

Informationswirtschaft 3

Aufwandsschätzung

Wolfgang H. Janko
Stefan Koch

Abteilung für Informationswirtschaft
Institut für Informationsverarbeitung und Informationswirtschaft
Wirtschaftsuniversität Wien
Augasse 2-6, A-1090 Wien, Österreich
Telefon: +43-1-31336-5206
E-mail: Stefan.Koch@wu-wien.ac.at
URL: <http://wwwai.wu-wien.ac.at/>

Inhaltsverzeichnis

1. Grundlagen
2. Modelle des Software-Entwicklungsprozesses
3. COCOMO
4. Putnam-Modell
5. Function Points

6. Nicht-algorithmische Verfahren

7. Einsatz in Organisationen, Datengewinnung und Validierung

8. Fallbeispiel für Aufwandsschätzung im Unternehmen

Grundlagen

- Entwicklung und Wartung von Software: Aufgabe von hoher Unsicherheit
- vor allem die zugrundeliegende Komplexität zu Beginn kaum abzuschätzen
- Ziel trotzdem: Zeit-, Kosten- und Personalplanung sowie die laufende Kontrolle, für Angebotslegungen, Business Plans, Investitionsentscheidungen,...
- rein subjektive Schätzungen durch Experten haben sich als zu fehleranfällig erwiesen, daher Versuch zur Objektivierung

- Modell zur Aufwandsschätzung: Vorhersagesystem für Attribute des zugrundeliegenden Prozesses (vor allem des Aufwands)
- besteht aus einem mathematischen Modell des Prozesses mit den Beziehungen zwischen den verschiedenen Attributen, einer Vorgehensweise zur Bestimmung unbekannter Parameter und einer Anleitung zur Interpretation der Ergebnisse
- Anforderungen: Validität der Schätzungen, Objektivität, einfache Handhabung, Robustheit in Hinsicht auf kleine Änderungen bei den Input-Parametern und Transportfähigkeit zwischen verschiedenen Umgebungen

- viele der existierenden Modelle enthalten Koeffizienten, die nur für eine bestimmte Entwicklungsumgebung Gültigkeit aufweisen (z.B. gefunden durch Regression einiger Projekte), daher ist die sogenannte Kalibrierung notwendig: mittels entsprechender Methoden (z.B. Regression) werden diese Parameter an die vorliegende Umgebung angepasst, Modell wird 'kalibriert' (dafür logischerweise notwendig: entsprechende Datenaufzeichnung über vergangene Projekte in der vorliegenden Umgebung)

Einteilung

- Einteilung von Modellen nach der Methode, mittels derer sie abgeleitet wurden (Conte et al., 1986)
 - Historisch-experimentelle Modelle: Expertenschätzungen, teilweise basierend auf historischen Daten (einfache Analogien)
 - Statistik-basierte Modelle: normalerweise mittels Regressionsanalyse bestimmte Beziehung zwischen Aufwand und anderen Parametern wie der Programmgröße (unterteilbar in lineare und nicht-lineare Modelle)
 - Theoretisch-basierte Modelle: Grundlage sind Annahmen über die Funktionsweise des menschlichen Geistes bei der Programmentwicklung oder postulierte mathematische Gesetze, denen der Software-Entwicklungsprozeß folgt

- Zusammengesetzte Modelle: Mischung von Schätzgleichungen, statistischen Methoden und Expertenurteil
- andere Einteilung z.B. nach der mathematischen Form der verwendeten Gleichungen, nach der Art der Daten, die zur Voraussage verwendet werden, und nach den Annahmen, die hinsichtlich des Software-Entwicklungsprozesses gemacht werden,...
- statische und dynamische Modelle: bei ersteren wird eine Variable wie zum Beispiel die Größe zur Berechnung der anderen herangezogen, während bei dynamischen Modellen alle Variablen interdependent sind
- Modelle mit einer und solche mit mehreren Variablen
- Einteilung nach Fenton

- Expertenmeinung beziehungsweise das Raten auf der Grundlage persönlicher Erfahrung
- Analogie als formalerer Ansatz zur Expertenmeinung unter Einbeziehung von direkten Vergleichen mit einem oder mehreren vergangenen Projekten
- Dekomposition in einzelne zu schätzende Teilbereiche
- Schätzgleichungen, d.h. mathematische Formeln, die Inputfaktoren wie zum Beispiel ein Maß für die Programmgröße in Beziehung zum Aufwand setzen
- Einteilung nach Arifoglu und Janko
 - Verfahren, die auf der Aufwandsschätzung einer Einheit beruhen

- Verfahren, die eine prozentuelle Aufteilung der Kosten des gesamten Softwaresystems der Organisation vornehmen
- Analogie-basierte Verfahren: Auf der Grundlage von Ähnlichkeiten zu früher entwickelten Softwaresystemen wird eine Schätzung abgeleitet.
- Verfahren, die auf parametrischen Gleichungen basieren: Aus Vergangenheitsdaten abgeleitete parametrische Gleichungen werden zur Schätzung für neue Entwicklungen herangezogen.
- Verfahren, die Methoden der künstlichen Intelligenz einsetzen

Modelle der Software-Entwicklung

- den meisten (konkreten) Aufwandsschätzungsmodellen liegt eine Vorstellung über die Software-Entwicklung zugrunde
- Software-Entwicklungsprozeß als Produktionsfunktion
 - einfachster Fall: Input einer solchen Funktion ist Aufwand zum Beispiel in Mannjahren und der Output ist das erstellte Software-Artefakt, gemessen anhand einer Metrik für die Größe wie zum Beispiel LOC
 - unterschiedliche Arten von Produktionsfunktion möglich (S bezeichnet jeweils die Größe der Software, E den Aufwand, a_1, \dots, a_n entsprechende Parameter):

- * linear: $S = a_0 + a_1E$ (empirisch nicht haltbar)
 - * quadratisch: $S = a_0 + a_1E + a_2E^2$
 - * Cobb-Douglas: $S = a_0E^{a_1}$
 - * translog-Produktionsfunktion: $\ln S = a_0 + a_1 \ln E + a_2 \ln^2 E, \dots$
- Problem: empirische Belege für sowohl steigende wie auch sinkende Skaleneffekte (analog zu Industrieproduktion: bei höherer Produktion sinken Grenzkosten), bei kleinen Projekten eher steigende, danach fallende - folglich könnte es Projektgröße mit maximaler Produktivität geben (kann über Data Envelopment Analysis untersucht werden - siehe Kapitel zum Vergleich von Produktivität)

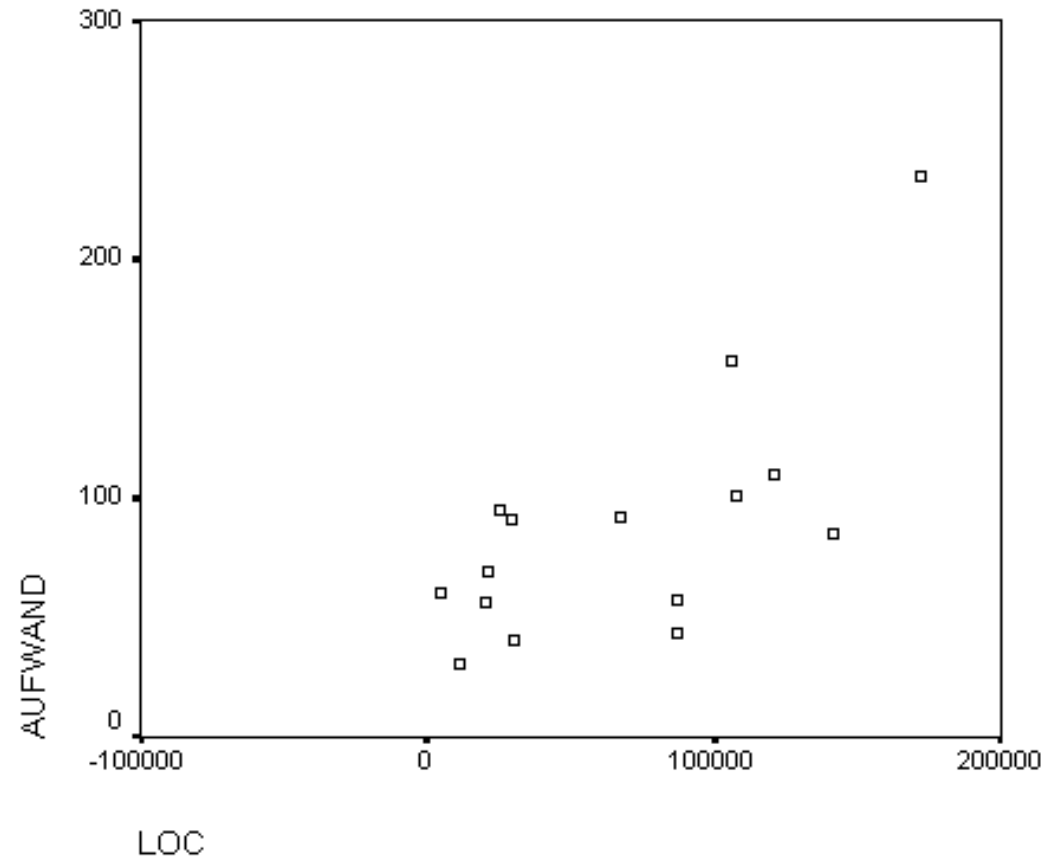


Abbildung 1: Scatterplot - LOC vs. Aufwand

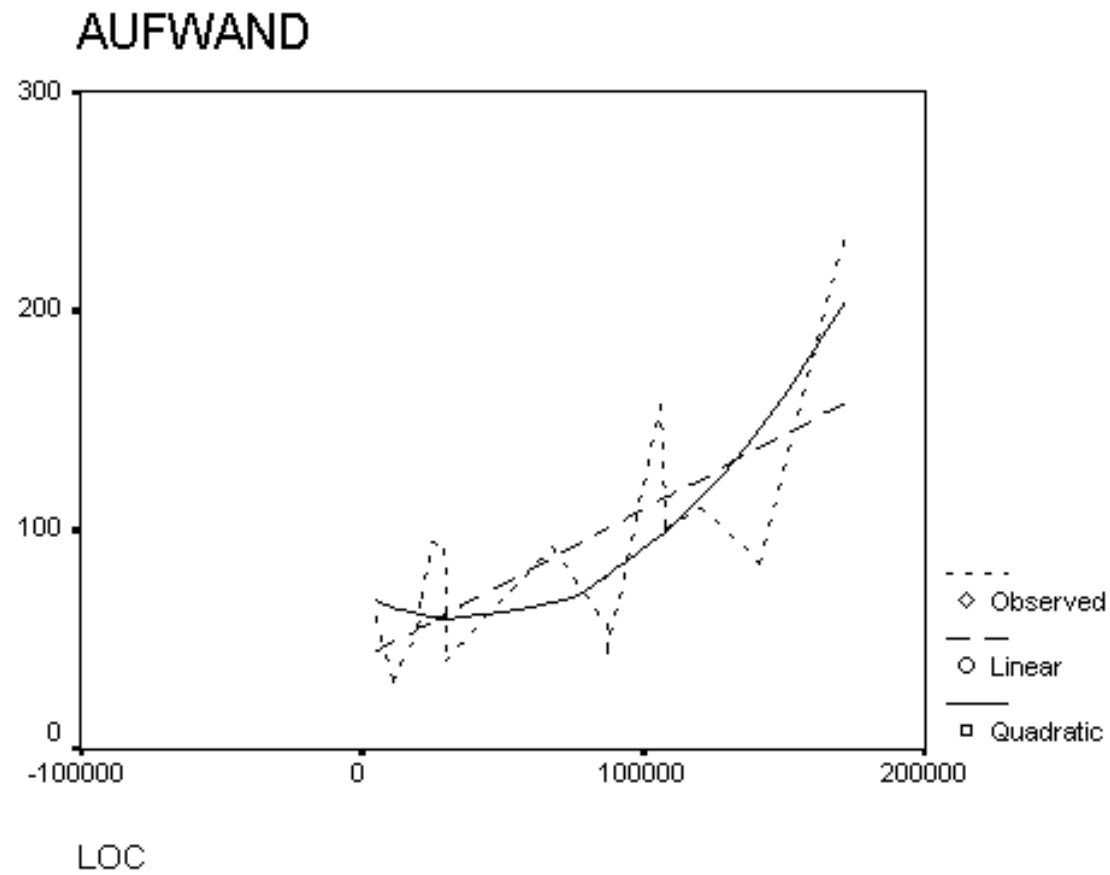


Abbildung 2: Regression - unterschiedliche Produktionsfunktionen

- Ansatz von Norden:

- publiziert 1960, eigentlich für allgemeine F&E-Projekte
- Entwicklungsprojekt: endliche Folge von zielgerichteten, zeitlich geordneten Aktivitäten, die sich auf eine homogene Menge von Problemelementen beziehen, um eine spezifizierte Menge von Zielen zu erreichen
- Projekt besteht folglich aus einer Menge von Problemen, die alle gelöst werden müssen
- Annahme: Anzahl der Probleme ist unbekannt aber endlich
- Annahme: Finden sowie Lösung von Problemen findet durch die kreative Leistung der eingesetzten Arbeitskräfte ('manpower') statt

- Annahme: Lösen eines Problem ist ein Ereignis, das ein Element aus der endlichen Menge der ungelösten entfernt, Auftreten dieser Ereignisse ist unabhängig und zufällig (folgt einer Poisson-Verteilung)
- Annahme: Anzahl der Personen, die zu einem bestimmten Zeitpunkt sinnvoll an einem Projekt mitarbeiten können, ist ungefähr proportional zu der Anzahl der Probleme, die zu diesem Zeitpunkt zur Lösung bereit sind
- Folge: gegen Ende eines Projekts, wenn nur mehr wenige Probleme vorhanden sind, nimmt die Anzahl der eingesetzten Mitarbeiter ab (bzw. sollte es)
- Annahme: Lernrate des Projektteams wird mittels einer linearen Funktion der Zeit t mit $p(t) = 2 a t$ modelliert

- Kosten $C(t)$ in Mannjahren des Projektes entstehen durch die Kumulierung der eingesetzten Arbeitskraft über die Zeit
- Kosten sind zu Beginn des Projektes Null und steigen monoton bis zum Gesamtaufwand K an
- Änderung dieser Kostenfunktion zu einem beliebigen Zeitpunkt, $dC(t)/dt$, ist die Anzahl der zu diesem Zeitpunkt involvierten Personen (d.h. im nächsten Zeitschritt kommt als Aufwand die Anzahl der in dem Zeitschritt eingesetzten Personen hinzu)
- diese Anzahl der nutzbringend involvierten Personen ist über einen Faktor k (Lernrate) proportional zur Anzahl der noch zu lösenden Probleme (Lernrate bestimmt eigentlich, wieviele davon schon bekannt sind, während des Projektfortschritts werden die Probleme durch Lernen entdeckt)

- diese Anzahl der noch zu lösenden Probleme kann über die Differenz zwischen den bisher angefallenen Kosten ($C(t)$) und der Gesamtanzahl der Probleme, wobei dafür der gesamte Projektaufwand K als Indikator verwendet wird, ausgedrückt werden, also $K - C(t)$

- Folge: kumulativer Aufwand zu einem Zeitpunkt t

$$C(t) = K[1 - \exp(-at^2)] \quad (1)$$

- Ableitung dieses Ausdrucks nach der Zeit gibt somit die Besetzung des Projekts mit Personen ('manpower function', Manpower-Funktion) an:

$$m(t) = C'(t) = 2 K a t \exp(-a t^2) \quad (2)$$

- dies ergibt eine Rayleigh-Kurve, daher wird das Modell auch Norden/Rayleigh-Modell genannt

- Verlauf: zu Beginn ein Wert von 0, dann ein Anstieg bis zu einem Gipfelpunkt t_d und schließlich ein stetiger Abstieg wiederum auf 0 (Parameter a hat dabei einen wichtigen Einfluß auf diesen Gipfelpunkt t_d , je größer a ist, desto steiler der Anstieg und früher der Gipfelpunkt)
- Es kann, wenn der Gipfelpunkt t_d erreicht, und damit zugleich die Besetzung mit Personen zu diesem Zeitpunkt m_0 bekannt ist, der Gesamtaufwand K , der für das Projekt anfällt, berechnet werden.

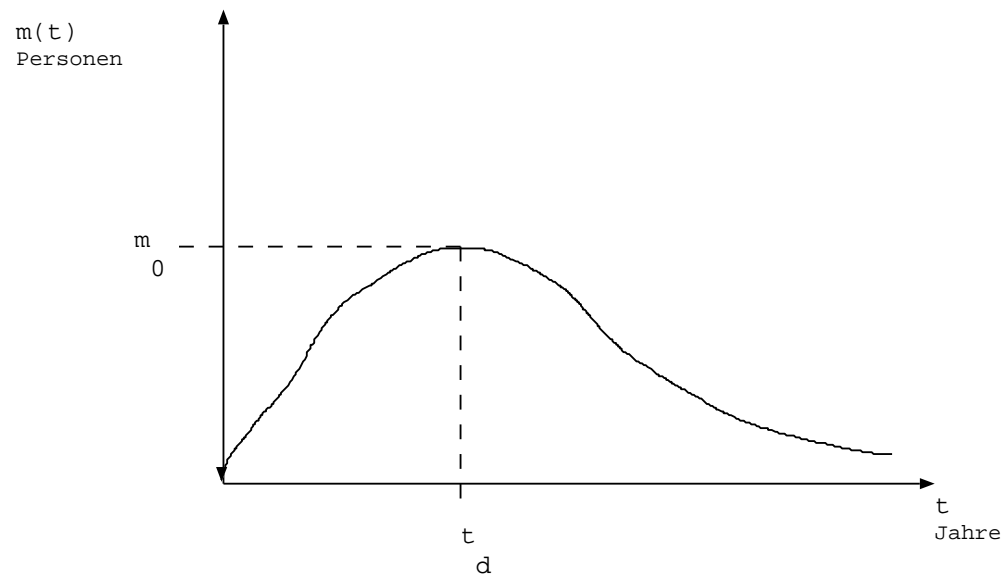
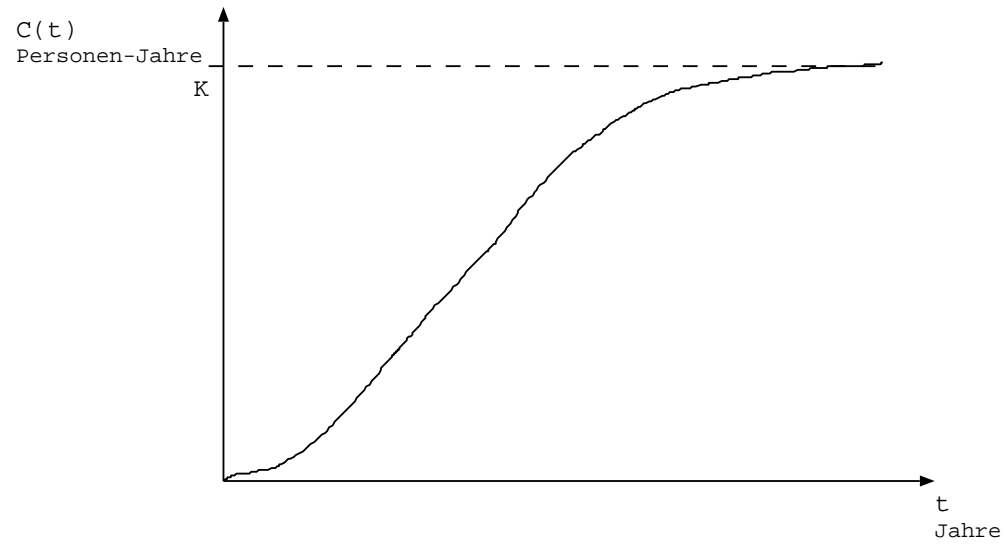


Abbildung 3: Rayleigh-Modell

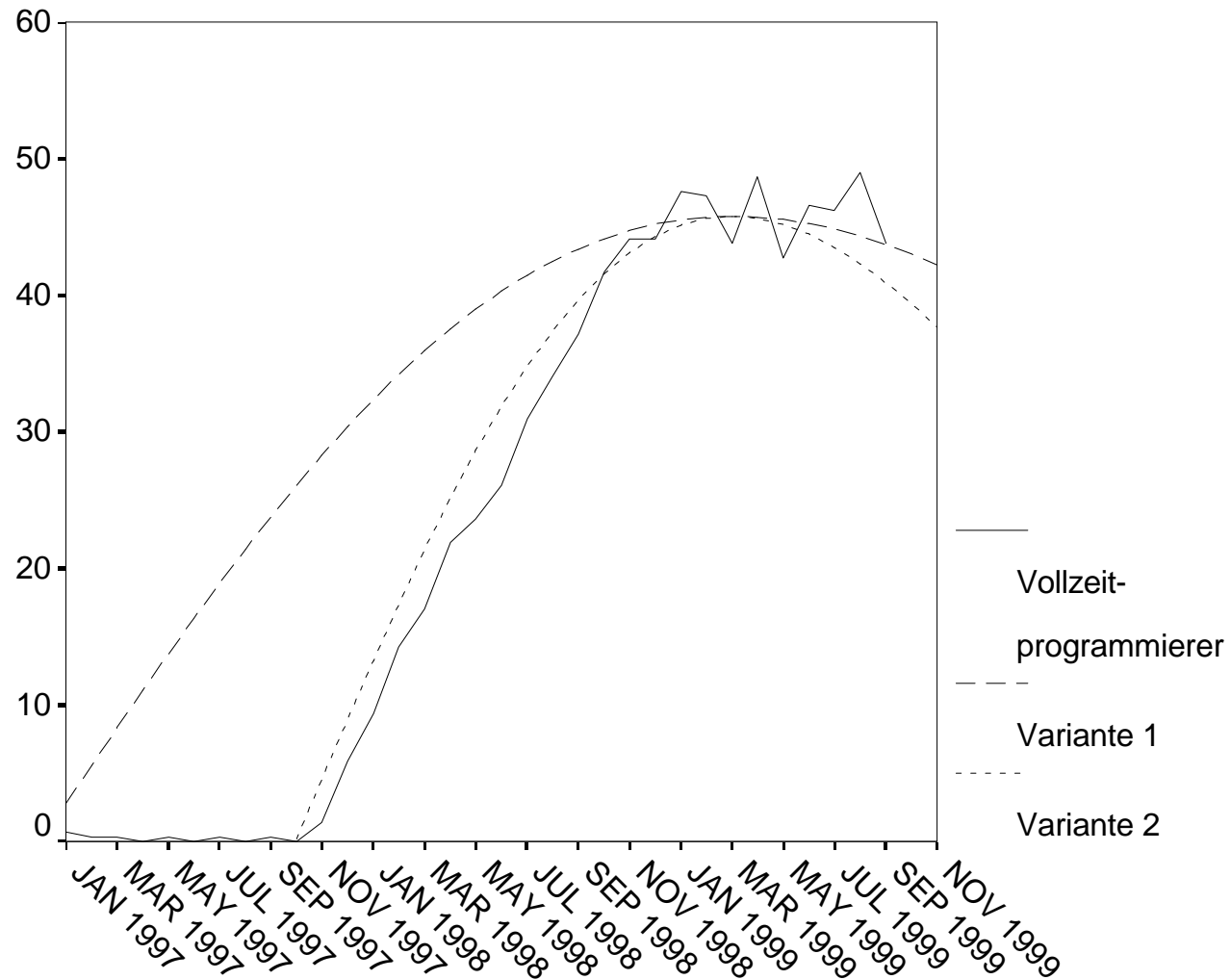


Abbildung 4: Rayleigh-Modell GNOME Projekt

- ähnlich Ansatz von Parr: Abfall im Entwicklungsaufwand gegen Ende eines Projekts wird ebenfalls durch Ausschöpfung des Problemraumes erklärt, jedoch existiert keine Lernrate, sondern bestimmte Abhängigkeiten zwischen den zu lösenden Problemen (zu jedem Zeitpunkt gibt es eine Menge von sichtbaren ungelösten Problemen, sinnvoller Input ist dazu proportional), je nach gewählten Abhängigkeiten kann man z.B. den Nutzen strukturierter Programmierung modellieren (zu Beginn werden durch Analyse etc. gleich mehr Probleme eröffnet, damit kann mehr Manpower angewandt werden - schnelleres Projektende), Unterschiede in der Kurvenform zu Norden vor allem in der Anfangsphase (Beginn eines Projekts nicht genau definiert, liegt vor $t = 0$)
- Software-Projekt als dynamisches System: einzelne Bestandteile beeinflussen einander wechselseitig, auch Faktoren wie die

Erwartungen der Projektleitung, die Reaktionen auf Verzögerungen oder die Motivation der Mitarbeiter werden explizit modelliert, wird beispielsweise simuliert, kann z.B. Brooks's Law zeigen

COCOMO

- COnstructive COst MOdel (Boehm, 1981)
- liegt in drei zunehmend detaillierteren Formen (inkludieren mehr und genauere Parameter) vor (Basic, Intermediate und Detailed)
- Grundannahmen
 - Wasserfall-Modell in der Software-Entwicklung
 - Existenz von (nur) drei verschiedenen Modi der Entwicklung
 - * Organic: Kleine Teams erfahrener Personen in bekanntem Umfeld, mit stabiler Entwicklungsumgebung und geringer Projektgröße

- * Embedded: Projekte relativ hoher Größe unter Existenz schwieriger Nebenbedingungen mit höherer Unsicherheit sowie einem höheren Innovationsgrad
- * Semi-detached: Mischung aus den beiden anderen Modi
- primärer Kostentreiber sind die ausgelieferten Source-Anweisungen ('delivered source instructions')
- gutes Management von Entwickler- wie auch Kundenseite
- weitgehende Konstanz der Anforderungsspezifikation während des Projektverlaufes
- Entwicklungsaufwand E in Mannmonaten wird über die Größe S der Software in Kilo-NCSS berechnet:

$$E = a_i S^{b_i} m(x) \quad (3)$$

- Parameter a_i sowie b_i : abhängig von Form des Modells und Modus der Entwicklung (von Boehm geschätzt basierend auf einer Datenbasis von 63 Projekten, teilweise durch Anwendung der Delphi-Methode mit Experten)

Modell	Modus	Entwicklungsaufwand
Basic	Organic	$E = 2.4S^{1.05}$
	Semi-detached	$E = 3.0S^{1.12}$
	Embedded	$E = 3.6S^{1.20}$
Intermediate	Organic	$E = 3.2S^{1.05}m(x)$
	Semi-detached	$E = 3.0S^{1.12}m(x)$
	Embedded	$E = 2.8S^{1.20}m(x)$

- $m(x)$: Einfluß von 15 Umgebungsfaktoren (dies sind Produktattribute wie geforderte Zuverlässigkeit des System,

Computerattribute wie Hauptspeicherengpässe, Personalattribute wie Programmiersprachenerfahrung und Projektattribute wie moderne Programmierpraktiken), jeder Faktor wird auf einer 6-teiligen Skala eingeschätzt, eine Tabelle liefert dann je nach Faktor und Einschätzung einen Wert von ca. 0.7 bis 1.3, diese Werte aller Faktoren werden dann alle miteinander multipliziert um $m(x)$ zu erhalten, wird erst in Intermediate COCOMO inkludiert, in Detailed COCOMO werden die jeweiligen Werte der Faktoren für die einzelnen Phasen der Software-Entwicklung getrennt festgelegt

- zusätzlich erlaubt COCOMO eine Schätzung der Entwicklungsdauer T (in Jahren) in Abhängigkeit vom Aufwand E (analoge Formel wie Schätzung des Aufwandes) sowie eine prozentuelle Aufteilung des Gesamtaufwandes auf die einzelnen Phasen (einfache Auflistung)
- Vorgehen bei der Schätzung:

1. Auswahl des Detaillierungsgrades (also des Modells): Basic, Intermediate oder Detailed
 2. Einordnung des Projekts in eine der drei Modi: Organic, Semi-detached oder Embedded
 3. Schätzung der Größe der zu erstellenden Software in LOC
 4. für die Modelle Intermediate oder Detailed: Bewertung der 15 Umgebungsfaktoren
 5. Anwendung der Formel (die zu verwendende Formel ergibt sich aus der Kombination Modell-Modus)
- Beispiel
 - Entwicklung einer neuen Anwendung an der WU für LV-Anmeldung und Verwaltung

- Basic Model wird gewählt
 - Modus Semi-detached wird gewählt
 - Grösse wird mit 23.000 LOC geschätzt
 - Umgebungsfaktoren nicht nötig, da Basic Model verwendet wird
 - ergibt Formel $E = 3.0 \times 23^{1.12}$ und damit einen Aufwand von 100,5 Mannmonaten
- Probleme
 - neuere Lebenszyklusmodelle
 - Verfügbarkeit von Informationen in unterschiedlichen Stadien der Entwicklung (LOC schwer zu Beginn zu schätzen)

- daher: **COCOMO II** (Boehm et al., 2000)
- jetzt auch Verwendung von Function Points oder Application Points für Größenschätzung möglich, diese werden in LOC umgerechnet (siehe unten)
- Bewertung von Reuse mittels eines nicht-linearen Modells
- Re-engineering und Konvertierung von Bestandteilen eines vorhandenen Systems werden bewertet
- sowohl Economies wie auch Diseconomies of Scale möglich
- 3 Modelle

- Application Composition Model: Dieses Modell kommt eher in frühen Phasen des Lebenszyklus zum Einsatz, wenn mittels Prototyping vorgegangen wird.
- Early Design Model: In diesen Phasen des Lebenszyklus werden verschiedene Architektur-Alternativen exploriert oder eine inkrementelle Entwicklung begonnen. Es ist bereits mehr Information verfügbar.
- Post-Architecture Model: Dieses Modell wird für die eigentliche Software-Entwicklung und -Wartung eingesetzt. Die Lebenszyklus-Architektur ist zu diesem Zeitpunkt fixiert. Die Granularität dieses Modells entspricht dem bisherigen COCOMO 81.

- Schätzung (ähnlich zu COCOMO 81):

$$Effort_{nominal} = A \times (Size)^E \quad (4)$$

- A : aufgrund der vorgenommenen Kalibrierung auf 2.94 gesetzt
- Skalen-Parameter E wird über fünf Skalenfaktoren (Precendentedness, Development Flexibility, Architecture/Risk Resolution, Team Cohesion und Process Maturity) abgeschätzt, jeder Faktor wird auf einer 6-teiligen Skala bewertet, eine Liste liefert die jeweilige Gewichtung SF_j , dann wird E durch die Formel

$$E = B + 0.01 \times \sum_{j=1}^5 SF_j \quad (5)$$

bestimmt (B momentan 0.91)

- Einfluß von Kostentreibern EM_i (insgesamt 17 verfügbar, etwas modernisiert, für die ersten Phasen der Entwicklung werden nur 7 verwendet):

$$Effort_{adjusted} = Effort_{nominal} \times \prod_{i=1}^{17} EM_i \quad (6)$$

Modell von Putnam

- basiert auf einer Modellierung des Software-Entwicklungsprozesses nach Norden/Rayleigh
- Untersuchung von 200 Projekten: Verlauf der Kurve in der Praxis bestätigt
- außerdem herausgefunden: Gipfelpunkt t_d liegt sehr nahe an dem Zeitpunkt, zu dem das System operativ gesetzt wird (damit entspricht t_d der Entwicklungszeit)
- Rayleigh-Gleichung dividiert durch die Zeit t mittels Logarithmus linearisiert ergibt eine Gerade mit Steigung $-1/2t_d^2$ und Interzept $\ln(K/t_d^2)$ - empirisch: Argument des Interzepts weist eine Beziehung zu den betrachteten Projekten auf

- Schwierigkeitsgrad ('difficulty') eines Projektes daher definiert mit

$$D = \frac{K}{t_d^2} \quad (7)$$

- Betrachtung der Änderung des Schwierigkeitsgrades D über dessen Gradienten ergibt: wenn die Entwicklungszeit verkürzt wird, so steigt die Schwierigkeit des Projektes dramatisch an
- Größenordnung der partiellen Ableitung von D nach t_d hat bei den betrachteten Projekten einen Wert von 8 für völlig neue Systeme mit vielen Interaktionen mit anderen Systemen, 15 für neue stand-alone Systeme, oder 27 für Systeme, die aus vorhandenen Komponenten erstellt werden (genannt Manpower Build-up, gibt die maximale Zufuhr an Mitarbeitern an)

- Beziehung mit dem Umfang S der Software in NCSS über die Software-Gleichung

$$S = C_k K^{1/3} t_d^{4/3} \quad (8)$$

- C_k : Umgebungsfaktor, der den Stand der angewendeten Technologie ausdrückt (kann z.B. aus alten Projekten geschätzt werden)

- Beispiel

- Grösse der Software wird mit 23.000 LOC geschätzt, Umgebungsfaktor ist 900
- Software soll in 3 Jahren einsatzbereit sein
- Software-Gleichung daher: $23000 = 900 \times K^{1/3} \times 3^{4/3}$

- Gesamtaufwand K ist folglich 206 Mannjahre
- wenn die Software nun doch schon in 2 Jahren einsatzbereit sein soll, ergibt sich eine Software-Gleichung von $23000 = 900 \times K^{1/3} \times 2^{4/3}$
- und ein neuer Gesamtaufwand K von 1043 Mannjahren

Function Points

- eigentlich eine Metrik zur Quantifizierung der Größe eines Software-Systems
- erster Schritt: betrachtetes System wird in seine Komponenten unterteilt, jede Komponente gehört zu einer Klasse und wird nach ihrer Komplexität mittels dreier Stufen gewichtet
- Klassen:
 - externe Inputs
 - externe Outputs
 - logische interne Dateien

- externe Schnittstellendateien
- externe Abfragen
- anhand einer Tabelle der jeweiligen Gewichte für jede Kombination Klasse/Schwierigkeit und entsprechender Summierung über alle Komponenten erhält man: Unadjusted Function Points (UFP)
- Einfluß von 14 Charakteristiken (Anforderungen an das System wie verteilte Verarbeitung oder hohe Transaktionsrate): jeder Faktor ist jeweils auf einer Skala von 0 bis 5 einzuschätzen, Aufsummierung ergibt Processing Complexity (PC) oder Technical Complexity Factor (TCF)
- Function Points (FP, Metrik für den notwendigen Aufwand zur Erstellung des Software-Systems und damit seine Größe)

$$FP = UFP \times (0.65 + (0.01 \times PC)) \quad (9)$$

- Korrelation zu LOC und tatsächlichem Entwicklungsaufwand empirisch bestätigt
- für verschiedene Sprachen Umrechnungsfaktoren von einem Function Point in die jeweils im Durchschnitt zur Implementierung notwendige Anzahl von LOC (z.B. C 128, C++ 55, PERL 27, Visual Basic 29, Assembler 320,...)
- Schätzung des Aufwandes basierend auf FP: entweder Umrechnung in LOC und dann Anwendung eines anderen Modells (z.B. COCOMO,...), oder Bestimmung einer direkten Beziehung FP zu Aufwand aus Vergangenheitsdaten (z.B. $E = 54 \times FP - 13390$ oder $\ln E = 1.00 \times (\ln FP) + 2.51$, siehe Produktionsfunktionen)
- Beispiel

- Entwicklung einer neuen Anwendung an der WU für LV-Anmeldung und Verwaltung
- externe Inputs: 1 einfacher (Noten), 2 schwierige (Ankündigung und Anmeldung) = $1 \times 3 + 2 \times 6 = 15$
- externe Outputs: 1 schwieriger (VVZ), 2 mittlere (Noten- und Anmeldeliste) = $1 \times 7 + 2 \times 5 = 17$
- logische interne Dateien: 1 schwierige (LV-Verzeichnis), 2 mittlere (Notenliste, Teilnehmerliste) = $1 \times 15 + 2 \times 10 = 35$
- externe Schnittstellendateien: 2 schwierige (zu VVZ im WWW und zu Noten-DB) = $2 \times 10 = 20$
- externe Abfragen: keine
- $UFP = 15 + 17 + 35 + 20 + 0 = 87$

- besondere Auswirkungen durch Anforderungen: Performance 3, Datenkommunikation 2, Transaktionsrate 4, Online-Eingaben 5, End User Effizienz 5, Online-Update 5 (alle anderen ohne Einfluss)
- $PC = 3 + 2 + 4 + 5 + 5 + 5 + 0 + \dots + 0 = 24$
- $FP = 87 \times (0.65 + (0.01 \times 24)) = 87 \times 0.89 = 77.43$
- Aufwandsschätzung: z.B. über Formel $\ln E = 1.00 \times (\ln 77.43) + 2.51 = 952.77$ oder Umrechnung in LOC (Programmiersprache Java) $LOC = FP \times 53 = 77.43 \times 53 = 4103.79$ und dann in COCOMO einsetzen
- Vorteile: frühzeitige Quantifizierbarkeit im Entwicklungsprozeß, von Sprache und Technologie unabhängig

- Weiterentwicklungen

- Symons (Function Point Mark II): kritisiert Gewichtefestlegung durch Befragungen bei IBM, geringe Bedeutung der internen Verarbeitungskomplexität und Aufsummierung der Function Points (insbesondere sollte ein integriertes System eine andere Komplexität haben als mehrere unabhängige, bei selben FP), Folgerung: Input-Datenelemente, Output-Datenelemente und referenzierte Entitäten aller Transaktionen werden gezählt
- Application Points (Boehm), vor allem für Application Composition (Framework, Middleware, GUI Builder, Development Tools): Anzahl von Screens, Reports und 3GL Komponenten gezählt, jeweils Gewichtung, Reuse-Faktor, Produktivität durch Programmierer-Erfahrung sowie CASE-Ausgereiftheit als Einflußfaktoren

Nicht-algorithmische Verfahren

- nicht-algorithmische ('machine learning') Methoden
- versuchen Methoden der künstlichen Intelligenz auf das Problem der Aufwandsschätzung anzuwenden
- Neuronale Netze (haben sich als sehr sensitiv auf Topologie-Entscheidungen wie Anzahl der Layers erwiesen, bieten wenig Erklärungswert und werden daher insbesondere vom Management kaum akzeptiert)

- Entscheidungsbaum ('decision tree') bzw. Regressionsbaum ('regression tree'): Klassifikation der Daten entlang der verschiedenen vorhandenen Dimensionen beziehungsweise Attributen, an jedem Blatt sind die Aufwandsdaten der jeweils entsprechend klassifizierten Projekte gespeichert, Vorteile: einfache Interpretation, mögliche Automatisierung, Behandlung diskreter Variablen

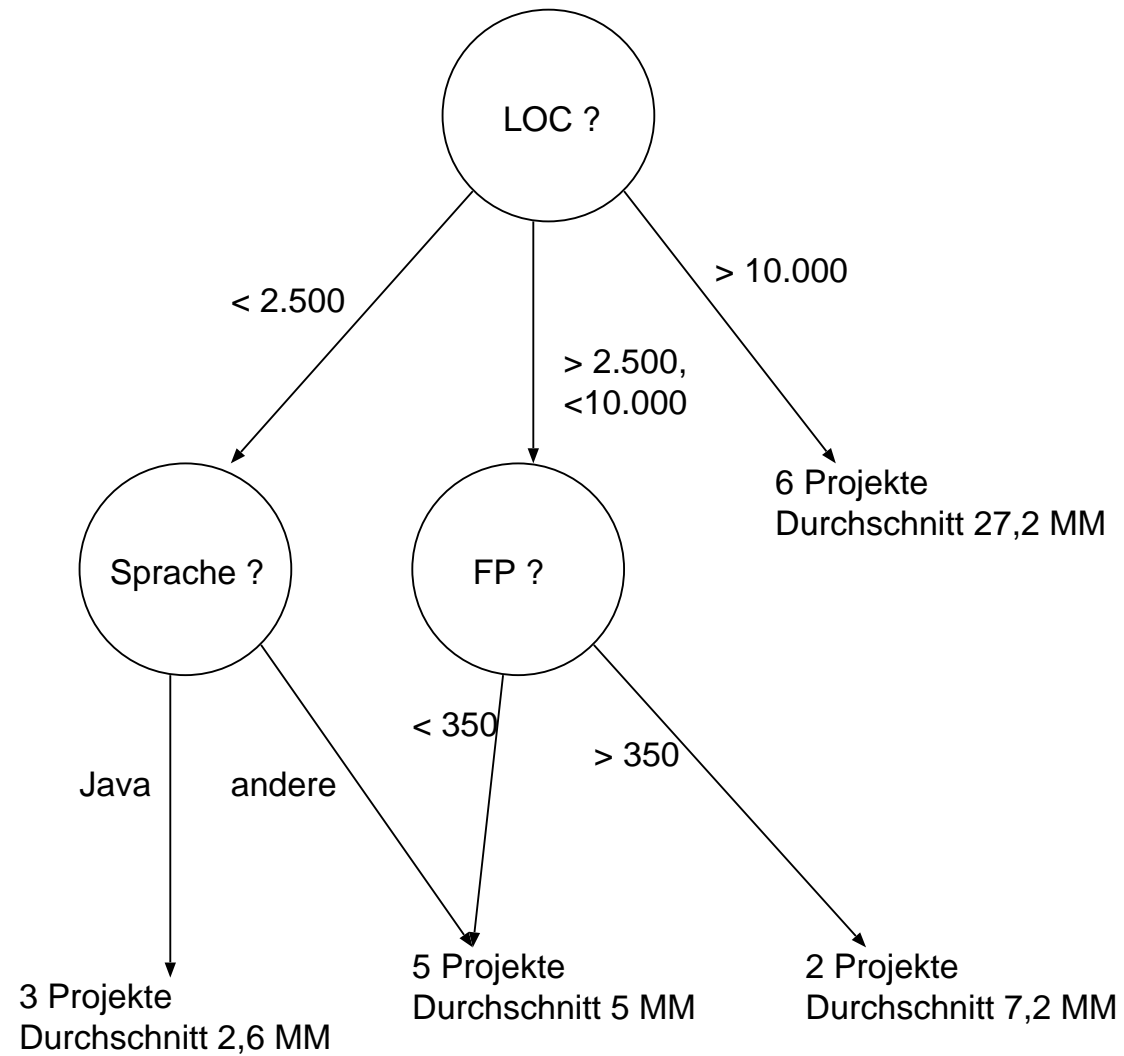


Abbildung 5: Regression Tree

- Case-based Reasoning: analogie-basierte Schätzung, jeder erfaßte Fall wird anhand der vorhandenen Information (Reihe von Attributen wie LOC, Function Points,...) charakterisiert, bei einem neuen Fall werden die ähnlichsten (z.B. 'Nearest Neighbor', Distanzmaße) gesucht und Schätzung (z.B. Mittelwert, eventuell mit Ähnlichkeit gewichtet) aus diesen (ähnlichster Fall? n ähnlichste Fälle? alle Fälle über bestimmter Ähnlichkeitsschranke?) abgeleitet

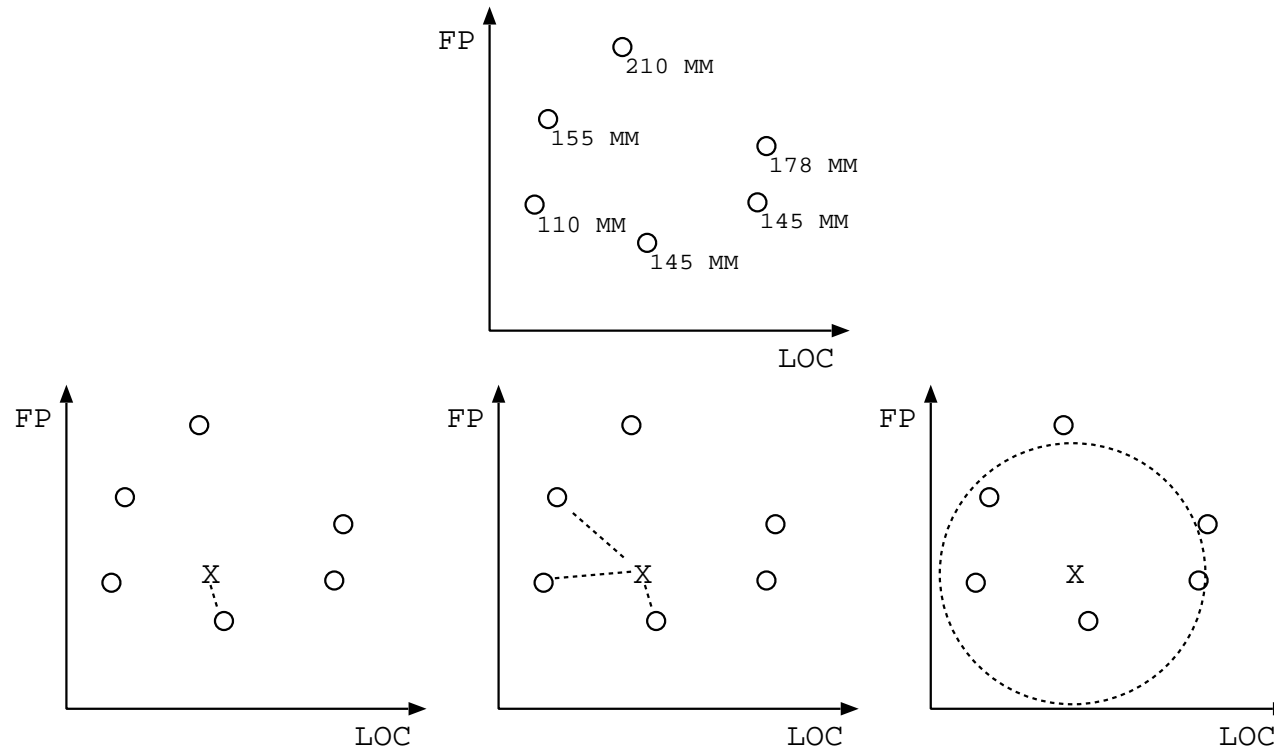


Abbildung 6: Analogie-basierte Schätzung

Einsatz in Organisationen, Datengewinnung und Validierung

- viele Modelle enthalten organisationsspezifische Koeffizienten, die mittels einiger Projekte geschätzt wurden: Kalibrierung notwendig
- dazu ist eine Datenaufzeichnung über vergangene Projekte notwendig (Projektdatenbank)
- oft nicht vorhanden, nicht einheitlich erhoben (z.B. LOC) oder nicht alle Attribute vorhanden (z.B. Function Points)
- falls vollständig: Anwendung von Methoden wie Regressionsanalyse oder Bayessche Analyse zur Kombination der vorliegenden Daten mit a priori Expertenwissen zu einem posteriori Modell

- Validierung: Prozeß, mit dem in einer gegebenen Umgebung die Genauigkeit des Systems durch empirische Mittel, das heißt durch den Vergleich des Verhaltens des Modells mit bekannten Datenpunkten, festgestellt wird (im Endeffekt Vergleich einer Schätzung mit dem tatsächlichen Aufwand)
- normalerweise Kalibrierung und Validierung nötig:
 - zuerst Kalibrierung, um Modell anzupassen, dann Validierung, um seine Qualität beurteilen zu können
 - man darf nicht Validierung anhand von Projekten durchführen, die man zur Kalibrierung verwendet hat (das Modell beinhaltet ja sonst schon das entsprechende Wissen), daher entweder
 - vorhandene Projekte zufällig in zwei Gruppen teilen ('holdout sample') oder

- jack-knifing (jeweils ein Fall wird aus der Datenbasis entfernt und unter Verwendung der verbleibenden anderen geschätzt)

- Kennzahlen:

- objektive Maßzahlen, um die Leistung verschiedener Modelle vergleichbar zu machen (welches Modell sagt den realen Aufwand besser vorher?)
- beruhen immer auf einem Vergleich von E (tatsächlicher Aufwand eines Software-Projektes) und \hat{E} (vom Modell geschätzter Aufwand)
- relativer Fehler RE ('relative error') bei einem Projekt

$$RE = \frac{E - \hat{E}}{E} \quad (10)$$

sowie durchschnittlicher relativer Fehler \overline{RE} über mehrere Projekte (Problem: starke Überschätzungen können durch starke Unterschätzungen über mehrere Projekte hinweg ausgeglichen werden)

- Größe des relativen Fehlers MRE ('magnitude of relative error')

$$MRE = |RE| = \left| \frac{E - \hat{E}}{E} \right| \quad (11)$$

sowie durchschnittliche Größe des relativen Fehlers \overline{MRE} (Problem: Überschätzungen werden wegen fehlender Schranke mehr bestraft als Unterschätzungen - der Aufwand kann maximal um 100% unter-, aber um mehrere hundert % bzw. unbegrenztüberschätzt werden)

- Anzahl der Voraussagen auf einem gewissen Niveau l , $PRED(l)$ ('prediction at level l '), definiert mit

$$PRED(l) = \frac{k}{n} \quad (12)$$

bei k Projekten mit $MRE \leq l$ in einer Datenbasis von insgesamt n Projekten ($PRED(0.25) = 0.83$: bei 83 Prozent der Projekte lagen die vorhergesagten Werte in einem Bereich von 25 Prozent der tatsächlichen Werte)

Fallbeispiel für Aufwandsschätzung im Unternehmen

- Problemstellung: Schätzung für ein neues Software-Projekt
- Beschreibung des Projektes
 - Datenbank über Kunden, Kontakte mit diesen, Verkäufen,...
 - CRM-Lösung
 - soll von Außendienstmitarbeitern zur Erfassung und für einfache Abfragen auf Laptops eingesetzt werden
 - intern für Auswertungen, Mailing-Aktionen,...
 - Programmiersprache Java

- objektorientiert (Klassen Mitarbeiter, Kunde, Produkt,...)
- Programmierer sind Java-erfahren, jedoch nicht mit CRM
- Schritt 1: Projektdatenbank erstellen
 - Annahme A: Organisation hat schon Projekte durchgeführt
 - Annahme B: noch keine Projektdatenbank vorhanden
 - Schritt 1a: Attribute für die einzelnen Projekte festlegen (z.B. in Abhängigkeit von angestrebten Schätzungsverfahren, wenn schon bekannt)
 - * Projektnummer (Schlüssel)
 - * Programmiersprache

- * LOC
- * Function Points
- * Erfahrung der Programmierer mit Sprache und Projekt
- * Aufwand
- Schritt 1b: Attribute definieren
 - * Projektnummer: Numerisch
 - * Programmiersprache: 1-3, 1...Java, 2...C++, 3...Perl
 - * LOC: ohne Kommentare und Leerzeilen, in jeweiliger Programmiersprache
 - * Function Points: Numerisch

- * Erfahrung der Programmierer mit Sprache und Projekt: von 1...keinerlei Erfahrung bis 10...Erfahrung in Sprache, mit Projektarbeit allgemein, Projektart und Kunden
- * Aufwand: in Mannmonaten
- Schritt 1c: Attributsausprägungen für alle Projekte soweit möglich aus Aufzeichnungen erheben
 - * Ergebnis

Case Summaries^a

	PROJ_NUM	LANG	LOC	FP	ERFAHR	AUFWAND
1	1	1	25000	500	3	95
2	2	1	20000	378	4	56
3	3	2	125000	.	9	110
4	4	2	70000	.	2	92
5	5	2	90000	.	.	57
6	6	1	87000	.	6	44
7	7	3	15000	79	5	91
8	8	3	72000	300	5	.
9	9	2	178000	2000	4	.
10	10	2	110000	1350	2	157
11	11	1	.	.	1	69
12	12	1	.	.	1	31
13	13	3	55000	210	6	101
14	14	2	31000	120	7	41
15	15	1	.	89	7	60
Total N	15	15	12	9	14	13

a. Limited to first 100 cases.

Abbildung 7: Schritt 1c

- Schritt 1d: notwendige Konvertierungen durchführen
 - * LOC zwischen Programmiersprachen nicht vergleichbar
 - * Beispiellösung: alles auf Java LOC konvertieren
 - * Vorgehensweise: mit Werten aus Literatur in Function Points und dann in Java umrechnen (Java: 1FP = 53LOC, C++: 1FP = 55LOC, Perl: 1FP = 27LOC, daher 1LOC C++ = $53/55$ (0.964) LOC Java und 1LOC Perl = $53/27$ (1.963) LOC Java)
 - * Ergebnis

Case Summaries

	PROJ_NUM	LANG	LOC	FP	ERFAHR	AUFWAND
1	1	1	25000	500	3	95
2	2	1	20000	378	4	56
3	3	2	120500	.	9	110
4	4	2	67480	.	2	92
5	5	2	86760	.	.	57
6	6	1	87000	.	6	44
7	7	3	29445	79	5	91
8	8	3	141336	300	5	.
9	9	2	171592	2000	4	.
10	10	2	106040	1350	2	157
11	11	1	.	.	1	69
12	12	1	.	.	1	31
13	13	3	107965	210	6	101
14	14	2	29884	120	7	41
15	15	1	.	89	7	60
Total N	15	15	12	9	14	13

Abbildung 8: Schritt 1d

- Schritt 1e: Erhebung unbekannter Attribute

- * fehlende LOC: nacherfassen oder aus Function Points, wenn bekannt, errechnen (Projekt 11 und 12: Nacherfassung, Projekt 15: 89 FP in Java LOC sind $89 \cdot 53$ LOC)
- * fehlende FP: nacherfassen (Analyse, Designdokumentation) oder aus LOC errechnen (Projekt 3,4,5: Nacherfassung, Projekt 6,11,12 Errechnung durch Java $LOC/53$)
- * Alternative zu Werten aus Literatur: eigenen Umrechnungsfaktor aus bekannten Datenpunkten berechnen (Projekt 1: $1FP = 50LOC$, 2: 52.9, 7: 372.7, 8: 471.1, 9: 85.8, 10: 80, 13: 514.1, 14: 249, Durchschnitt $1FP = 234.4LOC$), jedoch problematisch bei wenig Daten
- * fehlende Erfahrung: schätzen

- * fehlender Aufwand: nacherfassen oder mit Aufwandsschätzungsverfahren aus anderen Daten schätzen (äusserst fragwürdig, da später die Schätzungen bei diesen Projekten natürlich genau stimmen werden)
- * Ergebnis

Case Summaries

	PROJ_NUM	LANG	LOC	FP	ERFAHR	AUFWAND
1	1	1	25000	500	3	95
2	2	1	20000	378	4	56
3	3	2	120500	1500	9	110
4	4	2	67480	467	2	92
5	5	2	86760	420	4	57
6	6	1	87000	1642	6	44
7	7	3	29445	79	5	91
8	8	3	141336	300	5	85
9	9	2	171592	2000	4	235
10	10	2	106040	1350	2	157
11	11	1	21000	396	1	69
12	12	1	11000	208	1	31
13	13	3	107965	210	6	101
14	14	2	29884	120	7	41
15	15	1	4717	89	7	60
Total N	15	15	15	15	15	15

Abbildung 9: Schritt 1e

- Schritt 2: Vornahme erster Analysen

- Korrelationen: interessant sind LOC-FP, LOC-Aufwand, FP-Aufwand (welche Variablen haben tatsächlich Einfluß auf Aufwand?)

Correlations

		LANG	LOC	FP	ERFAHR	AUFWAND
LANG	Pearson Correlation	1,000	,551*	-,100	,282	,332
	Sig. (2-tailed)	.	,033	,723	,309	,226
	N	15	15	15	15	15
LOC	Pearson Correlation	,551*	1,000	,671**	,222	,696**
	Sig. (2-tailed)	,033	.	,006	,426	,004
	N	15	15	15	15	15
FP	Pearson Correlation	-,100	,671**	1,000	,114	,648**
	Sig. (2-tailed)	,723	,006	.	,685	,009
	N	15	15	15	15	15
ERFAHR	Pearson Correlation	,282	,222	,114	1,000	-,061
	Sig. (2-tailed)	,309	,426	,685	.	,830
	N	15	15	15	15	15
AUFWAND	Pearson Correlation	,332	,696**	,648**	-,061	1,000
	Sig. (2-tailed)	,226	,004	,009	,830	.
	N	15	15	15	15	15

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

Abbildung 10: Schritt 2

- Schritt 3: Aufwandsschätzung mit einfacher Produktionsfunktion
 - Auswahl eines Modell (einer Produktionsfunktion)
 - * zuerst Betrachtung über Scatterplots
 - * linear: $E = a_0 + a_1S$
 - * quadratisch: $E = a_0 + a_1S + a_2S^2$
 - Regression zur Bestimmung der Parameter
 - * linear: $E = 41.45 + 0.0007S$ ($R^2 = 0.48$)
 - * quadratisch: $E = 70.3 - 0.0006S + 0.000000008S^2$ ($R^2 = 0.62$)
 - Schätzung der Grösse der Software in LOC

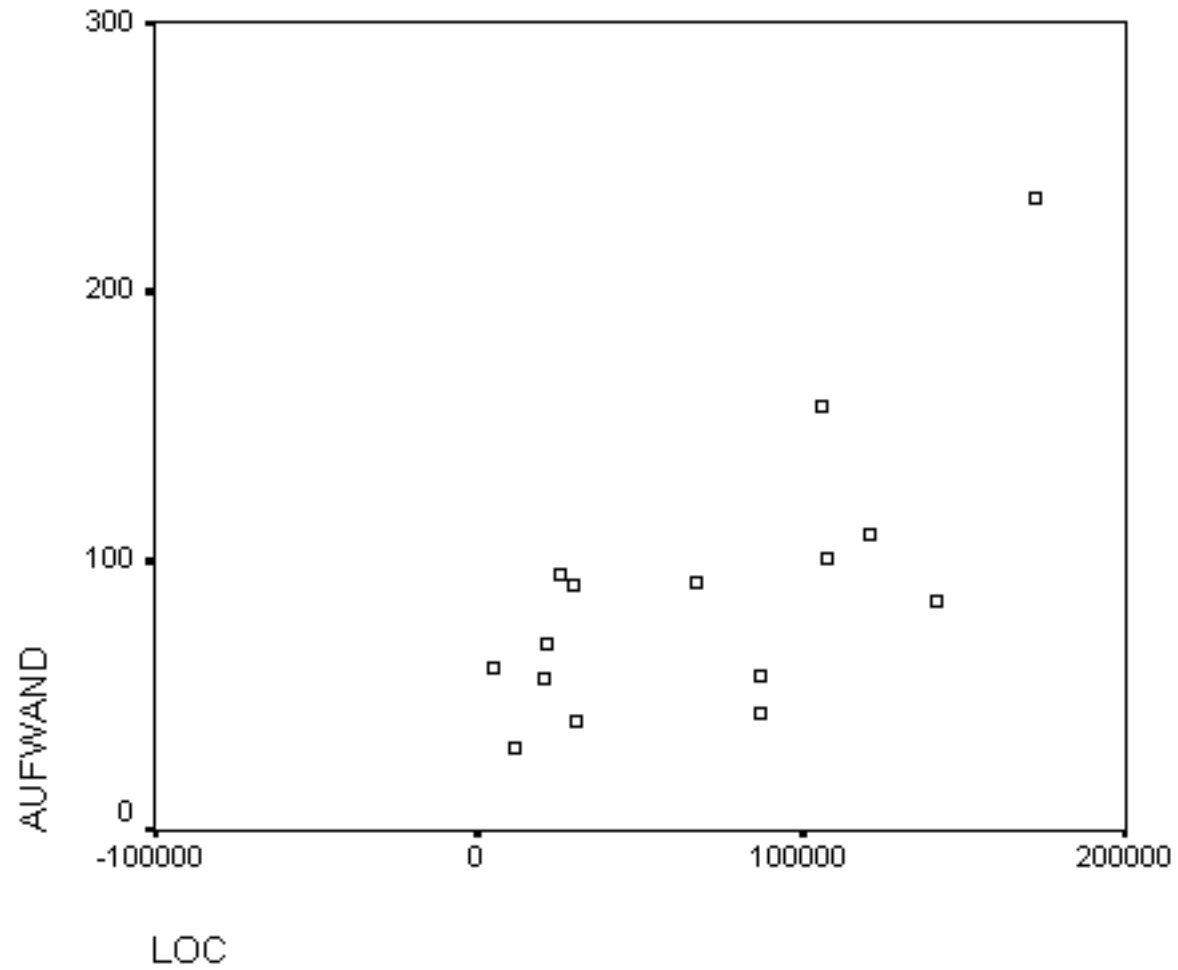


Abbildung 11: Schritt 3: Scatterplot

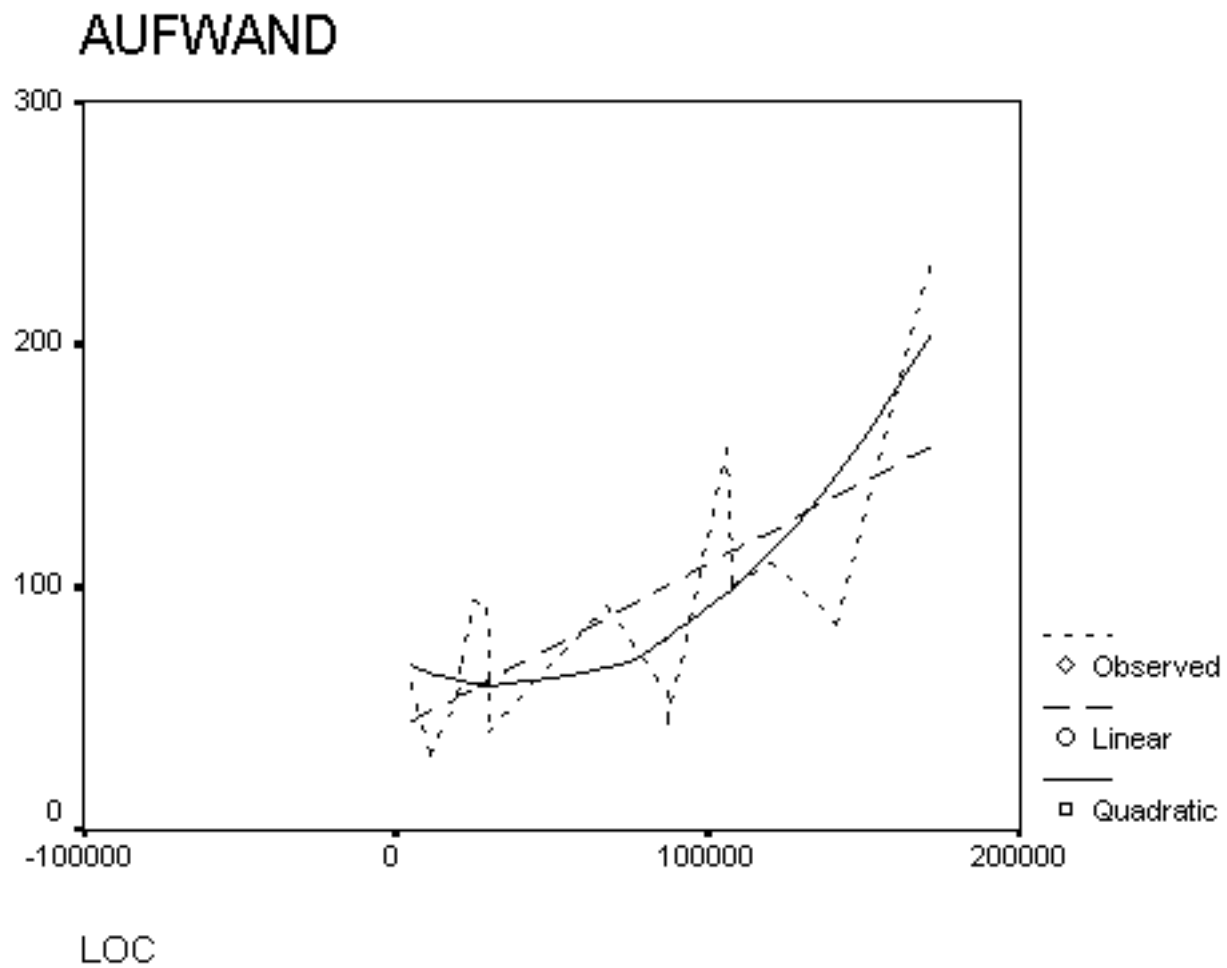


Abbildung 12: Schritt 3: Regression

- * beispielsweise aus Vergleichen mit alten Projekten
- * Schätzung in unserem Fall angenommen ca. 85000 LOC

- Schätzung

- * linear: $E = 41.45 + 0.0007 * 85000 = 100.95$

- * quadratisch: $E = 70.3 - 0.0006 * 85000 + 0.000000008 * 85000^2 = 77.1$

● Schritt 4: Aufwandsschätzung mit Function Points

- Schätzung Function Points

- * Errechnung aus LOC (da diese auf reiner Schätzung basieren eher fragwürdig, wird nicht verwendet)

- * Externe Inputs: Außendienstmitarbeiter erfassen Kunde, Auftrag, Kontakt (Average, Complex, Average, anhand vor allem Anzahl Datenelemente), Innendienst erfasst Produkte (Complex), Innendienst startet Auswertungen (Average), ergibt $3 \times \text{Average}(4)$ und $2 \times \text{Complex}(6) = 12 + 12 = 24$
- * Externe Outputs: Außendienstmitarbeiter bekommen Informationen zu Kunden/Ansprechpartner, bisherigen Aufträgen, bisherigen Kontakten, Produkten (alle Average, da multi-column), Innendienst bekommt Auswertungen (Complex), ergibt $4 \times \text{Average}(5)$ und $1 \times \text{Complex}(7) = 20 + 7 = 27$
- * Logische Interne Files (jede größere logische Datengruppe, anhand von Record Types und Datenelementen, Performance und Recovery, die von Applikation generiert, benutzt oder gewartet werden): jeweils für Kunden/Ansprechpartner,

Aufträge, Kontakte, Produkte (Average, Aufträge und Produkte Complex), dazu Mitarbeiter (Average), ergibt $3 \times \text{Average}(10)$ und $2 \times \text{Complex}(15) = 30 + 30 = 60$

* Externe Schnittstellendateien: eine Datei zu Rechnungswesen und eine zu Personalverrechnung für Boni (beide Complex), ergibt $2 \times \text{Complex}(10) = 20$

* Externe Abfragen (die auch sofort Output bringen): Management Informationssystem, OLAP-Abfragen, einmal über Mitarbeiter, Kunden und Produkte (Complex), ergibt $3 \times \text{Complex}(6) = 18$

* Unadjusted Function Points: Summe der bisherigen = $24 + 27 + 60 + 2$

* Technical Complexity Factor: Data Communications(5), Distributed Functions(3), Performance(4), Heavily Used Configuration(3), Transaction Rate(4), Online Data Entry(5), End User Efficiency(5), Online Update(5), Complex Processing(3), Reusability(4), Installation Ease(1), Operational Ease(3), Multiple Sites(2), Facilitate Change(5),
 $=0.65+(0.01 \times (5+3+4+\dots))=0.65+0.52=1.17$

* Adjusted Function Points= $149 \times 1.17=174$

- Aufwand nach Albrecht und Gaffney: $E = 0.355 \times FP - 88$

- Erstellung eines eigenen Modells (Regression, diesmal mit FP als unabhängiger Variable)

* linear: $E = 54.35 + 0.0527FP$ ($R^2 = 0.42$)

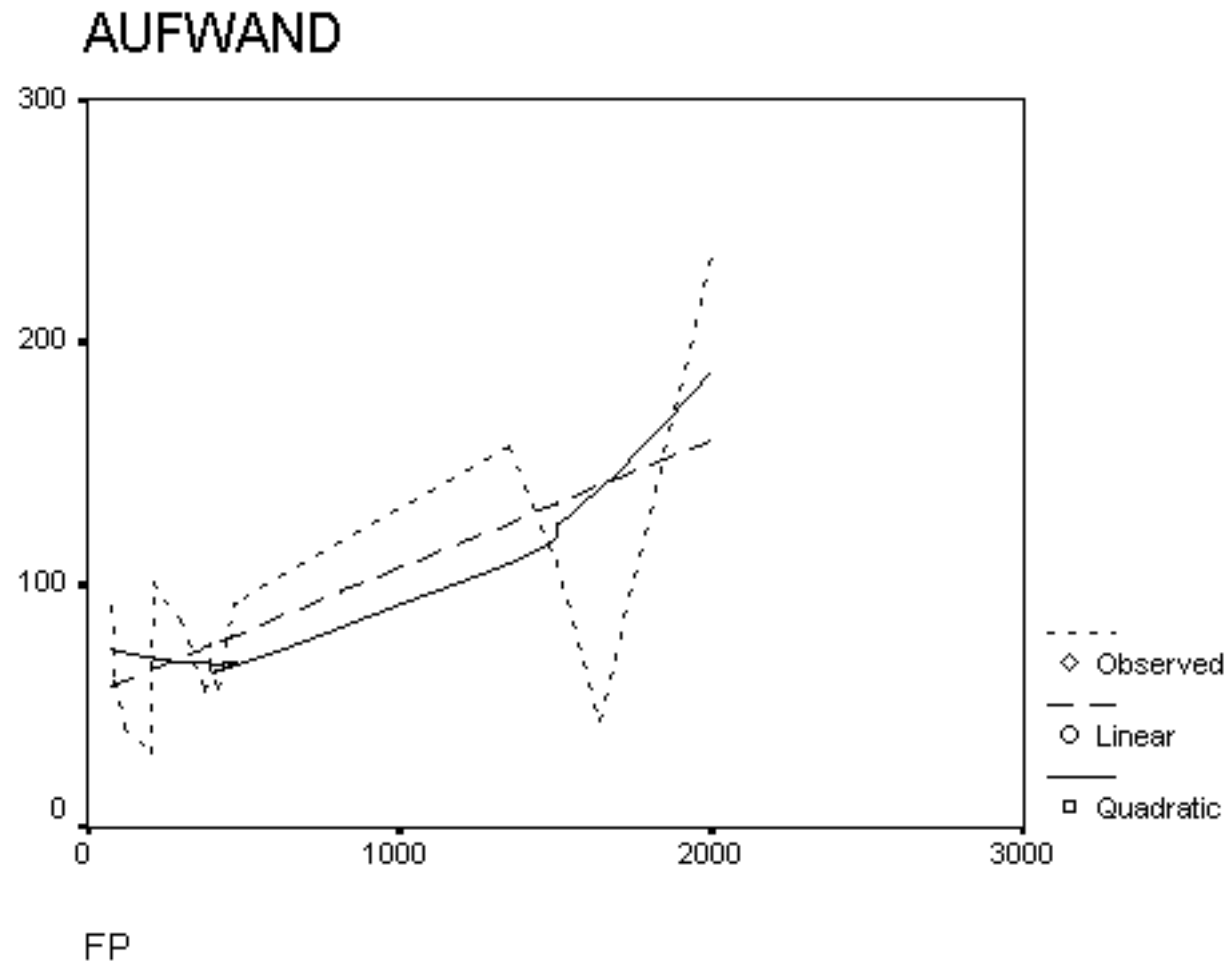


Abbildung 13: Schritt 4: Regression

* quadratisch: $E = 76.42 - 0.0416FP + 0.000049FP^2$ ($R^2 = 0.48$)

- Schätzung

* Albrecht und Gaffney: $E = 0.355 \times 174 - 88 = -26.23$

* Regression linear: $E = 54.35 + 0.0527 * 174 = 63.52$

* Regression quadratisch: $E = 76.42 - 0.0416 * 174 + 0.000049 * 174^2 = 70.39$

● Schritt 5: Aufwandsschätzung mit COCOMO

- Kalibrierung

* für COCOMO 81 müssen die Projekte auf die 3 Modi aufgeteilt werden und dann entsprechende Formeln geschätzt werden

- * daher: neue Variable Modus (1-3) für Projektdatenbank
 - * neue Gleichung für Organic: $E = 66.12S^{0.0027}$
 - * neue Gleichung für Semi-detached: $E = 22.36S^{0.286}$
 - * neue Gleichung für Embedded: $E = 0.17S^{1.377}$
- Schätzung Grösse, entweder
- * direkt LOC (wie oben): 85000 oder
 - * über FP (wie oben), umgerechnet in LOC mit hauseigenem Faktor (1FP entspricht in Java 234.4 LOC): $174 * 234.4 = 40800$
- COCOMO 81
- * Bestimmung Modus: Semi-detached (höhere Grösse, einige Nebenbedingungen, Programmierer jedoch z.T. erfahren)

- * Parameterbestimmung $m(x)$: Reliability low (0.88), Database Size high (1.08), Product Complexity nominal (1.00), Time execution constraint nominal (1.00), Main storage constr. high (1.06), Virtual machine volatility low (0.87), computer turnaround time low/interactive (0.87), Analyst capability nominal (1.00), Applications experience low (1.13), Programmer capability high (0.86), Virtual machine experience high (0.90), Programming language experience high (0.95), Use of modern prog. practices high (0.91), Use of software tools high (0.91), Required development schedule nominal (1.00)
- * daher $m(x) = 0.88 * 1.08 * 1 * 1 * 1.06 * \dots = 0.58$
- * verwendet wird LOC Schätzung (85000), also in Tausenden 85
- * Schätzung mit eigener Kalibrierung: $E = 22.36 * 85^{0.286} * 0.58 = 46.21$

* Schätzung mit vorgegebener Formel ($E = 3.0S^{1.12}m(x)$): $E = 3.0 * 85^{1.12} * 0.58 = 252.06$

- COCOMO II

* Skalenfaktoren: Precendentedness nominal (3.72), Development Flexibility high (2.03), Architecture/Risk Resolution nominal (4.24), Team Cohesion high (2.19), Process Maturity nominal (4.68), $E = 0.91 + 0.01 \times (3.72 + 2.03 + 4.24 + 2.19 + 4.68) = 1.0786$

* Umgebungsfaktoren: Required Reliability low (0.92), Data high (1.14), Product Complexity nominal (1.00), Developed for Reusabiliyt very high (1.15), Documentation nominal (1.00), Time execution constraint nominal (1.00), Main storage constr. high (1.05), Platform volatility low (0.87), Analyst capability

nominal (1.00), Programmer capability high (0.88), Personnel continuity nominal (1.00), Applications experience low (1.10), Platform experience high (0.91), Language and tool experience high (0.91), Use of software tools high (0.90), Multisite development very high (0.86), Required development schedule nominal (1.00), ergibt $0.92 * 1.14 * 1.00 * \dots = 0.684$

* verwendet wird Schätzung über FP (umgerechnet in LOC, in Tausenden)

* Schätzung: $Effort = 2.94 * 40.8^{1.078} * 0.684 = 109.57$

- Schritt 6: Aufwandsschätzung mit analogie-basierter Schätzung

- verwendet wird ANGEL

- Attribute bestimmen: signifikant korreliert mit Aufwand sind FP und LOC
- Auswahlregeln und Verknüpfung festlegen: die 3 naheliegendsten Fälle werden verwendet, alle Attribute gleichgewichtet
- Attribute für neues Projekt bestimmen: 85000 LOC und 174 FP (siehe oben)
- Schätzung: 83.33 (verwendet werden Projekte 5(Abstand 0.09), 13(0.1) und 4(0.13))
- Schritt 7: Vergleich, Validierung und Auswahl
 - Kalibrierung ist immer unter Verwendung aller Projekte passiert
 - eigentlich sollten Holdout Sample oder Jack-Knifing zur Anwendung kommen

Case Summaries

	PROJ_NUM	AUFWAND	LOCSCHLI	LOLIMRE	LOCSCHQU	LOQUMRE	
1	1	95	58,95	,38	60,30	,37	
2	2	56	55,45	,01	61,50	,10	
3	3	110	125,80	,14	114,16	,04	
4	4	92	88,69	,04	66,24	,28	
5	5	57	102,18	,79	78,46	,38	
6	6	44	102,35	1,33	78,65	,79	
7	7	91	62,06	,32	59,57	,35	
8	8	85	140,39	,65	145,31	,71	
9	9	235	161,56	,31	202,90	,14	
10	10	157	115,68	,26	96,63	,38	
11	11	69	56,15	,19	61,23	,11	
12	12	31	49,15	,59	64,67	1,09	
13	13	101	117,03	,16	98,77	,02	
14	14	41	62,37	,52	59,51	,45	
15	15	60	44,75	,25	67,65	,13	
Total	N	15	15	15	15	15	
	Minimum	1	31	44,75	,01	59,51	,02
	Maximum	15	235	161,56	1,33	202,90	1,09
	Mean	8,00	88,27	89,5036	,3959	87,7032	,3547

Abbildung 14: Schritt 7: Validierung (Schätzung mittels Produktionsfunktion basierend auf LOC - linear und quadratisch)

Case Summaries

		PROJ_NUM	AUFWAND	FPLI	FPLIMRE	FPQU	FPQUMRE
1		1	95	80,70	,15	67,87	,29
2		2	56	74,27	,33	67,70	,21
3		3	110	133,40	,21	124,27	,13
4		4	92	78,96	,14	67,68	,26
5		5	57	76,48	,34	67,59	,19
6		6	44	140,88	2,20	140,22	2,19
7		7	91	58,51	,36	73,44	,19
8		8	85	70,16	,17	68,35	,20
9		9	235	159,75	,32	189,22	,19
10		10	157	125,50	,20	109,56	,30
11		11	69	75,22	,09	67,63	,02
12		12	31	65,31	1,11	69,89	1,25
13		13	101	65,42	,35	69,84	,31
14		14	41	60,67	,48	72,13	,76
15		15	60	59,04	,02	73,11	,22
Total	N	15	15	15	15	15	15
	Minimum	1	31	58,51	,02	67,59	,02
	Maximum	15	235	159,75	2,20	189,22	2,19
	Mean	8,00	88,27	88,2853	,4315	88,5671	,4472

Abbildung 15: Schritt 7: Validierung (Schätzung mittels Produktionsfunktion basierend auf FP - linear und quadratisch)

Case Summaries

	PROJ_NUM	AUFWAND	COCOKAL	COKALMRE	COCOBOE	COBOEMRE	
1	1	95	66,70	,30	70,48	,26	
2	2	56	66,66	,19	55,76	,00	
3	3	110	123,26	,12	1131,06	9,28	
4	4	92	74,58	,19	335,58	2,65	
5	5	57	66,92	,17	260,28	3,57	
6	6	44	25,76	,41	446,05	9,14	
7	7	91	66,73	,27	83,69	,08	
8	8	85	153,48	,81	1369,64	15,11	
9	9	235	200,56	,15	1728,62	6,36	
10	10	157	103,31	,34	970,21	5,18	
11	11	69	53,41	,23	90,78	,32	
12	12	31	44,39	,43	44,00	,42	
13	13	101	318,41	2,15	568,07	4,62	
14	14	41	66,73	,63	85,00	1,07	
15	15	60	66,40	,11	12,23	,80	
Total	N	15	15	15	15	15	
	Minimum	1	31	25,76	,11	12,23	,00
	Maximum	15	235	318,41	2,15	1728,62	15,11
	Mean	8,00	88,27	99,8193	,4328	483,4301	3,9236

Abbildung 16: Schritt 7: Validierung (Schätzung mittels COCOMO - kalibriert und Original nach Boehm)

- Ergebnisse daher tendenziell eher zu gut
- Ergebnis

- Vergleich der Schätzungsergebnisse für neues Projekt

Modell	Ergebnis
LOC Regression linear	100.95
LOC Regression quadratisch	77.10
FP Albrecht/Gaffney	-26.23
FP Regression linear	53.52
FP Regression quadratisch	70.39
COCOMO 81 Original	252.06
COCOMO 81 kalibriert	46.21
COCOMO II	109.57
Analogie	83.33
Maximum (ohne FP A/G)	252.06
Minimum (ohne FP A/G)	46.21
Mittelwert (ohne FP A/G)	99.14