w_i für ELF und ILF

Definition

Feldgruppe: 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-wiederholtes Feld



- RET: 2
- DET: 8

Function Points zusammenfassen

ILF_2: 1 Datenbestand für Taxonomie (Name, Beschreibung, Oberklasse)

Parameter	RET	DET	Gewicht
ILF_1	2	8	
ILF_2	1	3	
Parameter	FTR	DET	Gewicht
EI_1			
EQ_1			
EQ_2			
EQ_3			
UFP			

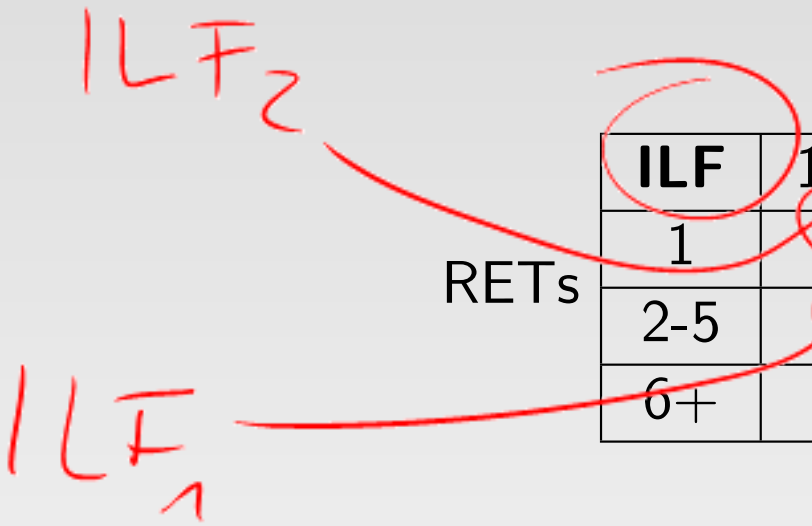
FP – Bestimmung der Komplexitätsgewichte w_i

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 – Werte der Komplexitätsgewichte w_i



DETs

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

DETs

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

Function Points zusammenfassen

Parameter	RET	DET	Gewicht
ILF_1	2	8	7
ILF_2	1	3	7
Parameter	FTR	DET	Gewicht
EI_1			
EQ_1			
EQ_2			
EQ_3			
UFP			

Gegeben ist ein Puzzle mit 500 Teilen. Entwickeln Sie eine Software, die dieses Puzzle löst. Wie lange werden Sie für die Entwicklung dieser Software brauchen?

- das Puzzle ist dreidimensional
- die Puzzle-Teile haben alle dieselbe Farbe
- **die Puzzle-Teile haben alle dieselbe Form**

Komplexitätsgewichte w_i für EI, EO, EQ

Definition

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

Komplexitätsgewichte w_i für EI, EO, EQ

Definition

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

Beispiel: der Kundenstammdatenbestand, der bei der Ausgabe von Kundendaten herangezogen wird

Komplexitätsgewichte w_i für EI, EO, EQ

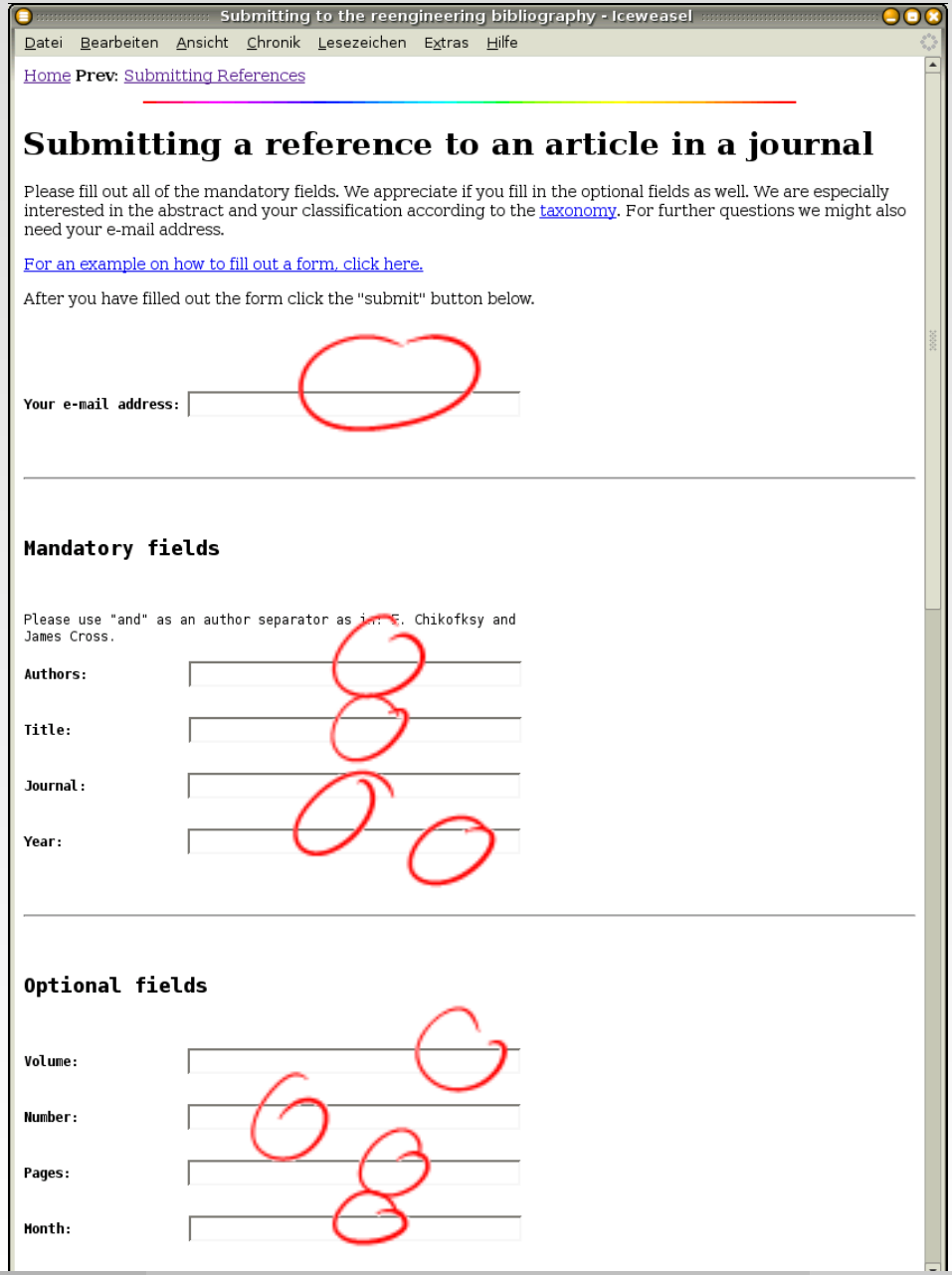
Definition

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

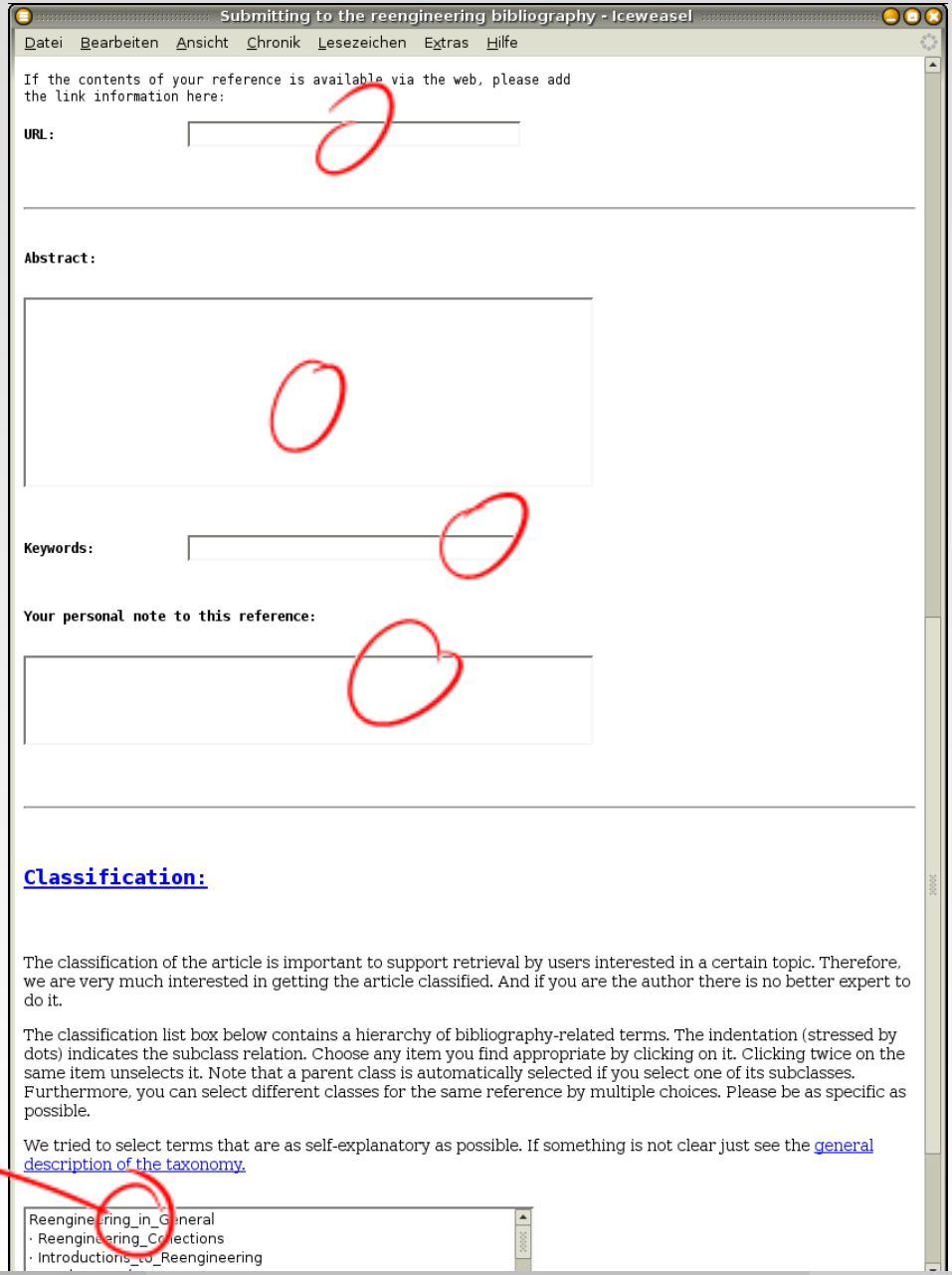
Beispiel: der Kundenstammdatenbestand, der bei der Ausgabe von Kundendaten herangezogen wird

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

Online-Bibliographie: Neuen Artikel einpflegen



Online-Bibliographie: Neuen Artikel einpflegen



Function Points zusammenfassen

- EI_1: Neuen Artikel einpflegen
- EQ_1: Beschreibung der Taxonomie
- EQ_2: Suche über Taxonomie
- EQ_3: Artikel über Suchmaske abfragen
- ILF_1: Referenzen
- ILF_2: Taxonomie-Metadaten

Parameter	RET	DET	Gewicht
ILF_1	2	8	7
ILF_2	1	3	7
Parameter	FTR	DET	Gewicht
EI_1	1	15	
EQ_1	1	1	
EQ_2			
EQ_3			
UFP			

FP – Werte der Komplexitätsgewichte w_i

DETs

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

DETs

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

DETs

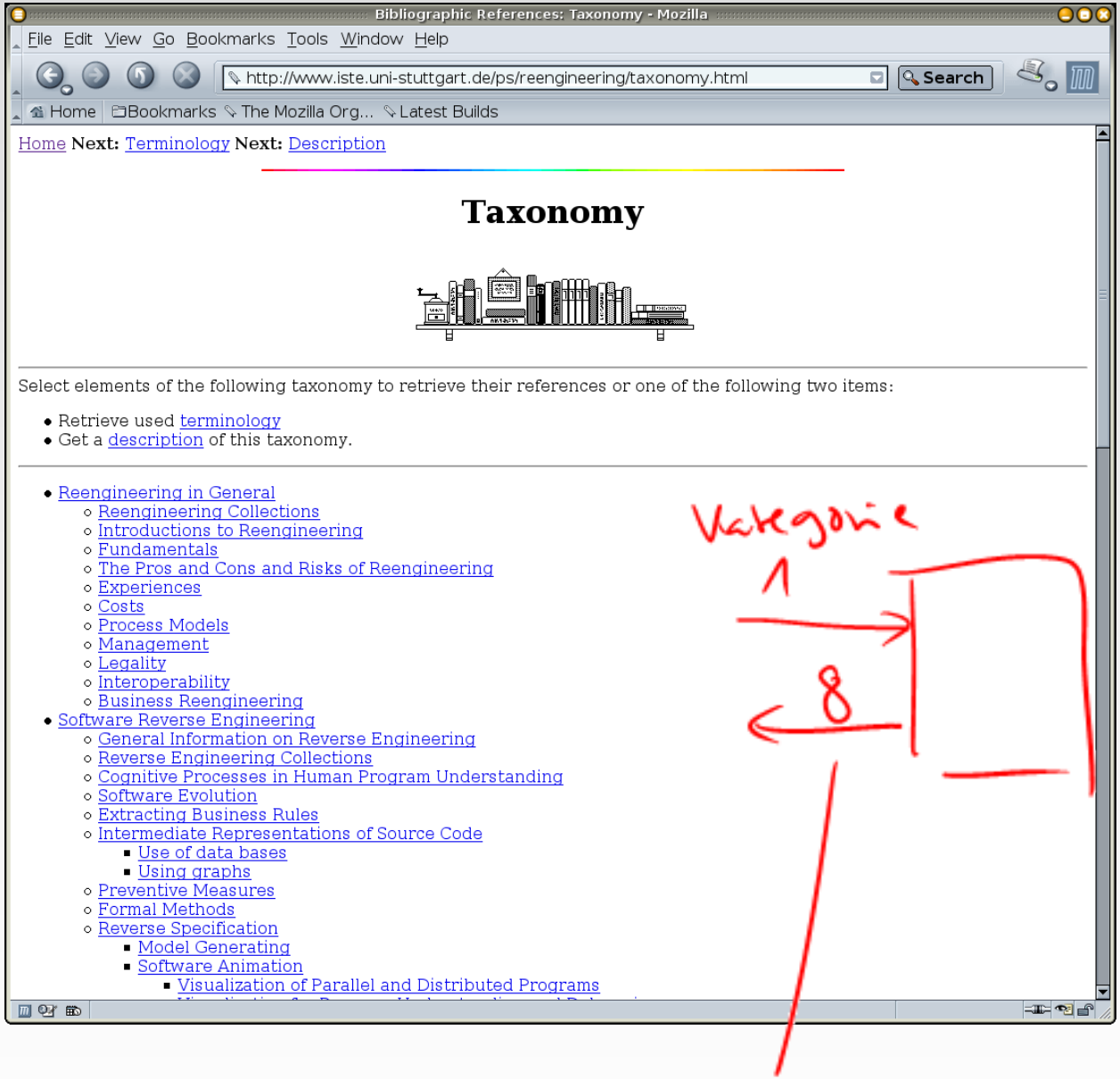
FTRs	**EQ**	1-5	6-19	20+
	0-1	3	3	4
	2-3	3	4	6
	4+	4	6	6

Function Points zusammenfassen

- EI_1: Neuen Artikel einpflegen
- EQ_1: Beschreibung der Taxonomie
- EQ_2: Suche über Taxonomie
- EQ_3: Artikel über Suchmaske abfragen
- ILF_1: Referenzen
- ILF_2: Taxonomie-Metadaten

Parameter	RET	DET	Gewicht
ILF_1	2	8	7
ILF_2	1	3	7
Parameter	FTR	DET	Gewicht
EI_1	1	15	3
EQ_1	1	1	3
EQ_2			
EQ_3			
UFP			

Online-Bibliographie: Taxonomiesuche



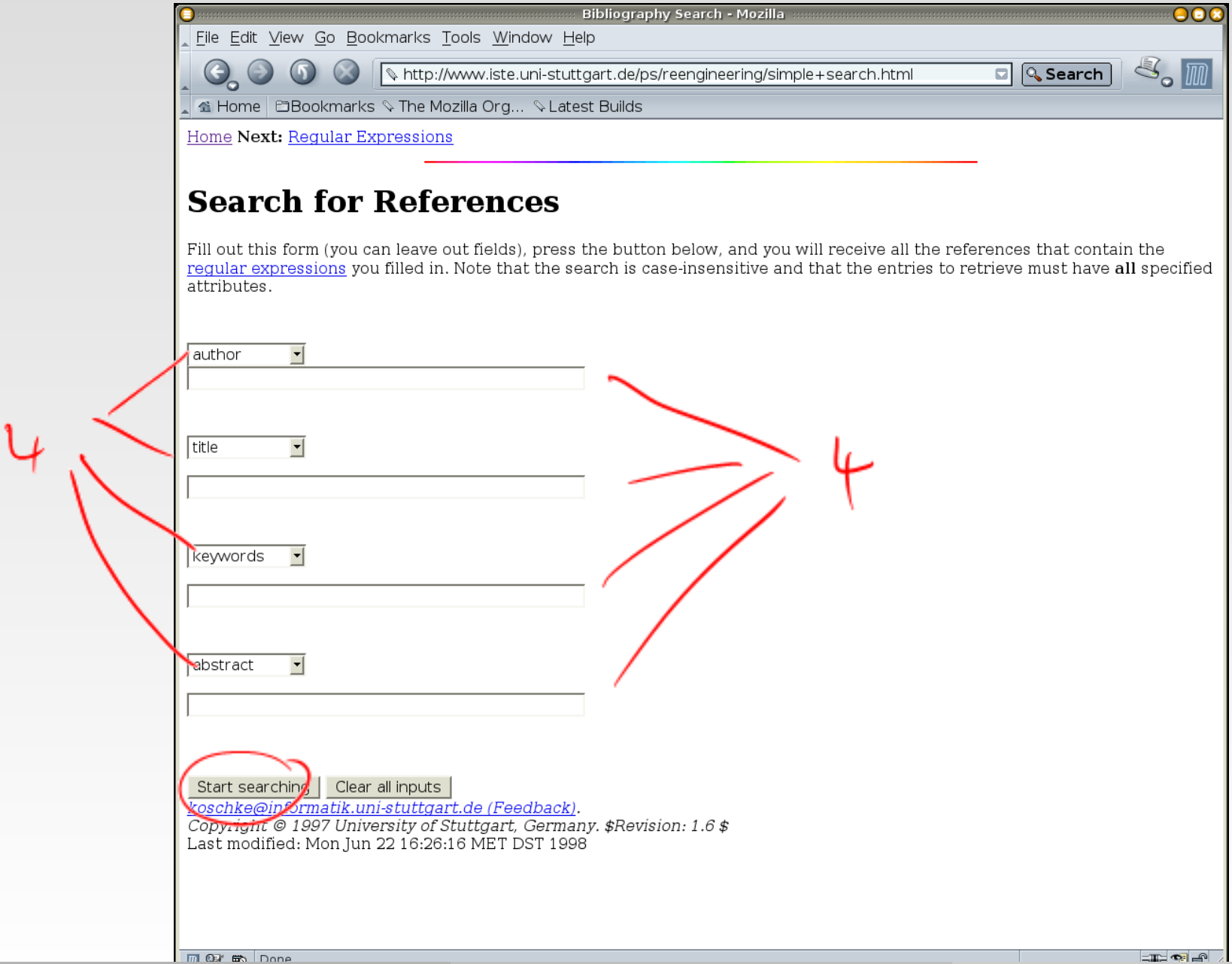
Annahme: Ein BibTeX-Eintrag habe max. 8 Einträge

Function Points zusammenfassen

- EI_1: Neuen Artikel einpflegen
- EQ_1: Beschreibung der Taxonomie
- EQ_2: Suche über Taxonomie
- EQ_3: Artikel über Suchmaske abfragen
- ILF_1: Referenzen
- ILF_2: Taxonomie-Metadaten

Parameter	RET	DET	Gewicht
ILF_1	2	8	7
ILF_2	1	3	7
Parameter	FTR	DET	Gewicht
EI_1	1	15	3
EQ_1	1	1	3
EQ_2	2	9	4
EQ_3			
UFP			

Online-Bibliographie: Suche nach Eigenschaften



Function Points zusammenfassen

- EI_1: Neuen Artikel einpflegen
- EQ_1: Beschreibung der Taxonomie
- EQ_2: Suche über Taxonomie
- EQ_3: Artikel über Suchmaske abfragen
- ILF_1: Referenzen
- ILF_2: Taxonomie-Metadaten

Parameter	RET	DET	Gewicht
ILF_1	2	8	7
ILF_2	1	3	7
Parameter	FTR	DET	Gewicht
EI_1	1	15	3
EQ_1	1	1	3
EQ_2	2	9	4
EQ_3	1	17	4
UFP			28

Gegeben ist ein Puzzle mit 500 Teilen. Entwickeln Sie eine Software, die dieses Puzzle löst. Wie lange werden Sie für die Entwicklung dieser Software brauchen?

- das Puzzle ist dreidimensional
- die Puzzle-Teile haben alle dieselbe Farbe
- die Puzzle-Teile haben alle dieselbe Form
- **alle Puzzle-Teile sind Würfel mit gleicher Kantenlänge**

FP – Gewichtete Function-Points

Systemmerkmale:

- Datenkommunikation
- Verteilte Verarbeitung
- Leistungsanforderungen
- Ressourcennutzung
- Transaktionsrate
- Online-Benutzerschnittstelle
- Benutzerfreundlichkeit
- Online-Verarbeitung
- Komplexe Verarbeitung
- Wiederverwendbarkeit
- Migrations-/Installationshilfen
- Betriebshilfen
- Mehrfachinstallationen
- Änderungsfreundlichkeit

Bewertung: 0 = kein Einfluss, 5 = starker Einfluss

Ordinal Skala

Konkrete Fragen I

- Are data communications required? **1**
- Are there distributed processing functions? **0**
- Is performance critical? **1**
- Will the system run in an existing, heavily utilized operational environment? **1**
- How frequently are transactions executed? **3**
- Does the system require on-line data entry? **4**
- Was the application designed for end-user efficiency? **0**

Konkrete Fragen II

- Are the master files updated on-line? **0**
- Is the internal processing complex? **1**
- Is the code designed to be reusable? **1**
- Are conversion and installation included in the design? **0**
- How effective and/or automated are start-up, back-up, and recovery procedures? **0**
- 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? **0**

FP – Gewichtete Function-Points

- TDI (Total Degree of Influence) = Summe der Bewertungen
  $\in [0 \ldots 15 * 5 = 75]$
- VAF (Value Adjustment Factor) = TDI/100 + 0,65
  $\rightarrow$ Gesamteinflussfaktor: 65% - 135%
- AFP (Adjusted Function Points) = UFP $\cdot$ VAF

FP – Gewichtete Function-Points

- TDI (Total Degree of Influence) = Summe der Bewertungen $\in [0 \ldots 15 * 5 = 75]$
- VAF (Value Adjustment Factor) = TDI/100 + 0,65
  $\rightarrow$ Gesamteinflussfaktor: 65% - 135%
- AFP (Adjusted Function Points) = UFP $\cdot$ VAF

Beispiel:

- TDI = 14
- VAF = 14/100 + 0,65 = 0,79
- AFP = 28 $\cdot$ 0,79 = 23 (es wird stets aufgerundet)

Kritik an der Function-Point-Methode

Einwand gegen Systemmerkmale:

- einige der Systemmerkmale sind heute eher anachronistisch
- durch Multiplikation von VAF und UAF findet eine fragwürdige Vermengung von technischen Faktoren und funktionalem Umfang sowie von quantitativen und qualitativen Aspekten statt
- VAF bringt kaum praktischen Nutzen:
  - COCOMO verwendet UAF als Eingangsgröße
  - COCOMO hat eigene technische und organisatorische Gewichtsfaktoren

$\rightarrow$ AFP sind heute eher unbedeutend (im Gegensatz zu UFP)

$\rightarrow$ bei Angaben von Function-Points nachfragen: AFP oder UFP?

Kritik an der Function-Point-Methode

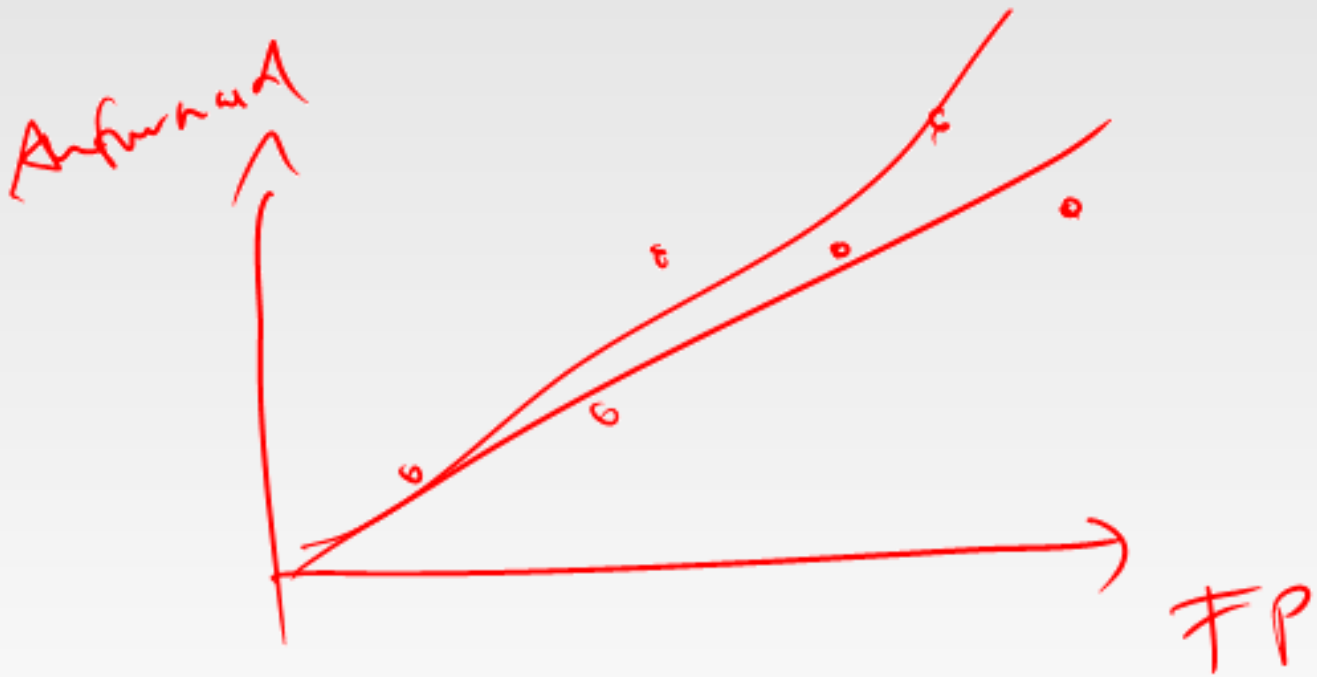
Allgemeinere Einwände:

- sehr stark auf Informationssysteme bezogen; Übertragbarkeit auf andere Arten, wie z.B. reaktive System oder Compiler, fragwürdig
- die Komplexität der Anfragen kann maximal um den Faktor 2 variieren; die Komplexität der Verarbeitungslogik geht nicht direkt ein

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs → Aufwand

- Erstellung einer neuen Erfahrungskurve
  (Zählen abgeschlossener Projekte, Regressionsanalyse)



[2] International Software Benchmarking Standards Group

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs $\rightarrow$ Aufwand

- Erstellung einer neuen Erfahrungskurve
  (Zählen abgeschlossener Projekte, Regressionsanalyse)
- Datenbank, z.B. ISBSG[2]:
  - 3GL-Projekte: $PM = 0,971 \cdot AFP^{0,351}$
  - 4GL-Projekte: $PM = 0,622 \cdot AFP^{0,405}$
  - basierend auf 662 Projekten: $PM = 0,38 \cdot AFP^{0,37}$

[2] International Software Benchmarking Standards Group

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs $\rightarrow$ Aufwand

- Erstellung einer neuen Erfahrungskurve
  (Zählen abgeschlossener Projekte, Regressionsanalyse)
- Datenbank, z.B. ISBSG[2]:
  - 3GL-Projekte: $\text{PM} = 0,971 \cdot \text{AFP}^{0,351}$
  - 4GL-Projekte: $\text{PM} = 0,622 \cdot \text{AFP}^{0,405}$
  - basierend auf 662 Projekten: $\text{PM} = 0,38 \cdot \text{AFP}^{0,37}$
- grobe Schätzung mit Faustregeln Jones (1996):
  - Entwicklungsdauer (Monate) $= \mathit{AFP}^{0,4}$
  - Personen = AFP / 150 (aufgerundet)
  - Aufwand (Personenmonate) = Personen $\cdot$ Entwicklungsdauer

[2] International Software Benchmarking Standards Group

FP – Umrechnung in Aufwand

Gesucht: Abbildung FPs $\rightarrow$ Aufwand

- Erstellung einer neuen Erfahrungskurve
  (Zählen abgeschlossener Projekte, Regressionsanalyse)
- Datenbank, z.B. ISBSG[2]:
  - 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 $\cdot$ Entwicklungsdauer

Beispiel (mit Jones-Schätzung):

- Entwicklungsdauer (Monate) $= 23^{0,4} = 3,5$
- Personen $= 23/150 \rightarrow 1$
- Aufwand (Personenmonate) $= 1 \cdot 3{,}5 = 3{,}5$

[2] International Software Benchmarking Standards Group

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 zwei „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 *Objects Points* =
  - 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
- primär 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

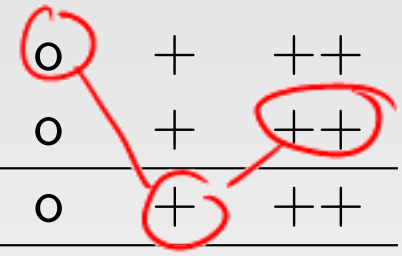
Spätere Schätzung $\rightarrow$ höhere Genauigkeit

COCOMO II – Early prototyping level

Eingaben:

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

Erfahrung/Fähigkeiten der Entwickler	- -	-	o	+	++
Reife/Fähigkeiten der CASE-Tools	- -	-	o	+	++
Median obiger Zeilen	- -	-	o	+	++
PROD [NOP/Monat]	4	7	13	25	50



- Wiederverwendungsanteil *%reuse* in Prozent

COCOMO II – Early prototyping level

Eingaben:

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

Erfahrung/Fähigkeiten der Entwickler	- -	-	o	+	++
Reife/Fähigkeiten der CASE-Tools	- -	-	o	+	++
Median obiger Zeilen	- -	-	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

FP → LOC

- 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

COCOMO II – Early design level

- Schätzung basiert auf Function Points (LOCs werden daraus abgeleitet)
- Personenmonate $PM_{NS} = A \cdot \text{KLOC}^{E} \; EM + \text{PM}_\text{m}$ bei nominalem Zeitplan
- $A = 2,94$ in initialer Kalibrierung (empirischer Wert)
- 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 $B = 0,91$

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 (empirischer Wert)
- 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 $B = 0,91$
- 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
  - $EM = \prod \textit{Effort-Multiplier}_\text{i}$

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 (empirischer Wert)
- 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 $B = 0,91$
- 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

$$PM_{NS} = A \cdot \text{KLOC}^{E} \cdot EM + \text{PM}_\text{m}$$

- 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

$$PM_{NS} = A \cdot \text{KLOC}^{E} \cdot EM + \text{PM}_{\text{m}} \text{ mit } EM = \prod \textit{Effort-Multiplier}_{\text{i}}$$

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 (Auslastung CPU)
STOR	Main storage constraints (Auslastung RAM)
PVOL	Platform volatility (Häufigkeit von Plattformänderungen)

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 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	≤2 Mo.	6 Mo.	1 J.	3 J.	6 J.
PLEX	≤2 Mo.	6 Mo.	1 J.	3 J.	6 J.
LTEX	≤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

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

Rechenbeispiel

Nominaler Aufwand

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

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 $B = 0,91$.

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 $B = 0,91$.

Annahme: es herrschen einfache Verhältnisse:

- $\rightarrow \forall i : w_i = 0 \Rightarrow E = 0,91$ (bester Fall)
- $\rightarrow$ nominale Effort-Multiplier = 1,00 (Normalfall) $\Rightarrow EM = 1,00$

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 $B = 0,91$.

Annahme: es herrschen einfache Verhältnisse:

$\rightarrow \forall i : w_i = 0 \Rightarrow E = 0,91$ (bester Fall)

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

Geschätzte Programmlänge = 100 [KLOC]

$\rightarrow PM_{NS} = 2,94 \times 100^{0,91} \times 1,0 = 194,24 \text{ Monate } \approx 16 \text{ Jahre}$

Entwicklungsdauer

Nominale Entwicklungsdauer (Kalenderzeit in Monaten)

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

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

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: $TDEV_{NS} = 3,67 \times 194,24^{0,28+0,2\times(0,91-0,91)} \approx 16$

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: $TDEV_{NS} = 3,67 \times 194,24^{0,28+0,2\times(0,91-0,91)} \approx 16$

Anzahl Entwickler

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

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: $TDEV_{NS} = 3,67 \times 194,24^{0,28+0,2\times(0,91-0,91)} \approx 16$

Anzahl Entwickler

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

Beispiel: $N = 194,24/16 = 12$

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 Monate? Geht das nicht schneller?“

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 Monate? Geht das nicht schneller?“

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

Abweichung von der nominalen Entwicklungdauer

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

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 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%:

$TDEV = 16 \times 75/100 = 12$ Monate

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 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%:

$TDEV = 16 \times 75/100 = 12$ Monate

Chef: „Super!“

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 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%:

$TDEV = 16 \times 75/100 = 12$ Monate

Chef: „Super!“

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

$PM = 2,94 \times 100^{0,91} \times 1,0 \times 1,43 = 277,76$

Erhöhung des Aufwands: um 43 %

Verkürzte Entwicklungsdauer

Chef: „Wieso 16 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%:

$TDEV = 16 \times 75/100 = 12$ Monate

Chef: „Super!“

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

$PM = 2,94 \times 100^{0,91} \times 1,0 \times 1,43 = 277,76$

Erhöhung des Aufwands: um 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	$\longrightarrow$	Verlässlichkeit, Datenbasisgröße, Komplexität, Dokumentation
Plattformkomplexität	$\longrightarrow$	Laufzeit-, Speicherbeschränkungen, Plattformdynamik
Fähigkeiten Personal	$\longrightarrow$	Fähigkeiten der Analysten/Entwickler, Kontinuität des Personals
Erfahrung Personal	$\longrightarrow$	Domänenerfahrung der Analysten/Entwickler, Erfahrung mit Sprache und Werkzeugen
Infrastruktur	$\longrightarrow$	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

1 **Albrecht 1979** Albrecht, Alan: Measuring application development productivity. In: *Proc. Joint SHARE/GUIDE/IBM Applications Development Symposium*, 1979, S. 83–92

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 IBM 1985 : *Die Function-Point-Methode: Eine Schätzmethode für IS-Anwendungsprojekte*. IBM-Form GE12-1618-1. 1985

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

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

0 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

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

2 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

3 Poensgen und Bock 2005 Poensgen, Benjamin ; Bock, Bertram: *Die Function-Point-Analyse. Ein Praxishandbuch*. Dpunkt Verlag, 2005. – ISBN 978-3898643320

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