Diplomarbeit

Aufwandsabschätzungen mit Softwaremetriken -

Theoretische Methoden und deren Praxiseinsatz

Michael Schusser

(duckula@uni-paderborn.de 100273.3323@compuserve.com)

Erklärung

Hiermit erkläre ich, daß die vorliegende Diplomarbeit selbständig und nur unter Benutzung der angegeben Literatur angefertigt habe.

Michael Schusser, 14.12.1996

Inhalt

1 Vorwort

Softwareprojekte werden heute immer noch ohne genaue Projektstruktur durchgeführt. Die Planung und Analyse der zu erstellenden Teile des Softwareprojektes erfolgt sehr häufig noch ohne gezieltes Vorgehen. Insbesondere im Bereich der Aufwandsabschätzungen werden Methoden eingesetzt, die keiner objektiven Überprüfung standhalten. Softwaremetriken sind heute immer noch umstritten. Sie sind für viele fremd und noch nicht sehr weit verbreitet; dies führt zur Skepsis.

Aufgrund eigener Erfahrungen mit Softwareprojekten habe ich die Diplomarbeit in diesem Bereich durchgeführt. Diese Diplomarbeit wurde in Zusammenarbeit von Herrn Prof. Dr. Szwillus an der Universität Gesamthochschule Paderborn und Herrn Dipl. Ing. Furchert von der Behringwerke AG Abteilung EMR-Automatisierungstechnik betreut. Die Abteilung der Automatisierungstechnik führt neben der Betreuung von Anwendern auch eigene Softwareprojekte für die am Standort befindlichen Firmen durch. Das im Rahmen dieser Diplomarbeit entstandene Tool, soll für die Verwaltung von Projekten und deren Metriken verwendet werden. Es soll eine Schätz- und Vergleichsbasis für Softwareprojekte liefern. Ein möglicher Einsatz währe z.B. die Überprüfung von externen Angeboten (z.B. sind die Angebote unrealistisch, können die Termine deshalb nicht eingehalten werden).

An dieser Stelle möchte ich meinen beiden Betreuern für die Unterstützung während meiner Diplomarbeit danken, ebenso Herrn Dr. Kuchler, der als Leiter der EMR-Abteilung diese Diplomarbeit ermöglicht hat. Natürlich bedanke ich mich auch für die Unterstützung der Kollegen der EMR-Abteilung, von denen ich einige schon in meiner Ausbildungszeit in der EMR-Abteilung kennengelernt habe.

Wetter-Mellnau, 14.12.1996

2 Aufgabenstellung

Die modernen Softwaresysteme haben eine Komplexität erlangt, die es nicht mehr erlauben, ein System ohne Planung zu erstellen. Um das System in dem erwarteten Zeit- und Kostenrahmen fertigzustellen, müssen neben dem eigentlichen produzieren der Software auch Verwaltungsaufgaben bewältigt werden, die meist in der vorherigen Zeitkalkulation nicht berücksichtigt wurden. Aber auch die Aufwandsabschätzungen des gesamten Softwareentwicklungsprozesses beruhen zum größten Teil noch auf eigenen Erfahrungswerten, die man aus früheren Projekten erhält. Viele dieser Erfahrungswerte können aber nicht direkt auf andere bzw. neue Projekte übertragen werden, da man die Aufgaben nicht vergleichen kann. Wer würde schon ein Brückenbauprojekt mit einem Hochhausbauprojekt vergleichen.

Die Aufwandsabschätzungen in anderen Bereichen wie z.B. im Baubereich sind heutzutage sehr genau. Für sie existieren sogar zum Teil direkte Aufwandsfaktoren (z.B. zwanzig Quadratmeter Fliesen verlegen benötigt fünf Personenstunden und Material). Ziel dieser Diplomarbeit soll nun die Entwicklung eines Tools sein, das ähnlich geringe Abschätzungsfehler für Softwareprojekte liefert.

Zu diesem Zweck sollen zunächst die theoretischen Methoden und deren Modelle untersucht werden. In diesem ersten Teil der Diplomarbeit wird unter den untersuchten Modellen eine Auswahl getroffen, die dann später im zweiten Teil, der Entwicklung des Tools verwendet werden. Diese Auswahl richtet sich nach der Anwendbarkeit der Methoden bzw. Modell im Praxiseinsatz.

Eines dieser Modelle sind die „Function points“. Diese Bewertungsmethode klassifiziert die einzelnen Funktionen in Gruppen. Diese werden gewichtet und zu einer Gesamtkomplexität zusammengefaßt. Die Treffsicherheit dieses Modells hängt natürlich von der Genauigkeit der Gewichtung der einzelnen Gruppen ab. Ein weiteres beliebtes Maß ist die Codegröße. Man findet sie in den Ausprägungen: ausgelieferte Zeilen Sourcecode (DSI) und unkommentierte Zeilen Sourcecode (NCSS). Dieses Modell liefert für Kodeoptimierungen meist ein negatives Produktivitätsmaß. Ein beliebtes Modell für die Abschätzung der Kostenentwicklung ist das CoCoMo-Modell. Es unterteilt die Aufgaben in Aufwandsklassen und liefert so eine Abschätzung über den gesamten

Entwicklungsprozeßes. Mit Hilfe von Faktoren kann man den Kostenaufwand bzw. Entwicklereinsatz ermitteln. Weiterhin sollen noch verschiedene Komplexitätsmodelle untersucht werden.

Nach der Analyse der ausgewählten Methoden und Modellen soll die Praxistauglichkeit untersucht werden. Zu diesem Zweck wird ein Tool entwickelt, das die Untersuchung von bereits durchgeführten Projekten verwaltet. Die so gewonnen Daten soll das Tool für neue Projekte verwenden, um die Abschätzungsverfahren zu verbessern. Hierbei kommen die untersuchten Methoden zum tragen. Die Verwaltung der Projekte wird durch das Tool in verschiedene Abschnitte untergliedert:

- Projektverwaltung: dient zum Verwalten der gesamten Projektkalkulation. Sie wird für die Erfassung bestehender Projekte verwendet. Bei der Erfassung neuer Projekte wird der Benutzer durch Daten aus alten und analysierten Projekten unterstützt.

- Analysetool: führt die Bewertung der Projekte und deren Projektteile durch und erstellt eine Bestandsliste für die Unterstützung bei der Eingabe neuer Projekte. Das Analysetool kann aber auch noch zur Überprüfung der Zeitkalkulation verwendet werden.

Die Projektverwaltung begleitet das Projekt während des gesamten Lebenszykluses. Dies ermöglicht zu jedem Zeitpunkt des Projektes einen genauen Soll/Ist-Vergleich. Auch eine Möglichkeit zur Nachkalkulation nach Projektabschluß ist gegeben. Die interne Projektstruktur soll frei definierbar sein (.z.B. Grobspezifikation/Vergabe, Feinspezifikation, Designphase, Implementierungsphase, Testphase, Nachkalkulation, etc.).

Das Analysetool vergleicht die einzelnen Komplexitätsmodelle mit- und untereinander. Abgeschlossene Projekte und deren Teilkomponenten werden anschließend in verschiedene Kategorien eingestuft und die Konstanten der einzelnen Komplexitätsmodelle werden angepaßt. Die Analyse liefert auch noch eine Vergleichsgrundlage zur Gegenüberstellung der Projekte untereinander. Dies ermöglicht z.B. die Untersuchung von Entwicklungstrategien, um deren Produktivitätssteigerungen zu ermitteln.

3 Metriken und Modelle

Das Gebiet der Metriken und Modelle für den Softwareengineeringbereich hat inzwischen einen großen Umfang erreicht, so daß es nicht sinnvoll und möglich ist, alles abzudecken. Aus diesem Grund werden in diesem Kapitel nur die wichtigsten bzw. diejenigen Metriken und Modelle beschrieben, die in dem geplanten Tool eingesetzt werden könnten. Außerdem ist es für die Einführung von Metriken nicht sinnvoll, zu viele Metriken gleichzeitig neu einzuführen. Da man so leichter die Ergebnisse der Metrikeinführung überschauen kann und gegebenenfalls Fehlentwicklungen leichter entgegenwirken kann.

3.1 Lines of Code

Eines der umstrittensten Metriken sind die „Lines of Code“ (LOC). Für die Anwendung dieser Metrik muß man zunächst genau festlegen, was alles zu den Lines of Code hinzugehört. Einige mögliche Kriterien für die Zählung sind:

- Nur ausführbare Zeilen
- Ausführbare Zeilen und Datendefinitionen
- Ausführbare Zeilen, Datendefinitionen und Kommentare
- Alle physikalischen Zeilen (NCSS)
- Zeilen unterteilt durch logische Trennzeichen (z.B. »;«)
- Ausgelieferte Zeilen Sourcecode (DSI)

Meist werden die „Lines of Code“ als KLOC (1000 LOC) angegeben. Dies gilt auch für die anderen Kodezeilen, wie z.B. KNCSS oder KDSI. Neben diesen eigentlichen Zählkriterien muß man noch zusätzlich für die Implementierung eine Form bzw. einen Stil festlegen. Diese Maßnahmen zielen auf eine Vergleichsbasis, anhand der dann später die einzelnen Projekte bzw. Projektteile untereinander verglichen werden können.

Ein großer Nachteil der Lines of Code ist das Fehlen von Gewichten für die Programmschwierigkeit. 10 KLOC eines einfachen Abfragesystems sind wesentlich

rascher realisiert als 10 KLOC einer Echtzeitanwendung. Man kann die Lines of Code auch immer nur in der gleichen Programmiersprache vergleichen. 20KLOC C-Kode kann man nicht mit 20KLOC C++ Kode vergleichen. Dieser Umstand ist natürlich für die Abschätzung des Entwicklungsaufwandes sehr hinderlich. Deshalb werden die Lines of Code meist nur als Hilfsmaß eingesetzt. Ein Beispiel dafür findet man im folgenden Kapitel über das CoCoMo-Modell.

3.2 CoCoMo-Modell

Das „constructive cost model“ kurz CoCoMo wurde von [Boehm 81] vorgestellt. Dieses Modell stellt die Menge des ausgelieferten Zeilen Sourcecode (DSI) in Relation zum Aufwand bzw. zur benötigten Entwicklungszeit:

$$P_M = a(Z)^e$$
$$T_{DEV} = c(P_M)^d$$

Gleichung 3-1

Der Aufwand P_M wird in Personenmonaten gemessen. Der Parameter Z liefert die Anzahl der ausgelieferten Zeilen Sourcecode in KDSI. Die benötigte Entwicklungszeit T_{DEV} wird ebenfalls in Monaten angegeben. Die Faktoren a, c, e, d hängen von der Schwierigkeitsklasse des Projektes ab. Diese werden im folgenden Text beschrieben.

Das CoCoMo-Modell gibt es in drei Ausprägungen:

- basic
- intermediate
- detailed

In diesem Kapitel wird nur das Modell „intermediate“ beschrieben. Die beiden anderen Ausprägungen werden durch [Boehm 81] beschrieben.

Für die Ermittlung der Faktoren a, c, e, d teilt [Boehm 81] die Projekte in drei verschiedene Schwierigkeitsklassen ein:

Organic: kleine Projekte, vertraute Applikationen

Semi-detached: liegt zwischen Organic und Embedded

Embedded: komplexe Organisation, enge Anbindung an Soft- und Hardware, viele Interaktionen, begrenzte Erfahrung bzw. Praxis

Typische Systeme der Embedded-Klasse sind Echtzeitsysteme (z.B. Transportsteuerungen, die die Wege bzw. Ziele der Pakete anhand eines Barkodes direkt wählen). Ein Vertreter der Organic-Klasse ist ein Auskunftssystem, das auf einer festen Datenbasis arbeitet.

Nach der Klassifizierung des Projektes, kann man mit Hilfe der Tabelle 3-1 die Faktoren ermitteln:

Klasse	a	e	c	d
Organic	3,2	1,05	2,5	0,38
Semi-detached	3,0	1,12	2,5	0,35
Embedded	2,8	1,20	2,5	0,32

Tabelle 3-1 : Schwierigkeitsklassen und Faktoren [Shepperd 95] p. 103

In dieser Ausprägung des CoCoMo-Modelles hat [Boehm 81] noch einen Korrekturfaktor K für den Aufwand P_M. Dieser wird aus der Bewertung der in Tabelle 3-2 aufgeführten Attribute A_i gebildet:

$$K = \prod_{i=1}^{15} A_i$$

Gleichung 3-2

Produktivitätsfaktoren	Klasse					
	V. low	Low	Nominal	High	V. high	E. high
PRODUCT ATTRIBUTES						
RELAY Required soft. Reliability	0,75	0,88	1,00	1,15	1,40	
DATA Database size		0,94	1,00	1,08	1,16	
CPLX Product complexity	0,70	0,85	1,00	1,15	1,30	1,65
COMPUTER ATTRIBUTES						
TIME Execution time constraint			1,00	1,11	1,30	1,66
STOR Main storage constraint			1,00	1,06	1,21	1,56
VIRT Virtual machine volatility		0,87	1,00	1,15	1,30	
VEXP Virtual machine experience	1,21	1,10	1,00	0,90		
TURN Computer turn-round ime		0,87	1,00	1,07	1,15	
PERSONAL ATTRIBUTES						
ACAP Analyst capability	1,46	1,19	1,00	0,86	0,71	
AEXP Application experience	1,29	1,13	1,00	0,91	0,82	
PCAP rogrammer capability	1,42	1,17	1,00	0,86	0,70	
LEXP Prog. Language experience	1,14	1,07	1,00	0,95		
PROJECT ATTRIBUTES						
MODP Use of modern programming practices	1,24	1,10	1,00	0,91	0,82	
TOOL Use of tools	1,24	1,10	1,00	0,91	0,83	
SCED Development schedule	1,23	1,08	1,00	1,04	1,10	

Tabelle 3-2 : Produktivitätsfaktoren [Boehm 81]

Mit Hilfe des oben ermittelten Korrekturfaktors K kann man den korrigierten Aufwand $P_{M'}$ ermitteln:

$$P_{M'} = KP_M$$

Gleichung 3-3

Für die Aufwandsschätzung des Projektes kann man nun nach folgendem Schema vorgehen:

1. Auswahl der Schwierigkeitsklasse des Projektes
2. Abschätzen der Projektgröße in KDSI und Ermittlung von P_M (Gleichung 3-1)
3. Ermittlung des Korrekturfaktors K
4. Berechnung des korrigierten Aufwandes $P_{M'}$
5. Berechnung der Durchführungszeit mit Hilfe von $P_{M'}$

Bei diesem Verfahren ist der Punkt 2 der Schwierigste. Für die Abschätzung der Projektgröße gibt es verschiedene Techniken:

- Bottom-Up Analyse
- Analyse über Analogien
- Orakel Analyse

Bottom-Up Analyse

Meist ist es am einfachsten, das Projekt durch seine Einzelkomponenten abzuschätzen. Durch Aufsummieren der Einzelschätzungen erhält man die gesamte Größe des Projektes. Bei dieser Technik muß man allerdings noch den Overhead, der durch die Verbindung der Einzelkomponenten bzw. Module anfällt, mit berücksichtigen. [Boehm 81] veranschlagt für den Overhead 20%, d.h. man muß die geschätzte Gesamtgröße des Projektes mit dem Faktor 1.2 korrigieren.

Analyse über Analogien

Verfügt man schon über Daten aus vorherigen Projekten, so kann man diese Technik anwenden. Hierbei versucht man in den alten Projekten Analogien und Unterschiede zu dem geplanten Projekt zu finden und einzustufen. Typische Vergleichspunkte sind z.B. die Komplexität des Projektes, die Qualitätsanforderungen oder das verwendete Programmiersystem.

	Altes Projekt	Neues Projekt
CMPLX[1]	nominal	nominal
DATA	nominal	nominal
RELY	high	high
Sprache	50% COBOL, 50% C	50% 4GL, 50% C
Größen-änderung		20 - 40%
Min.-Größe		4*1,2*(0,38*100/50)
Größe	4 KDSI	
Max.-Größe		4*1,4*(0,38*100/50)

Tabelle 3-3 : Beispiel Abschätzung mit Hilfe von Analogien [Shepperd 95] p. 107

Orakel-Analyse

Die letzte Technik wird als Orakel-Analyse bezeichnet. Sie wird nach folgendem Schema durchgeführt:

- Die Abschätzer erhalten die Spezifikation und das Schätzformular (siehe Abbildung 3-1)
- Es wird eine Diskussion über die geplanten Produktteile und deren Abschätzung durchgeführt.
- Jeder Teilnehmer führt seine eigene Schätzung durch
- Die Schätzungen werden in Schätzformulare eingetragen und den Teilnehmern wieder ausgehändigt.
- Auf den Schätzformularen wird nur der Durchschnitt aller Schätzungen und die persönliche Schätzung ausgewiesen. Alle anderen Schätzungen bleiben anonym.
- Die Ergebnisse der Schätzung werden von allen in einer Diskussion analysiert
- Die eigenen Schätzungen werden überarbeitet

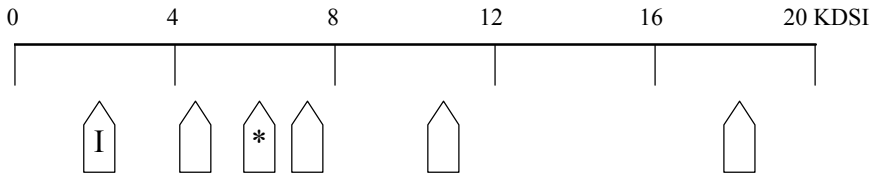
[1] Bezeichnungen aus Tabelle 3-2

- Das Verfahren wird solange wiederholt, bis die Streuung der Schätzungen hinreichend klein ist.

Projekt: **Banksystem** Datum: **04.07.1996**

Schätzer: **Mr. Bean**

Schätzrunde: **1**



Marke: Schätzung

I Eigene Schätzung

* Mittlere Schätzung

Abbildung 3-1 : Schätzformular [Shepperd 95] p. 108

3.2.1 Kalibrierung des CoCoMo-Modelles

Eine Kalibrierung des CoCoMo-Modelles geschieht durch die nicht lineare Regression. Als Basis für das Regressionsverfahren nimmt man die Gleichung 3-1 und Gleichung 3-3:

$$f_1(Z) = P_M = a(Z)^e$$
$$f_2(Z) = c(f_1(Z))^d = ca(Z)^{de} = g(Z)^p$$
mit $g = ca$ und $p = de$
$$f_3(Z) = Kf_2(Z) = Kg(Z)^p = K'(Z)^p$$
mit $K' = Kg$

Gleichung 3-4

Aus Gleichung 3-4 und den Daten aus abgeschlossenen Projekten (mit der Zuordnung: X_i = Ausgelieferte Lines of Code von Projekt i und Y_i = Aufwand in Personenmonaten für f_1 bzw. Entwicklungszeit für f_2 und f_3) kann man mit Hilfe der Summe der kleinsten Quadrate die Faktoren a, e, g, K´ und p ermitteln. Sei n die Anzahl der untersuchten Projekte:

$$a_1 = \sum_{i=1}^{n} (Y_i - f_1(X_i))^2 = \sum_{i=1}^{n} (Y_i - a(X_i)^e)^2$$
$$a_2 = \sum_{i=1}^{n} (Y_i - f_2(X_i))^2 = \sum_{i=1}^{n} (Y_i - g(X_i)^p)^2$$
$$a_3 = \sum_{i=1}^{n} (Y_i - f_3(X_i))^2 = \sum_{i=1}^{n} (Y_i - K'(X_i)^p)^2$$

Gleichung 3-5

Mit Hilfe der Regressionsmethoden kann man nun die benötigten Faktoren ermitteln und so das CoCoMo-Modell auf den eigenen Entwicklungsprozeß abstimmen.

3.3 Fuzzy-Logic Methode

Die Fuzzy-Logic Methode verwendet die Daten aus alten abgeschlossenen Projekten. Meist wird als Metrik Lines of Code verwendet, aber es können natürlich auch andere Metriken wie z.B. Function points verwendet werden. Die Projekte werden in fünf verschiedenen Klassen eingeteilt („Very Small“, „Small“, „Medium“, „Large“ und „Very Large“). Danach werden diese Klassen noch in fünf Unterklassen unterteilt. Diese Einteilung erfolgt nach dem in Tabelle 3-4 Beispiel. S_{XM} ist die mittlere Anzahl der Zeilen in der Klasse X und Δ_B die logarithmische Klassenbreite. Sie wird mit Hilfe des kleinsten (S_{MIN}) und größten (S_{MAX}) Projektes berechnet:

$$\Delta_B = \frac{\log(S_{MAX}) - \log(S_{MIN})}{2}$$

Gleichung 3-6

Unterklasse Klasse	Very Small LOC $= 10^{\log(S_{XM})-\Delta_B}$	Small LOC $= 10^{\log(S_{XM})-\Delta_B/2}$	Medium LOC S_{XM}	Large LOC $= 10^{\log(S_{XM})+\Delta_B/2}$	Very Large LOC $= 10^{\log(S_{XM})+\Delta_B}$
Very Small	1.148	1.514	2000	2630	3467
Small	4.570	6.025	8000	10.471	13.804
Medium	18.197	23.988	32.000	41.687	54.954
Large	72.444	95.499	128.000	165.958	218.776
Very Large	288.403	380.189	512.000	660.693	870.964

Tabelle 3-4 : Klasseneinteilung [Humphrey, 95] p. 104

Der eigentliche Schätzvorgang wird durch das Vergleichen der Charakteristiken des geplanten Projektes mit den abgeschlossenen Projekten durchgeführt. Hierbei wird die Klasse bestimmt, in der das zu schätzende Projekt am besten zu geordnet werden kann.

Nachteil dieser Methode ist, daß sie nur grobe Abschätzungen liefert und das man zur Verbesserung der Genauigkeit sehr viele Projekte verwalten muß und die Klassengrenzen ständig korrigiert werden müssen.

3.4 Halstead's Software science

Die Grundlage für viele moderne Modelle liefert die Software science von Halstead. Aus diesem Grund soll es etwas genauer als die anderen Modell vorgestellt werden.

3.4.1 Meßbare Eigenschaften von Algorithmen

Für die kommenden meßbaren Eigenschaften werden zunächst folgende Parameter eingeführt:

η_1	Anzahl der verschiedenen Operatoren, die in dieser Implementierung auftreten
η_2	Anzahl der verschiedenen Operanden, die in der betrachteten Implementierung vorkommen
N_1	Summe der verwendeten Operatoren, die in dieser Implementierung auftreten
N_2	Gesamte Anzahl der verwendeten Operanden, die in der betrachteten Implementierung vorkommen
$f_{1,j}$	Anzahl des j-ten Operators. [$j \in 1 \ldots \eta_1$]
$f_{2,j}$	Anzahl des j-ten Operanden. [$j \in 1 \ldots \eta_2$]

Man nennt η die Größe des Vokabulars. Es wird wie folgt gebildet:

$$\eta = \eta_1 + \eta_2$$

Gleichung 3-7

Die Programmlänge N wird durch folgende Beziehung beschrieben:

$$N = N_1 + N_2$$

Gleichung 3-8

Die Anzahl der verwendeten Operatoren bzw. Operanden wird durch folgende Summe bestimmt:

$$N_1 = \sum_{j=1}^{\eta_1} f_{1,j} \qquad N_2 = \sum_{j=1}^{\eta_2} f_{2,j}$$

Gleichung 3-9

Zur Ermittlung der minimalen Programmlänge untersuchte Halstead zunächst mit Hilfe der Kombinatorik den möglichen Aufbau eines Programmes. Hierbei fließen die oben erwähnten Operatoren und Operanden, deren Auftreten durch η_1 und N_1 bzw. η_2 und N_2 beschrieben wurden, mit ein. Man sieht natürlich sofort, daß folgende Beziehung zwischen der Vokabulargröße und der Programmlänge besteht:

$$\eta \leq N$$

Gleichung 3-10

bzw. für die Anzahl der möglichen Kombinationen ergibt sich folgende Ungleichung:

$$N \leq \eta^{\eta+1}$$

Gleichung 3-11

Da aber die Anzahl der möglichen Kombinationen für Programmiersysteme 2^N ist, kann man die Programmlänge ermitteln:

$$2^N = \eta_1^{\eta_1} \times \eta_2^{\eta_2}$$

Gleichung 3-12

Für die Abschätzung der Summe aller Operanden und Operatoren erhält man aus Gleichung 3-12:

$$N_G = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

Gleichung 3-13

Das Programmvolumen V wird aus der Menge der für die Operanden und Operatoren benötigten Bits und deren tatsächlichen Verwendung in der Implementation errechnet:

$$V = N \log_2 \eta$$

Gleichung 3-14

Das Volumen V wird in Bit gemessen. Es ist stark von dem jeweils verwendeten Programmiersystem abhängig. Systeme mit „mächtigen“ Operatoren kommen im Programm mit einer geringeren Anzahl von verwendeten Operatoren aus.

Das minimale Volumen (Potential Volume) V^* liefert ein Maß für das kleinste Programmvolumen eines Algorithmuses. Zu diesem Zweck geht man davon aus, daß man mit zwei Operatoren auskommt. Bei diesen handelt es sich um einen Funktionsaufruf bzw. Funktionsname und einer Zuweisung für das Ergebnis des Aufrufes. Die Zahl der Operanden kann man auf die Ein- und Ausgabe beschränken:

$$\eta_1^* = 2 \qquad \eta_2^* = |\text{Eingabe}| + |\text{Ausgabe}|$$

Gleichung 3-15

Nimmt man nun noch weiter an, daß jeder Operator bzw. Operand nur einmal verwendet wird:

$$N_1^* = \eta_1^* \qquad N_2^* = \eta_2^*$$

Gleichung 3-16

Durch einsetzen in Gleichung 3-14 folgt für das minimale Volumen V^*:

$$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$$

Gleichung 3-17

Auch dieses Volumen wird in Bit gemessen. Da man aber hier immer von zwei Operatoren ausgeht, ist dieses Volumen von der Wahl des Programmiersystems unabhängig.

Für spätere Analysen wird noch das „boundary Volume V^{**}“ definiert:

$$V^{**} = (\eta_1^* \log_2 \eta_1^* + \eta_2^* \log_2 \eta_2^*) \log_2 (\eta_1^* + \eta_2^*)$$

Gleichung 3-18

mit

$$\eta_1^* = 2 = \eta_1^* \log_2 \eta_1^*$$

Gleichung 3-19

folgt

$$V^{**} = (2 + \eta_2^* \log_2 \eta_2^*) \log_2 (2 + \eta_2^*)$$

Gleichung 3-20

Der Programm Level L wird aus dem Quotienten des minimalen Volumens und dem Programmvolumen gebildet:

$$L = \frac{V^*}{V}$$

Gleichung 3-21

Mit Hilfe des Levels kann man verschiedene Programmiersysteme miteinander vergleichen. Zu diesem Zweck verwendet man folgende Form von Gleichung 3-21:

$$V^* = L \times V$$

Gleichung 3-22

Da das minimale Programmvolumen systemunabhängig ist, sieht man sofort, daß die Levels der einzelnen Programmiersysteme das Programmvolumen beeinflussen. Die einfachsten Sprachen sind die sog. „potential languages“ oder Level 1 - Sprachen. Sie

haben den Vorteil, daß der Algorithmus nur aus vordefinierten Operatoren bzw. Funktionen besteht (vgl. Gleichung 3-16). Daher ist das Volumen V minimal. Level 1 - Sprachen können aber in der Praxis nicht realisiert werden.

Für die Abschätzung des Programm Levels L betrachtet man folgende Proportionen:

$$L \approx \frac{\eta_1^*}{\eta_1} \qquad L \approx \frac{\eta_2}{N_2}$$

Gleichung 3-23

Durch die Verbindung der beiden Teile von Gleichung 3-23 erhält man folgenden Schätzwert:

$$L_G = \frac{\eta_1^*}{\eta_1}\frac{\eta_2}{N_2}$$

Gleichung 3-24

Ein Hauptproblem bei den Abschätzungen ist die Ermittlung von η_2^*. Im einfachsten Fall erhält man η_2^* durch Zählung der Ein- und Ausgabeoperanden. Dies ist aber meist zu ungenau, deshalb werden „busy-on-entry“ und „busy-on-exit“ als Schätzgrundlage verwendet.

Die Nachrichtenmenge „intelligence content“ I ist wie folgt definiert:

$$I = \hat{L} \times V$$

Gleichung 3-25

Durch Einsetzten von Gleichung 3-14 und Gleichung 3-24 erhält man für die Nachrichtenmenge I folgende Beziehung:

$$I = \frac{2}{\eta_1}\frac{\eta_2}{N_2} \times (N_1 + N_2)\log_2(\eta_1 + \eta_2)$$

Gleichung 3-26

Für die Bewertung der Algorithmen werden die Algorithmenteile in „Program Purity's" eingeteilt. Unter „Program Purity" versteht man die Verfälschung der Algorithmen. Sie sind in sechs verschiedene Klassen eingeteilt. Ziel dieser Klassen ist es einen „unverfälschten" Algorithmus zu erzeugen. Eine weitere Anwendung finden diese Klassen in der Analyse der später beschriebenen Abschätzung von „Programmlänge" und „Erstellungszeit".

Impurity I : Complementary Operations

In dem Algorithmus existieren komplementäre Operationen (z.B. A+B-B+C→D). Moderene Programmsysteme erkennen solche Konstrukte und beheben sie automatisch (A+C→D).

Impurity II : Ambigous Operands

Mehrdeutige Operanden (z.B. A+B→C, C*C→C) verringern die Lesbarkeit des Kodes und erschweren dadurch das Verstehen. Abhilfe schafft hier das Entfernen der betroffenen Operation ((A+B)^2→D).

Impurity III : Synonomous Operands

Auch die Verwendung von mehren Operanden für den gleichen Inhalt führt zur Verringerung der Lesbarkeit (z.B. A+B→C_1, A+B→C_2, C_1*C_2→D). Abhilfe schafft auch hier die Beseitigung der betroffenen Operanden (z.B. A+B→C_1, C_1*C_1→D).

Impurity IV : Common Subexpressions

Gemeinsame Teilterme (z.B. (A+B)*(A+B)→C) werden normalerweise von den Programmsystemen automatisch erkannt und eliminiert. Falls diese nicht automatisch entfernt werden, muß man sie „von Hand" entfernen (z.B. (A+B)→D, D*D→C), damit sie nicht mit in die Kalkulation einfließen.

Impurity V : Unwarranted Assignment

Neben den oben erwähnten Common Subexpressions existiert noch ein komplementärer Fall. Hier wird einem Operanden nur einmal ein Wert zugewiesen und nur einmal

wiederverwendet (z.B. A+B→C, C^2→D). Zur Verringerung der Operandenzahl η_2 bzw. n_2 werden solche Operatoren entfernt (z.B. (A+B)^2→D).

Impurity VI : Unfactored Expressions

Terme, die noch nicht „ausmultipliziert“ sind, (z.B. (A+B)^2→C) können leichter verstanden werden, als solche die schon aufgelöst worden sind (z.B. A*A+2*A*B+B^2→C).

Für die Ermittlung des Programmieraufwandes, führte Halstead folgende Schritte durch:

1. Schritt: Für den Algorithmus die N Selektionen aus dem Vokabular mit η Elementen übernehmen.

2. Schritt: Für die Kodierung der Selektionen wird eine Hash-Tabelle verwendet. Auf dieser wird dann mit Hilfe der binären Suche auf das jeweilige Element zu gegriffen. Dies erfordert $\log_2 \eta$ Vergleiche.

3. Schritt: Aus den beiden vorherigen Schritten folgt, daß zur Generierung des Programms $N \times \log_2 \eta$ Denkentscheidungen „mental comparisons“ nötig sind.

4. Schritt: Das Volumen V ist definiert als:

$$V = N\log_2 \eta$$

Gleichung 3-27

Man sieht sofort, daß das Volumen die Anzahl der Denkentscheidungen ist.

5. Schritt: Jede Denkentscheidung benötigt eine Anzahl von Elementarentscheidungen ⇒ Programmlevel ist reziprok zur Programmschwierigkeit.

6. Schritt: Das Volumen V liefert die Anzahl der Denkentscheidungen. Der reziproke Programmlevel (1/L) lbeeinflusst die Anzahl der Elementarentscheidungen, die für eine Denkentscheidung benötigt wird. Die gesamte Anzahl der benötigten

Elementarentscheidungen E wird folgende Beziehung bestimmt:

$$E = \frac{V}{L}$$

Gleichung 3-28

bzw. Durch Auswertung von L folgt:

$$E = \frac{V^2}{V^*}$$

Gleichung 3-29

Aus dieser Beziehung sieht man leicht, daß der Aufwand quadratisch mit der Programmgröße steigt.

Für die Zeitdauer einer Elementarentscheidung verwendet man die Stroud'sche Zahl S. Sie zählt die Elementarentscheidungen, die ein spezieller Mensch pro Sekunde durchführen kann. Typische Werte für S sind:

$$5 \le S \le 20$$

Gleichung 3-30

Dies führt zu einem Zeitmaß für die Programmgröße:

$$T_G = \frac{E}{S} = \frac{V}{SL} = \frac{V^2}{SV^*}$$

Gleichung 3-31

Nach der Substitution bis auf die „Basisparameter", wird das Zeitmaß durch folgende Beziehung beschrieben:

$$T_G = \frac{\eta_1 N_2 N \log_2 \eta}{2 S \eta_2}$$

Gleichung 3-32

Für erste Schätzungen muß man die Programmlänge N durch die geschätzte Programmlänge N_G substituieren:

$$T_G = \frac{\eta_1 N_2 (\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2) \log_2 \eta}{2S\eta_2}$$

Gleichung 3-33

Für die Bewertung verschiedener Algorithmen, die mit dem gleichen Programmiersystem implementiert sind, wird der Sprachenlevel (Language Level) λ definiert:

$$\lambda = LV^*$$

Gleichung 3-34

Durch Substitution des Programmlevels L in Gleichung 3-28 kann man die Anzahl der Denkentscheidungen E wie folgt beschreiben:

$$E = \frac{(V^*)^3}{\lambda^2}$$

Gleichung 3-35

Der Sprachenlevel λ ist ein Maß für die Mächtigkeit des verwendeten Programmiersystems und wird ebenfalls für die im folgenden Kapitel beschriebenen Abschätzung der Programmlänge bzw. Programmierzeit verwendet.

3.4.2 Abschätzung der Programmlänge und Programmierzeit zu Beginn

Für Abschätzung der Programmlänge und deren Erstellungszeit benötigt man zwei Parameter:

η_2^*: die minimale Anzahl der Operanden

λ: der Sprachenlevel für das System, das verwendet wird.

Mit Hilfe der Purity-Klassen und der vorherigen Gleichungen kommt man zu folgendem Gleichungssystem:

$$\eta^* = 2 + \eta_2^*$$
$$\eta_2 = \eta_2^* \log_2(\frac{\eta^*}{2}) \frac{\eta_1 - 2}{\eta^*} + \eta_2^*$$
$$(\eta^* \log_2 \eta^*)^2 = \lambda(\eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2) \log_2(\eta_1 + \eta_2)$$

Gleichung 3-36

Dieses Gleichungssystem erlaubt es, die restlichen Parameter zu ermitteln. Das System wird in dem späteren Praxiseinsatz mit Hilfe von numerischen Näherungsverfahren ausgewertet. Die Beziehung zwischen η_2^*, λ und den anderen Parametern werden durch die folgenden Kurven dargestellt:

Die Anzahl der paarweise verschieden Operatoren η_1 kann man mit Hilfe des folgenden Bildes graphisch ermitteln:

Abbildung 3-2 : Paarweise verschiedene Operatoren [Halstead 77] p.76

Abbildung 3-3 stellt die Beziehung der bekannten Parameter zu der Anzahl der paarweise verschiedenen Operanden η_2:

Abbildung 3-3 : Paarweise verschiedene Operanden [Halstead 77] p. 77

Die nächste Darstellung (Abbildung 3-4) zeigt den Zusammenhang zwischen der minimalen Operandenanzahl, dem Sprachenlevel und der Vokabulargröße η:

Abbildung 3-4 : Zusammenhang zwischen η_2^*, λ und der Vokabulargröße η [Halstead 77] p. 78

Mit Hilfe der aus Abbildung 3-4 ermittelten Wert für η kann man nun die Programmlänge N ermitteln. Die folgende Grafik (Abbildung 3-5) stellt den Zusammenhang zwischen η_2^*, λ und der Programmlänge N dar:

Abbildung 3-5 : Beziehung zwischen η_2^*, λ und der Programmlänge N [Halstead 77] p. 79

Die Abbildung 3-5 bietet auch eine Vergleichsmöglichkeit für verschiedene Programmiersysteme. Da die Programmlänge ebenfalls von Sprachenlevel λ abhängt (vgl. Gleichung 3-21 und Gleichung 3-34).

Neben der Programmlänge N kann man auch noch das Programmvolumen V ermitteln. Es wird in „Bit“ gemessen:

Abbildung 3-6 : Programmvolumen V in „Bit“ [Halstead 77] p. 80

Durch die so ermittelten Parameter (vgl. Abbildung 3-6) läßt sich der Programmaufwand in Elementarentscheidungen ermitteln:

Abbildung 3-7 : Programmaufwand in Elementarentscheidungen [Halstead 77] p. 81

Mit Hilfe des Programmaufwandes E und der Stroud'schen Zahl S ist die Programmierzeit zu ermitteln. Für die Darstellung in Abbildung 3-8 wurde S = 18 Elementarentscheidungen pro Sekunde gewählt:

Abbildung 3-8 : Programmierzeit in Sekunden [Halstead 77] p. 83

3.4.3 Fehlervorhersage

Halstead liefert neben der Abschätzung für den Programmieraufwand auch noch eine Fehlerabschätzung. Diese Abschätzung basiert auf der Annahme, daß das menschliche Gehirn Informationen mit sog. Chunks[2] verarbeitet. Es können dabei 5 Chunks gleichzeitig verwaltet werden. Jedes Chunk entspricht dem Ergebnis eines Operanden η_2^*. Das Fassungsvermögen von den Chunks wird mit E_{crit} bezeichnet und kann mit Hilfe des entsprechenden minimalen Volumens V^*_{crit} berechnet werden. Für die Berechnung von V^*_{crit} führt man folgende Annahmen durch: $\eta_1^* = 2$ und $\eta_2^* = 6$ (Da jedes Chunk einen Operanden repräsentiert und ein „Rückgabe"-Operand). Darauf folgt für V^*_{crit}:

[2] Chunks sind Elemente einer bekannten und überschaubaren Grundmenge. Sie werden in der Psychologie eingesetzt. (Vgl. [Szwillus 94] Kap 3.1.4.1)

$$V^*_{crit} = (\eta^*_1 + \eta^*_2)\log_2(\eta^*_1 + \eta^*_2) = (2+6)\log_2(2+6) = 24$$

Gleichung 3-37

Mit Hilfe von Gleichung 3-35 kann man E_{crit} berechnen:

$$E_{crit} = \frac{(V^*_{crit})^3}{\lambda^2}$$

Gleichung 3-38

In [Halstead 77] wird für E_{crit} ein Wert von 3000 ermittelt. Er basiert auf der Annahme, daß für den Sprachenlevel $\lambda = 2.16$ (für die in [Halstead 77] untersuchte Englische Sprache) verwendet wird. Zusätzlich werden noch folgende Parameter definiert:

E_0 : mittlere Anzahl Elementarentscheidungen zwischen zwei Fehlern

B : Anzahl der ausgelieferten Fehler

$$B = L\frac{E}{E_0}$$

Gleichung 3-39

Durch Substitutionen und Umformungen erhält man folgende Beziehung:

$$B = \frac{E^{2/3}}{E_0}$$

Gleichung 3-40

Der Sprachenlevel λ tritt während der Umformung in Form einer kubischen Wurzel ($\sqrt[3]{\lambda}$) auf. Da die Wurzel gegen eins konvergiert, wird sie in Gleichung 3-40 nicht mit verwendet. Für Abschätzungen benutzt man für E_0 E_{crit}:

$$B_G = \frac{E^{2/3}}{3000}$$
$$B_G = \frac{V}{3000}$$

Gleichung 3-41

3.4.4 Modularisierung

Für das Problem der optimalen Modularisierung gibt es vier verschiedene Lösungsansätze:

1. Gleiche Längen
2. Minimierung des modularen minimalen Volumens
3. Module als fehlerfreie Programme
4. Das Konzept der „Chunks“

Jede dieser Lösungsansätze wird kurz vorgestellt. Hierfür führt man für die Anzahl der Module eines Programmes den Parameter M ein.

3.4.4.1 Gleiche Längen

Nimmt man an, daß $\eta_1 = \eta_2$ ist, so folgt für die Programmlänge N:

$$N = \eta \log_2 (\frac{\eta}{2})$$

Gleichung 3-42

Für jedes Modul wird die Vokabulargröße η_m und die Länge N_m definiert:

$$N_m = \eta_m \log_2 (\frac{\eta_m}{2})$$

Gleichung 3-43

Aus der Länge des Moduls N_m und der Anzahl der Module kann man die Programmlänge N ermitteln:

$$N = MN_m$$

Gleichung 3-44

Um die Beziehung zwischen η_m, M und η zu ermitteln, unterteilt man zunächst η in die zwei Klassen a und b. In Klasse a liegen alle Operanden und Operatoren (η_a), die in allen Modulen vorkommen. Hieraus ergibt sich für die Länge N_m:

$$N_m = \eta_a + \frac{\eta - \eta_a}{M}$$

Gleichung 3-45

Nimmt man weiterhin an, daß η_a genau durch η^* repräsentiert wird, erhält man folgende Gleichung:

$$\eta_m = \eta^* + \frac{\eta - \eta^*}{M}$$

Gleichung 3-46

Für Länge N_m folgt daraus:

$$N_m = (\eta^* + \frac{\eta - \eta^*}{M}) \log_2 (\frac{\eta^* + (\eta - \eta^*)/M}{2})$$

Gleichung 3-47

Durch die Beschreibung der Länge N_m durch Gleichung 3-47, der Beziehung der Anzahl der Module M in Gleichung 3-44 und der Programmlänge N durch Gleichung 3-42 gelangt man zu folgender Beziehung:

$$\eta \log_2 (\frac{\eta}{2}) = M(\frac{\eta - \eta^*}{M} + \eta^*) \log_2 (\frac{(\eta - \eta^*)/M}{2})$$

Gleichung 3-48

Mit Hilfe von η_2^* und λ kann man durch das Verfahren aus Kap. 3.4.2 den Wert von η ermitteln und somit auf die optimale Modulanzahl berechnen.

3.4.4.2 Minimierung des modularen minimalem Volumens

Bei der Betrachtung der Modularisierung fällt auf, daß die kleinsten Module am Effektivsten sind. Sie bestehen aus zwei Eingaben und einer Ausgabe ($\Rightarrow \eta_2^* = 3$). Würde man aber das Programm in Module dieser Größe aufteilen, so würde der Overhaed viel zu groß werden. Um nun die optimale Modulanzahl zu ermitteln, werden folgende Parameter eingeführt:

V_M^* : Vereinigtes minimales Volumen des modularisierten Programmes

V_m^* : Individuelles minimales Volumen eines mittleren Moduls

Diese beiden Volumenmaße stehen in folgender Beziehung:

$$V_M^* = MV_m^* + M \log_2 M$$

Gleichung 3-49

mit V_m^* gleich:

$$V_m^* = (\eta_1^* + \frac{\eta_2^*}{M}) \log_2 (\eta_1^* + \frac{\eta_2^*}{M})$$

Gleichung 3-50

folgt aus Gleichung 3-49 (unter der Annahme, daß $\eta_1^* = 2$):

$$V_M^* = M((2 + \frac{\eta_2}{M}) \log_2 (2 + \frac{\eta_2}{M}) + \log_2 M)$$

Gleichung 3-51

Für die Ermittlung der optimalen Modulanzahl M wird nun das Minimum für V_M^* gesucht. Dies kann man direkt oder mit Hilfe von numerischen Verfahren berechnen.

3.4.4.3 Module als fehlerfreie Programme

Eine weitere Strategie, die optimale Modulgröße zu ermitteln, ist die Programmlänge der Module zu beschränken. Als Schranke wird hier die Länge eines fehlerfreien Programmes angenommen. Die Länge bzw. die minimale Operandenzahl η_2^* kann man mit Hilfe der

Gleichungen aus dem Kapiteln 0 und 3.4.3 ermitteln. Gemäß Gleichung 3-40 und der Annahme, daß E_0 gleich E_{crit} ist, erhält man folgende Beziehung:

$$B = \frac{(V^*)^2 \lambda^{2/3}}{V_{crit}^3}$$

Gleichung 3-52

Für das weitere Vorgehen ersetzt man V_{crit} durch den in Gleichung 3-37 berechneten Wert und führt für B die Bedingung B < 1 ein:

$$1 > \frac{(V^*)^2 \lambda^{2/3}}{V_{crit}^3} = \frac{(V^*)^2 \lambda^{2/3}}{13824}$$

Gleichung 3-53

Durch Substitution mit Hilfe von Gleichung 3-17 und der Umbenennung von η_2^* in e_2 folgt folgende Ungleichung:

$$\lambda^{-1/3} \sqrt{13824} > (2 + e_2) \log_2 (2 + e_2)$$

Gleichung 3-54

Den Wert für e_2 kann man mit Hilfe eines numerischen Verfahrens ermitteln. Anhand des Wertes von e_2 kann man die optimale Modulgröße bzw. Modulanzahl berechnen:

$$M = \frac{\eta_2^*}{e_2}$$

Gleichung 3-55

Die so gewonnene Anzahl bezieht sich auf die Länge des fehlerfreien Programmes. In [Halstead 77] wurde für e_2 ein Wert von 7.76 ermittelt. Anhand dieses Wertes kann man nun die Anzahl der Module mit Hilfe von Gleichung 3-55 berechnen:

$$M = \frac{\eta_2^*}{7{,}76}$$

Gleichung 3-56

3.4.4.4 Das Konzept der Chunks

Das Konzept der Chunks wurde schon im Kapitel 3.4.3 vorgestellt. Analog zum vorherigen Kapitel versucht man, die Modulgröße auf ein Maß zu beschränken. Man nimmt an, daß der Mensch fünf Chunks im Kurzzeitgedächnis gleichzeitig verwalten kann. Dies entsprechen fünf Operanden in einem idealen Modul:

$$\eta_2^* = 5 + 1 = 6$$

Gleichung 3-57

Zusätzlich zu den fünf Operanden beinhaltet das Modul noch einen „Ausgabeoperand“ (vgl. Gleichung 3-57). Mit Hilfe dieser Operandenzahl kann man jetzt die Anzahl der Module berechnen:

$$M = \frac{\eta_2^*}{6}$$

Gleichung 3-58

3.4.5 Beispiel: Euklid’s Algorithmus

Die Anwendung der in den vorherigen Kapiteln beschriebenen Parameter und Abschätzungen soll mit Hilfe des folgenden Beispiels demonstriert werden.

Es soll der Euklidische Algorithmus für den größten gemeinsamen Teiler implementiert werden. Als Eingabe werden die beiden Zahlen A und B verwendet und das Resultat soll durch die Variable GCD ausgegeben werden.

Für die Abschätzung muß die minimale Anzahl der Operanden bestimmt werden:

$$\eta_2^* = |\text{Eingabe}| + |\text{Ausgabe}|$$
$$\eta_2^* = |\{A, B\}| + |\{GCD\}| = 3$$

Gleichung 3-59

Das minimale Volumen V^* kann mit diesen Werten berechnet werden:

$$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2) = (2 + 3) \log_2 (2 + 3) = 11.61\text{Bit}$$

Gleichung 3-60

Für die Abschätzung wird noch ein Sprachenlevel von $\lambda = 0.65$ angenommen. Aus dem System in Gleichung 3-36 wird nun die Anzahl der paarweise verschiedenen Operatoren η_1 und die der Operanden η_2 bestimmt. Das numerische Abschätzungsverfahren liefert für die Operatoren und Operanden folgende Werte:

$$\eta_1 = 8.5 \quad \eta_2 = 8.2 \quad \Rightarrow$$
$$\eta = \eta_1 + \eta_2 = 8.5 + 8.2 = 16.7$$

Gleichung 3-61

Aus diesen Werten folgt das Programm Volumen V:

$$V = \frac{\left(V^*\right)^2}{\lambda} = \frac{(11.61\text{Bit})^2}{0.65} = 207.37\text{Bit}$$

Gleichung 3-62

Mit Hilfe des Volumens V kann man die Anzahl der verwendeten Operatoren und Operanden N bestimmen:

$$N = \frac{V}{\log_2 \eta} = \frac{207.37\text{Bit}}{\log_2 (16.7)} = 51.05$$

Gleichung 3-63

Die Anzahl der benötigten Denkentscheidungen ist mit Hilfe des Sprachenlevels λ und des minimalen Volumens V^* zu berechnen:

$$E = \frac{(V^*)^3}{\lambda^2} = \frac{(11.61\text{Bit})^3}{0.65^2} = 3703.99$$

Gleichung 3-64

Nimmt man für die Stroud'sche Zahl S einen Wert von S = 18 Entscheidungen pro Sekunde an, erhält man folgenden Zeitaufwand für die Entwicklung:

$$T = \frac{E}{S} = \frac{3703.99}{18s^{-1}} = 205{,}78s$$

Gleichung 3-65

Nach diesen Abschätzungen soll nun der Algorithmus implementiert werden. Hierfür wird das Beispiel aus [Halstead 77] Seite 7 verwendet:

```
       IF (A = 0)
LAST:  BEGIN GCD := B; RETURN END;
       IF (B = 0)
       BEGIN GCD := A; RETURN END;
HERE:  G := A / B; R := A - B * G;
       IF(R = 0) GO TO LAST;
       A := B; B := R; GO TO HERE
```

Die Faktoren sind zu analysieren werden:

Operator	j	$f_{1,j}$
;	1	9
:=	2	6
() oder BEGIN ... END	3	5
IF	4	3
=	5	3
/	6	1
-	7	1
*	8	1
GO TO HERE	9	1
GO TO LAST	10	1
	$\eta_1 = 10$	$N_1 = 31$

Tabelle 3-5 : Anzahl der Operatoren

Operand	j	$f_{2,j}$
B	1	6
A	2	5
O	3	3
R	4	3
G	5	2
GCD	6	2
	$\eta_2 = 6$	$N_1 = 21$

Tabelle 3-6 : Anzahl der Operanden

Aus den beiden Tabellen kann man die beiden Parameter η und N berechnen:

$$\eta = \eta_1 + \eta_2 = 10 + 6 = 16$$
$$N = N_1 + N_2 = 31 + 21 = 52$$

Gleichung 3-66

Da das minimale Volumen V^* von der Implementierung unabhängig ist, kann es mit den Werten aus der Abschätzung berechnet werden:

$$V^* = (2 + \eta_2^*)\log_2(2 + \eta_2) = (2 + 3)\log_2(2 + 3) = 11.61\text{Bit}$$

Gleichung 3-67

Das Programm Volumen V wird durch die Anzahl der Operatoren und Operanden und der Vokabulargröße gebildet (Gleichung 3-14):

$$V = N\log_2(\eta) = 52\log_2(16) = 208\text{Bit}$$

Gleichung 3-68

Für die Ermittlung der Anzahl der Denkentscheidungen E wird Gleichung 3-29 verwendet:

$$E = \frac{V^2}{V^*} = \frac{(208\text{Bit})^2}{11.61\text{Bit}} = 3726.44$$

Gleichung 3-69

Wird auch hier wieder für die Stroud'sche Zahl S ein Wert von S = 18 Entscheidungen pro Sekunde angenommen, so erhält man den Zeitaufwand T:

$$T = \frac{E}{S} = \frac{3726.44}{18s^{-1}} = 207.02s$$

Gleichung 3-70

Zur Kontrolle kann der Sprachenlevel λ überprüft werden:

$$\lambda = \frac{(V^*)^2}{V} = \frac{(11.61\text{Bit})^2}{208\text{Bit}} = 0.65$$

Gleichung 3-71

Vergleicht man die geschätzten Werte mit den Werten aus der Implementation, erhält man folgende Abweichungen:

Parameter	Geschätzt	Implementation	Abweichung in %
η_2^*	3	3	0
η_1	8.5	10	15
η_2	8.2	6	36,67
η	16.7	16	4.37
N	51.05	52	1.83
V^*	11.61 Bit	11.61 Bit	0
V	207.37 Bit	208 Bit	0.31
E	3703.99	3726.44	0.60
$T_{S=18s}$	205.78 s	207.02 s	0.60

Tabelle 3-7 : Vergleich zwischen der Schätzung und der Implementation

Die Abweichungen wurden nach folgender Formel ermittelt:

$$Abweichung\% = \frac{|Implementation - Geschätzt|}{Implementation} 100\%$$

Gleichung 3-72

Die großen Abweichungen bei der Abschätzung der paarweise verschiedenen Operatoren η_1 und Operanden η_2 liegt in der geringen Anzahl der minimalen Operanden η_2^*, größere Werte verringern die prozentuale Abweichung.

3.4.6 Untersuchung des MS Access

Für die Abschätzung der Programmierzeit ist der Sprachenlevel λ von großer Bedeutung. Deshalb soll er für das MS Access bestimmt werden. Für diese Bestimmung müssen zunächst einige Konventionen festgelegt werden:

- Als Operatoren zählen neben den vordefinierten Operatoren auch die eigenen Funktionen
- SUB END SUB, FUNCTION END und PRIVATE FUNCTION END zählen jeweils als ein Operator
- IF THEN ENDIF bzw. IF THEN GOTO... zählen als ein Operator
- FOR TO NEXT zählt als ein Operator
- Objekteigenschaften zählen als Operanden
- Konstanten werden ignoriert
- Variablen Deklarationen in Funktionsköpfen werden ebenfalls gezählt
- Als minimale Operanden werden alle Felder in den Formularen gezählt.

Für die erste Analyse wurde das Beispiel „BESTELLG.MDB“ des MS Access 2.0 untersucht. Die Auswertung des Beispieles lieferte folgende Werte:

$$N_1 = 772 \quad \mathrm{N}_2 = 517 \quad \Rightarrow N = 1289$$
$$\eta_1 = 123 \quad \eta_2 = 97 \quad \Rightarrow \eta = 220$$
$$\eta_2^* = 45$$

Gleichung 3-73

Aus diesen Werten kann man nun die beiden Volumen V und V^* berechnen:

$$V = N \log_2(\eta) = 1289 \log_2(220) = 10030{,}17\mathrm{Bit}$$
$$V^* = (2 + \eta_2^*) \log_2(2 + \eta_2^*) = (2 + 45) \log_2(2 + 45) = 261{,}07\mathrm{Bit}$$

Gleichung 3-74

Aus den beiden Volumen ergibt sich dann der Sprachenlevel λ:

$$\lambda = \frac{(V^*)^2}{V} = \frac{(261{,}07\text{Bit})^2}{10030{,}17\text{Bit}} = 6{,}80$$

Gleichung 3-75

Die Varianz dieses Wertes kann man natürlich erst nach einer genügend großen Menge untersuchter Projekte ermitteln. Der in Gleichung 3-75 errechnete Wert für den Sprachenlevel soll in dem Kapitel Function point Methode für Vergleiche verwendet werden.

3.4.7 Vergleich mit den Lines of Code

Für den Vergleich mit den Lines of Code muß man zu nächst folgende Annahmen festlegen:

- Die Anzahl der paarweise verschiedenen Operatoren η_1 und Operanden η_2 ist gleich $\Rightarrow \eta_1 = \eta_2 = \eta / 2$
- Außerdem ist die Zahl der tatsächlich verwendeten Operatoren N_1 und Operanden N_2 gleich $\Rightarrow N_1 = N_2 = N / 2$

Damit folgt aus Gleichung 3-24, Gleichung 3-28 und den obigen Annahmen:

$$E = \frac{1}{4} N^2 \log_2 \eta$$

Gleichung 3-76

Die Verbindung zu den Lines of Code kann man durch folgende Beziehung beschreiben:

$$N = k \times S_S$$

Gleichung 3-77

Der Parameter S_S beinhaltet die Zeilen des Kodes und der Faktor k liefert die mittlere Anzahl von Operatoren und Operanden einer Kodezeile. Für die Bestimmung der Vokabulargröße η verwendet man folgende Beziehung:

$$N = \eta \log_2(\frac{\eta}{2})$$

Gleichung 3-78

Den Wert für die Vokabulargröße η kann man mit Hilfe eines geeigneten numerischen Verfahrens bestimmen. Da aber diese Art der Abschätzung des Aufwandes E zu ungenau ist, hat [Conte 86] für den Aufwand E folgende Näherung vorgeschlagen:

$$E \approx \frac{1}{4} N^{2+\alpha}$$

Gleichung 3-79

In dieser Beziehung wird der Term $\log_2 \eta$ durch N^{α} approximiert. Eine Abschätzung für α liefert die folgende Tabelle:

η	$\log_2 \eta$	$N = \eta \log_2(\eta / 2)$	α
64	6	320	0,31
128	7	768	0,29
256	8	1792	0,28
512	9	4096	0,26
1024	10	9216	0,25

Tabelle 3-8 : Werte von α für $N^{\alpha} = \log_2 \eta$ [Conte 86] p. 298

In diesem Bereich der Vokabulargröße η kann man für α einen Mittelwert von α = 0,28 ermitteln. Für größere Vokabulare muß man für α kleinere Werte annehmen. Durch Einsetzen des Wertes von α in Gleichung 3-79 erhält man die Näherung:

$$E \approx \frac{1}{4} N^{2,28}$$

Gleichung 3-80

Ersetzt man N durch Gleichung 3-77, erhält man die Näherungsbeziehung zwischen Aufwand E und den Zeilen des Kodes S_S:

$$E \approx \frac{1}{4} k^{2,28} S_S^{2,28} = c S_S^{2,28}$$

Gleichung 3-81

3.5 Function point Methode

Eine auf Halstead's Arbeiten basierende Methode ist die Function point Methode (FP), vgl. [Albrecht 83]. Sie wird als Produktivitätsmaß verwendet. Der Vorteil dieser Methode gegenüber den Lines of Code liegt in der Programmiersprachenunabhänigkeit. Die Ermittlung der Function points wird im folgendem Kapitel beschrieben.

3.5.1 Methode

Nach der Spezifikation werden die einzelnen Funktionen in folgende Klassen eingeteilt:

- Eingaben
- Ausgaben
- Abfragen (interaktive Funktionen, die Rückmeldungen benötigen)
- Externe Dateien (Zugriffsfunktionen auf im System gemeinsam genutzte Datenbestände)
- Interne Dateien (Dateien, die nur für das Programm sichtbar sind)

Nach dieser Klassifizierung werden die einzelnen Funktionen gewichtet. Hierfür werden sie in drei Schwierigkeitsklassen eingeteilt und die einzelnen Gewichte summiert (Die Summe wird als „unadjusted function count" (UFC) bezeichnet).

Funktionsklasse	Simple	Average	Complex
Eingaben	3	4	6
Ausgaben	4	5	7
Interne Dateien	7	10	15
Externen Dateien	5	7	10
Abfragen	3	4	6

Tabelle 3-9 : Gewichte der Funktionsklassen [Shepperd 95] p. 92

Für die Einteilung in die Schwierigkeitsklassen „Simple", „Average" und „Complex" untersucht man den Inhalt bzw. die geplante Verwendung von Dateien, Recordtypen (bzw. Datenbankfelder) und Datenelementen (Variablen, etc.) für jede Funktion. Anhand der Tabelle 3-10 erhält man eine Punktzahl für die Anzahl der Dateien, Recordtypen und Datenelementen. Aus diesen Punkten wird eine Summe gebildet. Mit Hilfe dieser Summe

ist durch Anwendung von Tabelle 3-11 die Schwierigkeitsklasse der untersuchten Funktion zu bestimmen. Nachdem alle Gewichte bestimmt sind, werden sie zu der obigen Summe UFC zusammengefaßt.

Funktionsklasse	Einordnung								
	#Dateien			#Recordtypen			#Datenelemente		
	1	2	3	1	2	3	1	2	3
Eingaben	0-1	2	≥3				1-4	5-15	≥16
Ausgaben	0-1	2-3	≥4				1-5	6-19	≥20
Interne Dateien				1	2-5	≥6	1-19	20-50	≥51
Ext. Dateien				1	2-5	≥6	1-19	20-50	≥51
Abfragen	Max(Eingabekomponenten, Ausgabekomponenten)								

Tabelle 3-10 : Einordnung der Funktionen in Schwierigkeitspunkte [Shepperd 95] p. 92

∑Punkte	Schwierigkeitsklasse
2-3	simple
4	average
5-6	complex

Tabelle 3-11 : Schwierigkeitsklassen [Shepperd 95] p. 93

i	GSC_i	Erklärung
1	Data communications	Datenaustausch mit externen Programmen
2	Distributed data proccessing	Verteilte Applikationen
3	Performance	Leistungsanforderungen durch den Kunden
4	Heavily used configuration	Leistungsanforderung bei stark ausgelastetem System
5	Transaction rate	Geplante Durchsatzrate der Daten
6	Online data entriy	Interaktive Benutzung
7	End user efficiency	Anforderung an die Benutzungsschnittstelle
8	On-line update	Forderung bezüglich sofortiger Updates der Daten
9	Complex processing	Komplexe interne Berechnungen
10	Re-usability	Wiederverwendbarkeit von Systemteilen
11	Installation ease	Anforderung an die Installation des Programmes
12	Operation ease	Leichtigkeit der Bedienung
13	Multiple sites	Anzahl der zu unterstützenden Systemplattformen
14	Facilitate change	Einfache Veränderbarkeit der Daten des Systems

Tabelle 3-12 : „general system characteristics“ (GSC) [Shepperd 95] p. 94

Neben diesen Klassen existieren noch weitere Faktoren, die die Anforderungen an das zu erstellende System beeinflussen. Sie werden als „general system characteristics“ (GSC) bezeichnet (Vgl. Tabelle 3-12).Jede dieser GSC’s wird ein Wert zu geordnet. Er beschreibt die Beeinflussung des Systems. Der Wertebereich liegt zwischen 0 (keine Beeinflussung)

und 5 (essentielle Beeinflussung). Aus der Summe der einzelnen GSC's kann man einen Korrekturfaktor VAF ermitteln:

$$TDI = \sum_{i=1}^{14} CSG_i$$
$$VAF = 0{,}65 + (0{,}01 * TDI)$$

Gleichung 3-82

Der VAF kann Werte zwischen 0,65 und 1,35 annehmen. Er bildet zusammen mit der Summe UFC die „function points" FP:

$$FP = UFC * VAF$$

Gleichung 3-83

Typische Werte für VAF hängen von der Systemanforderung ab. Einprozessor Systeme ohne Zeitschranken haben Werte für VAF von VAF = 0,65 ~ 0,85. Dies führt zum Verkleinern der UFC's. Für verteilte Systeme mit sehr kritischen Zeitschranken nimmt VAF Werte von VAF = 1,15 ~ 1,35 an. Analog steigt hier die Zahl der Funktionspunkte UFC an. Für den mittleren Fall nehmen die GSC's Werte zwischen 2,5 und < 3 an.

Der so gewonnene Wert für FP wird mit einem Produktivitätsfaktor P multipliziert, um so den Aufwand T in Personenmonaten zu berechnen:

$$T = \frac{FP}{P}$$

Gleichung 3-84

Der Produktivitätsfaktor P muß für jedes Team vorher bestimmt werden. In die Skalierung GSC's kann man ebenfalls das „Know How" des Teams miteinfließen lassen.

3.5.2 Beispiel

Die Anwendung der „Function point"-Methode soll nun anhand eines Beispiels demonstriert werden:

Aus der Spezifikation eines Banksystems hat man folgende Anforderungen entnommen:

Eingaben:

- Einen neuen Kunden aufnehmen
- Löschen eines Kunden
- Zahlungstransaktion
- Eine getätigte Transaktion wieder zurücknehmen
- Benutzerdefinierte Reporte über Kunden generieren

Ausgaben:

- Warnung bei überzogenen Konten bzw. Krediten
- Benutzerdefinierte Reports ausgeben

Abfragen:

- Kontostandabfrage

Interne Dateien:

- Kundendatenbank

Nach der Klassifizierung erfolgt nun die Einteilung in die Komplexitätsklassen gemäß Tabelle 3-11 und Tabelle 3-10:

Funktion	Klasse	#Dateien	#Record-typen	#Daten-elemente	Kom-plexität
Einen neuen Kunden aufnehmen	Eingaben	1	-	10	Simple
Löschen eines Kunden		1	-	2	Simple
Zahlungstransaktion		1	-	2	Simple
Eine getätigte Transaktion wieder zurücknehmen		1	-	2	Simple
Benutzerdefinierte Reporte über Kunden generieren		1	-	0	Simple
Warnung bei überzogenen Konten bzw. Krediten	Ausgaben	1	-	4	Simple
Benutzerdefinierte Reports ausgeben		1	-	8	Simple
Kontostandabfrage	Abfragen	1	-	3	Simple
Kundendatenbank	Interne Dateien	-	1	12	Simple

Tabelle 3-13 : Einteilung der Funktionen in Komplexitätsklassen [Shepperd 95] p. 95

Die in Tabelle 3-13 gewonnenen Komplexitätsklassen ergeben mit den in Tabelle 3-9 zugeordneten Gewichten den UFC:

Funktion	Klasse	Komplexität	Gewicht
Einen neuen Kunden aufnehmen	Eingaben	Simple	3
Löschen eines Kunden		Simple	3
Zahlungstransaktion		Simple	3
Eine getätigte Transaktion wieder zurücknehmen		Simple	3
Benutzerdefinierte Reporte über Kunden generieren		Simple	3
Warnung bei überzogenen Konten bzw. Krediten	Ausgaben	Simple	4
Benutzerdefinierte Reports ausgeben		Simple	4
Kontostandabfrage	Abfragen	Simple	3
Kundendatenbank	Interne Dateien	Simple	7
UFC = Summe			33

Tabelle 3-14 : Ermittlung der unadjusted function count (UFC)

Nach der Bestimmung des Wertes für UFC folgt nun die Klassifizierung des geplanten Systems anhand der 14 GSC's:

i	General system characteristic	DI	Komentar
1	Data communications	3	„Online“-Kommunikation für Abfragesystem und Batch-System für Reports
2	Distributed data proccessing	3	Verteilte Berechnung und Online-Datentransfer in eine Richtung
3	Performance	3	Antwortzeiten für die Online-Abfragen sind in den Geschäftszeiten kritisch. Antwortzeit durch ein Batch gestarteten Reports unkritisch, da die Antwort erst am nächsten Geschäftstag erwartet wird.
4	Heavily used configuration	2	Das Programm wird auf einem bestehendem System eingesetzt. Hier wird es einige Zeit- und Sicherheitseinschränkungen geben.
5	Transaction rate	4	Die Höhe der Transaktionsrate wird in der Designphase festgelegt.
6	Online data entriy	5	Mindestens 30% der Transaktionen werden„Online“ eingegeben.
7	End user efficiency	4	Für die Bestimmung der Effizienz werden noch weitere Analysen benötigt.
8	On-line update	4	Die Hauptdatenbasis wird durch „Online“-Update im Fehlerfall restauriert.
9	Complex processing	1	Sehr geringe Komplexität der Berechnungen
10	Re-usability	0	Keine Wiederverwendung geplant
11	Installation ease	0	Keine Maßnahmen geplant
12	Operation ease	1	Effektive Start-, Backup- und Rettungsprozeduren erforderlich.
13	Multiple sites	0	Nur eine Systemplattform
14	Facilitate change	0	Nur feste Abfragemuster, keine benutzerdefinierte Abfragen möglich.
	TDI	30	

Tabelle 3-15 : Systemkarakterisierung [Shepperd 95] p: 96

Aus Tabelle 3-15 kann man Korrekturfaktor VAF berechnen:

$$VAF = 0{,}65 + \left(0{,}01 * TDI\right) = 0{,}65 + \left(0{,}01 * 30\right) = 0{,}95$$

Gleichung 3-85

Jetzt kann man die „Function points“ FP ermitteln:

$$FP = VAF * UFC = 0{,}95 * 33 = 31{,}35\text{fp}$$

Gleichung 3-86

Nimmt man nun an, daß das Entwicklungsteam eine Produktivität von 10,45 fp/Monat hat, kann man die Zeit berechnen, die das Team für das Bankprogramm benötigt:

$$T = \frac{FP}{P} = \frac{31{,}35\text{fp}}{10{,}45\,\text{fp}/\text{Monat}} = 3\text{Monate}$$

Gleichung 3-87

3.5.3 Vergleich mit den Lines of Code

Für den Vergleich mit den Lines of Code müssen zunächst für die einzelnen Funktionen und deren Schwierigkeitsklassen die mittlere Kodegröße gebildet werden. Sie werden mit S_{FG} bezeichnet, wobei der Index F für die Funktionsklasse („E“ für Eingabe, „A“ für Ausgabe, „I“ für interne Dateien, „F“ für externe Dateien und „Q“ für Abfragen) und G für die Schwierigkeitsgrade („S“ für Simple, „A“ für Average und „C“ für Complex). Dies ergibt dann folgendes Schema:

Funktionsklasse		Anzahl	mtl. LOC	Gesamt
Eingaben	Simple		S_{ES}	
	Average		S_{EA}	
	Complex		S_{EC}	
Ausgaben	Simple		S_{AS}	
	Average		S_{AA}	
	Complex		S_{AC}	
Interne Dateien	Simple		S_{IS}	
	Average		S_{IA}	
	Complex		S_{IC}	
Externen Dateien	Simple		S_{FS}	
	Average		S_{FA}	
	Complex		S_{FC}	
Abfragen	Simple		S_{QS}	
	Average		S_{QA}	
	Complex		S_{QC}	
			Gesamt S_S =	

Tabelle 3-16 : Ermittlung der Lines of Code

Der so erhaltene Abschätzung kann z.B. als Basis für das CoCoMo-Modell verwendet werden.

3.5.4 Vergleich mit der Methode von Halstead

Zu Vergleich der beiden Methoden soll zunächst das obige Beispiel nach Halstead analysiert werden.

3.5.4.1 Umsetzung des Beispiels aus Kap. 3.5.2

Zunächst müssen die minimalen Operanden η_2^* abgeschätzt werden. Tabelle 3-13 liefert die benötigten Daten für die Abschätzung in Tabelle 3-17:

Funktion	Klasse	#Dateien	#Record-typen	#Daten-elemente	min. Op. η_2^*
Einen neuen Kunden aufnehmen	Eingaben	1	-	10	11
Löschen eines Kunden		1	-	2	3
Zahlungstransaktion		1	-	2	3
Eine getätigte Transaktion wieder zurücknehmen		1	-	2	3
Benutzerdefinierte Reporte über Kunden generieren		1	-	0	1
Warnung bei überzogenen Konten bzw. Krediten	Ausgaben	1	-	4	5
Benutzerdefinierte Reports ausgeben		1	-	8	9
Kontostandabfrage	Abfragen:	1	-	3	4
Kundendatenbank	Interne Dateien	-	1	12	13
				Gesamt η_2^*	52

Tabelle 3-17 : Ermittlung der Anzahl minimaler Operanden η_2^*

Für die Abschätzung des Aufwandes wird zusätzlich der Sprachenlevel λ benötigt. Er wird bei diesem 4GL-System mit $\lambda = 3{,}11$ angenommen. Für die Stroud'sche Zahl S werden S = 18 Elementarentscheidungen pro Sekunde angenommen. Somit erhält man folgenden Zeitaufwand:

$$E = \frac{\left((2+\eta_2^*)\log_2(2+\eta_2^*)\right)^3}{\lambda^2} = \frac{\left((2+52)\log_2(2+52)\right)^3}{3.11^2} = 3102923{,}13$$

$$T = \frac{E}{S} = \frac{3102923{,}13}{18 s^{-1}} = 172384{,}62 s$$

$$T = 2{,}99 \text{Monate}$$

Gleichung 3-88

3.5.4.2 Programmsystem MS Access

Das vorherige Beispiel soll jetzt mit dem im Kapitel 3.4.6 (Gleichung 3-75) ermittelten Sprachenlevel λ abgeschätzt werden. Für die Stroud'sche Zahl S werden S = 18 Elementarentscheidungen pro Sekunde angenommen. Dies führt zu folgendem Zeitaufwand:

$$E = \frac{\left((2+\eta_2^*)\log_2(2+\eta_2^*)\right)^3}{\lambda^2} = \frac{\left((2+52)\log_2(2+52)\right)^3}{6{,}80^2} = 649043{,}75$$
$$T = \frac{E}{S} = \frac{649043{,}75}{18 s^{-1}} = 36057{,}99 s$$
$$T = 0{,}63 \text{Monate}$$

Gleichung 3-89

Vergleicht man die beiden Ergebnisse aus Gleichung 3-87 und Gleichung 3-89, so erhält man für das MS Access eine um Faktor 4,8 höhere Produktivität als bei dem 4GL-System aus dem Kap. 3.5.2. Welche Produktivitätswerte am genausten die Wirklichkeit wiedergeben, kann nur durch Untersuchung weiterer abgeschlossener Projekte entschieden werden. Dies soll nach der Konstruktion des Tools durchgeführt werden (vgl. Kap. 7).

3.5.5 Kalibrierung der Function points auf die Entwicklungsabteilung

Für den Einsatz der „Function point"-Methode muß zunächst die Produktivität des Entwicklerteams bestimmt werden. Eine einfache Bestimmung des Produktivitätsfaktors p ist meist zu ungenau. Für einzelne Programmgruppen (z.B. Datenbankanwendungen oder Echtzeitsysteme) kann man mit Hilfe der linearen Regression eine hinreichend genaue Kalibrierung für die Function points erzielen. Für das Tool sind folgende Kalibierungsstrategien geplant:

- Die Erstkalibierung erfolgt mit einem abgeschlossenen Projekt, durch Bewertung der Function points mit den Systemeinflüssen und ohne deren Betrachtung. (VAF = 1)
- Für mehrere Projekte kann zwischen linearer und quadratischer Regression gewählt werden

- Für die Kompensation der Lernkurve können die Projekte je nach Alter verschieden gewichtet werden
- Die Projekte können in Projektklassen eingeteilt werden und die Produktivität kann für jede Klasse ermittelt werden

3.6 Design Metriken

Die Design Metriken können erst in der Designphase angewendet werden. Sie sind für eine frühe Aufwandsschätzung deshalb nicht geeignet. Da sie aber in der Beurteilung der Qualität bzw. der Fehlerhäufigkeit des Entwicklungsprozesses von Bedeutung sind, wurden sie mit in das Kapitel Metriken und Modelle aufgenommen.

3.6.1 McCabe's Cyclomatic Complexity

Der Ansatz von McCabe verwendet den Kontrollfluß in Programmen. Für die Bestimmung der Komplexität wird ein Graph aus dem Sourcecode konstruiert, in dem die Anweisungen als Knoten erscheinen und die Kanten mögliche Folgen der Anweisungen bilden:

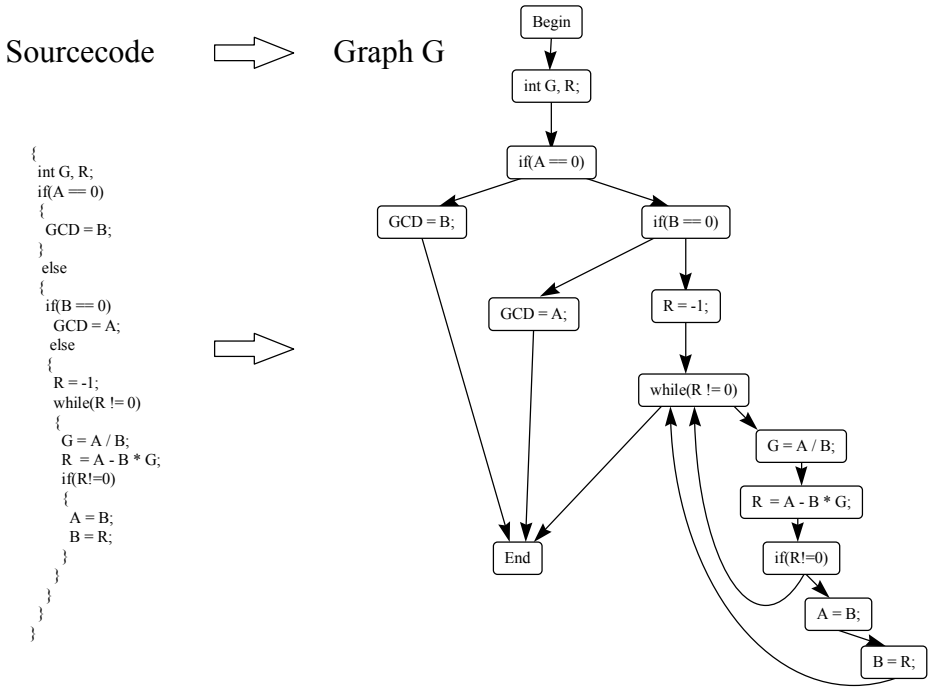


Abbildung 3-9 : Konstruktion des Graphen

Dieser Graph hat eine Quelle („Begin“) und eine Senke („End“). Die möglichen Wege in dem Graphen repräsentieren den Kontrollfluß. Für die Komplexität hat McCabe für einen Graphen G bzw. den Sourcecode folgendes Maß eingeführt:

$$V(G) = e - n + 2p$$
mit
$e = |\text{Kanten von } G|$
$n = |\text{Knoten von } G|$
$p = |\text{nichtverbundene Teile des Graphen } G|$

Gleichung 3-90

Der Wert von V liefert ein Maß für die binären Entscheidungen d des Programmes:

$$V(G) = d + 1$$
$$d = V(G) - 1$$

Gleichung 3-91

Zu den binären Entscheidungen zählen „IF THEN ELSE“ -Konstrukte und Schleifen (FOR oder WHILE). Höherwertige Entscheidungen, z.B. „CASE“-Konstrukte werden bei der Konstruktion des Graphen automatisch auf binäre Entscheidung abgebildet.

Das Modell von McCabe macht zwischen geschachtelten „IF THEN ELSE“-Konstrukten, einfachen „IF THEN ELSE“-Konstrukten und Schleifen keinerlei Unterschiede in der Bewertung der Komplexität. Als Metrik für den praktischen Einsatz bietet sich nur der Einsatz in der Fehlerabschätzung. [Troster 92] hat eine sehr starke Korrelation zwischen V(G) und der Fehlerrate gefunden. Er verwendet das Modell von McCabe für die Ermittlung der Fehlerrate im Programm. Der größte Nachteil diese Modelles liegt in dem späten Stadium des Entwicklungprozesses, an dem das Verfahren angewandt werden kann. Deshalb ist es für diese Zwecke unbrauchbar.

3.6.2 Strukturmetriken

Die Metrik von McCabe geht von dem Modell aus, daß jedes Modul eine Einheit bildet. Strukturmetriken bewerten im Gegensatz dazu die Beziehungen der Module miteinander.

Die Kohäsion und die Kopplung der Module werden untersucht. Für die Bewertung der Prozeduren und Module wurden folgende Parameter eingeführt:

Fan-in : (FI) Anzahl der Module bzw. Prozeduren, die ein gegebenes Modul bzw. Prozedur aufrufen.

Fan-out: (FO) Anzahl der Module bzw. Prozeduren, die von einem gegebenen Modul bzw. Prozedur aufgerufen werden.

[Henry 81] definiert aus diesen beiden Parametern die Struktur-Komplexität C_P:

$$C_P = (FI \times FO)^2$$

Gleichung 3-92

Eine hybride Form der Struktur-Komplexität HC_P definierte [Henry 90]:

$$HC_P = C_{ip} \times (FI \times FO)^2$$

Gleichung 3-93

C_{ip} ist die interne Komplexität von dem Modul bzw. der Prozedur P. Sie kann entweder nach [Henry 81] in Lines of Code gemessen werden oder mit Hilfe der Metrik von McCabe (V(G)).

[Card 90] führte die System-Komplexität C_t ein:

$$C_t = S_t + D_t$$

Gleichung 3-94

Mit S_t = Struktur-Komplexität (auf Modulebene) und D_t = Daten-Komplexität (auf Modulebene). Die relative System-Komplexität wird mit Hilfe der Anzahl der Module im System n bestimmt:

$$C = \frac{C_t}{n}$$

Gleichung 3-95

Die Struktur-Komplexität S wird durch den Fan-out der einzelnen Module bestimmt. Die Funktion f(i) liefert den Fan-out von Modul i:

$$S = \frac{\sum_{i=1}^{n} f^2(i)}{n}$$

Gleichung 3-96

Die Daten-Koplexität auf Modulebene D_i wird zusätzlich durch die Anzahl der I/O-Variablen in Modul i bestimmt. Sie wird als V(i) bezeichnet:

$$D_i = \frac{V(i)}{f(i)+1}$$

Gleichung 3-97

Aus Gleichung 3-97 folgt die Daten-Komplexität D:

$$D = \frac{\sum_{i=1}^{n} D_i}{n}$$

Gleichung 3-98

[Card 90] schätzten mit Hilfe der Komplexitäten die Fehlerrate des Systems ab:

$$\text{Fehler Rate} = -5{,}2 + 0{,}4 \times \text{Komplexität}$$

Gleichung 3-99

Für die Komplexität C verwendet [Card 90] die Struktur- und Datenkomplexität aus Gleichung 3-96 und Gleichung 3-98:

$$C = S + D$$

Gleichung 3-100

Die Abschätzung der Fehlerrate ist für spätere Stufen der Aufwandsabschätzung wichtig, wird aber in den frühen Stufen der Metrikeinführung keine Rolle spielen.

3.7 Personal Software Process

Der Personal Software Process (PSP) wie er von [Humphrey 95] beschrieben wird, ist genaugenommen weder eine Metrik noch ein Modell. Er verwendet Modelle bzw. Metriken aus vorherigen Kapiteln. Ziel des PSP ist die Ausreifung des Entwicklungsprozesses. [Humphrey 95] unterteilt die Reifegrade bzw. Evolutionsstufe in 7 Abschnitte (vgl. Abbildung 3-10). Sie werden an dieser Stelle nur kurz vorgestellt; dem interessierten Leser sei auf [Humphrey 95] verwiesen.

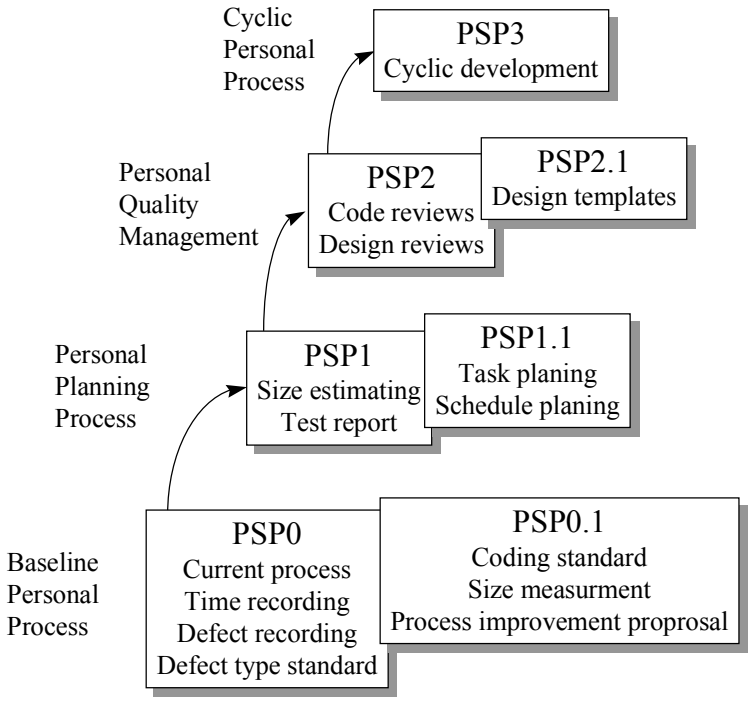


Abbildung 3-10 : PSP Evolution [Humphrey 95] p. 11

PSP0: THE BASELINE PROCESS

Die erste Stufe führt einige grundlegende Messmethoden ein. Sie werden in den Entwicklungsphasen eingesetzt und sollen für die Protokollierung von Compilerzeiten und Entwicklungsfehlern verwendet werden. Für die Fehlerprotokollierung werden zusätzlich Fehlertypen definiert. Die PSP0 Stufe kann durch das Einführen von Kodierungsstandards, Messung von Programmgröße und Dokumentationstandards für Probleme und Erfahrungen in den Prozeßphasen zu dem PSP0.1 erweitert werden.

PSP1 : THE PERSONAL PLANNING PROCESS

Durch zusätzliche Planungsschritte wird aus dem PSP0 der PSP1. Es wird eine Programmgrößen- und Resourcen-Schätzung eingeführt. Zusätzlich wird dem PSP0 ein Test-Bericht hinzugefügt. Um einen Überblick über den Projektstand zu erhalten, wird die Planung von Aufgaben- und Terminplänen im PSP1.1 eingeführt.

PSP2 : PERSONAL QUALITY MANAGEMENT PROCESS

Fehler früh zu entdecken und zu beheben bzw. zu vermeiden ist eines der Ziele des PSP. Zu diesem Zweck werden in dem PSP1 Überprüfungstechniken zum einen für das Design und zum anderen für den in der Implementierung produzierten Kode eingeführt. PSP2.1 führt zusätzlich noch Design-Richtlinien ein.

PSP3 A CYCLIC PERSONAL PROCESS

Der PSP3 ist die letzte Stufe des PSP. Er stellt für große Projekte die Abstraktionsstufe des PSP2 dar. Große Projekte werden in einem iterrativen Prozeß entwickelt. Man beginnt mit einem kleinen Kern, auf dem man den PSP2 anwendet und erweitert ihn im jedem Iterrationsschritt. Auf dem erweiterten Projektteil wird dann wieder der PSP2 angewendet.

Zum Erreichen der einzelnen PSP-Stufen hat Humphrey ein Kurs entwickelt. Er ist Bestandteil von [Humphrey 95].

Die Einführung des PSP läßt sich nicht durchführen, da gerade die Protokollierung der gemachten Fehler auf starken Widerstand stößt. Dies ist nicht unbedingt ein Problem der Softwareentwickler, sondern ein generelles Metalitätsproblem. Ähnliche Probleme gab es bei einem früheren Projekt ([Koch 93a] und [Koch93b]). In diesem Projekt waren die Mitarbeiter gegen die Einführung eines automatischen Röntgenbuches, das die Protokolle der Aufnahmen automatisch erstellen sollte.

4 Einführung von Metriken

Für die Einführung der Metriken habe ist die Entscheidung auf zwei Metriken gefallen. Der Einsatz wird im folgenden Kapitel 4.1 beschrieben. Die Untersuchungsergebnisse werden als Grundlage für die Entwicklung des Tools verwendet.

4.1 Erste Untersuchungen der Projekte

Für die Untersuchung der Projekte muß zunächst ein Rahmen festgelegt werden. Für diese Phase stehen Projekte, die mit dem Microsoft Access erstellt wurden, zur Verfügung. Für die Metriken fiel die Entscheiidung auf die Lines of Code und die Function points, da sie einen relativ einfachen Einsatz versprechen.

Die Lines of Code werden nach folgenden Kriterien bestimmt:

- Für Mircosoft Access werden die Lines of Code als S_{MA} bezeichnet.
- Der Kode bzw. das komplette Projekt wird durch das Add-In „Datenbank-Dokumentierer“ erzeugt und in eine Tabelle abgelegt (vgl. Kap. 4.1.1).
- Es werden nur diejenigen Zeilen erfaßt, die als folgende Typen ausgewiesen werden. Typ „Code“, ohne Leer- oder Kommentarzeilen. Typ „Aktion“ ohne Leer- und Kommentarzeilen. Der Typ „Tabelle“, „Spalte“, „Abfrage“, „SQL-Anweisung“ werden als Datendefinition mit gezählt.

Folgende Annahmen werden für die Erfassung der Function points getroffen:

- Für Mircosoft Access werden der „unadjusted function count“ mit UFC_{MA} bezeichnet.
- Der Kode bzw. das komplette Projekt wird durch das Add-In „Datenbank-Dokumentierer“ erzeugt und in eine Tabelle abgelegt (vgl. Kap. 4.1.1).
- Als interne Dateien zählen alle Tabellen des Projektes.
- Externe Dateien können nicht automatisch erfaßt werden.

- Es können nur statische Abfragen gezählt werden. Dynamisch erzeugte Abfragen können nur per Hand erfaßt werden.
- Formulare werden als Ein-, Ausgabefunktion oder als Abfragefunktion gewertet. Die verknüpften Steuerelemente werden als Datenelemente gezählt. Verknüpfte Abfragen oder Tabellen werden als Dateien erfaßt.
- Berichte werden als Abfragefunktion gewertet.
- Die unter der Rubrik „Abfrage“ definierten Abfragen werden nicht gezählt, da sie schon in die Komplexität der Formulare und Berichte einfließen.
- Für die Ermittlung der Korrekturfaktoren GSC wird das Formular aus dem Anhang (Kapitel 8.2) verwendet.

Für die ersten Schätzungen eignen sich nur die Function points, da eine Abschätzung nach Lines of Code ohne Erfahrungswerte nicht durchführbar ist.

4.1.1 Verfahren zur Kode-Erfassung von Microsoft Access Projekten

Für die automatische Erfassung der Mircosoft Access Projekte muß das ausgewählt Projekt vorbereitet werden. Nach dem das Projekt geladen wurde, muß man das Add-In „Datenbank-Dokumentierer“ aufrufen.

Abbildung 4-1 : Auswahl aller Objekte für die spätere Analyse

Dort wählt man alle Objekttypen und Objekte aus und stellt folgende Optionen ein:

Abbildung 4-2 : Alle Komponenten des jeweiligen Objektes ausgeben

Hat der Dokumentierer den Prozeß abgeschlossen, muß man das Ergebnis als Tabelle speichern. Es erscheint nach dem Abspeichern die Tabelle „Objektdefinitionen“ im Projekt.

Abbildung 4-3 : Neu erzeugte Tabelle „Objektdefinitionen“

Diese Tabelle dient als Basis für die Ermittlung der Lines of Code S_{MA} und der Function points UAC_{MA}. Einen Auszug findet man in Tabelle 4-1. Anhand der Spalte „Object Type“, „ID“ und „ParentID“ kann man die benötigen Informationen für die beiden Metriken ermitteln.

ID	ParentID	Object Type	Name	Extra1
...				
17952	13482	Steuerelement	Eingebettet103	Objektfeld
17953	17952	Eigenschaft	EreignisprozPräfix:	Eingebettet103
...				
17988	0	Formular	Ereignisse anzeigen - Unterformular	""
...				
17995	17988	Code	7	Sub Anzahl_ AfterUpdate ()

Tabelle 4-1 : Tabellenauszug aus der Konvertierung eines MS Access Projektes

4.2 Strategien zur Kalibrierung und Validierung der Metriken

Die Kalibrierung der ausgewählten Metriken soll mit Hilfe der Regressionsanalyse durchgeführt werden. Für die Function points bietet sich da die lineare bzw. quadratische Regression an. Zu diesem Zweck werden die Function points FP als X_i und die jeweils benötigte Zeit als Y_i angegeben. Aus diesen Messwertpaaren (X_i,Y_i) kann man dann die Parameter für das ausgewählte Regressionsverfahren berechnen. Für die Ermittlung der Regessionsparameter benötigt man ebenfalls die folgenden statistischen Kenngrößen: (Sei n die Anzahl der Meßwerte bzw. Projekte)

- Mittelwerte: $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$
- Varianz der X-Werte: $S_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$
- Kovarianz: $S_{XY} = \frac{1}{n-1}(\sum_{i=1}^{n} X_i Y_i - n\overline{XY})$

Die lineare Regression wird durch die folgende Regressionsgerade beschrieben:

$$Y = b_0 + b_1 X$$

Gleichung 4-1

Der Regressionskoeffizient b_1 wird mit Hilfe der Kovarianz S_{XY} und der Varianz der X-Werte bzw. der Function points S_X^2 und der Koeffizient b_0 durch die Mittelwerte bestimmt:

$$b_1 = \frac{S_{XY}}{S_X^2}$$
$$b_0 = \overline{Y} - b_1 \overline{X}$$

Gleichung 4-2

Die quadratische Regressionsanalyse wird analog nach dem Schema aus Kapitel 3.2.1 Kalibrierung des CoCoMo-Modelles durchgeführt. Die Funktion f_1 ist wie folgt definiert:

$$f_1(X) = b_2 X^2 + b_1 X + b_0$$

Gleichung 4-3

Daraus folgt die Summe der kleinsten Quadrate a:

$$a = \sum_{i=1}^{n} (Y_i - (b_2 X_i^2 + b_1 X_i + b_0))^2$$

Gleichung 4-4

Die Berechnung der kleinsten Quadrate führt, durch Differation der einzelnen Koeffizienten, zu folgendem Gleichungssystem:

$$b_0 n + b_1 \sum_{i=1}^{n} X_i + b_2 \sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} Y_i$$
$$b_0 \sum_{i}^{n} X_i + b_1 \sum_{i=1}^{n} X_i^2 + b_2 \sum_{i=1}^{n} X_i^3 = \sum_{i=1}^{n} Y_i X_I$$
$$b_0 \sum_{i}^{n} X_i^2 + b_1 \sum_{i=1}^{n} X_i^3 + b_2 \sum_{i=1}^{n} X_i^4 = \sum_{i=1}^{n} Y_i X_i^2$$

Gleichung 4-5

Die aus dem Gleichungssystem gewonnenen Koeffizienten beschreiben mit Gleichung 4-3 die Regressionsparabel.

4.3 Erste Ergebnisse

Für die Ermittlung der Function points wurden für die Klassifizierung der Funktionen folgende Kriterien angesetzt:

Abfrage: Funktionen, die Daten selektieren und das Ergebnis in einer Listenform ausgeben.

Ausgabe: Funktionen, die einzelne Daten ausgeben

Eingabe: Neu Eintragen, Ändern, Löschen werden zu den Eingabefunktionen gezählt

Ext. Datei: Funktionen, die mit externen Datenquellen Informationen austauschen. Z.B. Datenaustausch über Modem, Einlesen von Logdateien von Prozeßleitsystemen.

Int. Datei: Datenbanktabellen und andere Dateien. Bei den Datenbanktabellen bestimmt die Anzahl der Felder den Schwierigkeitsgrad.

Anhand dieser Kriterien wurden zwei Projekte untersucht. Die Schätzung wurde aufgrund des Pflichtenheftes durchgeführt. Für die Nachkalkulation wurden die Access-Projekte in eine Tabelle konvertiert (Vgl. Kap. 4.1.1). Nach der Konvertierung wurden mit Hilfe eines Berichtes die Funktionen und deren verknüpften Komponenten ausgegeben. Die Ergebnisse der Projekte kann man aus Tabelle 4-2 und Tabelle 4-3 entnehmen.

Funktion	Schwierig-keit	Gewicht	Anzahl Geschätzt	Punkte Geschätzt	Anzahl Kalk.	Punkte Kalk.
Abfrage	Simple	3	20	60	21	63
	Average	4	6	24	5	20
	Complex	6				
Ausgabe	Simple	4	2	8	2	8
	Average	5				
	Complex	7				
Eingabe	Simple	3	2	6	6	18
	Average	4				
	Complex	6				
Ext. Datei	Simple	5				
	Average	7				
	Complex	10				
Int. Datei	Simple	7	6	42	8	56
	Average	10				
	Complex	15				
		Summe :	36	140	42	165

Tabelle 4-2 :Analyseergebnisse Projekt 1

Projekt 2 weist eine große Differenz zwischen Schätzung und Nachkalkulation auf. Gründe hierfür sind die erheblichen Änderungen, die während des Projektes und in der Nachkalkulation durchgeführt werden mußten. Es hat sich herausgestellt, daß eine Nachkalkulation über einfache Abfragen zu ungenau ist. Deshalb wird für die automatische Nachkalkulation des Tools ein verbesserter Mechanismus gefordert. Außerdem existierte für die Änderungen keine vollständige Dokumentation. Hierfür sollte das Tool eine schnelle und einfache Möglichkeit bieten, den Lebenszyklus der einzelnen Komponenten zu dokumentieren.

Funktion	Schwierig-keit	Gewicht	Anzahl Geschätzt	Punkte Geschätzt	Anzahl Kal.	Punkte Kal.
Abfrage	Simple	3	18	54	37	111
	Average	4	5	20	26	104
	Complex	6				
Ausgabe	Simple	4	1	4	8	32
	Average	5			6	30
	Complex	7				
Eingabe	Simple	3	15	45	85	255
	Average	4			16	64
	Complex	6				
Ext. Datei	Simple	5	4	20		
	Average	7				
	Complex	10				
Int. Datei	Simple	7	9	63	69	483
	Average	10	4	40	12	120
	Complex	15				
		Summe :	56	246	259	1199

Tabelle 4-3 : Analyseergebnisse Projekt 2

Wegen der großen Differenzen zwischen Schätzung und Nachkalkulation ist eine erste Bestimmung der Produktivität sehr ungenau und deshalb zu diesem Zeitpunkt nicht machbar. Sie muß bis nach der Verbesserung der Nachkalkulation verschoben werden. Die Bestimmung der Einflußfaktoren (GSC's) ist recht schwierig. Die Bewertungsskala von 0 bis 5 erwies sich als sehr unpraktisch, deshalb wurde versucht, die Zahlen durch Erklärungen auszutauschen und in einen Fragebogen zu kleiden. Der letzte Stand befindet sich im Anhang (Kap. 8.2).

5 Entwicklung des Tools

Dieses Kapitel beschreibt die Entwicklung des Tools. Für die Entwicklung wurde als CASE-Tool das Object Engineering Tool OEW 2.0 der Firma Innovative Software verwendet. Die Implementierung wurde mit Hilfe von Borland C++ Version 4.52 für Windows durchgeführt. Die folgenden Unterkapitel beschreiben die Entwicklung. Hierbei wurde auf die komplette Dokumentation des Entwicklungsprozesses verzichtet und es wurden nur die wichtigsten Punkte vorgestellt, da dies den Rahmen dieser Publikation sprengen würde (das CASE-Tool generierte in der Implementierungphase schon ca. 1500 Seiten).

5.1 Anforderungsbeschreibung

Auf der Basis der Aufgabenbeschreibung und den Erfahrungen in der Einführung der Metriken ist diese Anforderungsbeschreibung für das Tool entstanden.

5.1.1 Definition des Einsatzbereiches (Problembereich)

Die Einführung von Metriken für den Softwareentwicklungsprozeß erfordert die Schaffung einer gemeinsamen Datenbasis für die Verwaltung der Projektdaten. Die Projekte werden zu diesem Zweck in Projektteile untergliedert. Der gesamte Lebenslauf jedes dieser Teile wird protokolliert und das Durchlaufen der einzelnen Entwicklungsphasen erfaßt. Neben der Projektverwaltung soll das Tool verschiedene Modelle und Metriken verwalten können. Jedem Projekt werden Modelle zugeordnet. Den Projektteilen können Metriken zugewiesen werden, die aus den Modellen des Projektes gewonnen werden. Vom Modell abhängigen Korrekturfaktoren werden dem Projekt direkt zugeordnet. Eine Nachkalkulation nach Projektabschluß ergänzt die in den Projektteilen durchgeführten Schätzungen der zugeordneten Metriken.

Die den Projektteilen zugeordneten Metriken werden für die Kalibrierung der Modelle verwendet. Die Kalibrierung liefert Anhaltspunkte für Aufwandsschätzungen neuer Projekte bzw. Projektteile. Die Genauigkeit der Schätzung kann durch Analysen überprüft werden. Die Analysenergebnisse werden als Diagramme dargestellt. Es können mehrere

Analysenergebnisse gleichzeitig dargestellt werden, was den Vergleich der einzelnen Modelle untereinander erlaubt.

Zur Unterstützung der Nachkalkulation der Projekte bietet das Tool eine Import-Funktion an. Diese liefert Daten über das Projekt bzw. den Projektteilen aus den verwendeten Entwicklungsumgebungen. Der Datenaustausch erfolgt über eine DDE-Verbindung (mit einem eigenen Protokoll) und einer speziellen Tabelle aus einer ODBC-Datenquelle. Nach Abschluß des Importvorgangs kann der Benutzer das Projekt wählen, indem die Daten übernommen werden sollen. Die mit dem Projekt verknüpften Modelle ermitteln aus diesen Daten die Werte für die betroffenen Metriken.

Die Kalibrierung der einzelnen Modelle erfolgt in der Modellverwaltung. Zu diesem Zweck wird die Datenbasis der Projektverwaltung analysiert. Neben der Kalibrierung liefert die Analyse noch Vorschlagswerte für die in dem Modell verwendete Metrik. Die Vorschlagswerte werden in einzelne Klassen organisiert.

5.1.2 Globale Ziele des Systems (erwartete Vorteile)

Das Tool schafft eine gemeinsame Basis für bestehende und noch zu entwickelnde Softwareprojekte, welche die Grundlage für bessere Aufwandsabschätzungen liefert. Die Erfassung der Projekte bzw. deren Lebenszyklus unterstützt die Dokumentation des Entwicklungsprozesses. Weiterhin wird eine einheitliche Form für die Erfassung bzw. Dokumentation des Prozesses eingeführt. Dies erhöht die Vergleichbarkeit der Daten untereinander und steigert die Genauigkeit der Aufwandsschätzung.

5.1.3 Systemfunktionalität aus Benutzersicht

5.1.3.1 Sicht des Benutzers

Der Benutzer kann neue Projekte neu erzeugen oder bestehende öffnen. Das geöffnete Projekt verwaltet neben einer Kurzbeschreibung des Projektes zusätzlich den Projektstand. Für spätere Importvorgänge wird noch eine Kennung für das Importtool angegeben. Das System bietet dem Benutzer eine Auswahl von Modellen, die er mit dem Projekt verknüpfen kann. Die Modellparameter können für jedes Projekt einzeln bearbeitet werden. Vor jedem Bearbeiten der Parameter überprüft das Tool das gesamte Projekt automatisch.

Für Abschätzungen wird das Projekt in Projektteile untergliedert. Das System bietet dem Benutzer die Möglichkeit, neue Projekteile zu erzeugen und bestehende zu öffnen oder aus dem Projekt wieder zu entfernen. Die Projektteile bestehen aus den Stammdaten und einem dynamischen Metrikteil. Die Stammdaten werden aus den folgenden Komponenten gebildet:

1. Name des Teiles

2. Interne Bezeichnung für den Importvorgang

3. Zeitangaben für die einzelnen Entwicklungsphasen

Der Metrikteil besteht aus den Bereichen Schätzung und Kalkulation. Für die Verknüpfung der Metriken mit dem Projektteil liefert das System eine Liste aller möglichen Metriken. Sie wird aus den dem Projekt verknüpften Modellen gewonnen.

Wurden Importdaten in das System transferiert, so können sie für ein ausgewähltes Projekt abgerufen werden. Das System unterscheidet zwischen bestehenden und neuen Projektteilen. Der Benutzer kann angeben, ob die neuen Projektteile zusätzlich mit Metriken eingefügt werden sollen und ob bestehende Teile aktualisiert werden sollen.

Der Benutzer kann Modelle kalibrieren und für spätere Projektteile Schätzklassen definieren. Diese Schätzklassen bestehen aus einem Namensteil und metrikabhänigen Parametern, welche durch das Tool immer aktualisiert werden.

Das System bietet dem Benutzer die Möglichkeit, Analysen zu definieren. Zu diesem Zweck wird eine Vergleichsbasis durch Auswahl eines Projektes oder Modells bestimmt. Als Vergleichsdaten können dann Modelle aus allen installierten Modellen bzw. Projekte aus allen im System verwalteten Projekten gewählt werden. Die Darstellung erfolgt grafisch.

5.1.3.2 Sicht des Exportierers

Obwohl der Exportierer kein realer Benutzer ist, erscheint er dem Tool als solcher. Er holt sich einen Verweis auf eine Importtabelle und sperrt diese. Nach Abschluß des Datentransfers in diese Tabelle gibt er sie wieder frei.

In dem Kapitel 5.4.2 wird ein Beispiel eines Exportieres vorgestellt. Er wurde direkt in die Oberfläche des MS Access 2.0 (German) als Add-In integriert.

5.1.4 Produktmodell

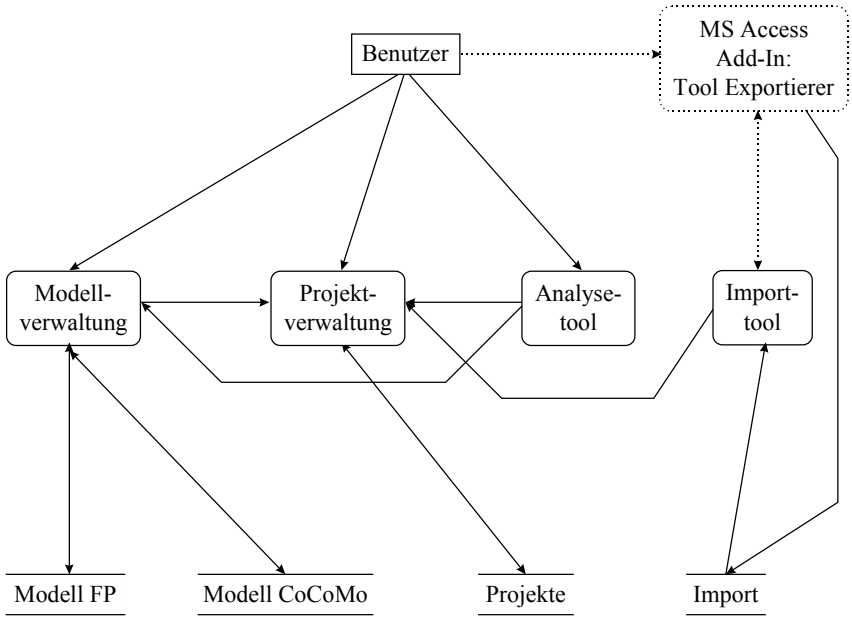


Abbildung 5-1 : Produktmodell

5.1.4.1 Erläuterungen

5.1.4.2 Terminatoren

Benutzer

Der Benutzer legt neue Projekte an und verknüpft Modelle mit den Projekten. Er fügt neue Projektteile ein oder löscht sie wieder. Den Projektteilen weist er Metriken zu und fügt in sie Schätz- und Kalkulationsdaten ein. Der Benutzer startet die Kalibrierung und bearbeitet Schätzklassen. Ebenso werden von ihm Analysen neu erzeugt oder bestehende aufgerufen.

Exportierer (Beispiel Add-In für MS Access 2.0)

Der Exportierer fordert eine Importtabelle an und sperrt diese. Nach dem Datentransfer in die Tabelle hebt er diese Sperrung wieder auf.

5.1.4.3 Prozesse

Projektverwaltung

Die Projektverwaltung nimmt vom Benutzer die Projekt- und Modelldaten entgegen, verwaltet die Projektteile und deren Metriken, überprüft das Projekt und sammelt für die Modelle die Metrikdaten der Projektteile. Sie liefert für die Modellverwaltung und dem Analysetool eine Zusammenfassung des Projektes und untergliedert nach den verknüpften Projekten. Die Projektverwaltung fordert von dem Importtool die aktuellen Importdaten an und fügt sie in das Projekt ein.

Modellverwaltung

Die Modellverwaltung kalibriert die Modelle. Zu diesem Zweck fordert es von der Projektverwaltung Zusammenfassungen aller Projekte an, in denen das angegebene Modell verknüpft ist. Diese Daten liefern zusätzlich Werte für Schätzklassen, die von der Modellverwaltung modellspezifisch umgesetzt werden.

Analysetool

Das Analysetool erhält vom Benutzer Anweisungen über die Vergleichsbasis (Modell oder Projekt) und den verwendeten Projekten bzw. Modellen. Aus diesen Informationen fordert das Analysetool Daten aus der Modell und Projektverwaltung an und kreiert eine Grafik.

Importtool

Exportierer fordern von dem Importtool Informationen über eine passende Importtabelle an. Nach Abschluß des Transfers übergibt der Exportierers die Kontrolle der Tabelle wieder an das Importtool. Diese verwendet das Importtool als Datenbasis für die nächste Anforderung von Importdaten durch die Projektverwaltung. Nach Abschluß des Importvorganges leert das Importtool die betroffene Tabelle.

5.1.5 Qualitätsanforderungen des Programms

Installation des Programmes

Skala	Minuten (Zeitbedarf für die Installation ohne externe Module)
Test	Installation durch einen Entwickler der Abteilung anhand der Installationsanweisung
Schlechtester Fall	60 min
Plan	25 min
Bester Fall	15 min
Jetzt	-

Installation eines Exportierers

Skala	Minuten (Zeitbedarf für die Installation eines Exportieres)
Test	Installation durch einen Entwickler der Abteilung anhand der Installationsanweisung
Schlechtester Fall	30 min
Plan	10 min
Bester Fall	5 min
Jetzt	7 min

Anlegen eines Projektes

Skala	Minuten (Zeitbedarf für die Neuanlage eines Projektes mit anschließender Bearbeitung der Modelle)
Test	Arbeiten mit der Oberfläche unter normaler Last (lokaler Rechner und verwendeter ODBC-Datenbank)
Schlechtester Fall	10 min
Plan	5 min
Bester Fall	2 min
Jetzt	-

Bearbeiten eines Projektes

Skala	Minuten (Zeitbedarf mit öffnen und speichern des Projektes)
Test	Öffnen eines bestehenden Projektes
Schlechtester Fall	5 min
Plan	3 min
Bester Fall	1 min
Jetzt	-

Bearbeiten eines verknüpften Modells.

Skala	Sekunden (Zeit für die Auswahl des Projektes und deren Bearbeitung)
Test	Bearbeiten unter normalen Einsatzbedingungen durch einen eingewiesenen Benutzer
Schlechtester Fall	400 sec
Plan	120 sec
Bester Fall	40 sec
Jetzt	-

Anlegen eines Projektteiles

Skala	Minuten (Zeitbedarf für die Neuanlage eines Projektteiles mit anschließender Bearbeitung der Metriken)
Test	Arbeiten mit der Oberfläche unter normaler Last (lokaler Rechner und verwendeter ODBC-Datenbank). Es existieren für den Projektteil Schätzklassen
Schlechtester Fall	5 min
Plan	3 min
Bester Fall	1 min
Jetzt	-

Bearbeiten eines Projektteiles

Skala	Minuten (Zeitbedarf mit öffnen und speichern des Projektteiles)
Test	Öffnen eines bestehenden Projektteiles, bei geöffneten Projekt
Schlechtester Fall	10 min
Plan	4 min
Bester Fall	3 min
Jetzt	-

Bearbeiten einer verknüpften Metrik

Skala	Sekunden (Zeitbedarf für Auswahl und Bearbeitung)
Test	Auswahl einer Metrik bei geöffnetem Projektteil
Schlechtester Fall	120 sec
Plan	60 sec
Bester Fall	40 sec
Jetzt	-

Importieren in ein Projekt

Skala	Sekunden/Projektteil (Zeitbedarf / Projektteil bei interaktiven Importprozess)
Test	Auswahl eines mittleren Projektes. Messung ab Start Importprozess
Schlechtester Fall	200 sec/pt
Plan	60 sec/pt
Bester Fall	15 sec/pt
Jetzt	-

Kalibrieren eines Modells

Skala	Sekunden (Starten der Kalibrierung)
Test	Durchschnittlicher Benutzer öffnet Modell und startet den Kalibrierungsprozess
Schlechtester Fall	120 sec
Plan	30 sec
Bester Fall	15 sec
Jetzt	-

Bearbeiten von Schätzklassen

Skala	Sekunden (Ändern der Parameter)
Test	Bearbeiten bei ausgewählter Schätzklasse
Schlechtester Fall	60 sec
Plan	20 sec
Bester Fall	5 sec
Jetzt	-

Erstellen von Analysen

Skala	Sekunden (Auswahl einer Vergleichsbasis und den zugehörigen Modellen bzw. Projekten, ohne Aufbereitungszeit)
Test	Feldversuch mit Entwickler bei Durchschnittlicher Projekt- und Modellanzahl)
Schlechtester Fall	120 sec
Plan	30 sec
Bester Fall	15 sec
Jetzt	-

5.1.6 Umgebungsbeschreibung

5.1.6.1 Erwartete Benutzergruppen

Erwartete Benutzergruppen sind die Entwickler der Automatisierungstechnik der EMR-Abteilung. Für den als Benutzer auftretenden Exportierer des Entwicklungssystemes existiert zur Zeit nur das Add-In von MS Access 2.0 (German).

5.1.6.2 Oragnisatorische Einbindung in existierende Strukturen

Das System soll in den Softwareentwicklungsprozess der Abteilung integriert werden. Da zur Zeit als Entwicklungssystem MS Access 2.0 (German) eingesetzt wird, erfolgt der Ersteinsatz kurz vor Fertigstellung des Pflichtenheftes. Die mit dem MS Access definierten Funktionen bzw. Formulare, Berichte etc. werden als Schätzung in das Tool importiert und dienen für die spätere Nachkalkulation als Zählpunkte. Während der Projektabwicklung werden die Zeiten der importierten Projektteile aktualisiert. Nach Abschluß des Projektes werden die Projektdaten aus dem MS Access-Projekt nochmals importiert, wobei. nur die Zählpunkte als Kalkulation aktualisiert und neue Projektteile zu Dokumentationszwecken ohne Metriken importiert werden.

5.1.6.3 Hardware

Das System benötigt folgende Hardwarekomponenten:

- Windowsfähiger PC mit mindestens 8 MByte RAM Hauptspeicher und ein Farbbildschirm
- Netzwerkkomponenten für den Zugriff auf das Gebäude- bzw. Werksnetz
- 2 MByte Festplattenkapzität für das Programmsystem
- mindestens 20 MByte Festplattenkapzität für die ODBC-Datenbasis

5.1.6.4 Andere Software

Das Tool basiert auf dem System Windows 3.1x. Aus diesem Grund wird für das System eine Version 3.11 und aufwärts gefordert. Zusätzlich muß der Zugriff auf das Gebäude-

bzw. Werksnetz durch geeignete Treiber gewährleistet sein. Die Datenbasis des Systems benötigt ein ODBC-fähiges Datenbanksystem. Der verwendete ODBC-Treiber muß mindestens Level 1 unterstützen. Für den Import von Projektdaten werden entsprechende Tools[3] benötigt, die mit dem System via DDE und ODBC kommunizieren.

5.2 Analysephase

Die Analysephase soll an dieser Stelle nicht komplett vorgestellt werden. Dem interessierten Leser sei auf die vom CASE-Tool generierten Daten auf der beigefügten CD (vgl. Kap. 8.6) verwiesen. Die folgenden Kapitel stellen typische Eckpunkte dieser Phase dar. Neben der Abschätzung der Function points wird das Datenmodell präsentiert. Es wird in der Designphase auf Objekte abgebildet.

Für die Analyse wurden folgende Vereinbarungen getroffen:

- Als Modelle werden CoCoMo und die Function point Methode umgesetzt.
- Als Metriken werden Lines of Code und Function points realisiert.
- Das Modell CoCoMo liefert Lines of Code als Metrik
- Die Metrik Function points werden als Metrik der Function point Methode verwendet.
- Das Importtool verwendet das in Kap. 5.4.2 beschriebe Add-In als Exportierer

5.2.1 Erste Schätzung mit Function points

Die Analysephase lieferte die in Tabelle 5-1 aufgeführten Funktionen. Mit Hilfe der aus Tabelle 5-2 ermittelten Werte für die GSC läßt sich folgende Abschätzung für das Tool erstellen:

[3] Ein Importtool ist direkt in das Microsoft Access 2.0 integriert worden. Die Beschreibung befindet sich in Kapitel 5.4.2

$$VAF = 0{,}65 + (0{,}01 * TDI) = 0{,}65 + (0{,}01 * 18) = 0{,}83$$
$$FP = UFC * VAF = 164 * 0{,}83 = 136{,}12\text{fp}$$

Gleichung 5-1

Funktion	Art	Dateien	Record-typen	Daten-elemente	Punkte
Projekt erzeugen	Eingabe			2	3
Projekt bearbeiten	Eingabe			5	3
Projekt löschen	Eingabe			2	3
Projektteil erzeugen	Eingabe			3	3
Projektteil bearbeiten	Eingabe			15	3
Projektteil löschen	Eingabe			2	3
Projektteile importieren	Abfrage			8	3
Modellparameter Function point bearbeiten	Eingabe			14	3
Modellparameter CoCoMo bearbeiten	Eingabe			19	4
Metrik LOC bearbeiten	Eingabe			2	3
Metrik Function points	Eingabe			7	3
Modell Function point	Abfrage			4	3
Schätzklasse Function points	Eingabe			5	3
Modell CoCoMo	Abfrage			9	3
Schätzklasse CoCoMo	Eingabe			2	3
Analyse	Abfrage			2	3
LOC	Tabelle		1	5	7
FP	Tabelle		1	9	7
ParamCoCoMo	Tabelle		1	10	7
ParamFP	Tabelle		1	5	7
KorrCoCoMo	Tabelle		1	20	7
KorrFP	Tabelle		1	18	7
Modell	Tabelle		2	2	7
Projekt	Tabelle		3	8	7
Pitem	Tabelle		4	25	10
Analyse	Tabelle		2	2	7
Aconn	Tabelle		1	2	7
Mconn	Tabelle		1	2	7
Class	Tabelle		2	2	7
ClassLOC	Tabelle		1	3	7
ClassFP	Tabelle		1	6	7
ImportAccess	Tabelle		2	7	7
				Gesamt :	164

Tabelle 5-1 : Abschätzung des Tools

i	GSC_i	Erklärung
1	3	Importierer, die über DDE kommunizieren
2	2	ODBC-Datenbank muß nicht auf dem lokalen Rechner sein.
3	2	Änderungen der Projektteile dürfen nicht zu lange dauern.
4	0	Hauptlast liegt bei der ODBC-Datenbank
5	0	Hauptlast liegt bei der ODBC-Datenbank
6	1	Da nur als Einzelplatzversion geplant
7	4	Dateneingabe darf nicht zu lange dauern
8	4	Automatische Überprüfung der Daten vorgesehen
9	1	Regressionsanalyse aufwendig.
10	0	Keine Wiederverwendung geplant
11	0	Keine Angaben über die Installation
12	0	Keine Angaben definiert
13	0	Einzelplatzversion auf nur einem Zielsystem
14	1	Analyse frei definierbar
TDI =	18	

Tabelle 5-2 : Schätzung GSC

Die Nachkalkulation befindet sich in der Datenbasis des fertigen Tools (Vgl. Kap. 8.6). Aufgrund der geringen Anzahl von bisher untersuchten Projekten wird hier auf eine Zeitschätzung verzichtet.

5.2.2 Datenmodell

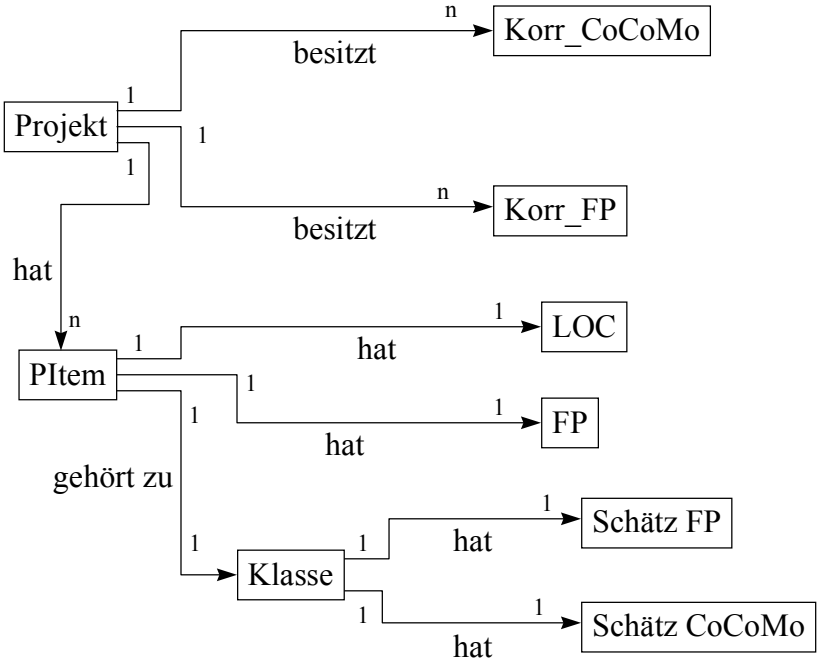


Abbildung 5-2 : Datenmodell (Teil 1)

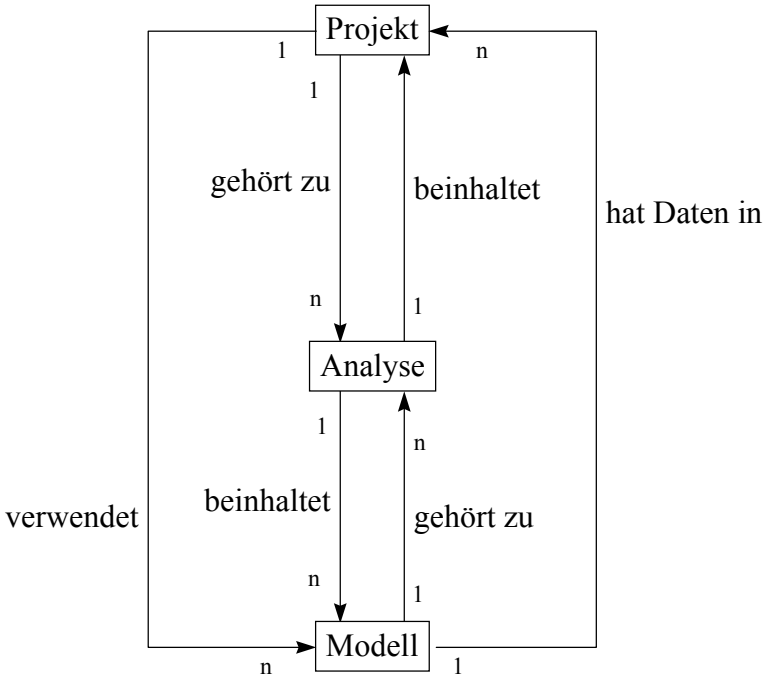


Abbildung 5-3 : Datenmodell (Teil 2)

Aus Gründen der Übersichtlichkeit wurde das Datenmodell auf die Abbildung 5-2 und Abbildung 5-3 verteilt. Das vorgestellte Datenmodell mußte für den Einsatz in der objektorientierten Entwicklungsumgebung in einer passenden Form umgewandelt werden. Die Umsetzung wird in dem folgenden Kapitel beschrieben.

5.3 Designphase

In diesem Kapitel werden einige für die Entwicklung wichtige Designentscheidungen vorgestellt. Teile dieser Entscheidungen überschneiden sich mit der Analysephase, da sie aber zum Teil für das Design wichtig sind, werden sie in den folgenden Kapiteln beschrieben.

5.3.1 Umsetzung des Datenbankmodells in das objektorientierte Design.

Abbildung 5-4 : Datenmodell objektorientiert

Die Umsetzung des Datenmodells ist in Abbildung 5-4 dargestellt. Zu diesem Zweck wurde zunächst die ODBC-API gekapselt. Sie wurde in zwei Komponenten unterteilt:

- Database; dient zur Verwaltung von Verbindungen zu den Datenbanken
- Connection; stellt eine logische Verbindung zu einer Datenbank dar. Sie wird von den Objekten mit Datenbankzugriff verwendet.

Alle Objekte mit Datenbankzugriff werden von dem Objekt DB_Item abgeleitet. Es beinhaltet neben dem Verweis auf Connection noch Slots für die Zustände, die das Sichern der Daten regelt. Die genaue Beschreibung des Objektes kann man aus der von CASE-Tool generierten Dokumentation entnehmen (Vgl. Kap. 8.6).

5.3.2 Organisation der Modelle und Metriken

Ein Modell besteht aus den in Tabelle 5-3 beschriebenen Komponenten. Zukünftige Modelle werden durch Vererbung der Klasse Modell gebildet. Metriken werden durch die in Tabelle 5-4 dargestellten Komponenten definiert. Für die Modellverwaltung wird zusätzlich für jedes Modell eine Präsentationskomponente konstruiert. Sie werden von dem Objekt PresModell abgeleitet. Die Beziehung zu den Modellen kann man aus Abbildung 5-5 entnehmen.

Slot	Kommentar
ID	Eindeutige ID
Name	Eindeutiger Name
Metrik	Wird von DB_ModellData abgeleitet
Anzeige der Modellparameter	Wird von ViewModell abgeleitet
Parametersatz	Der Parametersatz wird von DB_Item abgeleitet
Kalibrierungsschnittstelle	Unterteilt in Vorbereitung, Daten importieren und Abschluß
Generische Ausgabe von Schätz- und Kalkulierten Daten	Wird für das Analysetool verwendet.
Operatoren für die Containerverwaltung	Dienen für die Organisation der Modelle in den Projekten

Tabelle 5-3 : Elemente des Modells

Abbildung 5-5 : Modelle und Metriken

Slot	Kommentar
ID	Eindeutige ID
Name	Eindeutiger Name
metrikabhänigen Teil	Verwaltet die Metrikdaten; unterteilt in Schätzung und Kalkulation
Anzeige	Stellt die Daten dar und gibt dem Benutzer eine Möglichkeit, die Daten zu bearbeiten
Operatoren für die Containerverwaltung	Dienen für die Organisation der Modelle in den Projekten

Tabelle 5-4 : Elemente der Metriken

5.4 Importieren von Projektdaten aus den Entwicklungstools

Für die Nachkalkulation wurde eine Importmöglichkeit für bestehende Projekte gefordert. Aus diesem Grund ist die folgende Schnittstelle definiert worden. Eine Realisierung für das Microsoft Access 2.0 befindet sich im Kapitel 5.4.2.

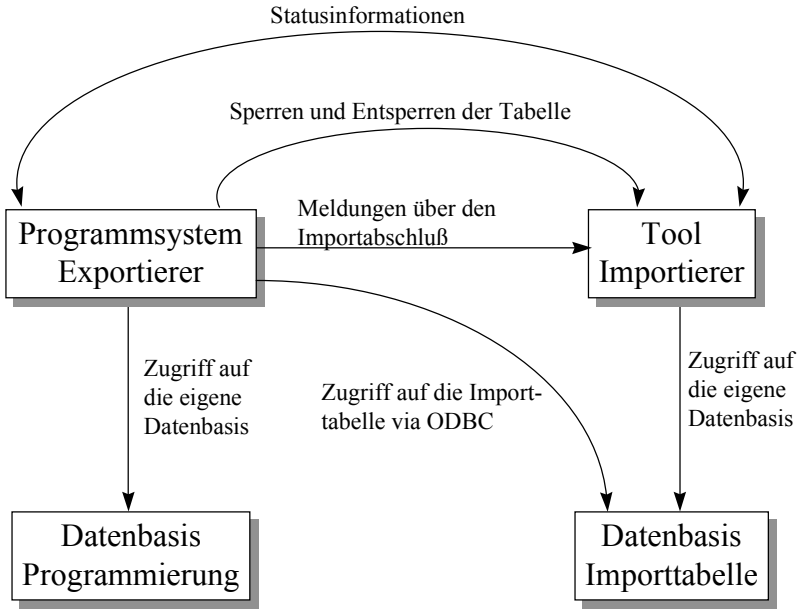


Abbildung 5-6 : Kommunikation zwischen Programmsystem und Tool

Für den eigentlichen Importvorgang werden zwei Prozesse benötigt. Ein Prozeß befindet sich im Tool (Tool Importierer) und regelt die Kommunikation mit dem Fremdprozeß (Programmsystem Exportierer). Die genaue Kommunikation wird im folgenden Kapitel erläutert.

Der Fremdprozeß arbeitet nach folgendem Schema:

1. Es werden die zu exportierenden Daten vorbereitet
2. Die Kommunikation mit dem Toolprozeß wird aufgebaut
3. Es wird der Status des Tools abgefragt („Ready“ : Datenbasis bereit für Importvorgang, „intern“ Datenbasis durch das Tool gesperrt, „Extern“ : Datenbasis durch einen Exportierer gesperrt „Extern“)
4. Falls Status ungleich „Ready“ noch mal versuchen, sonst Importsystem setzen und Informationen über die ODBC-Datenquelle holen
5. Einen Zugriffspfad bilden und Importtabelle sperren.
6. Daten in die Importtabelle transferieren
7. Tabelle wieder freigeben und Zugriffspfad löschen
8. Meldung über den Importabschluß an das Tool leiten
9. Verbindung abbauen
10. Fremdprozeß beenden

5.4.1 Kommunikation zwischen Tool und Exportierer

Die Kommunikation erfolgt mit Hilfe des DDE-Protokolls. Dem Exportierer wird die Rolle des Clients zugewiesen und das Tool erscheint als Server. Als Verbindungstyp wird der „Cold Link“ gewählt. Dieser Modus reicht für die Kommunikation völlig aus. Eine Beschreibung dieser Verbindungsart befindet sich im Anhang (Kap. 8.3).

Der Clientprozeß fragt mit Hilfe der sog. „Requests“ Daten vom Server ab und kann Befehle „Execute“ an den Server senden, der diese dann ausführt. In den folgenden Tabellen werden die einzelnen Requests und Befehle vorgestellt.

Request	Aktion	Resultat
System	Liefert alle unterstützten Importsysteme	Liste der unterstützten Systeme getrennt durch „|“. Z.B. „MSAccess 2.0|ROSY|QSP WIN|RKA"
Source	Liefert den Namen der Datenquelle und die Tabelle	Format : “Quellennamen|User|Passwort|Tabellenname“. Z.B."MetrikDB|admin||ImportAccess“
Status	Liefert Informationen über das System	Es wird eine Zeichenkette aus allen Servervariablen nach folgenden Schema erzeugt: „NAME=WERT|NAME=WERT....“

Tabelle 5-5 : Requests vom Server

Execute	Aktion	Parameter	Format
ImportBeendet	Weist den Server an, daß der Import erfolgreich beendet wurde	Keine	ImportBeendet
ImportAbbruch	Der Import wurde abgebrochen	Keine	ImportAbbruch
Set	Setzen einer Variablen	Name und Wert der Variablen	Set(NAME=WERT)
LockTable	Sperrt oder gibt die Importtabellen wieder frei	„ON“ oder „OFF“	LockTable(ON) LockTable(OFF)

Tabelle 5-6 : Befehle, die der Server unterstützt

Der Server unterstützt folgende Variablen, die mit dem Request „Status“ abgefragt und mit dem Befehl „Set“ gesetzt werden können, falls sie beschreibbar sind.

Name	Wert	Zugriff
ProjectName	Name der Importierten Datenbank	RW
Import	Zustand der Importmöglichkeit (Ready, Extern, Intern)	RO
ImportSys	Name des Importsystems (Teil aus Request System)	RW
DBFormat	Ausgabeformat der Datenquellenangabe	RW
DBEngine	Verwendete ODBC-Engine	RO

Tabelle 5-7 : Servervariablen

5.4.2 Exportieren für das Microsoft Access 2.0 (German)

Als Basis für diesen Exportierer wurde das Add-In „Datenbankdokumentierer“ des Microsoft Access 2.0 verwendet. Dieser bereitet die Daten für den Export vor. Der Zugriff auf die Importtabelle erfolgt durch eine Verknüpfung in die Datenbank des Dokumentierers. Hierfür wird die Tabelle als SQL-Datenbanktabelle verwendet. Da man bei der Verknüpfung einer ODBC-Datenquelle in eine Access-Datenbank keine ODBC-Datenquellen auf Basis der Access Datenbankengine durchführen kann, wurde eine

Möglichkeit geschaffen, diese Tabelle als Access Datenbanktabelle direkt zu verknüpfen. Aus diesem Grund wurden die Variablen „DBFormat“ und „DBEngine“ eingeführt.

5.4.2.1 Erste Version des Exportierens

Für die Entwicklung der ersten Version des Exportierers wurde ein DDE-Server entwickelt, der das Verhalten des Importprozesses simuliert. Gestartet wird der Exportierer durch den Menüaufruf „Tool Exportierer“.

Abbildung 5-7 : Aufruf des Exportierers

Nach dem Aufruf erscheint der in Abbildung 5-8 dargestellte Dialog. Mit Hilfe der Pfeiltasten kann man den Exportvorgang Schritt für Schritt durchführen. In der nächsten Version wird die Funktion der Pfeiltasten durch den Dialog selbst durchgeführt.

Abbildung 5-8 : Start Exportierer

Gestartet wird der Exportierungsvorgang durch die rechte Pfeiltaste. Dieser startet den internen Datenbankexportierer (Abbildung 5-9).

Abbildung 5-9 : Der interne Datenbankdokumentierer hat seine Arbeit abgeschlossen

Im nächsten Schritt wird die DDE-Verbindung zum Tool aufgebaut (Abbildung 5-10).

Abbildung 5-10 : Aufbau DDE-Verbindung zum Tool

Nach dem erfolgreichen Aufbau der DDE-Verbindung erfolgt die Überprüfung des Tool-Zustandes. Bei ODBC-Datenquellen mit Access Datenbankengine muß der Server auf das Access-typische Format zur Datenquellenbezeichnung umgeschaltet werden (Abbildung 5-11).

Abbildung 5-11 : Überprüfung des Zustandes des Servers

Die Verknüpfung der Importtabelle wird jetzt durchgeführt. Hierfür wird auf dem Server die Tabelle gesperrt und die Quellenbezeichnung vom Server angefordert (Abbildung 5-12).

Abbildung 5-12 : Importtabelle wurde verknüpft

Jetzt können die Daten in die Importtabelle transferiert werden (Abbildung 5-13).

Abbildung 5-13 : Daten werden in die Importtabelle kopiert

Nach Abschluß des Transferierens wird die letzte Phase des Exportvorganges eingeleitet. Es wird die Verknüpfung der Importtabelle gelöst und auf dem Server die Sperrung wieder aufgehoben. Zum Schluß wird eine Fertigmeldung an den Server gesendet (Abbildung 5-14).

Abbildung 5-14 : Abschluß des Exportvorganges

6 Das Tool

Nach der im vorherigen Kapitel beschrieben Entwicklung des Tools wird im folgenden die Bedienung und Installation des Tools vorgestellt.

6.1 Beschreibung des Tools

6.1.1 Hauptkomponenten

Abbildung 6-1 : Hauptfenster des Tools

Das Tool verwaltet die Projekte und Analysen als Dokumente, die man neu erzeugen, öffnen, speichern, umbenennen, schließen und löschen kann. Da die Modelle fest im Tool integriert sind, kann man sie nur öffnen, speichern und schließen. Dank der Multidokumentoberfläche können gleichzeitig beliebig viele[4] Dokumente und Sichten auf diese Dokumente geöffnet und dargestellt werden.

Datei/Neu	Dokumente werden durch Angabe des Dokumententyps und dem Namen neu erzeugt (Vgl. Abbildung 6-2).

[4] Genaugenommen wird die Anzahl nur durch den Speicher und der sich in der Datenbasis gespeicherten Dokumente begrenzt.

Datei/Öffnen	Bestehende Dokumente können durch Auswahl des Dokumententyps und dem entsprechenden Namen aus einer Auswahlliste geöffnet werden (Vgl. Abbildung 6-3).
Option/ Einstellungen	Globale Projekteinstellungen können mit Hilfe des in Abbildung 6-4 dargestellten Dialogs vorgenommen werden. Der Benutzer kann festlegen, welche Modelle automatisch mit einem neu erzeugten Projekt verknüpft werden sollen. Für den Importvorgang können folgende Vorgaben ausgewählt werden: Der Importvorgang verläuft im interaktiven Modus (d.h., daß unter anderem die Benennung des neuen Projektteiles beeinflußt werden kann). Metriken von bestehenden Projektteilen können aktualisiert werden und neue Projektteile werden mit Metriken eingefügt.

Abbildung 6-2 : Dokument neu erzeugen

Abbildung 6-3 : Dokument öffnen

Abbildung 6-4 : Einstellungen

6.1.2 Projektverwaltung

Nachdem ein Projekt neu erzeugt oder geöffnet wurde, erscheint das Hauptfenster des Projektes (siehe Abbildung 6-5). Der Menüpunkt »Projekt« wird in die Menüleiste eingefügt. Über ihn können Projekte umbenannt und gelöscht werden. Falls sich im System Importdaten befinden, können sie über Projekt/Importieren in das Projekt importieren (Vgl. Kap. 6.1.2.2).

Abbildung 6-5 : Projekt-Hauptfenster

Die Projektanzeige ist in zwei Bereiche unterteilt. In dem oberen Bereich werden die Stammdaten des Projektes verwaltet. Zu den Stammdaten zählt eine Kurzbeschreibung des Projektes und eine Bezeichnung des Importierers (interner Name). Der Projektstand kann als Phase ausgewählt werden. Für spätere Auswertungen kann man den Projektbeginn und das Projektende angeben. Durch betätigen des jeweiligen Buttons neben den Datumsfelder öffnet als Eingabehilfe ein Kalenderblatt.

Die Überprüfung des gesamten Projektes kann manuell durch den Button »Überprüfen« gestartet werden. Sie überprüft die Projektteile auf Datenkonsistenz. Hat die Überprüfung Fehler entdeckt, so werden sie in einem Fenster aufgelistet. Typische Fehlerquellen sind Projekteile, die Metriken besitzen und bei denen die Angaben für die verwendete Zeit des Projektteiles fehlen.

Abbildung 6-6 : Modellverknüpfungen bearbeiten

Der untere Bereich des Projektfensters ist für die Verwaltung der mit dem Projekt verknüpften Modelle vorgesehen. Mit Hilfe des Buttons »Bearbeiten« wird der Dialog »Verknüpfungen bearbeiten« geöffnet (Abbildung 6-6). Die in dem Tool installierten Modelle werden als „Verknüpfte“ und „Nicht Verknüpfte“ dargestellt. Durch Anwahl mit der Maus und den Buttons »Zufügen« und »Entfernen« können die Modelle mit dem Projekt verknüpft bzw. die Verknüpfung mit dem Projekt wieder gelöscht werden.

Abbildung 6-7 : Parameter CoCoMo-Modell

Der Button »Anzeigen« öffnet ein markiertes Modell. Die geöffnete Anzeige ist vom ausgewählten Modell abhängig. Vor dem Öffnen der Anzeige wird automatisch die Projektüberprüfung gestartet. In der jetzigen Version hat das Tool zwei Modelle installiert.

Wurde das CoCoMo-Modell ausgewählt, erscheint die in Abbildung 6-7 dargestellte Übersicht der Schätzparameter.

Der linke Bereich ist für die eigentlichen Parameter des CoCoMo-Modells zuständig. Der Benutzer kann die Schwierigkeitsklasse des Projektes bestimmen. Die Produktivitätsfaktoren können durch Öffnen der jeweiligen Dialoge (Button »Produktattribute«, »Resourcenattribute« und »Projektattribute«) eingestellt werden. Die Auswahl in diesen Dialogen erfolgt über vorgegebene Werte. Die so erhaltenen Werte liefern mit den Metrikdaten (hier Lines of Code) aus den Projektteilen die Aufwandsabschätzung. Sie ist unterteilt in geschätzte und kalkulierte Metrikdaten.

Abbildung 6-8 : Modelldarstellung Function point

Die Anzeige für das Function point Modell (Abbildung 6-8) ist ähnlich aufgebaut. Die Bestimmung der Korrekturfaktoren (Abbildung 6-9) erfolgt durch direkte Eingabe der Korrekturziffer (0 bis 5) oder durch einen Dialog, der mit Hilfe des jeweiligen Buttons »...« am rechten Ende der Zeile geöffnet wird. Der Dialog zeigt die jeweilige Bedeutung des Zahlenwertes des betroffenen Korrekturfaktors. Die Aufwandsabschätzung ist auch hier wieder in geschätzte und kalkulierte Metrikdaten unterteilt. Das Function point Modell verwendet Function points als Metrik in den Projektteilen.

Abbildung 6-9 : Korrekturfaktoren Function point

6.1.2.1 Projektteile

Projektteile können mit Hilfe der Menüpunkte »Projekt/Projektteil neu erzeugen« »Projekt/Projektteil öffnen« neu erzeugt bzw. geöffnet werden. Bei der Neuerzeugung können ausgewählte Metriken automatisch miterzeugt werden. Die Liste der möglichen Metriken wird aus der mit dem Projekt verknüpften Modellen gewonnen. Die Angabe eines Typs liefert Vorschlagswerte für die Schätzung. Diese werden für jede Metrik als Schätzklasse in der Modellverwaltung definiert und bei der Modellkalibrierung aktualisiert.

Abbildung 6-11 : Bestehende Projektteile öffnen

Abbildung 6-10 : Projektteile neu erzeugen

Die Sicht auf das geöffnete Projektteil ist wie in Abbildung 6-12 dargestellt in verschiedene Bereiche unterteilt. Wie in dem Projekthauptfenster kann man auch hier mit

einer Bemerkung den Projektteil kurz beschreiben. Der „interner Name“ wird von dem Importierer gesetzt und dient ihm als Erkennungsmerkmal für weitere Importvorgänge. Den Stand kann man mit einem Editfeld mit Kalenderblatt anpassen. Mit Hilfe des Buttons »Eigenschaften« kann man die Schätzklasse verändern. Dies hat Auswirkungen auf die spätere Kalibirierung der Schätzklassen.

Abbildung 6-12 : Projektteil

Die Angabe der Phasenzeiten erfolgt in einer Zusammenfassung oder wie in Abbildung 6-13 für die jeweils ausgewählte Phase. Die Zeitangabe erfolgt in dem Format Monate - Tage - Stunden. Intern werden die Zeiten als Stunden abgelegt und mit den über »Optionen/Einstellungen« festgelegten Umrechnungsfaktoren entsprechend umgewandelt.

Abbildung 6-13 : Ansicht einzelne Phase

Der untere Teil der Projektteilanzeige ist für die Metriken vorgesehen. Sie können mit Hilfe des Buttons »Hinzufügen/löschen« beliebig hinzugefügt oder gelöscht werden. Die Anzeige einer bestimmten Metrik erfolgt durch Auswahl aus der Combo-Box.

Die Anzeige der Metrik Lines of Code (Abbildung 6-14)besteht aus den geschätzten und den kalkulierten Zeilen. Die von dem CoCoMo-Modell geforderten KDSI werden vom Modell selbständig umgerechnet.

Abbildung 6-14 : Anzeige Lines of Code

Abbildung 6-15 : Anzeige Function points

Die Metrik Function points (Abbildung 6-15)besteht aus Klasse der Funktion und der Angabe der Anzahl von Dateien, Recordtypen und Datenelemente der Funktion, jeweils als Schätzung und Kalkulation. Die Metrik berechnet bei vollständigen Angaben die Punktezahl als Schätzung und Kalkulation.

6.1.2.2 Importierer

Als letzte Sicht auf das Projekt kann man den Import öffnen. Diese Sicht läßt sich nur öffnen, wenn im System Importdaten vorhanden sind. Abbildung 6-16 zeigt den Importierer. Hier kann der Benutzer bestimmen, wie neue Projektteile in das Projekt eingefügt werden sollen. Für eine automatische Nachkalkulation des Projektes empfiehlt es sich, neue Projektteile ohne Metriken einzufügen. Dies erhöht die Genauigkeit der Kalkulation. Durch markieren von »Einfügen interaktiv« erhält der Benutzer die Möglichkeit, bei neuen Projektteilen einzugreifen. In dem Abschnitt Modelle kann bestimmt werden, wie die Metriken eingefügt werden sollen. Hierbei hat man die Möglichkeit, sie als Schätzung und Kalkulation einfügen zu lassen. Es werden nur die markierten Metriken eingefügt. Sollen zusätzlich noch die Metriken bestehender Projektteile aktualisiert werden, muß man dieses markieren.

Abbildung 6-16 : Improtierer

Für die im vorherigen Absatz erwähnte Möglichkeit neue Projektteile interaktiv einzufügen, erzeugt das Tool den in Abbildung 6-17 dargestellten Dialog. In ihm kann der Benutzer den Namen des neuen Projektteiles verändern oder das Projektteil vom Importvorgang durch »Übergehen« ausschließen.

Abbildung 6-17 : Interaktiver Importmodus

6.1.3 Modelle

In dieser Version des Tools sind die Modelle CoCoMo und Function point installiert. Sie können als Dokument mit Typ Modell geöffnet, gespeichert und geschlossen werden. Hauptaufgabe dieses Dokumententyps liegt in der Kalibrierung des Modells und der Schätzklassen. Wird beim Öffnen das CoCoMo-Modell ausgewählt, so erscheint das in Abbildung 6-18 dargestellte Fenster. Es zeigt alle Faktoren des CoCoMo-Modelles an und bietet den Benutzer an, diese Werte zu aktualisieren (Button »Werte aktualisieren«). Dann

wird ein Prozeß gestartet, der alle Projekte, mit denen das CoCoMo-Modell verknüpft ist, überprüft und die entsprechenden Daten sammelt. Gleichzeit werden bei diesem Vorgang auch alle Schätzklassen aktualisiert, die mit dem Modell verbunden sind. Die Bearbeitung der Schätzklassen wird durch den Button »Typen bearbeiten« aktiviert. Abbildung 6-19 zeigt den Dialog für die Bearbeitung. In ihm können neue Klassen definiert werden oder bestehende Überarbeitet werden.

Abbildung 6-18 : Modell CoCoMo

Abbildung 6-19 : Schätzklassen für CoCoMo

Die Anzeige für das Function point Modell ist ähnlich aufgebaut. Abbildung 6-20 stellt das Anzeigefenster dar. Neben der mittleren Anzahl von Functionspunkten pro Personenmonat, wird auch noch eine zweite Produktivitätsfunktion erzeugt. Sie wird durch die quadratische Regression bei dem Kalibirierungsvorgang gebildet. Die Bearbeitung der Schätzklassen kann analog zu der des CoCoMo-Modells durchgeführt werden. Hier werden, wie in Abbildung 6-21 dargestellt, statt den Lines of Code die Function points spezifischen Parameter präsentiert.

Abbildung 6-20 : Modell Function point

Abbildung 6-21 : Schätzklassen Function points

6.1.4 Analysen

Analysen werden als Grafik erzeugt. Nach dem neuen anlegen einer Analyse muß zunächst eine Vergleichsbasis gewählt werden. Für ein Modell als Vergleichsbasis erfolgt die Auswahl des Modells über »Analyse/Modelle bearbeiten« (Abbildung 6-22). Projekte können über »Analyse/Projekte bearbeiten « (Abbildung 6-23) der Analyse hinzugefügt werden. Bei einem Projekt als Vergleichsbasis erfolgt die Auswahl der Analysekomponenten analog.

Abbildung 6-23 : Projekte auswählen

Abbildung 6-22 : Modell auswaählen

Die fertig definierte Analyse wird wie in Abbildung 6-24 dargestellt. Sie kann abgespeichert werden und erneut aufgerufen werden. Änderungen der Projektdaten werden automatisch in der Analyse dargestellt.

Abbildung 6-24 : Analyse

6.2 Installation des Tools

Die Installation des Tool muß nach folgenden Schritten durch geführt werden. Für den Vorgang wird angenommen, das sich die CD im Laufwerk D: befindet und das Tool auf die Platte C: in daß Verzeichnis Tool installiert wird.

1. Verzeichnis „C.\Tool“ anlegen.

2. Dateien von Verzeichnis „D:\Tool“ in das angelegte Verzeichnis kopieren und den Schreibschutz der kopierten Dateien entfernen.

3. Dateien von Verzeichnis „D:\Tool\Windows“ in das Windows-Verzeichnis kopieren und den Schreibschutz der kopierten Dateien entfernen.

4. Dateien von Verzeichnis „D:\Tool\Windows\System“ in das Windows Systemverzeichnis kopieren und den Schreibschutz entfernen.

5. Definieren der ODBC-Datenquelle „MetrikDB“:

   Auswahl MS Access 2.0 Treiber

Abbildung 6-25 : ODBC-Treiberkonfiguration

- Datenbank Tool\MetrikDB.mdb
- Benutzer Admin, Kein Passwort

Abbildung 6-26 : Definition der Datenquelle

6. Starten des Programmes

6.3 Installation des Importierers für MS Access 2.0 (German)

1. Accsss beenden

2. Alte WZLIB.mda und WZLIB.ldb sicheren

3. Dateien aus dem Verzeichnis Tool\Access.200 in das Access-Verzeichnis kopieren und den Schreibschutz entfernen.

INI-Datei MSACC20.ini wie folgt ändern:
Abschnitt [Menu Add-Ins]:
&Add-In-Manager==Wm_Entry()
Tool &Exportierer==Exp_Start()
&Datenbank-Dokumentierer==Doc_PrintDataBase(0)
..

Abschnitt [Documentor]:

Option00=829

Option03=391

Option01=799

7 Zusammenfassung

Abschließend kann man sagen, daß die Einführung von Metriken schwierig ist. Es traten nicht nur Probleme bei der eigentlichen Umsetzung der Metriken auf, sondern es zeigten sich auch Bedenken über den Sinn und Zweck der Metriken. Einige hielten die Metriken für zu ungenau und blieben bei den alten Abschätzverfahren.

Die Function points erwiesen sich Für den Einstieg in die Metriken als gut geeignet, da man erste Schätzungen schon ab dem Pflichtenheft durchführen konnte. Nach der genauen Definition, was wie gezählt werden soll, kam man sehr schnell mit dem Bewertungsverfahren zurecht. Die Anwendung der anderen Modelle und Metriken erwies sich als zu umständlich. Probleme mit den Function points gab es nur mit der automatischen Umsetzung in der „pre Tool Phase“. In dieser Zeit liefen die Analysen noch mit einfachen SQL-Abfragen, die in der Nachkalkulation immer zu viele Funktionen zählte. Die am Anfang der Diplomarbeit erhofften schnellen Ergebnisse hinsichtlich der Genauigkeit stellten sich nicht ein. Vielmehr wurde klar, daß der Weg bis zu diesen Genauigkeiten über viele Projekte laufen muß. Welchen Einfluß der Wechsel auf neue Entwicklungstools auf diesen Prozeß haben wird, muß noch in späteren Untersuchungen geklärt werden.

Die für das Tool verwendete ODBC-Datenbasis hemmte die Entwicklung. Teilweise gab es Probleme mit der Funktionalität der ODBC-API. Selbst die in der Dokumentation beschriebenen Beispiele liefen nicht korrekt ab. Dieser Umstand zwang zu zeitaufwendigen Workaround, die die Entwicklung des Tools verzögerten.

Das entstandene Tool kann man als gemeinsame Datenbasis für weitere Metriken und Modelle sehen. Die Projekte werden in einer unabhängigen Form verwaltet. Die einzige Abhängigkeit zu den Entwicklungstools besteht in der Schaffung von geeigneten Exportierern. Die unabhängige Art der Verwaltung der Projekte und deren Teile bietet eine Vergleichsbasis über die kompletten Projekte, die eine Abteilung im Lauf der Zeit und durch Einsatz neuer Tools entwickelt wurden. Das Tool erhöht auch die Genauigkeit der automatischen Nachkalkulation durch selektives Importieren der Projektteile und deren Metriken.

Wie sieht die Zukunft aus?

Nach und nach sollten weitere Metriken in den Entwicklungsprozeß eingeführt werden. Ein Einsatz von Phasenmetriken würde die einzelnen Entwicklungsphasen hinsichtlicht der Aufwandabschätzung transparenter machen. Auch müßten Metriken für die Wiederverwenbarkeit von Projektteilen erarbeitet werden.

8 Anhang

8.1 Abfragen erste Untersuchungen

Für die erste Untersuchung wurde mit Hilfe des Dokumentierers aus dem MS Access eine Tabelle „Objektdefinition“ erzeugt. Diese Tabelle besteht aus folgenden Spalten:

- ID
- ParentID
- Object Type
- Name
- Extra1
- Extra2
- Extra3

Für die Analyse wurde zunächst eine Auswahlabfrage „Typen“ erzeugt:

```
SELECT DISTINCTROW Objectdefinition.ID,
                   Objectdefinition.Name
       FROM Objectdefinition
       WHERE (([Objectdefinition]![Object Type])="Formular"
          Or  ([Objectdefinition]![Object Type])="Bericht"
          Or  ([Objectdefinition]![Object Type])="Tabelle"));
```

Die Abfrage „Typen“ liefert alle Elemente, die gezählt werden können. Für die Function points findet man dort alle möglichen Funktionen gemäß den in Kapitel 4.3 getroffenen Definitionen. Für die Ermittlung der Komplexität der einzelnen Funktionen wird die folgende Abfrage „Komplett“ angewandt:

```
SELECT DISTINCTROW Typen.Name,
                   Objektdefinition.[Object Type],
                   Objektdefinition.Extra1
      FROM Objektdefinition, Typen
      WHERE ((([Objektdefinition]![Object Type])
                              ="Steuerelement")
        And (([Objektdefinition]![Extra1])<>"Linie"
        And  ([Objektdefinition]![Extra1])
                    <>"Bezeichnungsfeld")
        And  (([Objektdefinition]![ParentID])=[Typen]![ID]))
        Or  ((([Objektdefinition]![Object Type])
                             <>"Steuerelement"
        And   ([Objektdefinition]![Object Type])="Spalte")
        And  (([Objektdefinition]![ParentID])=[Typen]![ID]))
      ORDER BY Typen.Name;
```

Die Ergebnistabelle liefert für jede Funktion die zugehörigen Datenelemente, die zur Bestimmung der Komplexität herangezogen werden kann.

8.2 Formular Korrekturfaktoren

GSC	0	1	2	3	4	5
Datenaustausch mit externen Programmen	Reine Batch-Applikation	Dateneingabe oder -ausgabe von einer entfrenten Station (remote entry)	Dateneingabe und -ausgabe von einer entfernten Station (remote entry and printing)	Die Applikation besitzt ein spezielles Subsystem zur Datenein- und ausgabe an einer entfernten Station	Applikationen mit einem signifikanten Anteil von Ein- und Ausgabe an einer entfernten Station	Applikation, die zu wesentlichen Teilen auf einer entfernten Station abgewickelt werden
Verteilte Applikationen	Nur ein einzelner Prozessor vorhanden	Die Applikation läuft zwar nur auf einem Prozessor, bereitet allerdings Daten für andere Prozessoren auf	Die Applikation verteilt sich auf ein paar (wenige) Prozessoren	Die Applikation läuft auf mehreren Prozessoren	Die Applikation läuft auf vielen Prozessoren	Die Applikation wird waährend der Laufzeit dynamisch auf einem von mehreren Prozessoren geladen und ausgeführt
Leistungsanforderungen durch den Kunden	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Leistungsanforderung bei stark belastetem System	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Geplante Durchsatzrate der Daten	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Online Dateneingabe	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Anforderung an die Benutzungsschnittstelle	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Summe F1	*0	*1	*2	*3	*4	*5

Abbildung 8-1 : Formular Systembeeinflussung Teil 1

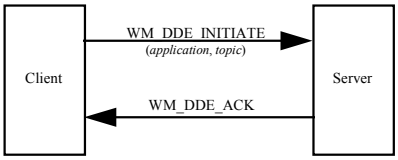
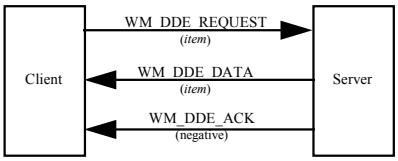
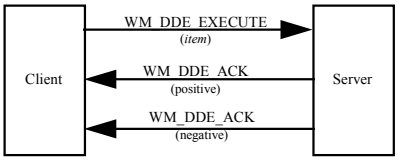
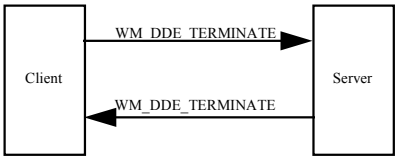
	0	1	2	3	4	5
Forderung bezüglich sofortiger Updates der Daten	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Komplexe interne Berechnungen	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Wiederverwendbarkeit von Systemteilen	X < 10%	10% ≤ X < 20%	20% ≤ X < 30%	30% ≤ X < 40%	40% ≤ X < 50%	50% ≤ X
Anforderung an die Installation des Programmes	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Bereitstellung von Standardfunktionen (z.B. Backup)	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Anzahl der zu unterstützenden Systemplattformen	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Einfache Veränderbarkeit der Daten des Systems (z.B. Kunden definierbare Abfragen)	Nicht vorhanden oder vernachlässigbar klein	Sehr klein	Moderat, aber vorhanden	Durchschnittliche Stärke	Signifikanter Einfluß	Sehr starker Einfluß
Übertrag F1	*0	*1	*2	*3	*4	*5
Gesamtsumme	*0	*1	*2	*3	*4	*5
TDI = VAF = 0,65+(0,01*TDI) = 0,65+(0,01*) =						

Abbildung 8-2 : Formular Systembeeinflussung Teil 2

Das Formular basiert im wesentlichen auf den Angaben von [Dreger 89].

8.3 DDE-Kommunikation

Die DDE-Kommunikation zwischen dem Exportierer und dem Tool wird durch die Verbindungsart „Cold Link“ durchgeführt. Sie Arbeit nach dem von [Petzold 90] beschriebenen Verfahren:

Verbindungsaufbau	 Abbildung 8-3 : [Petzold 90] p. 811	Der Client schickt die Nachricht WM_DDE_INITIATE (*application*, *topic*) zum Server. Dieser bestätigt dann die Verbindung durch die Nachricht WM_DDE_ACK.
Requests	 Abbildung 8-4 : [Petzold 90] p. 812	Der Client fordert Daten vom Server mit der Nachricht WM_DDE_REQUEST (*item*) an. Unterstützt der Server das Item, so werden die entsprechenden Daten mit Hilfe der Nachricht WM_DDE_DATA (*item*) zum Client verschickt. Falls das Item nicht vom Server unterstützt wird, erhält der Client die Quittung WM_DDE_ACK (*negative*).
Befehle	 Abbildung 8-5	Die Übermittlung von Befehlen an den Server läuft analog zu der Anforderung von Daten. In diesem Fall wird aber der Befehl positiv oder negativ quittiert.
Verbindungsabbau	 Abbildung 8-6 : [Petzold 90] p. 812	Der Verbindungsabbau durch den Client wird durch verschicken der Nachricht WM_DDE_TERMINATE eingeleitet und durch zurücksenden von WM_DDE_TERMINATE durch den Server abgeschlossen.

Der Serverprozeß wird in Abbildung 8-7 dargestellt. Sie beschreibt einen einfachen Automaten, der auf die Eingaben durch das DDE-Protokoll reagiert.

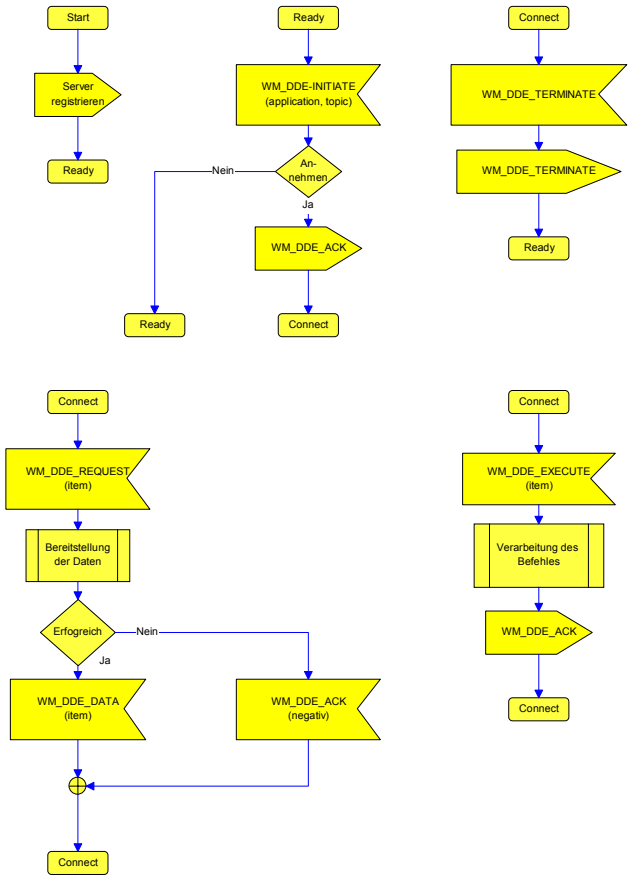


Abbildung 8-7 : Serverprozeß

8.4 Literatur

[Albrecht 83] A.J. Albrecht, J.R. Gaffney
Software function, source lines of code, and development effort prediction: a Software Science validation
IEEE Transactions on Software Engineering, 9(6):639-648, 1983

[Boehm 81] B. W. Boehm
Software Engineering Economics
Prentice-Hall; Englewood Cliffs, NJ, 1981

[Card 90] D. N. Card, Robert L. Glass
Measuring Software Design Quality Metrics
Englewood Cliffs, N. J. Prentice-Hall, 1990

[Conte 86] S.D. Conte, H.E.Dunsmore, V. Y. Shen
Software engineering metrics and models
The Benjamin/Cummings Publishing Company, Inc. 1986

[Dreger 89] J. Brian Dreger
Function Point Analysis
Prentice Hall 1989

[Ebert 96] Christof Ebert, Reiner Dumke
Software-Metriken in der Praxis
Springer 1996

[Elzer 94] Peter Elzer
Management von Softwareprojekten: eine Einführung für Studenten und Praktiker
Vieweg 1994

[Fenton 96] Norman E. Fenton, Shari Lawrence Pfleeger
Software Metrics
A Practical and Rigorous Approach
Second Edition
International Thomson Computer Press, 1996

[Halstead 77] Maurice H. Halstead
Elements of Softwae Science
Elsevier - Noth Holland 1977

[Henry 81] S. M. Henry, D. Kafura
Software Structure Metrics Based on Information Flow
IEEE Transactions on Software Engineering
Vol. SE-7, pp 510-518, 1981

[Henry 90] S. M. Henry, C. Selig
Prediction Source-Code Complexity at the Design Stage
IEEE Software
pp 37-44 März 1990

[Humphrey 95] Watts S. Humphrey
A Discipline for Software Engineering
Addison-Wesley, 1995

[Koch 93a] Jörg Koch, Michael Schusser
EDV-unterstützte Konstanzprüfung in der Filmverarbeitung
Krankenhaus Technik, 19. Jahrgang, Ausgabe März 1993, Seite 50- 52
ecomed - Verlagsgesellschaft

[Koch 93b] Jörg Koch, Michael Schusser
Archivierung, Dokumentation und Bearbeitung von Röntgen-aufnahmen
Krankenhaus Technik, 19. Jahrgang, Ausgabe April 1993, Seite 60- 62
ecomed - Verlagsgesellschaft

[Kreyszig 88] Erwin Kreyszig
Statistische Methoden und ihre Anwendungen
Vandenhoeck & Ruprecht in Göttingen 7. Auflage 1988

[MacCabe 76] T. J. McCabe
A Complexity Measure
IEEE Transactions on Software Engineering
Vol. 2 No. 4, pp 149-157, Dezember 1976

[Möller 93] K. H. Möller, D. J. Paulish
Software-Metriken in der Praxis
Oldenbourg 1993

[Petzold 90] Charles Petzold
Programming Windows:
the Microsoft guide to writing applications for Windows 3
Microsoft Press, 1990

[Shepperd 95] Martin Shepperd
Foundations of software measurment
Prentice Hall, 1995

[Szwillus 94] Prof. Szwillus
Vorlesung Benutzungsschnittstellen
Paderborn WS 93/94

[Troster 92] J. Troster
Assessing Design-Quality Metrics on Legacy Software
Software Engineering Process Group
IBM Canada Ldt. Laboratory
New York Ontario, September 1992

8.5 Abbildungsverzeichnis

8.6 Die beigefügte CD

Die dieser Diplomarbeit beigefügten CD beinhaltet neben dem Tool noch einige zusätzliche Informationen. Sie sind in folgenden Verzeichnissen organisiert:

Verzeichnis	Bedeutung
DOK\HTML	HTML-Version der Diplomarbeit
DOK\PS	Text Diplomarbeit als PS-Datei
ENTWICK\ANALYSE	Generierte Ausgaben des CASE-Tools für die Analysephase
ENTWICK\DESIGN	Generierte Ausgaben des CASE-Tools für die Designphase
PPT4VIEW	Powerpoint 4 Projektor Installation
TOOL	Installationsverzeichnis des Tools (siehe Install.DOC)
VORTRAG	Powerpoint-Folien des Vortrages
WWW	Kopien einiger interessanten WWW-Seiten

Für die Analyse und das Design existieren Versionen als Postscript (PS), HTML (HTML) und als Helpfile (HLP)