



Fakultät für Informatik  
Institut für Verteilte Systeme (IVS)

## **Diplomarbeit**

# **Qualitative Bewertung von Software-Messwerkzeugen zur Aufwandschätzung**

Themensteller: Prof. Dr.-Ing. habil. Reiner Dumke

vorgelegt von: Corinna Benecke  
Matrikelnummer 91418

Betreuer: Prof. Dr.-Ing. habil. Reiner Dumke  
Otto-von-Guericke-Universität Magdeburg  
Postfach 4120  
39106 Magdeburg

# Inhaltsverzeichnis

1	Einführung .....	4
1.1	Hintergrund und Zielstellung .....	4
1.2	Aufbau der Arbeit.....	6
2	Die Aufwandschätzung im Bereich der Software-Messung.....	7
2.1	Das Schätzproblem.....	7
2.2	Grundlagen der Aufwandschätzung .....	8
2.3	Methoden der Aufwandschätzung.....	11
2.3.1	Function-Point-Methode .....	12
2.3.2	COCOMO 81-Methode.....	17
2.3.3	COCOMO II-Methode .....	19
2.3.4	Object-Point-Methode .....	23
2.3.5	Data-Point-Methode .....	26
2.3.6	Error-Projection-Methode .....	29
2.3.7	UseCase-Point-Methode.....	30
2.3.8	Testfallmethode .....	33
2.4	Toolunterstützung bei der Aufwandschätzung.....	35
2.5	Zusammenfassung .....	37
3	Qualitative Anforderungen an Software-Messtools.....	38
3.1	Der Begriff Qualität im Bereich der Softwareentwicklung.....	38
3.2	Qualität der Software-Maße und -Metriken .....	39
3.3	Anforderungen an ein Software-Messwerkzeug .....	41
4	Das Messwerkzeug SoftCalc .....	43
4.1	Historie und grundlegende Arbeitsweise .....	43
4.2	Benötigte Eingabedaten.....	46
4.3	Datenmodell und Architektur des Programms SoftCalc .....	47
4.4	Ablauf einer Aufwandschätzung.....	48
4.5	Das SoftCalc Schätzverfahren.....	50
4.6	Ermittlung des Produktumfangs und des Aufwands in SoftCalc.....	52
4.7	Die Einflussfaktoren in SoftCalc.....	53
4.8	Die SoftCalc Benutzeroberfläche.....	58
4.9	Erzeugte Ausgabedaten .....	60
5	Die qualitative Bewertung des Programms SoftCalc.....	61
5.1	Messbeispiele .....	61
5.2	Auswertung .....	67
5.2.1	Prüfung gegen die Qualitätsanforderungen an Software-Produkte.....	67
5.2.2	Prüfung gegen die Anforderungen an ein Software-Messwerkzeug.....	69
5.2.3	Fazit.....	71
6	Zusammenfassung und Ausblick .....	72

Abbildungsverzeichnis .....	74
Tabellenverzeichnis.....	75
Literaturverzeichnis.....	76
Anhang A .....	78
Anhang B.....	81
Anhang C.....	85
Selbständigkeitserklärung .....	99

# 1 Einführung

## 1.1 Hintergrund und Zielstellung

Unser heutiges Leben ist ohne Informatik undenkbar. Keine andere Wissenschaft hat es in so kurzer Zeit geschafft, einen derart großen Einfluss auf unseren Alltag zu gewinnen. Viele Lebens- und Arbeitsbereiche werden mittlerweile durch Informatiksysteme unterstützt und wir erwarten aufgrund eines gewachsenen Qualitätsbewusstseins, dass die diesen Systemen zugrundeliegende Software bestimmte Funktionsweisen korrekt erfüllt.

Die Anfänge der Entwicklung in der Informationstechnologie waren durch bedeutende Fortschritte im Bereich der Hardware gekennzeichnet. Die mitgelieferte Software wurde nur als Anhängsel betrachtet.

Durch verbesserte Ingenieurtechniken konnten viele Probleme im Bereich der Hardware gelöst werden und es wurden immer leistungsfähigere Maschinen geschaffen. Die Entwicklung der Software konnte damit jedoch nicht Schritt halten.

In den 1960er Jahren wurden die Probleme im Bereich der Software immer gravierender und prägten den Begriff der "Softwarekrise". Fehler häuften sich, viele Projekte scheiterten an mangelhafter Software und das Projektmanagement war eher unterentwickelt. Die Problematik wird sehr treffend durch folgende Aussage E. W. Dijkstras (The Humble Programmer, 1972) beschrieben:

*“The major cause [of the software crisis] is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem. In this sense the electronic industry has not solved a single problem, it has only created them...”*

In dieser Zeit entstand die Forderung nach einem ingenieurmäßigen Ansatz bei der Software-Entwicklung ("Software Engineering"). Ziel war es, einheitliche Normen und Standards zu definieren, Tätigkeiten voneinander abzugrenzen und die Entwicklung als solche zu disziplinieren, um eine bestimmte Qualität der Software-Produkte gewährleisten zu können. Dieser Prozess dauert bis heute an und gewinnt durch die zunehmende Industrialisierung der Informationstechnologie und möglichen Gesetzesänderungen (s. z.B. Produkthaftung) immer mehr an Bedeutung.

Einen Aspekt dieses ingenieurmäßigen Ansatzes - der Nutzung von Prinzipien des Ingenieurwesens - stellt die Verwendung eines Maßsystems dar. Laut [Dumke 01] besteht das Maßsystem beim Software Engineering aus einer Menge von vorhandenen Software-Maßen, -Kennzahlen und -Metriken, die sich auf alle wesentlichen Aspekte der Software-Entwicklung, -Pflege und -Wartung beziehen. Die Anwendung dieses Maßsystems wird als Software-Messung bezeichnet und erfolgt ggf. unter Verwendung von speziellen Tools, die der Ausprägung der jeweiligen Messstrategie, der Aufbereitung der Messobjekte oder der statistischen Auswertung bzw. Darstellung der Messergebnisse dienen [Dumke 01]. Mit der Software-Messung werden Messziele unterschiedlichster Art verfolgt, wie zum Beispiel die Bestimmung der Größe eines Programms oder die Bestimmung der Produktivität verschiedener Entwicklungsteams.

Eine Form der Software-Messung stellt die Merkmalsabschätzung (*estimation*) dar. Sie erfolgt mit Hilfe einer vorgegebenen Formel oder einer allgemeinen Berechnungsvorschrift. Unter den Bereich der Merkmalsabschätzung fallen zum Beispiel die Methoden zur Aufwandschätzung. Die Aufwandschätzung ist eines der grundlegenden Themen des Software Engineering und stellt innerhalb der betrieblichen Organisation einen wesentlichen Bestandteil des Projektmanagements (software project management) dar. Auch im Bereich der Softwareentwicklung, -wartung und -pflege sind Kosten- und Leistungsschätzungen unabdingbar. Ohne derartige Betrachtungen liegen die geplanten Software-Projekte oftmals weit über dem veranschlagten Zeit- und Kostenplan bzw. sind im schlechtesten Fall gänzlich zum Scheitern verurteilt.

Mit dieser Diplomarbeit sollen die speziellen Qualitätsanforderungen an ein Software-Messwerkzeug zur Aufwandschätzung erarbeitet werden und das Programm SoftCalc als repräsentativer Vertreter eines solchen Messtools einer qualitativen Bewertung unterzogen werden. SoftCalc wird im Rahmen der Merkmalsabschätzung eingesetzt. Mit dem Tool sollen automatisierte Aufwandschätzungen von Software-Projekten unterschiedlichster Art möglich sein. Ziel dieser Arbeit ist es, das Programm zu analysieren und zu bewerten, um im Ergebnis festzustellen, ob es die Anforderungen an ein Software-Messwerkzeug im Sinne der Software-Messung, hier der Aufwandschätzung, erfüllt.

## 1.2 Aufbau der Arbeit

Die Diplomarbeit untergliedert sich in folgende Teilbereiche (*Kapitel*):

(2) *Die Aufwandschätzung im Bereich der Software-Messung*

Dieses Kapitel beinhaltet theoretische Ausführungen zum Thema Aufwandschätzung und liefert eine kurze Beschreibung der im Tool SoftCalc verwandten Aufwandschätzmethoden.

(3) *Qualitative Anforderungen an Software-Messtools*

Basierend auf der DIN-Norm 9126 werden die allgemeinen Anforderungen, die an die Softwarequalität gestellt werden, beschrieben. Zudem werden die Qualitätsanforderungen an die Software-Messung und ihre Werkzeuge herausgearbeitet.

(4) *Der SoftCalc-Ansatz*

In diesem Kapitel wird das Software-Messwerkzeug SoftCalc näher untersucht. Zum Einen werden die theoretischen Grundlagen, auf denen das Programm basiert, beschrieben und zum Anderen die Architektur, die Benutzeroberfläche und Handhabung, sowie die Implementierung dargestellt.

(5) *Die qualitative Bewertung von SoftCalc*

In diesem Abschnitt der Arbeit wird die praktische Relevanz des Tools untersucht und das Software-Produkt gegen die qualitativen Anforderungen geprüft und bewertet.

(6) *Zusammenfassung und Ausblick.*

## 2 Die Aufwandschätzung im Bereich der Software-Messung

*Wenn Sie messen können worüber Sie reden und es in Zahlen ausdrücken, dann wissen Sie etwas darüber. Wenn Sie es aber weder messen noch in Zahlen ausdrücken können, ist ihr Wissen dürftig und unbefriedigend. Es mag sein, dass Sie zu begreifen beginnen, aber Sie haben noch lange nicht den Stand erreicht, bei dem wir von Wissenschaft reden können.*

*Lord Kelvin*

### 2.1 Das Schätzproblem

Wie bereits in Kapitel 1.1 dieser Arbeit erläutert wurde, sind Merkmalsabschätzungen, wie zum Beispiel Aufwandschätzungen, unerlässlich, wenn die Softwareentwicklung als Software Engineering, also als Ingenieursdisziplin, und nicht als Ad-hoc-Programmierung wahrgenommen werden möchte. Nur in Ausnahmefällen werden Projekte ohne Endtermin gestartet. Ohne entsprechende Aufwandschätzungen wird die Termineinhaltung zum Zufallsprinzip, denn niemand kann ohne Grundlage beurteilen, welche Ressourcen benötigt werden und ob diese überhaupt vorhanden sind.

Die Aufwandschätzungen, die erstmals in einem sehr frühen Projektstadium erfolgen sollten, liefern Zahlen zu Kosten, Terminen und Dauer eines Software-Projekts. Ermittelt werden diese Größen aus dem Aufwand, bei dem es sich aus betriebswirtschaftlicher Sicht im allgemeinen um den Werteverbrauch an Gütern und Dienstleistungen in einer bestimmten Verbrauchsperiode handelt. Aus Sicht der Informatik wird mit dem Begriff Aufwand der Verbrauch an Ressourcen (Personal, Hard- und Software) zur Lösung eines Problems beschrieben.

Der Bereich der Aufwandschätzungen bildet einen wesentlichen Bestandteil des Projektmanagements. Nach [Dumke 01] beinhaltet das Projektmanagement (software project management) die planenden, kontrollierenden und steuernden Aktivitäten für die termingerechte Bereitstellung der Ressourcen und der kostengerechten Realisierung eines Software-Produkts.

Doch worauf sollen die Aufwandschätzungen basieren?

Software stellt ein immaterielles, abstraktes und einzigartiges Produkt dar. Sie ist Ergebnis geistiger Arbeit und lässt sich aufgrund unserer Individualität schwer auf einen gemeinsamen Nenner bringen. Die Produktivität und Qualität der Arbeit differiert nicht nur zwischen einzelnen Entwicklern, sondern kann auch von persönlichen Umständen beeinflusst werden. Doch nicht nur der "menschliche Faktor" spielt hier eine bedeutende Rolle. Ebenso müssen die Faktoren, die z.B. im Bereich der Software, Hardware oder in der betrieblichen Organisation eine Rolle spielen, in die Betrachtungen einbezogen werden. Demzufolge müssen abhängig von dem zu entwickelnden Software-Produkt die verschiedenen empirischen Relative, die einen Einfluss auf den jeweiligen Projekterfolg haben, analysiert werden, um zuverlässige numerische Relative wie z.B. die Projektdauer in Personenmonaten ermitteln zu können.

Es ist unschwer zu erkennen, dass sich die Entwicklung von Software wesentlich von der Entwicklung in der traditionellen Fertigung unterscheidet. Zudem unterliegt sie ständigen Änderungen, z.B. durch die Entwicklung neuer Technologien im Hard- und Softwarebereich.

## **2.2 Grundlagen der Aufwandschätzung**

Ziel der Aufwandschätzung ist die Vorhersage von Aufwand in Personenmonaten bzw. Personentagen, um daraus Kosten, Dauer, Termine und Personalbedarf abzuleiten. Die Aufwandschätzung ist Teil der Merkmalsabschätzung, die eine mögliche Messmethode der Software-Messung darstellt. Einsatzgebiet ist nicht nur die Entwicklung, sondern auch die Pflege oder die Wartung eines Software-Produkts. Das Software-Produkt stellt in diesem Zusammenhang die Gesamtheit aller Softwarekomponenten, die als Ganzes entwickelt, vertrieben, angewendet und gewartet werden, dar [Dumke 01]. Komponenten eines Software-Produkts können z.B. der Quellcode, das Benutzerhandbuch oder der Pflege- und Wartungsplan sein.

Da Software-Produkte meist sehr heterogen sind, entstanden viele Aufwandschätzmethoden innerhalb eines engen und spezifischen Umfeldes. Eine allgemeingültige Formel zur Aufwandschätzung lässt sich aus ihnen nicht ableiten, da sie auf ihren individuellen Gegebenheiten basieren. Allein die Ansiedlung der Software-Projekte in verschiedenen Umfeldern, wie

z.B. Integration, Neuentwicklung, Migration oder Reengineering, machen eine Vereinheitlichung des Problems Aufwandschätzung schwierig und im Augenblick nicht lösbar.

Formal basieren Schätzungen auf

$$\text{Schätzwert } P = f(M, \Sigma),$$

wobei  $M = \{\mu_1, \mu_2, \dots, \mu_n\}$  als Menge der in die Schätzung eingehenden gegebenen bzw. gemessenen Größen und  $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$  als Menge der eingehenden geschätzten Größen, wobei auch mehrere Merkmale gleichzeitig vorhergesagt werden können, d.h. es gilt  $P = \{\pi_1, \pi_2, \dots, \pi_k\}$  [Dumke 01].

Der Begriff Aufwand als Kennzahl der Schätzung wird im allgemeinen mit

$$\text{Aufwand} = \frac{\text{Produktgröße}}{\text{Produktivität}}$$

definiert.

Demzufolge ist im ersten Schritt einer Aufwandschätzung die Software-Produktgröße, d.h. der Umfang des Schätzobjekts, zu ermitteln. Diese Größe stellt die Grundlage für alle weiteren Vorgehensweisen dar. In der Praxis existieren zwei grundlegende Herangehensweisen, um die Größe bzw. den Umfang der zu entwickelnden Software zu ermitteln: Entweder anhand der funktionalen Vorgaben (*funktionales Software-Maß*) oder anhand des Programm-Umfangs (*technisches Software-Maß*). Software-Maße stellen gemäß der Messtheorie eine mit einer Maßeinheit versehene Skala dar und beziehen sich auf die Attribute von einem oder mehreren Objekten oder Komponenten der Software-Entwicklung, -Wartung oder -Anwendung, wie zum Beispiel die Größe eines Programms [Dumke 01]. Beispiele für Software-Maße, auf denen eine Aufwandschätzung beruhen kann, sind Lines of Code, Object-Points, Data-Points oder Function-Points. Sie werden in den folgenden Kapiteln detailliert beschrieben.

Im zweiten Schritt einer Aufwandschätzung erfolgt die Schätzung des konkreten Anteils einer Menge von Einflussfaktoren, die ebenfalls Software-Maße darstellen. Die verschiedenen Aufwandschätzmethoden berücksichtigen unterschiedliche Einflussfaktoren. Eine nähere Erläuterung erfolgt in Kapitel 4.

In diesem Teil der Aufwandschätzung wird versucht eine Beziehung zwischen der gemessenen Produktgröße und den bei der Realisierung auftretenden Einflussfaktoren herzustellen.

Zur Verdeutlichung des Zusammenhangs zwischen Produktgröße (Ergebnis aus Schritt 1) und Einflussfaktoren findet sich nachfolgend eine vereinfachte Darstellung der verschiedenen Niveaustufen einer Schätzung aus [Dumke 01]:

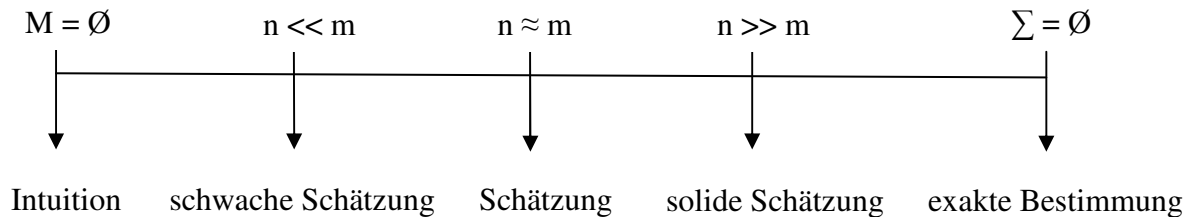


Abb. 1: Niveaustufen einer Schätzung

Die Stufe der exakten Bestimmung ist nur theoretisch möglich, da es keine Softwareerstellung ohne Einflussfaktoren gibt, so lange der menschliche Faktor eine Rolle spielt (s. Kapitel 2.1).

Letztendlich werden aus dem anhand der Schätzformel ermittelten Aufwand die Merkmale Dauer, Kosten, Termine und Personalbedarf abgeleitet.

Die Ergebnisse der Aufwandschätzung werden auf die verschiedenen Projektphasen verteilt und bilden die Grundlage für den Projektgesamtplan. Diese Verteilung erfolgt prozentual auf die verschiedenen Projektaktivitäten. Bei konventionellen Projekten verteilt sich der Aufwand im Allgemeinen wie folgt [vgl. KunDum 07]:

<i>Projektaktivität</i>	<i>prozentualer Kostenanteil</i>
Management	5 %
Anforderungen	5 %
Design	10 %
Codierung und Test	30 %
Integration und Test	40 %
Bereitstellung	5 %
Umgebung	5 %

Tab. 1: Prozentuale Kostenverteilung in konventionellen Software-Projekten

An dieser Stelle ist darauf hinzuweisen, dass eine Aufwandschätzung während des Projektverlaufs kein einmaliger Vorgang bleiben darf. Vielmehr müssen die Schätzgrößen während des

Projektverlaufs verfeinert und die Aufwandschätzung kontinuierlich fortgeführt werden, denn eine Schätzung widerspiegelt immer nur die Software-Charakteristika zu einem bestimmten Zeitpunkt. Die erstmalige Aufwandschätzung erfolgt in einem frühen Projektstadium, in dem noch nicht alle Faktoren bekannt sind. Die Ergebnisse werden jedoch durch fortlaufende Schätzungen verbessert und gewinnen dadurch an Genauigkeit. Damit ist ein Abgleich zwischen Ist- und Soll-Zustand des Projekts möglich und der Projektfortschritt wird sichtbar und kann im Rahmen des Projektcontrollings überwacht und gesteuert werden.

### **2.3 Methoden der Aufwandschätzung**

Innerhalb der Aufwandschätzung existieren verschiedene Vorgehensweisen, um zu einem Schätzergebnis zu gelangen. Grundsätzlich wird zwischen empirischen und algorithmischen Schätzansätzen unterschieden.

Die Klassifikation einer Methode erfolgt im allgemeinen nach folgenden Basismethoden der Aufwandschätzung:

- *Analogiemethode*

Um den Aufwand für das zu entwickelnde Software-Produkt zu schätzen, werden bei der Analogiemethode bereits abgeschlossene und vergleichbare Projekte herangezogen. Die Schätzung beruht im Ergebnis auf den individuellen Erfahrungen des Schätzers und ist für andere Beteiligte kaum nachvollziehbar.

- *Relationsmethode*

Die Relationsmethode basiert auf dem gleichen Ansatz wie die Analogiemethode. Allerdings erfolgt der Vergleich mit bereits abgeschlossenen Projekten über formal herausgearbeitete Indizes.

- *Gewichtungsmethode*

Bei der Gewichtungsmethode werden objektive und subjektive Einflussfaktoren mit einem bestimmten Gewicht versehen, je nachdem welchen Einfluss der Faktor auf den Aufwand hat. Der Gesamtaufwand wird mit einer vorgegebenen mathematischen Formel errechnet und ergibt sich aus der Summe der Einzelbewertungen.

- *Multiplikatormethode*  
Bei der Multiplikatormethode erfolgt die Schätzung über eine Multiplikation der geschätzten Anzahl an Projekteinheiten mit dem vorher festgelegten Aufwand pro Einheit.
- *Prozentsatzmethode*  
Bei der Prozentsatzmethode schätzt man den Gesamtaufwand nach Abschluss der ersten Phase oder nach dem tatsächlichen Aufwand einer beliebigen Phase und schließt daraus auf den Gesamtaufwand. Dabei wird eine prozentuale Verteilung des Aufwands auf die einzelnen Phasen des Projektverlaufs aus bereits abgeschlossenen Projekten zugrunde gelegt.
- *Methode der parametrischen Schätzgleichungen*  
Durch Korrelationsanalysen vergangener Projekte werden die Faktoren, die einen wertmäßigen Einfluss auf den Gesamtaufwand haben, ermittelt. Im Ergebnis werden die Einflussfaktoren, die die höchste Korrelation besitzen, zu einer Schätzgleichung zusammengefasst.

Momentan existieren zu jeder Basismethode verschiedene Ausprägungen. In das Messwerkzeug SoftCalc, das mit dieser Arbeit betrachtet werden soll, fließen acht verschiedene dieser Ausprägungen - sechs zu der *Gewichtungsmethode* und zwei zur Methode der *parametrischen Schätzgleichungen* - ein. Diese werden in den folgenden Kapiteln kurz beschrieben. Eine weitergehende Betrachtung existierender Methoden wäre in diesem Kontext nicht zielführend.

### **2.3.1 Function-Point-Methode**

Die Function-Point-Methode ist eine Methode zur Messung der Größe bzw. des Umfangs einer Anwendung aus Benutzersicht und wurde 1979 von A. J. Albrecht bei IBM entwickelt. Sie wird seit 1981 bei der IBM Deutschland und einigen großen deutschen IT-Entwicklern eingesetzt. 1986 wurde die International Function-Point Users Group (IFPUG) zum Zweck einer internationalen Standardisierung dieser Methode gegründet (<http://www.ifpug.org/>). Die Methode zählt zu den Ausprägungen der *Gewichtungsmethode* und findet ihre Anwendung sowohl bei der Entwicklung neuer Systeme, als auch bei der Weiterentwicklung bzw. Erweiterung bereits vorhandener Software-Produkte.

Im Gegensatz zu dem technischen Software-Maß Lines of Code (LOC), das eine Maßeinheit für den Codeumfang darstellt, soll mit dem Software-Maß Function-Points die Funktionalität der Software abgebildet werden. Aus diesem Grund erfolgt die Betrachtung aus Benutzersicht. Im Allgemeinen erfolgt eine Ermittlung der gewichteten Function-Points nach folgendem Muster [BuFab 00]:

**Schritt 1:** Zähltyp festlegen

Unterschieden werden die drei Zähltypen Neuentwicklung, Weiterentwicklung und Anwendungssystem.

**Schritt 2:** Umfang der Zählung und Systemgrenze festlegen

Um eine Zählung durchführen zu können, muss durch Festlegung der Systemgrenze geklärt werden, welche Funktionalität dem eigenen Produkt und welche externen Anwendungen zugerechnet wird:

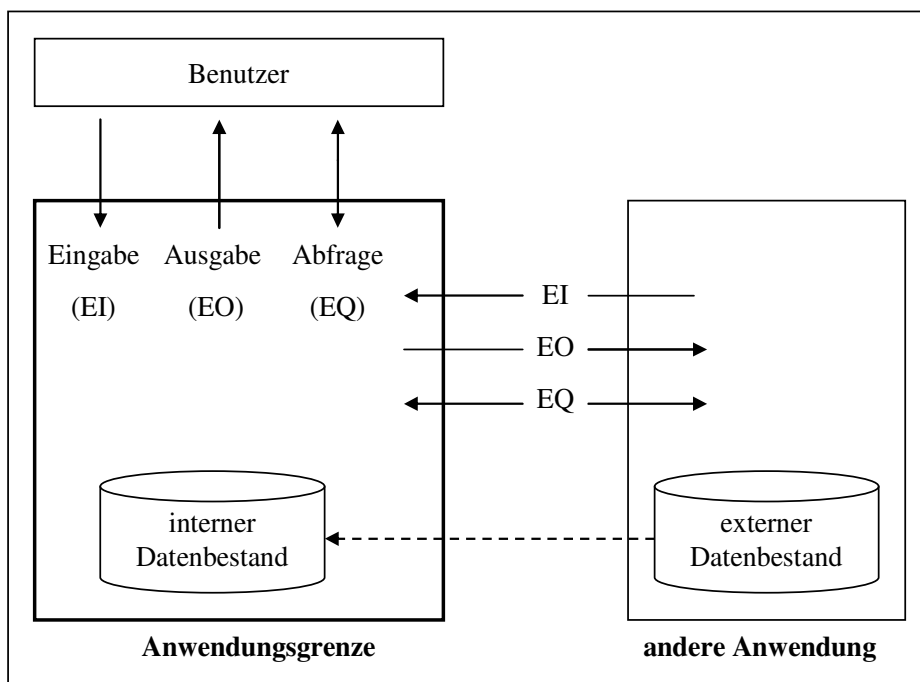


Abb. 2: Festlegen der Systemgrenze

**Schritt 3:** Zählung der ungewichteten Function-Points nach IFPUG 4.1

Bei der Zählung der ungewichteten Function-Points werden anhand der Datenbestände fünf Funktionstypen unterschieden und aufgrund ihrer Komplexität klassifiziert und gewichtet:

Funktionstyp		Komplexität	Gewicht
<i>Daten- Function- Points</i>	<b>ILF</b> (Internal Logical Files) Datenbestände, die innerhalb der Systemgrenzen der Anwendung gepflegt werden	gering	7
		mittel	10
		hoch	15
	<b>EIF</b> (External Interface Files) Datenbestände, die außerhalb der Systemgrenzen (von anderen Anwendungen) gepflegt werden	gering	5
		mittel	7
		hoch	10
<i>Transaktions- Function- Points</i>	<b>EI</b> (External Inputs) externe Eingabedaten	gering	3
		mittel	4
		hoch	6
	<b>EO</b> (External Output) externe Ausgabedaten	gering	4
		mittel	5
		hoch	7
	<b>EQ</b> (External Inquiries) externe Abfragen	gering	3
		mittel	4
		hoch	6

Tab. 2: Funktionstypen und ihre Einordnung nach IFPUG 4.1

Das Ergebnis dieses Schrittes stellt den Umfang des Software-Produkts ohne Berücksichtigung der spezifischen Einflussfaktoren dar.

**Schritt 4:** Wert der Einflussfaktoren bestimmen

Zur Ermittlung der gewichteten Function-Points werden nach IFPUG 4.1 die vierzehn Einflussfaktoren berücksichtigt:

- 1.) Datenkommunikation (*Data Communications*)
- 2.) verteilte Verarbeitung (*Distributed Data Processing*)
- 3.) Leistungsfähigkeit (*Performance*)
- 4.) begrenzte Kapazität (*Heavily Used Configuration*)
- 5.) Transaktionsrate (*Transaction Rate*)
- 6.) Online-Dateneingabe (*Online Data Entry*)
- 7.) Benutzerfreundlichkeit (*End-User Efficiency*)
- 8.) Online-Update (*Online Update*)

- 9.) komplexe Verarbeitung (*Complex Processing*)
- 10.) Wiederverwendbarkeit (*Reusability*)
- 11.) Installationsanforderungen (*Installation Ease*)
- 12.) Betriebsanforderungen (*Operational Ease*)
- 13.) Mehrfachinstallation (*Multiple Sites*)
- 14.) Änderungsfreundlichkeit (*Facilitate Change*)

Jeder Einflussfaktor hat einen Wertebereich von Null bis Fünf:

- 0 Einfluss nicht vorhanden oder vernachlässigbar klein,
- 1 Einfluss sehr klein,
- 2 Einfluss moderat aber vorhanden,
- 3 Einfluss in durchschnittlicher Stärke,
- 4 signifikanter Einfluss und
- 5 sehr starker Einfluss.

Die Summe der Werte aller Faktoren bildet den Gesamteinflussfaktor TDI (*Total Degree of Influence*), der die allgemeine Funktionalität der Anwendung darstellt. Um die gewichteten Function-Points ermitteln zu können, muss aus dem Gesamteinflussfaktor TDI der Wertkorrekturfaktor VAF (*Value Adjustment Factor*) mit

$$VAF = (TDI * 0.01) + 0.65$$

berechnet werden.

Hierbei ist anzumerken, dass die Anwendung des VAF in der Praxis nicht unumstritten ist, da es sich bei der Summe der ungewichteten Function-Points um ein funktionales Maß handelt und der VAF hingegen ein technisches Maß darstellt. Dadurch werden spezifische Maße einander gleichgesetzt und nicht aufgrund ihrer Unterschiede klar voneinander abgegrenzt. Aus meiner Sicht sollten die vierzehn Einflussfaktoren jedoch nicht unberücksichtigt bleiben, da die ungewichteten Function-Points andernfalls keine ausreichende Planungsgrundlage bilden würden. Meine eigenen Praxiserfahrungen haben gezeigt, dass die Realisierung der technischen Anforderungen einen nicht unerheblichen Teil der Gesamtentwicklungszeit ausmachen.

**Schritt 5:** Ermittlung der gewichteten Function-Points

Allgemein stellen die gewichteten Function-Points das Produkt aus ungewichteten Function-Points (Schritt 3) und dem Wertkorrekturfaktor VAF (Schritt 4) dar. Eine verfeinerte Berechnung erfolgt entsprechend Tabelle 3 nach Zähltypen:

Zähltyp	Berechnungsvorschrift
Neuentwicklungsprojekt	$\mathbf{DFP = (UFP + CFP) * VAF}$ <p>DFP: FP eines Neuentwicklungsprojekts            UFP: ungewichtete FP            CFP: FP aus Bestandsübernahmen            VAF: Einflussfaktor</p>
Weiterentwicklungsprojekt	$\mathbf{EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB)}$ <p>EFP: FP eines Weiterentwicklungsprojekts            ADD: hinzugefügte FP            CHGA: ungewichtete FP, die durch Änderung an bestehender Funktionalität entstehen            CFP: FP aus Bestandsübernahmen            VAFA: Einflussfaktor nach der Erweiterung            DEL: ungewichtete FP, die durch Löschung von bestehender Funktionalität entstehen            VAFB: Einflussfaktor vor der Erweiterung</p>
Anwendungssystem	<p><i>Initialisierung durch Neuentwicklungsprojekt:</i></p> $\mathbf{AFP = ADD * VAF}$ <p><i>Aktualisierung durch Weiterentwicklungsprojekt:</i></p> $\mathbf{AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] * VAFA}$ <p>AFP: FP nach Initialisierung            ADD: ungewichtete FP, die neu hinzugefügt werden            UFPB: ungewichtete FP vor Erweiterung            CHGA: ungewichtete FP, die durch Änderung an bestehender Funktionalität entstehen (Betrachtung nach Änderung)            CHGB: ungewichtete FP, die durch Änderung an bestehender Funktionalität entstehen (Betrachtung vor Änderung)            DEL: ungewichtete FP, die durch Löschung bestehender Funktionalität entstehen            VAF: Einflussfaktor der Anwendung            VAFA: Einflussfaktor nach der Erweiterung</p>

Tab. 3: Ermittlung der gewichteten Function-Points nach Zähltypen

Zur Berechnung des Aufwands wird die Summe der gewichteten Function-Points mittels Erfahrungskurve interpoliert. A. J. Albrecht legte hierbei die IBM-Erfahrungskurve zugrunde. Es muss innerhalb des jeweiligen betrieblichen Umfelds über Zählung abgeschlossener Pro-

jekte und Regressionsanalyse eine eigene Erfahrungskurve erstellt werden, um die Schätzgenauigkeit zu erhöhen und auf die unternehmenseigene Umgebung abzustimmen. Andernfalls können die Ergebnisse der Function-Point-Methode zu stark von der Wirklichkeit abweichen und sind keine seriöse Planungsgrundlage.

### 2.3.2 COCOMO 81-Methode<sup>1</sup>

Das algorithmische Kostenschätzmodell COCOMO (*CO*nstructive *CO*st *MO*del) wurde erstmals 1981 von Dr. Harry W. Boehm vorgestellt. Das Modell fällt unter die Kategorie der *parametrischen Schätzgleichungen*. Die Aufwandschätzung erfolgt unter Berücksichtigung der geschätzten Source Lines of Code und firmenspezifischer Parameter, die aus 63 Projekten ermittelt wurden. COCOMO 81 unterscheidet drei Modelle: Organic, Semi-detached und Embedded. Die Verwendung des jeweiligen Modells, hängt von der Komplexität des Projektes ab. Durch die Berücksichtigung von insgesamt 15 Kosten- bzw. Einflussfaktoren (*cost drivers*) in den Schätzformeln des Modells können die Berechnungen an die spezifischen Bedingungen eines Unternehmens angepasst werden. Zudem können durch die Kalibrierung einzelner Kostenfaktoren verschiedene Szenarien durchgespielt werden, wodurch der Einfluss der einzelnen Faktoren auf den möglichen Projektverlauf erkennbar wird. Die Einflussfaktoren werden anhand eines Fragenkatalogs empirisch auf einer Skala mit *very low* (VL), *low* (L), *nominal* (N), *high* (H), *very high* (VH) oder *extra high* (XH) bewertet und einem bestimmten Zahlenwert zugeordnet.

Für alle drei Modelle erfolgt die Berechnung des Aufwands unter Berücksichtigung der Einflussfaktoren mit der Formel:

$$\text{Aufwand (PM)} = a * (\text{KLOC})^b * c_i$$

PM	=	Personenmonate
a, b	=	Konstanten abhängig vom Modell:
		Organic            a = 3.2, b = 1.05
		Semi-detached a = 3.0, b = 1.12
		Embedded        a = 2.8, b = 1.20
KLOC	=	Kilo Lines of Code (geschätzte Produktgröße)
c <sub>i</sub>	=	Kostenfaktoren (mit i = 1, ..... ,15)

---

<sup>1</sup> Harry M. Sneed verwendet für COCOMO 81 den Begriff COCOMO I.

Die benötigte Entwicklungszeit in Monaten resultiert aus:

$$TIME_{dev} = 2.5 * (Aufwand)^d$$

d = modellabhängige Konstante mit:

0.38 bei Organic

0.35 bei Semi-detached

0.32 bei Embedded

In dem Modell COCOMO 81 werden folgende Einflussfaktoren (*cost drivers*) berücksichtigt:

<i>Produkt-attribute</i>	RELY	Required Reliability	geforderte Zuverlässigkeit
	DATA	Database Size	Größe der Datenbasis
	CPLX	Product Complexity	Komplexität des Produkts
<i>Rechner-eigenschaften</i>	TIME	Execution Time Constraint	benötigte Rechenzeit
	STOR	Main Storage Constraint	benötigter Speicherplatz
	VIRT	Virtual Machine Volatility	Systemänderungen
	TURN	Computer Turnaround Time	Rechnerwechsel
<i>Personal-attribute</i>	ACAP	Analyst Capability	Fähigkeit zur Analyse
	LEXP	Programming Language Experience	Erfahrungen mit der Programmiersprache
	PCAP	Programmer Capability	Programmierfähigkeit
	AEXP	Applications Experience	Anwendungserfahrungen
	VEXP	Virtual Machine Experience	Erfahrung mit dem System
<i>Projekt-attribute</i>	TOOL	Use of Software Tools	Anwendung von SW-Tools
	MODP	Modern Programming Practices	moderne Programmierpraktiken
	SCED	Required Development Schedule	erforderliche Entwicklungszeit

Tab. 4: Kostenfaktoren des Modells COCOMO 81

Sie fließen mit folgenden Faktoren, die aus der empirischen Sammlung von Datenpunkten ermittelt wurden, in die Schätzgleichung ein:

Driver	Gewicht des Einflussfaktors (Driver)					
	<i>very low</i>	<i>low</i>	<i>nominal</i>	<i>high</i>	<i>very high</i>	<i>extra high</i>
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	

PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SCED	1.23	1.08	1.00	1.04	1.10	

Tab. 5: Kalibrierung der Einflussfaktoren des Modells COCOMO 81

### 2.3.3 COCOMO II-Methode

Beim COCOMO II handelt es sich um eine Weiterentwicklung des Kostenschätzmodells COCOMO 81. Um mit der laufenden Entwicklung in der Informationstechnologie Schritt halten zu können, wurde 1995 eine Überarbeitung des ursprünglichen Modells vorgestellt. Die bisherigen 15 Einflussfaktoren (*cost drivers*) wurden auf 17 erweitert. Zudem kamen weitere 5 Faktoren (*scale factors*) hinzu. Die Kalibrierung dieser Faktoren beruht derzeit auf 161 Datenpunkten, die aus abgeschlossenen Software-Projekten ermittelt wurden.

Das Schätzmodell COCOMO II orientiert sich stärker an den Anforderungen moderner Projekte. Die Schätzung basiert zwar weiterhin auf der Codegröße in Source Lines of Code, ihr können jedoch entweder die Quellcodezeilen aus bestehenden Projekten oder die Anzahl der nach IFPUG 4.1 ermittelten ungewichteten Function-Points zugrunde liegen. Zudem widmet sich COCOMO II der Problematik von wiederverwendetem Code. Darin liegen die Vorteile der Methode, denn es sind Schätzungen in den Bereichen Neuentwicklung, Wartung und Reengineering möglich.

COCOMO II unterscheidet insgesamt drei Modelle für unterschiedliche Projektphasen: Application Composition, Early Design und Post Architecture. Im Folgenden wird nur auf das Modell Post Architecture näher eingegangen, da in dieses Modell alle COCOMO II-Einflussfaktoren einfließen.

Das Modell Post Architecture wird nach der Entwicklung der gesamten Projektarchitektur verwendet. Da zu diesem Zeitpunkt genauere Informationen zum Projekt zur Verfügung stehen, erfolgt die Schätzung unter Berücksichtigung von insgesamt 17 Kostenfaktoren neben den 5 allgemeinen Skalierungsfaktoren. Grundlage bilden die aus den ungewichteten Function-Points oder historischen Quelledaten ermittelten Source Lines of Code. Die einzelnen

Einflussfaktoren werden mit einem Fragenkatalog empirisch bewertet (s. a. Kapitel 2.3.2) und einem Zahlenwert aus Tabelle 7 (Skalierungsfaktoren) oder 8 (Kostenfaktoren) zugeordnet, um an die Gegebenheiten des jeweiligen Projekts angepasst werden zu können. Dafür folgendes Beispiel:

Dem Produktattribut RELY (Required Software Reliability/geforderte Zuverlässigkeit des Produkts) wird bspw. anhand nachstehender Frage ein äquivalenter Zahlenwert zugeordnet:

Welche der aufgeführten Auswirkungen kann ein Softwarefehler haben?

geringfügige Fehler	gering, leicht behebbar	moderate	hoher finanzieller Verlust	Risiko für Menschenleben	-
very low	low	nominal	high	very high	extra high
= 0.82	= 0.92	= 1.00	= 1.10	= 1.26	n/a

Tab. 6: Beispiel einer Kalibrierung eines Produktattributs

Ähnlich dem vorstehenden Beispiel werden die nachstehenden Skalierungs- und Kostenfaktoren bewertet:

<i>Skalierungsfaktoren</i>	PREC	Precedentedness	Erstentwicklungsform
	FLEX	Development Flexibility	Entwicklungsdynamik
	RESL	Architecture/Risk Resolution	Risikomanagement
	TEAM	Team Cohesion	Teamstabilität
	PMAT	Process Maturity	Prozessgüte nach dem CMM
<i>Produktattribute</i>	RELY	Required Software Reliability	geforderte Zuverlässigkeit
	DATA	Database Size	Größe der Datenbasis
	CPLX	Product Complexity	Komplexität des Produkts
	RUSE	Developed for Reusability	Entwicklung für die Wiederverwendung
	DOCU	Documentation Match to Life-Cycle Needs	entwicklungsbegleitende Dokumentation
<i>Rechner-eigenschaften</i>	TIME	Execution Time Constraint	benötigte Rechenzeit
	STOR	Main Storage Constraint	benötigter Speicherplatz
	PVOL	Platform Volatility	Platfordmdynamik
<i>Personalattribute</i>	ACAP	Analyst Capability	Fähigkeit zur Analyse
	PCAP	Programmer Capability	Programmierkenntnisse
	PCON	Personnel Continuity	Personalstabilität
	APEX	Applications Experience	Anwendungserfahrungen
	PLEX	Platform Experience	Plattdormerfahrungen
	LTEX	Language and Tool Experience	Erfahrungen mit der Programmiersprache
<i>Projektattribute</i>	TOOL	Use of Software Tools	Anwendung von CASE-Tools
	SITE	Multisite Development	Entwicklungsteamverteilung
	SCED	Required Development Schedule	erforderliche Entwicklungszeit

Tab. 7: Skalierungs- und Kostenfaktoren des Modells COCOMO II

Dabei fließen die Einflussfaktoren des Modells Post Architecture nach COCOMO II.2000 jeweils mit folgenden Zahlenwerten in die Schätzung ein:

Driver	Symbol	Gewicht des Einflussfaktors (Driver)					
		<i>very low</i>	<i>low</i>	<i>nominal</i>	<i>high</i>	<i>very high</i>	<i>extra high</i>
PREC	SF <sub>1</sub>	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	SF <sub>2</sub>	5.07	4.05	3.04	2.03	1.01	0.00
RESL	SF <sub>3</sub>	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	SF <sub>4</sub>	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SF <sub>5</sub>	7.80	6.24	4.68	3.12	1.56	0.00

Tab. 8: Kalibrierung der Skalierungsfaktoren SF<sub>1-5</sub> mit COCOMO II.2000

Driver	Symbol	Gewicht des Einflussfaktors (Driver)					
		<i>very low</i>	<i>low</i>	<i>nominal</i>	<i>high</i>	<i>very high</i>	<i>extra high</i>
RELY	EM <sub>1</sub>	0.82	0.92	1.00	1.10	1.26	
DATA	EM <sub>2</sub>		0.90	1.00	1.14	1.28	
CPLX	EM <sub>3</sub>	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	EM <sub>4</sub>		0.95	1.00	1.07	1.15	1.24
DOCU	EM <sub>5</sub>	0.81	0.91	1.00	1.11	1.23	
TIME	EM <sub>6</sub>			1.00	1.11	1.29	1.63
STOR	EM <sub>7</sub>			1.00	1.05	1.17	1.46
PVOL	EM <sub>8</sub>		0.87	1.00	1.15	1.30	
ACAP	EM <sub>9</sub>	1.42	1.19	1.00	0.85	0.71	
PCAP	EM <sub>10</sub>	1.34	1.15	1.00	0.88	0.76	
PCON	EM <sub>11</sub>	1.29	1.12	1.00	0.90	0.81	
APEX	EM <sub>12</sub>	1.22	1.10	1.00	0.88	0.81	
PLEX	EM <sub>13</sub>	1.19	1.09	1.00	0.91	0.85	
LTEX	EM <sub>14</sub>	1.20	1.09	1.00	0.91	0.84	
TOOL	EM <sub>15</sub>	1.17	1.09	1.00	0.90	0.78	
SITE	EM <sub>16</sub>	1.22	1.09	1.00	0.93	0.86	0.80
SCED	EM <sub>17</sub>	1.43	1.14	1.00	1.00	1.00	

Tab. 9: Kalibrierung der Kostenfaktoren EM<sub>1-17</sub> mit COCOMO II.2000

Die kalibrierten Faktoren werden in folgende Modellgleichungen eingesetzt:

### COCOMO II – Aufwandschätzung

$$PM = A \cdot \text{Size}^E \times \prod_{i=1}^{17} EM_i + PM_{\text{Auto}}$$

wobei:

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

$$PM_{\text{Auto}} = \frac{\text{Adapted SLOC} \times \left(\frac{AT}{100}\right)}{ATPROD}$$

Symbol	Beschreibung
A	kalibrierbarer Aufwandskoeffizient, momentan bei 2.94
AT	Prozentsatz der angepassten SLOC, die durch Reengineering automatisch "erschlossen" worden sind
ATPROD	Produktivität der automatischen Erschließung bzw. Übersetzung
B	kalibrierbarer Skalierungsexponent für den Aufwand, momentan bei 0.91
E	Aufwandsexponent
EM	Aufwandsmultiplikator: 7 Early Design / 17 Post Architecture
PM	Aufwand in Personenmonaten für den neuen und angepassten Code
PM <sub>AUTO</sub>	Aufwand in Personenmonaten für die automatische Codeerschließung
SF	5 Skalierungsfaktoren
SLOC	Quellcodezeilenanzahl

Tab. 10: Symbolbeschreibung COCOMO II-Aufwandschätzung

### COCOMO II – Entwicklungszeitschätzung

$$TDEV = \left[ C \times (PM_{NS})^F \right] \times \frac{SCED\%}{100}$$

wobei:

$$F = (D + 0.2 \times [E - B])$$

Symbol	Beschreibung
B	Basisexponent für die Skalierung, momentan bei 0.91
C	kalibrierbarer Entwicklungszeitkoeffizient, momentan bei 3.67
D	kalibrierbarer Entwicklungszeitexponent, momentan bei 0.28
E	Skalierungsexponent für die Aufwandsgleichung
F	Skalierungsexponent für die Entwicklungszeit
PM <sub>NS</sub>	Personenmonate geschätzt ohne SCED-Kostentreiber und ohne PM <sub>Auto</sub>
SCED%	Prozentsatz an Entwicklungszeitverkürzung im Verhältnis zur "normalen" Entwicklungszeit
TDEV	Entwicklungszeit in Kalendermonaten

Tab. 11: Symbolbeschreibung COCOMO II-Entwicklungszeitschätzung

## COCOMO II – Umfangschätzung

$$\text{Size} = \left(1 + \frac{\text{REVL}}{100}\right) \times (\text{New KSLOC} + \text{Equivalent KSLOC})$$

wobei:

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC} \times \left(1 - \frac{\text{AT}}{100}\right) \times \text{AAM}$$

$$\text{AAM} = \begin{cases} \frac{\text{AA} + \text{AAF} \times (1 + [0.02 \times \text{SU} \times \text{UNFM}])}{100}, & \text{for } \text{AAF} \leq 50 \\ \frac{\text{AA} + \text{AAF} + (\text{SU} \times \text{UNFM})}{100}, & \text{for } \text{AAF} > 50 \end{cases}$$

$$\text{AAF} = (0.4 \times \text{DM}) + (0.3 \times \text{CM}) + (0.3 \times \text{IM})$$

Symbol	Beschreibung
AA	Prozentsatz des Assessments und der Assimilation
AAF	Anpassungsjustierungsfaktor
AAM	Anpassungsjustierungsmodifikator
AT	Prozentsatz der angepassten KSLOC aus dem Reengineering
CM	prozentualer Anteil modifizierten Codes
DM	prozentualer Anteil modifizierten Designs
IM	Prozentsatz der integrierten (angepassten) Software
KSLOC	Kilo Lines of Code
REVL	Prozentsatz von Anforderungserweiterungen
SU	Prozentsatz des Softwareverständnisses
UNFM	Programmiererunvertrautheit mit der Software

Tab. 12: Symbolbeschreibung COCOMO II-Umfangschätzung

### 2.3.4 Object-Point-Methode

Die Object-Point-Methode wurde erstmals in den 1990er Jahren vorgestellt. Eine mögliche Variante wurde von Harry M. Sneed entwickelt. Die Methode ist eine Abwandlung der Function-Point-Methode, mittels derer die Schätzung bei einer objektorientierten Entwicklung ermöglicht werden soll. Zugrunde gelegt wird eine Objektmodellierung mit Objekt-, Kommunikations- und Prozessmodell. Grundlage der Schätzung sind die Object-Points, die sich aus den

Klassen-, Sequenz-, Kollaborations- und UseCase-Diagrammen der Objektmodellierung ergeben. Aus ihnen wird der Umfang des Software-Produkts berechnet.

Zur Ermittlung der Object-Points werden im Objektmodell *Class-Points*, im Kommunikationsmodell *Message-Points* und im Prozessmodell *Process-Points* gezählt.

Die **Class-Points** bilden die Summe aller Attribute, Methoden und Relationen jedes Objektes unter Berücksichtigung einer Wiederverwendungsrate. Diese Rate gibt Auskunft, ob ein Objekt neu ist oder ob es aus einer Klassenbibliothek abgeleitet wurde. Zum Beispiel beträgt die Wiederverwendung bei einem neu definierten Objekt 0 %. Die Class-Points werden nach folgender Vorschrift ermittelt:

$$\text{Class-Points} = \{ \text{Attribute} + (\text{Relationen} * 2) + (\text{Methoden} * 3) \} * \text{Wiederverwendungsrate}$$

Die Summe der Class-Points stellt das Software-Maß für den Klassenentwicklungsaufwand einschließlich Entwurf, Codierung und Modultest dar.

Die **Message-Points** ergeben sich aus den Nachrichten bzw. Schnittstellen zwischen den Objekten. Es werden die Nachrichten je Objekt hinsichtlich Parameter, Sender und potenzieller Empfänger analysiert. Zusätzlich zur Wiederverwendungsrate ist die relative Komplexität der Nachrichten zu beachten. Es erfolgt eine Klassifizierung mit niedrig (75 %), mittel (100 %) oder hoch (125 %). Die Summe der Message-Points wird wie folgt berechnet:

$$\text{Message-Points} = \{ \text{Parameter} + (\text{Quellen} * 2) + (\text{Ziele} * 2) \} * \text{Komplexitätsrate} * \text{Wiederverwendungsrate}$$

Die Summe der Message-Points ist Software-Maß für die Integration und den Testaufwand.

Die **Process-Points** werden durch eine Zählung der Prozesse ermittelt. Dabei werden vier verschiedene Typen mit unterschiedlicher Gewichtung unterschieden:

- Batchprozesse mit keiner oder geringer menschlicher Interaktion (Gewichtung 2),
- Onlineprozesse mit erheblicher menschlicher Interaktion (Gewichtung 4),
- Systemprozesse wie Backup und Recovery (Gewichtung 6) und
- Realtimeprozesse, die durch externe Ereignisse gesteuert werden (Gewichtung 8).

Wie bei der Zählung der Message-Points wird die Komplexität des Prozesses berücksichtigt. Die Klassifizierung erfolgt analog zu den Message-Points. Folgende Vorschrift ergibt die Summe an Process-Points:

$$\text{Process-Points} = (\text{Prozesstyp} + \text{Varianten}) * \text{Komplexität}$$

Im Ergebnis wird die **Summe der ungewichteten Object-Points** wie folgt berechnet:

$$\text{Object-Points} = \text{Class-Points} + \text{Message-Points} + \text{Process-Points}$$

Die **Justierung der ungewichteten Object-Points** erfolgt mit einem Qualitätsfaktor und zehn Einflussfaktoren.

Der *Qualitätsfaktor* stellt den arithmetischen Mittelwert von bis zu 12 Qualitätsfaktoren dar und sollte keinesfalls vernachlässigt werden, da die Qualitätsanforderungen an ein Software-Produkt einen signifikanten Einfluss auf die Höhe des Aufwands haben [Sneed 95.2]. Bei den ggf. zu betrachtenden Qualitätsfaktoren handelt es sich um Korrektheit, Zuverlässigkeit, Konformität, Benutzbarkeit, Integrität, Performance, Effizienz, Testbarkeit, Portabilität, Wartbarkeit, Wiederverwendbarkeit, und Datenunabhängigkeit, die mit ihrem prozentualen Erfüllungsgrad in die Berechnung einfließen. Die qualitätsjustierte Anzahl an Object-Points ist das Produkt aus der ungewichteten Anzahl an Object-Points und dem Qualitätsfaktor.

Zudem werden die folgenden 10 *Einflussfaktoren* berücksichtigt:

- 1) Verfügbarkeit von Groupware zur Unterstützung des Projektteams,
- 2) Qualität der Benutzeroberfläche der Entwicklungsumgebung,
- 3) Zuverlässigkeit des Netzwerks in dem die Entwickler arbeiten,
- 4) Prozessreife innerhalb des Teams für die Softwareentwicklung,
- 5) technische Unterstützung des Projektteams,
- 6) Anwendungsgrad objektorientierter Methoden,
- 7) objektorientiertes Niveau der verwendeten Sprache,
- 8) Verfügbarkeit objektorientierter CASE-Tools,
- 9) Anwesenheit eines Object Repository und
- 10) Grad der Testautomatisierung.

Jeder Einflussfaktor wird mit einem Faktor zwischen 0 und 4 gewichtet:

- |   |                         |
|---|-------------------------|
| 0 | keine Erfüllung         |
| 1 | geringe Erfüllung       |
| 2 | teilweise Erfüllung     |
| 3 | hohe Erfüllung          |
| 4 | vollständige Erfüllung. |

Der aus den 10 gewichteten Einflussfaktoren resultierende Gesamteinflussfaktor wird mit

$$1 - (0.01 * \text{Gesamtwicht})$$

berechnet. Im schlechtesten Fall ergibt die Summe der Einflussfaktoren 0 und hat somit keinerlei Einfluss auf die ungewichteten Object-Points (Gesamteinflussfaktor = 1). In allen anderen Konstellationen wird der Aufwand in jedem Fall gemindert (Gesamteinflussfaktor < 1). Die einflussjustierten Object-Points stellen das Produkt aus der Anzahl der qualitätsjustierten Object-Points und dem Gesamteinflussfaktor dar.

Aus dieser Größe, der Summe an justierten Object-Points, wird der Aufwand in Personenmonaten abgeleitet. Dies erfolgt über eine spezifische Produktivitätstabelle, die die eigenen Bedingungen widerspiegeln sollte. Diese Tabelle gibt Auskunft, wie viele getestete Codezeilen in einem Personenmonat produziert werden. Damit wird unterstellt, dass Object-Points zu Codezeilen in Relation stehen. Harry M. Sneed bezieht sich in dieser Aussage auf Capers Jones, der eine Äquivalenz zwischen Function-Points und Codezeilen annimmt [Sneed 95.2].

### 2.3.5 Data-Point-Methode

Die Data-Point-Methode ist eine von Harry M. Sneed entwickelte Variante der Function-Point-Methode zur Aufwandschätzung anhand des Datenmodells (Entity-Relationship-Modell). Die Methode wurde 1990 erstmals veröffentlicht und basiert auf den Informationen, die aus der Datenmodellierung im Rahmen der objektorientierten Softwareentwicklung gewonnen werden können.

Die Data-Points werden anhand folgender Informationen aus dem Datenmodell ermittelt:

- *Datenentitäten* und ein Teil ihrer Attribute,
- *Schlüssel* der Datenentitäten,
- *Beziehungen* zwischen den Datenentitäten untereinander und
- *Sichten* auf die Datenentitäten und einen Großteil ihrer Attribute

und anhand ihrer Komplexität gewichtet. Diese Wichtung erfolgt in zwei Schritten:

- 1.) Eine Datensicht ist die Summe ihrer Attribute. Zu dieser Summe wird abhängig von der Komplexität der Oberfläche ein Gewicht von 2 (niedrig), 4 (mittel) oder 8 (hoch) addiert.
- 2.) Zu jeder Datensicht wird die zweifache Anzahl ihrer Beziehungen zu den Datenentitäten addiert.

Mit der nachstehenden Abbildung wird die vorab beschriebene Ermittlung der Data-Points veranschaulicht:

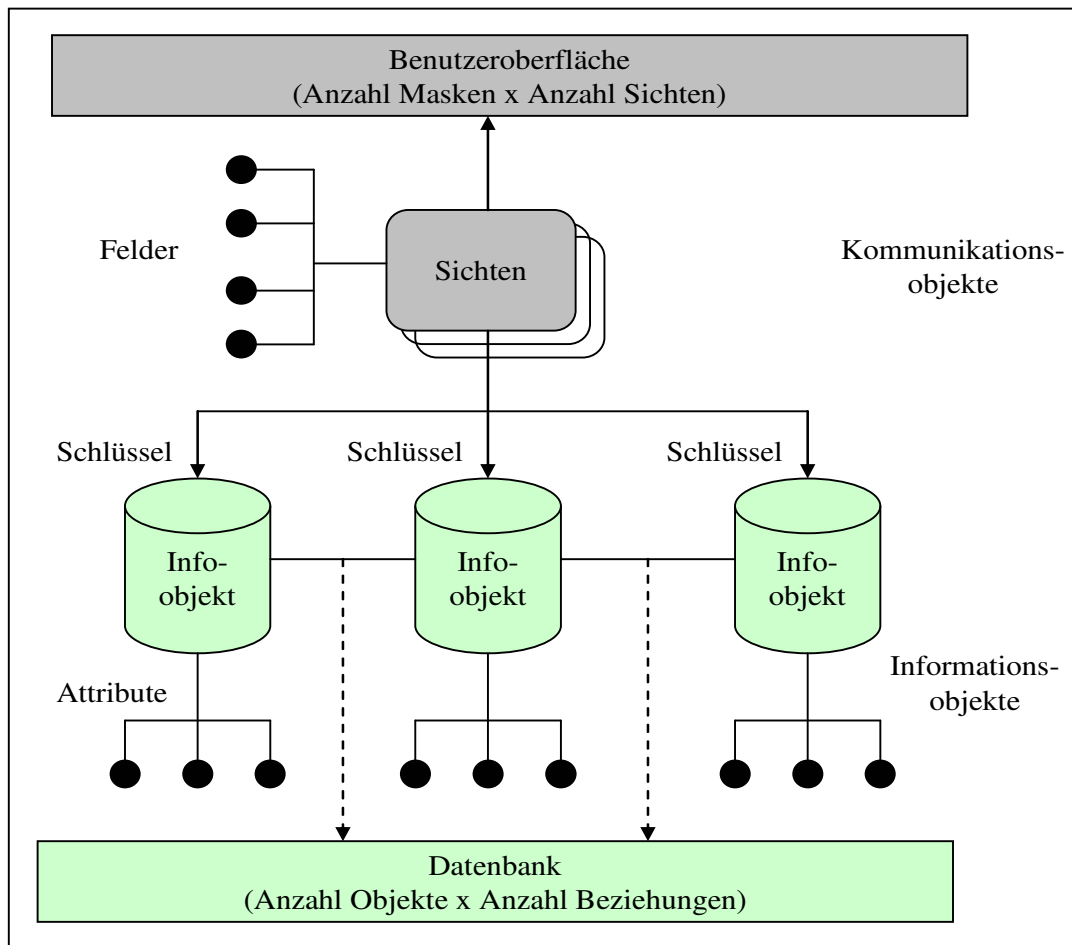


Abb. 3: Data-Point-Ableitung von Harry M. Sneed [Sneed 05]

Wie aus Abbildung 3 ersichtlich wird zwischen zwei Arten von Data-Points unterschieden: *Information-Points* und *Communication-Points*.

Bei den Informations-Points handelt es sich um die Summe der Data-Points für jede Datenentität, die aus dem Datenmodell gewonnen werden:

$$\text{Information-Points} = \sum \text{Entität} [\text{Attribute} + (\text{Schlüssel} * 4) + (\text{Beziehung} * 2)]_{1:n}$$

Die Communication-Points stellen hingegen die Summe der Data-Points für jede Datensicht dar und werden aus dem Kommunikationsmodell abgeleitet:

$$\text{Communication-Points} = \sum \text{Sicht} [\text{Attribute} + (\{2,4,8\}) + (\text{Beziehung} * 2)]_{1:m}$$

Die Summe aus Information-Points und Communication-Points ergibt die ungewichteten Data-Points, die mittels Qualitäts- und Einflussfaktoren justiert werden. Anstelle der bei der Function-Point-Methode verwendeten 14 Einflussfaktoren werden bei der Data-Point-Methode 8 Qualitäts- und 10 Einflussfaktoren berücksichtigt.

Als Qualitätsfaktoren dienen folgende Software-Charakteristika:

- 1.) Zuverlässigkeit
- 2.) Sicherheit
- 3.) Effizienz
- 4.) Datenunabhängigkeit
- 5.) Benutzerfreundlichkeit
- 6.) Übertragbarkeit
- 7.) Integrität
- 8.) Wartbarkeit

Die Multiplikation der ungewichteten Data-Points mit dem arithmetischen Mittelwert der zuvor genannten Einzelfaktoren ergibt die qualitätsjustierte Anzahl an Data-Points.

Bei den 10 Einflussfaktoren handelt es sich im Einzelnen um:

Einflussfaktor	Gewicht				
	0	2	4	6	8
Projektverteilung	mehrere Länder	mehrere Orte	ein Ort	mehrere Räume	ein Raum
Projekterfahrung	keine	mäßig	mittel	gut	sehr gut
Systemkenntnisse	keine	mäßig	mittel	gut	sehr gut
Arbeitsbedingungen	schlecht	mäßig	mittel	gut	sehr gut
Werkzeugausstattung	keine	teils	mittel	viel	voll
Teamzusammengehörigkeit	keine	wenig	mittel	gut	sehr gut
Prozessreife	Chaos	wiederholbar	definiert	gemanagt	optimiert
Spezifikation	informal	strukturiert	semi-formal	formal	streng formal
Sprachgeneration	1	2	3	4	5
Testautomation	0	1-25 %	25-50 %	50-75 %	75-100 %

Tab. 13: Einflussfaktoren der Data-Point-Methode [Sneed 05]

Das Produkt aus qualitätsjustierten Data-Points und dem Einflussfaktor, der sich aus dem Durchschnitt aller 10 Einflussfaktoren ergibt, stellt die einflussjustierten Data-Points dar. Die-

se Anzahl wird mittels einer Produktivitätstabelle, die individuell ermittelt werden muss, in den entsprechenden Aufwand in Personenmonaten überführt (s. a. Kapitel 2.3.4).

### **2.3.6 Error-Projection-Methode**

Bei der Error-Projection-Methode handelt es sich um eine von Harry M. Sneed entwickelte Methode, die auf seinen eigenen Erfahrungen sowie auf Erfahrungen aus dem Bereich der Fehleranalyse basiert. Sie soll innerhalb der Wartungsphase eines Software-Produkts zum Einsatz kommen. Mit dieser Methode wird die Restfehlerrate betrachtet, d.h. die Zahl an Fehlern, die sich nach Auslieferung des Software-Produkts an den Kunden noch im Produkt befindet.

In der Praxis sind fehlerfreie Produkte nicht existent und im Allgemeinen kann auf 1.000 Anweisungen mit einer Fehlerrate zwischen 0.1 und 3.0 gerechnet werden. Laut [Thaller 00] schwanken die Werte der Fehlerrate über einen weiten Bereich.

Bezogen auf die Software-Lebensphase der Anwendung ist die Fehlerhäufigkeit eines Software-Produkts zu Beginn am höchsten und nimmt tendenziell ab, wenn sich das Anforderungsprofil nicht ändert. Jede Änderung am Software-Produkt birgt die Gefahr einer neuerlichen Erhöhung der Fehlerrate.

Fehler werden anhand ihrer unmittelbaren Auswirkung klassifiziert. Schwerwiegende und ggf. minderschwere Fehler müssen umgehend beseitigt werden, z.B. um den Produktionsablauf nicht zu gefährden und Ausfallzeiten und finanzielle Verlustwerte zu minimieren.

Nach Harry M. Sneed ist es bei der Schätzung des gesamten Wartungsaufwands unerlässlich, auch den Aufwand für derartige Fehlerbehebungen zu berücksichtigen, um im Fehlerfall über entsprechende Ressourcen (z.B. Personal) verfügen zu können und damit den Schaden in Grenzen zu halten.

Basis der Schätzung sind laut [Sneed 05]

- die Systemgröße vor und nach dem letzten Release,
- die geschätzte Anzahl der Fehler für die nächste Zeitperiode und
- die Fehlerbehebungsproduktivität.

Die *Systemgröße* ergibt sich aus einer statischen Analyse des Quellcodes.

Die *geschätzte Anzahl an Fehlern für die nächste Zeitperiode* wird aus der geschätzten Fehlerdichte und der Testüberdeckungsrate ermittelt. Die Fehlerdichte stellt hierbei das Produkt aus der Fehlerdichte der letzten Zeitperiode und der Wachstumsrate dar. Der Testüberdeckungsgrad ist ein Maß für die Qualität der vor dem Release stattfindenden Tests. Je höher der Testüberdeckungsgrad ist, desto niedriger ist die Anzahl der vom Kunden gefundenen Fehler.

Das erste Zwischenergebnis der Schätzung ist die absolute Fehleranzahl, die im aktuellen Release erwartet wird und die das Produkt der justierten Fehlerdichte (geschätzte Anzahl an Fehlern) und der Systemgröße darstellt.

Im nächsten Schritt der Schätzung wird die *Fehlerbehebungsproduktivität* ermittelt. Sie basiert auf dem Fehlerbehebungsaufwand in der vorangegangenen Zeitperiode und der veränderten Systemkomplexität. Der Fehlerbehebungsaufwand ist ein Erfahrungswert, der die Anzahl der Personentage, die für eine Fehlerbehebung benötigt wurden, darstellt. Dieser Wert wird mit der veränderten Systemkomplexität justiert und ergibt die voraussichtlich erwarteten Personentage pro Fehlerkorrektur.

Der innerhalb der Wartung zu berücksichtigende Aufwand nach der Error-Projection-Methode ermittelt sich im Abschluss nach folgender Vorschrift:

$$\text{Aufwand in Personentagen} = \text{absolute Fehleranzahl} * \text{Fehlerbehebungsaufwand}.$$

Diese Aufwandskennzahl stellt den mittleren Fehleraufwand dar und bezieht sich auf die gesamte, erwartete Fehleranzahl einer bestimmten Zeitperiode.

### **2.3.7 UseCase-Point-Methode**

Diese Methode wurde 1993 von Gustav Karner vorgestellt und speziell für eine UML-basierte Softwareentwicklung entwickelt. Sie stellt wie die Object-Point- und die Data-Point-Methode eine Variante der Function-Point-Methode dar. Die UseCase-Point-Methode ermöglicht eine schnelle und frühzeitige Schätzung des zu erwartenden Aufwandes für Software-Entwicklungsprojekte. Basis der Schätzung sind Grob-Spezifikationen unterschiedlichen Formats und unterschiedlicher Granularität.

Die Ermittlung der UseCase-Points basiert auf folgender Berechnung:

$$UCP = UUCP * TCF * ECF$$

UUCP = Unadjusted UseCase Points (unjustierte UseCase-Points)

TCF = Technical Complexity Factor (technischer Komplexitätsfaktor)

ECF = Environmental Complexity Factor (Komplexitätsfaktor der Umgebung).

Im ersten Schritt werden aus den Anwendungsfällen (UseCases) die *unjustierten UseCase-Points* (UUCP) ermittelt. Dazu wird zunächst die Anzahl der Schritte aller UseCase-Szenarien gezählt, klassifiziert (einfach, durchschnittlich oder komplex) und gewichtet (5, 10, 15). Die Summe der gewichteten UseCases aller Kategorien bilden die *Unadjusted UseCase Weight* (UUCW). Zudem werden alle an den UseCases beteiligten Akteure betrachtet. Diese werden ebenfalls klassifiziert (einfach, durchschnittlich oder komplex) und gewichtet (1, 2, 3). Die Summe der gewichteten Akteure der einzelnen Kategorien ergibt die *Unadjusted Actor Weight* (UAW).

Die Summe der unjustierten UseCase-Points berechnet sich aus

$$UUCP = UUCW + UAW.$$

Im nächsten Schritt wird der *technische Komplexitätsfaktor* (TCF), der auf den folgenden 13 technischen Einflussfaktoren basiert, ermittelt:

Technischer Faktor	Beschreibung	Gewichtung
T <sub>1</sub>	Distributed System	2.0
T <sub>2</sub>	Performance	1.0
T <sub>3</sub>	End User Efficiency	1.0
T <sub>4</sub>	Complex Internal Processing	1.0
T <sub>5</sub>	Reusability	1.0
T <sub>6</sub>	Easy to Install	0.5
T <sub>7</sub>	Easy to Use	0.5
T <sub>8</sub>	Portability	2.0
T <sub>9</sub>	Easy to Change	1.0
T <sub>10</sub>	Concurrency	1.0
T <sub>11</sub>	Special Security Features	1.0
T <sub>12</sub>	Provides Direct Access for Third Parties	1.0
T <sub>13</sub>	Special User Training Facilities Are Required	1.0

Tab. 14: Technische Komplexitätsfaktoren der UseCase-Point-Methode

Die Höhe des Einflusses der einzelnen Faktoren wird über die Bewertung ihrer Komplexität vorgenommen.

Ein technischer Faktor ohne Einfluss auf das Produkt wird mit 0 bewertet, einer mit durchschnittlichem Einfluss mit 3 und einer mit starkem Einfluss mit 5. Die Summe der gewichteten und mit ihrer Komplexität multiplizierten technischen Einflussfaktoren ergibt den technischen Gesamtfaktor TTF (*Technical Total Factor*).

Der benötigte technische Komplexitätsfaktor TCF wird mit nachstehender Formel berechnet:

$$TCF = 0.65 + (0.01 * TTF)$$

Im nächsten Schritt wird der *Komplexitätsfaktor der Umgebung* (ECF) ermittelt. Grundlage bilden nach Karner die folgenden 8 Umgebungsfaktoren, die die Erfahrung und das Wissen des jeweiligen Entwicklerteams widerspiegeln:

Umgebungsfaktor	Beschreibung	Gewichtung
E <sub>1</sub>	Familiarity with UML	1.5
E <sub>2</sub>	Part-time Workers	-1.0
E <sub>3</sub>	Analyst Capability	0.5
E <sub>4</sub>	Application Experience	0.5
E <sub>5</sub>	Object-Oriented Experience	1.0
E <sub>6</sub>	Motivation	1.0
E <sub>7</sub>	Difficult Programming Language	-1.0
E <sub>8</sub>	Stable Requirements	2.0

Tab. 15: Umgebungsfaktoren der UseCase-Point-Methode

Den Einfluss des einzelnen Faktors auf das aktuelle Projekt widerspiegelt die Komplexität mit Wert 1 (starker negativer Einfluss), Wert 3 (durchschnittlicher Einfluss), Wert 5 (starker positiver Einfluss) oder Wert 0 (kein Einfluss). Der gewichtete und mit seiner Komplexität multiplizierte Umgebungsfaktor bildet den kalkulierten Faktor.

Die Summe aller kalkulierten Umgebungsfaktoren bildet den Gesamtfaktor der Umgebung ETF (*Environmental Total Factor*).

Der Komplexitätsfaktor der Umgebung (ECF) wird mit folgender Berechnungsvorschrift ermittelt:

$$ECF = 1.4 + (- 0.03 * ETF)$$

Unter Berücksichtigung eines Produktivitätsfaktors wird der Aufwand aus den so ermittelten UseCase-Points UCP folgendermaßen berechnet:

$$Aufwand = UCP * Produktivität$$

Der Produktivitätsfaktor resultiert aus den Arbeitsstunden, die ein Entwicklerteam pro Use-Case-Point benötigt und wird nach jedem abgeschlossenen Projekt neu bestimmt:

$$\text{Produktivität}_{\text{neu}} = \frac{\text{Zeitbedarf}_{\text{des}_{\text{aktuellen}_{\text{Projekts}}}}}{\text{UCP}_{\text{des}_{\text{aktuellen}_{\text{Projekts}}}}}$$

### 2.3.8 Testfallmethode<sup>2</sup>

Die Testfallmethode wurde von Harry M. Sneed entwickelt. Einsatzgebiet sind Integrations- und Installationsprojekte. Mit der Methode soll der Aufwand für den Test und die Integration der gelieferten Software-Produkte geschätzt werden. Der Testaufwand wird aus der Anzahl der Testfälle, die benötigt werden um alle Funktionen des Software-Produkts zu testen, abgeleitet. Zur Ermittlung des benötigten Aufwands sind Testfälle auf drei semantischen Stufen zu unterscheiden:

- Stufe 1: Fachkonzeptebene,
- Stufe 2: Systementwurfsebene und
- Stufe 3: Komponentenebene.

Im Ergebnis werden dadurch drei Aufwandskennzahlen klassifiziert:

- Systemtestaufwand (Stufe 1),
- Integrationsaufwand (Stufe 2) und
- Komponententestaufwand (Stufe 3).

Die Ermittlung des Aufwands auf der jeweiligen Stufe vollzieht sich im Einzelnen wie folgt:

#### **Stufe 1:** Testfälle der Fachkonzeptebene (Systemtestaufwand)

Die Testfälle der Fachkonzeptebene werden aus dem Fachkonzept oder der Systemdokumentation ermittelt. Der Text des Konzepts oder der Dokumentation wird analysiert und in einen Entscheidungsbaum überführt. Der arithmetische Mittelwert aller Wege durch diesen Entscheidungsbaum ergibt die Anzahl funktionaler Testfälle. Die nicht-funktionalen Testfälle sind das Produkt aus der Anzahl der Qualitätsanforderungen und der Anzahl der Anwendungsfälle. Die Summe der funktionalen und nicht-funktionalen Testfälle ergibt die Anzahl der insgesamt benötigten Testfälle. Der benötigte Systemtestaufwand wird unter Berücksich-

---

<sup>2</sup> Im Tool SoftCalc und der zugehörigen Dokumentation wird die Methode unter dem englischen Begriff „Test-Point“ aufgeführt.

tigung des Faktors (*(Alt-)Aufwand pro Testfall*), der die Produktivität je Testfall vergangener Projekte widerspiegelt, wie folgt berechnet:

$$\text{Systemtestaufwand} = \text{Systemtestfälle} * (\text{Alt-})\text{Aufwand pro Testfall}$$

**Stufe 2:** Testfälle der Systementwurfsebene (Integrationsaufwand)

Die Testfälle der Systementwurfsebene werden aus der Systementwurfsdokumentation abgeleitet und zählen zum „Grey-Box“-Testansatz. Es werden die UML-, Sequenz- und Zustandsdiagramme der Dokumentation analysiert, um die Anzahl der Pfade und die Anzahl der Zustandsübergänge zu identifizieren. Das Ergebnis der Analyse, die Summe der Integrationstestfälle, bildet die Menge aller Methodenaufrufsfolgen und Objektzustandsübergänge. Der benötigte Integrationsaufwand ergibt sich aus folgender Vorschrift:

$$\text{Integrationsaufwand} = \text{Integrationstestfälle} * (\text{Alt-})\text{Aufwand pro Testfall.}$$

Bei dem Faktor *(Alt-)Aufwand pro Testfall* handelt es sich um den Zeitbedarf pro Testfall aus bereits abgeschlossenen Projekten.

**Stufe 3:** Testfälle der Komponentenebene (Komponententestaufwand)

Die Testfälle der Komponentenebene werden aus dem Quellcode ermittelt und gehören damit zum „White-Box“-Testansatz. Aus der Pfadüberdeckung wird die minimale Anzahl an benötigten Komponententestfällen berechnet. Diese semantische Stufe hat nur Relevanz, wenn zumindest Teile des Quellcodes vorhanden und zudem frei zugänglich sind. Der Komponententestaufwand wird wie in den Stufen 1 und 2 unter Berücksichtigung eines Erfahrungswerts aus vergangenen Projekten (*(Alt-)Aufwand pro Testfall*) ermittelt:

$$\text{Komponententestaufwand} = \text{Komponententestfälle} * (\text{Alt-})\text{Aufwand pro Testfall.}$$

Der jeweils ermittelte Aufwand kann nach Harry M. Sneed noch entsprechend justiert werden, um z. B. projekt- und prozessspezifische Einflussfaktoren zu berücksichtigen. Dies liegt im Ermessen des Schätzers.

In Harry M. Sneeds Fallstudie zu einem Integrationsprojekt wurde der ermittelte Testaufwand mit einem Qualitäts- und einem Komplexitätsfaktor sowie mit den fünf Skalierungsfaktoren der COCOMO II-Methode justiert. Durch die Berücksichtigung der Komplexität und Qualität ist eine Abweichung von bis zu  $\pm 60\%$  möglich. Die Einflussfaktoren der COCOMO II-

Methode erhöhen den Testaufwand in keinem Fall. Sie mindern ihn um bis zu 25 %, wenn alle Skalierungsfaktoren mit dem höchsten Gewicht (5) angesetzt werden.

Der justierte Testaufwand wird nach folgender Vorschrift berechnet:

$$\text{Justierter Testaufwand} = \text{unjustierter Testaufwand} * EF * QF * KF$$

$$EF = 1 - [\text{Summe der Skalierungsfaktoren} / 100]$$

$$QF = 0.5 / \text{Systemqualitätskoeffizient}$$

$$KF = \text{Systemkomplexitätskoeffizient} / 0.5.$$

## 2.4 Toolunterstützung bei der Aufwandschätzung

Im folgenden Kapitel werden abschließend zu den vorab beschriebenen Aufwandschätzmethoden Gründe für eine Toolunterstützung bei der Aufwandschätzung erläutert und Beispiele für derartige Software-Werkzeuge genannt.

Nach [Dumke 00] ist ein Software-Messtool ein Software-Werkzeug, welches Komponenten eines Software-Produktes bzw. der Software-Entwicklung in ihrer Quellform oder in ihrer transformierten Form einliest und nach vorgegebenen Verarbeitungsvorschriften, die durch die jeweilige Software-Metrik definiert werden, numerisch oder symbolisch auswertet. Es dient der Ausprägung der jeweiligen Messstrategie, der Aufbereitung der Messobjekte oder der statistischen Auswertung oder Darstellung der Messergebnisse. Diese Art von Software-Werkzeugen wird als CAME-Tool (Computer Assisted Software Measurement and Evaluation) bezeichnet.

Für eine Aufwandschätzung mittels CAME-Tools gibt es vielfältige Gründe. An dieser Stelle sollte nicht vergessen werden, dass die Aufwandschätzung ihren Anwendungsbereich in der Softwareentwicklung, -anwendung und -wartung findet. Mit Verzicht auf Toolunterstützung würde zu einem gewissen Teil die Daseinsberechtigung der Softwareentwicklung in Frage gestellt werden.

Der Einsatz von Software-Werkzeugen sollte dazu beitragen, die Produktivität zu steigern und die Qualität innerhalb des Software-Entwicklungsprozesses zu erhöhen. In diesem Zusammenhang ist das Capability-Maturity-Modell (CMM) zu erwähnen. Mit diesem Modell wird der Reifegrad der Qualität des Software-Entwicklungsprozesses in einem Unternehmen oder

einer Organisation beurteilt. In der folgenden verkürzten Darstellung des Modells wird aufgezeigt, auf welcher Niveaustufe sich der Softwareentwicklungsprozess des Unternehmens oder der Organisation befindet, wenn Software-Messwerkzeuge eingesetzt werden:

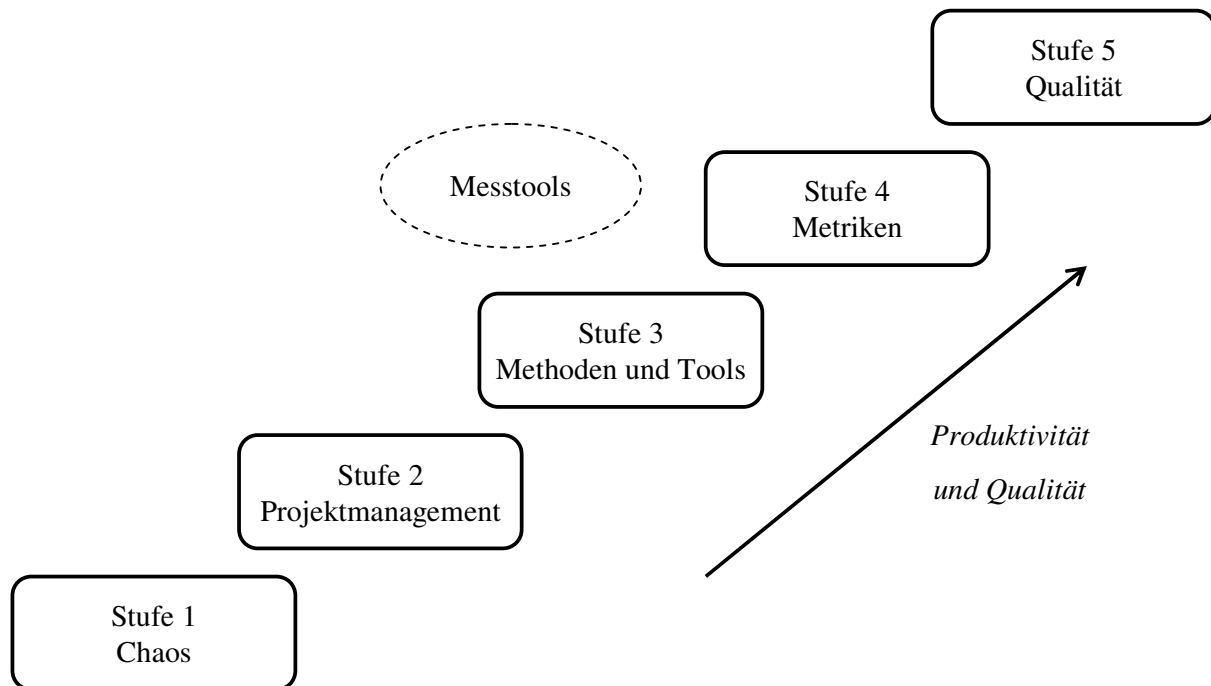


Abb. 4: verkürzte Darstellung des CMM-Modells nach Arthur aus [Dumke 96]

Ohne Toolunterstützung würde sich der Aufwand für die Schätzung erheblich erhöhen, denn aufgrund der Vielzahl an Messungen und Auswertungen wäre der Arbeitsumfang bei manueller Durchführung schwer zu bewältigen. Das würde nicht zur Akzeptanz des Schätzverfahrens beitragen und die Bereitschaft, ein derartiges Verfahren in die Unternehmenskultur zu integrieren, würde merklich sinken. Beispielsweise ist eine statische Quellcodeanalyse ohne entsprechende Werkzeuge nur bei sehr kleinen Codeumfängen ökonomisch vertretbar.

Hier ist eine Messung Capers Jones interessant, die der Frage „Welche Toolstützung führt zu welchen Produktivitätssteigerungen?“ nachging. Die Messungen wurden an einem speziellen Projekt mit mehr als einer Million C-Codeanweisungen vorgenommen. Beim Aspekt „Pläne, Schätzungen und Budgets“ wurde ohne Verwendung von Tools ein Aufwand von 15 Personenmonaten gemessen. Die Anwendung von Tools ergab hingegen einen gemessenen Aufwand in Höhe von 3 Personenmonaten.<sup>3</sup>

<sup>3</sup> Das Beispiel wurde [Dumke 96] entnommen.

Zur Schätzkultur eines Unternehmens gehört es auch, Erfahrungswerte aus bereits abgeschlossenen Projekten als Grundlage für zukünftige Aufwandschätzungen zu erfassen, um damit die Schätzgenauigkeit zu erhöhen. Dafür ist ebenso eine Toolunterstützung hilfreich.

## **2.5 Zusammenfassung**

Im Kapitel 2 dieser Arbeit wurden die theoretischen Grundlagen der Aufwandschätzung sowie die Schätzmethoden, die vom Software-Messwerkzeug SoftCalc unterstützt werden, beschrieben.

Jede dieser Aufwandschätzmethoden hat ihre speziellen Anwendungsbereiche und versucht, mit Hilfe verschiedener Software-Maße die Charakteristika, die jeweils von Bedeutung sind, zu berücksichtigen. Aufgrund der Vielfalt und Komplexität des Themengebiets lassen sich die Methoden nur in ihrem Grundgerüst auf einen gemeinsamen Nenner bringen. Eine allgemeingültige Formel zur Aufwandschätzung gibt es nicht und das Software-Maßsystem unterliegt ständigen Änderungen. Daraus resultiert eine zu große Methodenvielfalt mit jeweils geringer Verbreitung, was zu Akzeptanzproblemen führt.

Sobald jedoch Aufwandschätzmethoden konsequent und kontinuierlich angewendet werden, liefern sie Schätzergebnisse, die eine solide Projektplanung ermöglichen. Die Schätzverfahren müssen in die Unternehmensstruktur eingebettet sein und dürfen nicht als lästiges Übel betrachtet werden, denn jede Schätzung ist besser als keine.

Abschließend ist festzustellen, dass zwar jede Aufwandschätzmethode gewisse Nachteile hat, aber die Durchführung der Aufwandschätzung dennoch dazu beiträgt, den geforderten ingenieurmäßigen Ansatz bei der Softwareentwicklung zu verwirklichen und den Begriff „Software Engineering“ auf eine solide Basis zu stellen.

### 3 Qualitative Anforderungen an Software-Messtools

#### 3.1 Der Begriff Qualität im Bereich der Softwareentwicklung

Wie bereits in der Einführung dieser Arbeit dargelegt wurde ist es unumgänglich, bei der Softwareentwicklung bestimmte Vorgehensweisen zu definieren, damit z.B. die zu entwickelnden Software-Produkte die an sie gestellten qualitativen Anforderungen erfüllen. Die Qualität eines Software-Produktes darf nicht auf zufälligen Gegebenheiten beruhen, sondern muss Ergebnis eines definierten Prozesses sein. Nach DIN-Norm 55350 ist die (Software-) Qualität die Gesamtheit der Merkmale und Merkmalswerte eines materiellen oder immateriellen Gegenstandes der Betrachtung bezüglich ihrer Eignung, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen [Dumke 96]. Diese Anforderungen gelten ebenso für Software-Werkzeuge zur Aufwandschätzung, d.h. auch bei der Entwicklung von Software-Messwerkzeugen sind die allgemeinen Anforderungen an die Softwarequalität nach ISO 9126 zu beachten. Grundsätzlich sind damit die 6 in Abbildung 5 dargestellten Kriterien zu betrachten, um die Softwarequalität eines Produkts beurteilen zu können:

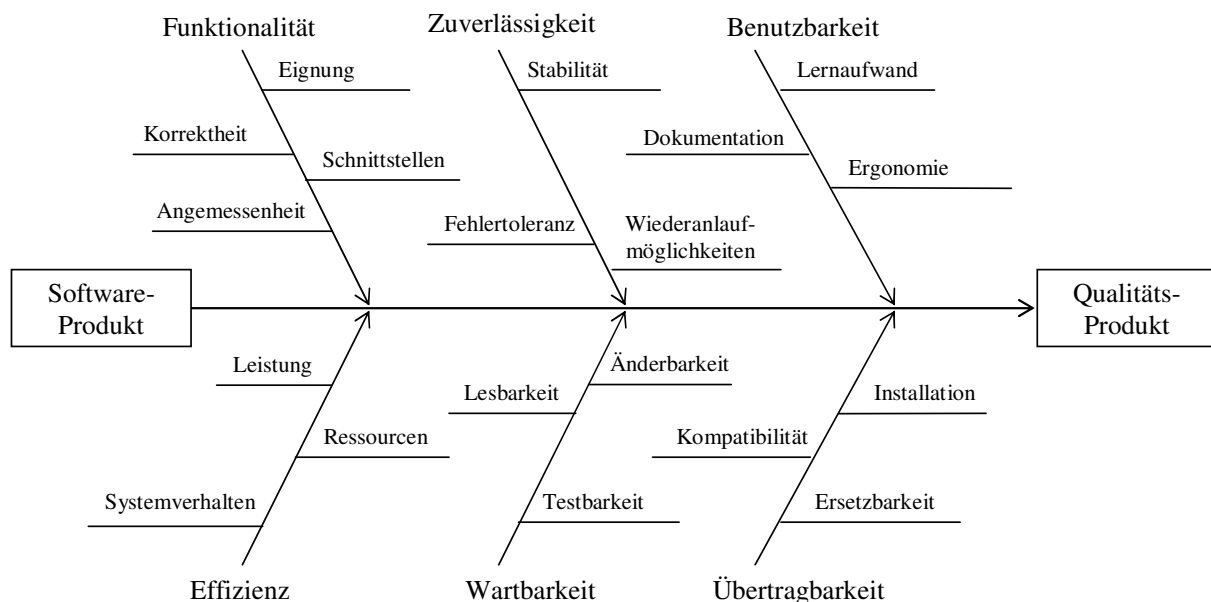


Abb. 5: Qualitätskriterien nach ISO 9126 aus [Dumke 96]

Auf eine detailliertere Beschreibung dieser Kriterien wird an dieser Stelle verzichtet, da sich die Arbeit auf die konkreten Qualitätsanforderungen an ein Software-Messwerkzeug im Sinne der Aufwandschätzung bezieht, die in den folgenden Kapiteln eingehend betrachtet werden.

## 3.2 Qualität der Software-Maße und -Metriken

„Der Sinn des Messens ist ein Vergleich einer neuen Situation mit einer bereits bekannten Situation unter Verwendung eines bestimmten Maßes, um Analogieschlüsse zu ermöglichen“ [Ligg 02]. „Durch Messungen am Software-Produkt, Statistiken und quantitativen Aussagen zur Güte der Software kann man die Software-Entwicklung entscheidend verbessern. Nur wenn die Bedingungen, unter denen Software entsteht, definiert, verbessert und kontrolliert und die Auswirkungen getroffener Maßnahmen durch Messungen überprüft werden, kann auf Dauer eine nachhaltige Verbesserung der Situation erreicht werden.“ [Thaller 00]

Im Bereich der Software-Messung fehlt es aufgrund der Komplexität und Vielschichtigkeit der Softwareentwicklung an standardisierten und allgemeingültigen Softwaremetriken und -maßen. Software-Maße werden mitunter für ein Messziel innerhalb eines speziellen Entwicklungs- und Produktumfeldes definiert und können dadurch keine Allgemeingültigkeit erlangen. Die Korrektheit und Eignung eines Software-Maßes bestimmt die Qualität der Messmethode und letztlich des Software-Messtools.

„Messen ist der Prozess, der den Eigenschaften von Objekten der realen Welt mit Hilfe einer Metrikvorschrift Zahlen oder Zeichen so zuordnet, dass diese Objekte anhand von definierten Regeln charakterisiert werden. Nur wenn die Metrikvorschrift tatsächlich die Objekteigenschaft misst, die sie zu messen vorgibt, handelt es sich um ein Maß.“ [EbDu 96]

Bei einem Software-Maß handelt es sich nach [Dumke 00] um eine mit einer Maßeinheit versehene Skala, die in dieser Form ein Software-Attribut bewertet bzw. messbar macht und die die empirische Bedeutung des jeweiligen Software-Attributs veranschaulicht. Eine Software-Metrik ist hingegen eine Abstandsfunktion, die Attributen von Software-Komponenten Zahlen (-bereiche) zuordnet.<sup>4</sup>

Die Software-Maße und -Metriken sind Teil des Maßsystems beim Software Engineering, das sich auf alle wesentlichen Aspekte der Software-Entwicklung, -Anwendung und -Wartung bezieht und in seiner Anwendung aufgrund von Beobachtungen, Analysen, und kontrollierten

---

<sup>4</sup> Die Begriffe Software-Maß und –Metrik werden in der Literatur meist synonym verwendet.

Experimenten durch die Definition und den Einsatz neuer Maße oder Metriken kontinuierlich erweitert, angepasst und verbessert wird [Dumke 00].

Die Erweiterung des Maßsystems erfolgt mittels Softwaremessstrategie, die sich allgemein in folgenden Schritten vollzieht:

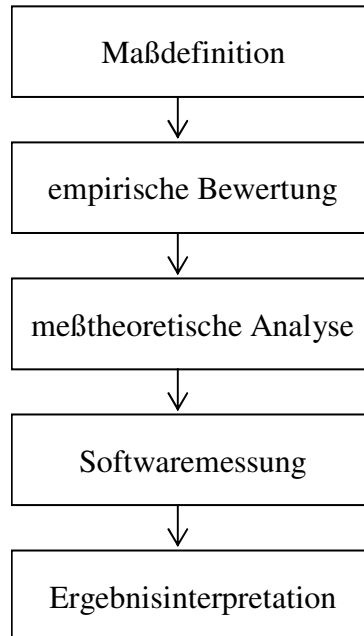


Abb. 6: Allgemeinste Schritte einer Softwaremessstrategie aus [Dumke 00]

Die Maßdefinition sollte unter anderem dazu beitragen, die Qualität des gemessenen Software-Produkts quantitativ beurteilen zu können. Ob ein Software-Maß für den jeweiligen Anwendungsbereich geeignet ist, entscheidet sich an der Validierungseigenschaft des Maßes.

In der Informatik bezeichnet Validierung die dokumentierte Beweisführung, dass ein System die Anforderungen in der Praxis erfüllt. Bezogen auf die Softwarequalität wird unter Validierung die Prüfung der Eignung (Korrektheit) beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck verstanden.

Bei der Prüfung der Validierung eines Software-Maßes sind folgende Probleme zu lösen [Dumke 00]:

- *Das Repräsentanzproblem:*  
Die Forderung, dass das Maß eine zahlenmäßige Darstellung (als Messwert bzw. numerisches Relativ) für die gemessene Eigenschaft liefert.
- *Das Eindeutigkeitsproblem*

Die Gleichartigkeit (Eindeutigkeit) des Verhaltens von mehreren Repräsentationen für ein und dieselbe (empirische) Eigenschaft.

- Das *Bedeutungsproblem*:

Die Beibehaltung der richtigen (empirischen) Bedeutung im Sinne der Interpretationen trotz zulässiger Messwerttransformationen.

- Das *Skalierungsproblem*:

Die Art und Weise der Bestimmung der jeweiligen Skalierung für die Software-Metrik bzw. für das Maß. Dabei wird eine Zuordnung gemäß der bekannten Skalentypen von einem sogenannten empirischen Relativ zu einem numerischen Relativ vorgenommen. Generell gilt, dass stets ein höheres Skalenniveau auf ein niederes zurückgeführt werden kann.

### **3.3 Anforderungen an ein Software-Messwerkzeug**

Ein Messtool stellt nach [Dumke 00] ein Software-Werkzeug dar, welches Komponenten eines Software-Produktes oder der Software-Entwicklung in ihrer Quellform oder transformierten Form einliest und nach vorgegebenen Verarbeitungsvorschriften, die durch die jeweilige Software-Metrik definiert werden, numerisch oder symbolisch auswertet. Sie dienen der Ausprägung der jeweiligen Messstrategie, der Aufbereitung der Messobjekte oder der statistischen Auswertung bzw. Darstellung der Messergebnisse.

Aus dieser Definition leiten sich die qualitativen Anforderungen ab, um sie zu erfüllen, sollte ein Software-Messwerkzeug im Sinne der Software-Messung generell über folgende Eigenschaften verfügen [Dumke 96]:

- 1.) Es sollte alle Messdaten rechnergestützt ermitteln und für eine weitere Verarbeitung zeitlich oder komponentenbezogen speichern.
- 2.) Die Messergebnisse sollten in den allgemein üblichen Diagrammformen aus der statistischen Analyse präsentiert werden.
- 3.) Zur Auswertung der Messergebnisse sollten die jeweils gültigen statistischen Methoden und Verfahren anwendbar sein.
- 4.) Die Eingabe der Messobjekte bzw. die Speicherung der Messdaten sollte jeweils eine Kopplung zu vorhandenen Werkzeugen, wie z.B. CASE-Werkzeugen, Analysewerkzeugen usw. ermöglichen.

- 5.) Das Messwerkzeug selbst sollte der jeweiligen Software-Entwicklungsmethodologie entsprechen und somit die Entwicklungskomplexität insgesamt nicht wesentlich vergrößern.
- 6.) Es sollte sich an allgemeinen Standards zur Software-Messung orientieren.

Hinsichtlich der zugrunde gelegten Aufwandschätzverfahren sollten zusätzlich folgende Kriterien betrachtet werden:

- 1.) *Schätzgenauigkeit* (größtmögliche Übereinstimmung der berechneten Soll-Werte aus der Aufwandsschätzung mit den sich im Projektverlauf ergebenden Ist-Werten)
- 2.) *Frühzeitigkeit* (Ausprägungen der relevanten Einflussfaktoren in Aufwandsschätzungen sollen in möglichst frühen Phasen des Entwicklungsprozesses für das Projektmanagement erkennbar sein)
- 3.) *Detaillierbarkeit* (in der Schätzmethode muss die Berechnung bis auf kleine überschaubare Einzelaktivitäten untergliedert werden können)
- 4.) *Eindeutigkeit* (bei gleichartiger Anwendung der Schätzverfahren durch unterschiedliche Personen muss ein annähernd gleiches Ergebnis für den Aufwand ermittelt werden)
- 5.) *Erfassbarkeit* (nur Faktoren, deren Schätzwerte quantitativ oder qualitativ beurteilt werden können, sollen in die Aufwandsschätzung einfließen)
- 6.) *Objektivität* (in die Aufwandsschätzung, sollen keine Faktoren, die nur durch subjektive Einschätzung bestimmt werden können, einfließen)
- 7.) *Transparenz* (die Darstellung und die Kalkulation im Schätzverfahren muss für Dritte inhaltlich und rechnerisch interpretationsfrei nachvollziehbar sein)
- 8.) *Stabilität* (Änderungen von Eingabeparametern der Aufwandsschätzung dürfen nicht zu Auswirkungen in der Kalkulation führen, die weit über den empirisch nachgewiesenen Einfluss dieser Parameter auf den Aufwand hinausgehen)
- 9.) *Flexibilität* (das Projektmanagementverfahren der Aufwandsschätzung muss zu unterschiedlichen Zeitpunkten bei unterschiedlichem Kenntnisstand innerhalb des Entwicklungsprozesses eingesetzt werden können)
- 10.) *Benutzerfreundlichkeit* (das Verfahren für die Schätzung muss einfach anwendbar und standardisierbar sein, und nur mit den verfügbaren Parametern auskommen, die auch nachweislich einen relevanten Einfluss auf die Schätzung haben)

Für die Prüfung der vorgenannten Kriterien wurde das Software-Messwerkzeug SoftCalc als repräsentativer Vertreter eines Aufwandschätzwerkzeugs ausgewählt. In den folgenden Kapiteln wird das Werkzeug beschrieben sowie geprüft und analysiert.

## 4 Das Messwerkzeug SoftCalc

### 4.1 Historie und grundlegende Arbeitsweise

Das Programm SoftCalc ist eine Weiterentwicklung des Werkzeugs PC-Calc, das im Rahmen eines europäischen Metrikprojekts durch Harry M. Sneed konzipiert und 1992 fertig gestellt wurde. Ziel des Projektes war die Erstellung eines Metrikleitfadens. Die Prototypentwicklung des Programms PC-Calc diente der Veranschaulichung verschiedener Vorgehensweisen im Bereich der Aufwandschätzung. Im dem Programm wurden fünf verschiedene Aufwandschätzmethoden berücksichtigt.

Mit Weiterentwicklung des Konzepts und Implementierung des Programms SoftCalc soll ein breiteres Spektrum an Aufwandschätzmethoden und Projektarten erfasst werden. Das Programm ist Teil des Software-Messungsarbeitsplatzes für Prüfung, Evaluierung und Berechnung von Produkten und Projekten, der in seiner Gesamtheit von Harry M. Sneed konzipiert wurde. Dieser Arbeitsplatz besteht aus den Werkzeugen SoftAudit, SoftCalc und SoftEval. Aufgabe dieser Werkzeuge sind im Einzelnen:

- Prüfen und Messen von Software-Entwürfen und Quellcode (*SoftAudit*)
- Akkumulieren und Evaluieren von Software-Metriken (*SoftEval*)
- Schätzung der Entwicklungs-, Wartungs-, Migrations- und Testkosten (*SoftCalc*)

Das Programm SoftCalc unterstützt die Projektarten Prototyp, Entwicklung, Evolution, Sanierung, Migration, Integration und Test und unterscheidet die Produkttypen Standard-PC-System, integriertes, verteiltes, eingebettetes und Internet-System. SoftCalc integriert acht bekannte Schätzmethoden (s. Kapitel 2.3):

- (1) Function-Point
- (2) Data-Point
- (3) UseCase-Point
- (4) COCOMO I
- (5) COCOMO II
- (6) Error-Projection
- (7) Object-Point
- (8) Test-Point

Laut Harry M. Sneed sollte zur Erhöhung der Schätzgenauigkeit die Aufwandschätzung mit mindestens drei verschiedenen Schätzmethoden erfolgen, um ein Zielintervall mit Ober- und Untergrenze zu entwickeln.

Dem Programm SoftCalc liegt ein algorithmischer Schätzansatz zugrunde und theoretisch basieren die Aufwandschätzungen auf dem sogenannten Teufelsquadrat, das mit nachstehender Abbildung veranschaulicht wird:

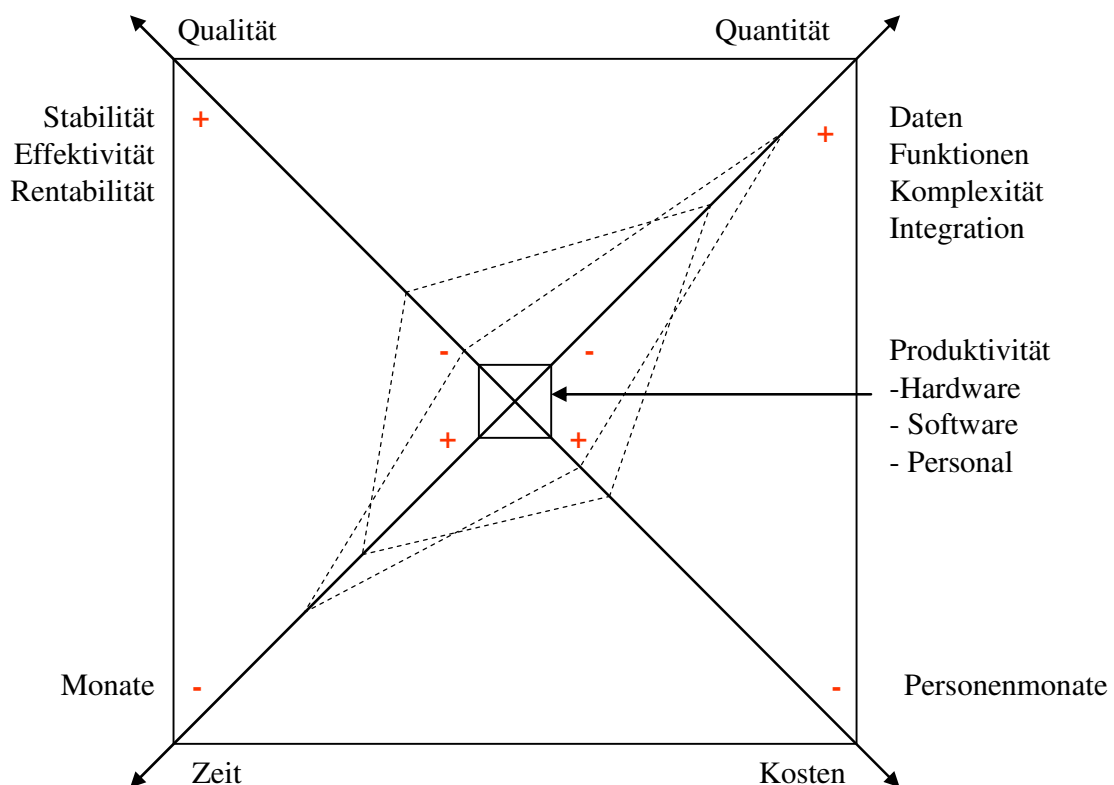


Abb. 7: Teufelsquadrat nach Sneed [Sneed 05]

Die Abbildung stellt die wichtigsten Eckpunkte einer Aufwandschätzung dar und veranschaulicht folgendes Prinzip:

Wird z.B. nur die Komplexität und damit der Umfang des Software-Produkts erhöht, geht dies zu Lasten der Qualität, die sich dadurch mindert. Wird hingegen die Qualität erhöht, steigen die Kosten. Es dürfen nur zwei Faktoren gleichzeitig verändert werden, um das Verhältnis der Faktoren zueinander nicht zu Lasten eines einzigen Faktors zu verzerren.

Die Berechnungen basieren auf der Formel:

$$\text{Aufwand} = \frac{\text{Qualität} * \text{Quantität}}{\text{Produktivität}}$$

Wenn Quantität und Qualität vorgegeben werden, folgen Aufwand und Dauer aus dem Verhältnis von Quantität und Qualität zur Produktivität (*Vorwärtskalkulation*). Wenn hingegen Aufwand und Dauer vorgegeben werden, folgen Quantität und Qualität aus dem Verhältnis von Aufwand und Dauer zur Produktivität (*Rückwärtskalkulation*). Dabei bildet nach Harry M. Sneed die Produktivität die einzige Konstante und steht im Mittelpunkt der Betrachtungen. Sie stellt die Quantität eines Mitarbeiters pro Zeiteinheit dar.

Die Aufwandschätzungen des Software-Werkzeugs SoftCalc basieren auf der geschätzten Größe des Software-Produkts und der Systemqualität, wobei Erfahrungen aus bereits abgeschlossenen Projekten gleicher Art berücksichtigt werden.

Aufgabe des Programms ist es, aus den Größen-, Komplexitäts-, Qualitäts- und Produktivitäts-Maßen die voraussichtlichen Zeit- und Kostenanforderungen eines Projekts zu ermitteln. Diese Software-Maße repräsentieren die Objekte, Anwendungsfälle, Schnittstellen, Komponenten und Testfälle des Produkts bzw. Projekts. Aus ihnen werden folgende Größen berechnet:

- der minimal benötigte Aufwand,
- die minimal benötigte Zeit,
- die optimale Teamgröße,
- die zu erwartende Fehleranzahl (Restfehlerwahrscheinlichkeit) und
- der jährliche Wartungsaufwand.

Die Größe des Software-Produkts als Ausgangsbasis der Aufwandschätzung mit SoftCalc wird anhand der gewählten Schätzmethode ermittelt. Diese Größe wird mit den spezifischen Einflussfaktoren der jeweiligen Methode und folgenden zusätzlichen Faktoren des Tools SoftCalc justiert:

- 1.) Änderungsrate
- 2.) Qualitätsfaktor
- 3.) Produktivitätsfaktor
- 4.) Ressourcenfaktor
- 5.) Risikofaktor.

## 4.2 Benötigte Eingabedaten

Die Aufwandschätzung mit dem Programm SoftCalc basiert je nach Vorhandensein auf der Analyse der Anforderungsspezifikation, Entwurfsdokumentation, Quellcodeanalyse und Testdokumentation.

Die Daten dieser Analysen werden mit einer relationalen Datenbank abgebildet. Sie besteht aus 12 Tabellen und bildet das Kernstück des Programms SoftCalc.

Bevor eine Aufwandschätzung mit dem Programm möglich ist, sind die Datenbanktabellen mit entsprechenden Werten zu füllen. Die Eingaben können manuell über die Benutzeroberfläche des Programms erfolgen. In einige Tabellen können Daten über eine XML-Systemimportschnittstelle importiert werden, was mit Abbildung 8 veranschaulicht wird:

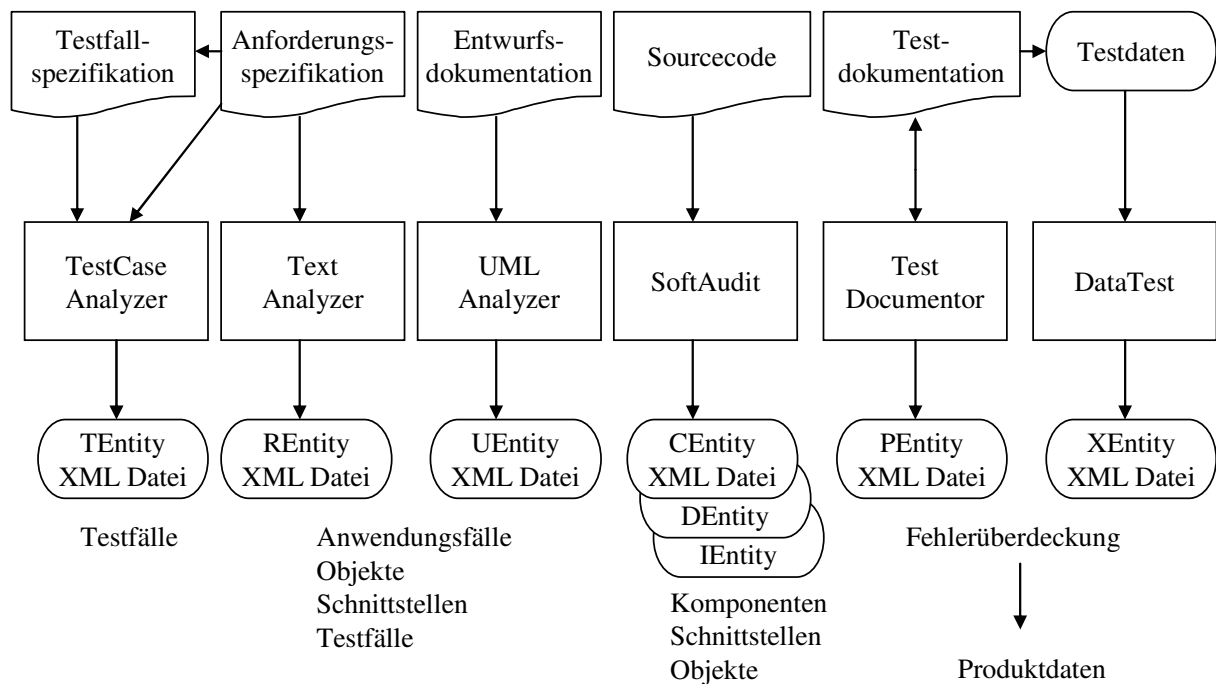


Abb. 8: Datenimporte in SoftCalc [Versionsbeschreibung S. 53]

In der Abbildung wird auch ersichtlich, mit welchen Software-Werkzeugen, die jeweiligen XML-Dateien für die Importschnittstelle erzeugt werden können. Die XML-Importdateien aus Abbildung 8 werden in Anhang A detailliert beschrieben.

### 4.3 Datenmodell und Architektur des Programms SoftCalc

Das Programm SoftCalc wurde mit einer Delphi-Entwicklungsumgebung implementiert. Delphi ist Derivat von Object Pascal.

Die Implementierung basiert auf dem in Abbildung 9 veranschaulichten Datenmodell:

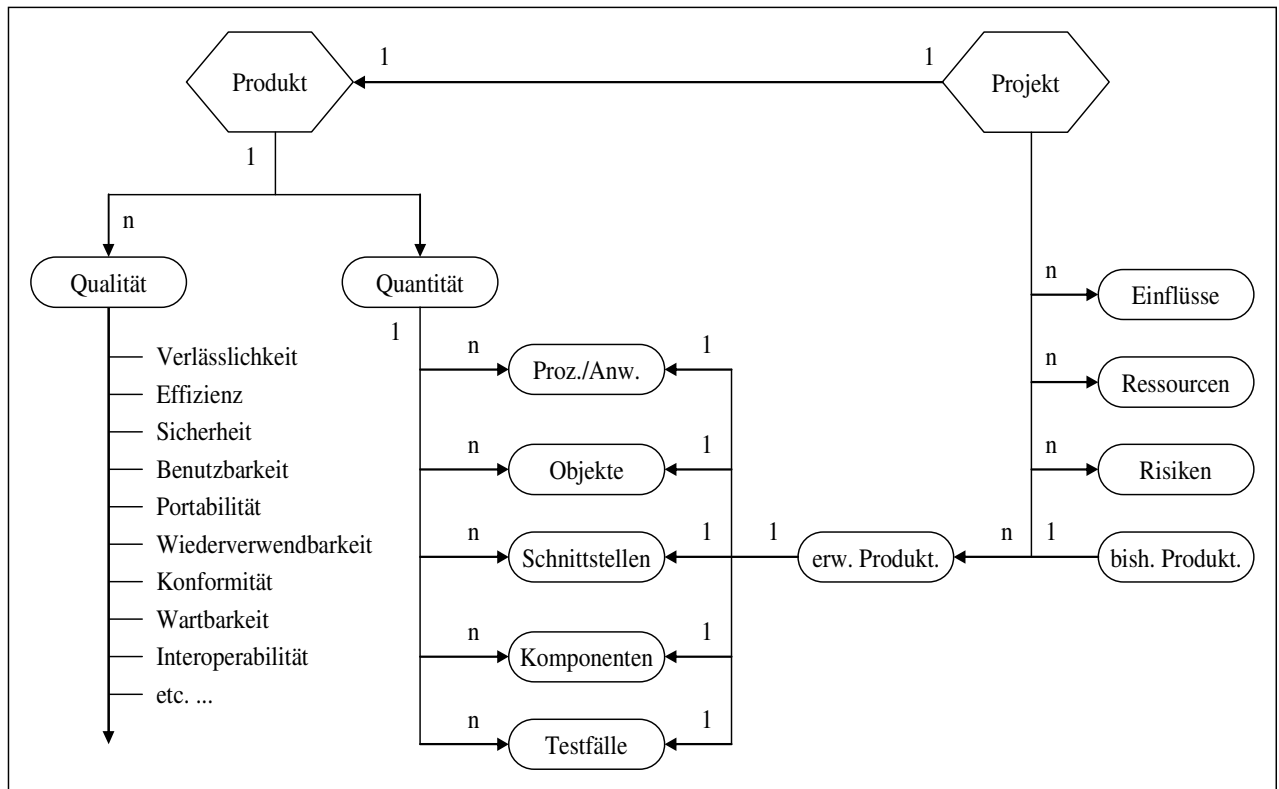


Abb. 9: SoftCalc-Datenmodell [Versionsbeschreibung S. 12]

Grundlage der Architektur des Programms SoftCalc ist eine relationale Datenbank, die wie im vorhergehenden Kapitel bereits erwähnt wurde, aus 12 Tabellen besteht.

Mit diesen Tabellen sollen folgende Objekte bzw. Merkmale abgebildet werden:

Objektbezeichnung	Beschreibung...
Produkt	... des Produktes (Projektzieles)
Projekt	... der Projektbedingungen
Objekt	... der Objekte des Zielsystems
Schnittstellen	... der Schnittstellen des Zielsystems
Anwendungsfälle	... der Geschäftstransaktionen im System
Komponenten	... der existierenden Software-Komponenten
Testfall	... der Testfälle für das Zielsystem
Qualitätsanforderungen	... der Qualitätskriterien, die das Zielsystem erfüllen soll
Produktivitätswerte	... der Produktivitätsdaten bereits abgeschlossener Projekte

Ressourcen	... der Eigenschaften, wie Verfügbarkeit, der Ressourcen Hardware, Software, Personal
Einflussfaktoren	... der speziellen Einflussfaktoren der einzelnen Aufwandschätzmethoden
Risiken	... der möglichen Projektrisiken

Tab. 16: Übersicht über die in den Datenbanktabellen abgebildeten Objekte

Die einzelnen Attribute der Datenbanktabellen werden in Anhang B erläutert.

#### 4.4 Ablauf einer Aufwandschätzung

Im folgenden Kapitel wird der Ablauf einer Aufwandschätzung mit dem Programm SoftCalc aus Anwendersicht beschrieben:

##### 1.) *Projekt definieren*

Zur Definition eines neuen Projektes sind die folgenden Schritte erforderlich:

- Eingabe des Projekt- und Produktnamens
- Auswahl des Produkttyps (Standalone-PC, Integrated, Distributed, Internet, Embedded)
- Auswahl des Projekttyps (Prototype, Development, Evolution, Migration, Integration)
- Auswahl der Projekttechnologie (Original, Procedural, Relational, Object-oriented, Web-based)
- Eingabe eines Kurznamens des Produkts
- Eingabe der Obergrenze Zeit in Personenmonaten ( $0 \geq \text{Monate} \leq 9999$ )
- Eingabe des max. möglichen Aufwands (PMS) ( $0 \geq \text{PMS} \leq 9999$ )
- Eingabe der Änderungsrate der Anforderungen (Requirement Change Rate) ( $0.00 \geq \text{RCR} \leq 1.00$ )

##### 2.) *Import der Produktivitätstabelle*

Eine Aufwandschätzung ist mit dem Programm SoftCalc nur möglich, wenn Produktivitätsdaten vergangener Projekte vorhanden sind. Diese sollten im XML-Format importiert werden, können jedoch auch manuell über die Benutzeroberfläche eingegeben werden. Sind keine Daten aus bereits abgeschlossenen Projekten vorhanden, kann ersatzweise die Produktivitätstabelle des im Lieferumfang enthaltenen Beispielprojekts „Auftragsbearbeitung“ genutzt werden.

### 3.) *Import der Systemmetriken*

Für die Vorwärts- und Rückwärtskalkulation sind die verschiedenen XML-Dateien (vgl. Abbildung 8) mit Anwendungsfällen, Datenobjekten, Systemschnittstellen und Testfällen zu importieren. Aus diesen werden die benötigten Systemmetriken bzw. -maße abgeleitet.

### 4.) *Anpassen der Systemmetriken*

Statt eines Imports ist auch die manuelle Eingabe bzw. die Anpassung folgender Werte möglich:

- Prozesse oder Anwendungsfälle
- Datenobjekte
- Systemschnittstellen
- Komponenten
- Testfälle

### 5.) *Definition der Systemqualität*

In SoftCalc wird eine Menge benutzerspezifischer Qualitätsmerkmale angeboten, die auf ISO-9126 basieren. Der Anwender kann seine erwartete Rate zu jedem Qualitätsmerkmal angeben. Die Summe dieser Merkmale wird zur Berechnung des Qualitätsfaktors verwendet.

### 6.) *Ermittlung der Projekteinflüsse*

Jede in SoftCalc mögliche Aufwandschätzmethode berücksichtigt eigene Einflussfaktoren (siehe Kapitel 2.3). Nach Auswahl der Schätzmethode sind die jeweils relevanten Einflussfaktoren durch den Anwender zu kalibrieren und den projekt- und unternehmensspezifischen Gegebenheiten anzupassen. Andernfalls werden für die Einflussfaktoren der jeweiligen Schätzmethode Standardwerte verwendet. Die Berücksichtigung der einzelnen Faktoren wird im Kapitel 4.7 beschrieben.

### 7.) *Anpassung der Projektressourcen*

Ressourcen variieren in ihrer Verlässlichkeit, Verfügbarkeit und Produktivität. Diesem Umstand soll mit der Berücksichtigung eines entsprechenden Einflussfaktors Rechnung getragen werden. Zur Ermittlung dieses Einflussfaktors sind durch den Anwender die folgenden Ressourcen zu bewerten:

- Software-Ressourcen
- Hardware-Ressourcen
- Personal-Ressourcen.

Die Bewertung ist optional, da die Verwendung des Ressourcenfaktors nicht zwingend erforderlich ist.

#### 8.) *Analyse der Projektrisiken*

Durch den Anwender können Projektrisiken erfasst werden, wobei es Sache des Anwenders ist, die Risiken zu identifizieren und die Auswirkung auf den Aufwand des Projekts einzuschätzen. Wie beim Ressourcenfaktor handelt es sich um einen optionalen Einflussfaktor.

Mit Schritt 8 hat der Anwender sämtliche Voraussetzungen für eine Aufwandschätzung mit dem Programm SoftCalc erfüllt.

An dieser Stelle ist zu erwähnen das das Programm SoftCalc in drei Modi *manuell*, *halbautomatisch* und *automatisch* benutzbar ist.

Der Modus wird durch die Projektvoraussetzungen in Bezug auf vorhandene Dokumentationen und der Art des Projekts bestimmt. Beispiele für Art des Projekts sind Neuentwicklung (manueller oder halbautomatischer Modus) oder Migrationsprojekt (automatischer Modus).

Der manuelle Modus kommt zum Einsatz wenn kein Import von Eingabedaten möglich ist, da es an entsprechenden Spezifikationen, Dokumentationen und/oder Analysetools mangelt. In diesem Fall sind sämtliche Daten über die Benutzeroberfläche des Programms SoftCalc einzugeben. Im halbautomatischen Modus werden Teile der benötigten Eingabedaten über die XML-Systemschnittstelle importiert. Im automatischen Modus sind die wenigsten manuellen Eingaben erforderlich.

## **4.5 Das SoftCalc Schätzverfahren**

Das Software-Messwerkzeug SoftCalc berechnet Aufwand und daraus resultierend Kosten, Dauer, Personalbedarf und den jährlichen Wartungsaufwand für jede ausgewählte Schätzmethode in folgenden zehn Schritten:

- 1.) Basierend auf der Methode (COCOMO I, COCOMO II, Function-Point, Object-Point, Test-Point, Error-Projection, UseCase-Point oder Data-Point) wird der Umfang des Software-Produkts berechnet und mit dem Qualitätsfaktor justiert. Ergebnis dieses Schrittes ist die *qualitätsjustierte Größe des Software-Produkts*.
- 2.) Die ermittelte qualitätsjustierte Größe wird an die Änderungsrate der Anforderungen angepasst. Es wird davon ausgegangen, dass sich die Anforderungen während eines Projekts um 1 bis 50% erhöhen. Diese Änderungsrate wird prozentual zu der qualitätsjustierten Größe hinzugerechnet.
- 3.) Aus der qualitätsjustierten und an die Änderungsrate der Anforderungen angepassten Größe wird in Abhängigkeit von der gewählten Schätzmethode der *Aufwand in Personenmonaten* abgeleitet.
- 4.) Die ermittelte Aufwandsgröße aus Schritt 3 wird mit den *spezifischen Einflussfaktoren der jeweiligen Schätzmethode* angepasst.
- 5.) Das Ergebnis aus Schritt 4 wird mit dem *Ressourcenfaktor* multipliziert. Diesem Faktor liegen die individuellen Ressourcen (Personal, Hardware und Software) zugrunde. Für jede Ressource wird ein individueller Ressourcenfaktor ermittelt, der als Produkt aus der relativen Produktivität, seiner Verlässlichkeit und Verfügbarkeit gebildet wird. Der Gesamtfaktor ist der arithmetische Mittelwert der individuellen Ressourcenfaktoren.
- 6.) Die Aufwandsgröße aus Schritt 5 wird an den *Risikofaktor* angepasst. Die individuellen Risikofaktoren sind jeweils das Produkt aus Risikoauswirkung, Eintrittswahrscheinlichkeit und (1 – Risikoreduzierung).
- 7.) In diesem Schritt wird der *minimale Zeitbedarf* des Projekts berechnet:

$$\text{Zeit} = \text{Systemtyp} * (\text{Aufwandsgröße aus Schritt 6})^{\text{Exponent}}$$

Systemtyp:

Standalone Systeme	= 2.0
Integrierte Systeme	= 2.4
Verteilte Systeme	= 2.8
Eingebettete Systeme	= 3.2

Der Exponent variiert programmintern zwischen 0.35 und 0.43 abhängig vom unjustierten Aufwand.

8.) Für die Höhe der *minimal zu erwartenden Kosten* werden die realen Kosten eines Projekts ähnlichen Aufwands aus der Produktivitätstabelle angenommen.

9.) Die Berechnung der *optimalen Teamgröße* erfolgt gemäß der Formel:

$$\text{Minimalaufwand} / \text{Minimalzeit} + 1$$

10.) Der *jährliche Wartungsaufwand* wird aus Änderungsrate und Produktqualität berechnet:

$$\text{Wartungsaufwand} = 1.2 * (\text{Aufwand} * \text{jährliche Änderungsrate}) * (2 - \text{Qualitätsfaktor})$$

#### 4.6 Ermittlung des Produktumfangs und des Aufwands in SoftCalc

Basis einer Aufwandschätzung ist immer der Umfang bzw. die Größe des zu entwickelnden Software-Produkts. Die Ermittlung ist abhängig von den produkt- und projektspezifischen Gegebenheiten und unterscheidet sich von Aufwandschätzmethode zu Aufwandschätzmethode.

In Tabelle 17 wird die jeweilige Berechnung des Produktumfangs und die daraus resultierende Ableitung des Aufwands im Programm SoftCalc kurz veranschaulicht:

Aufwandschätzmethode	Ermittlung Umfang	Ableitung Aufwand
COCOMO I	<ul style="list-style-type: none"> <li>- wenn Quellcode vorhanden:</li> <li>- <math>\text{KDSI}^5 = (\text{Anweisungen} * \text{Komponentenänderungsrate}) / 1.000</math></li> <li>- andernfalls Zählung im Systemmodell:</li> <li>- <math>\text{KDSI} = (\text{Anweisungen} * \text{Systemänderungsrate}) / 1.000</math></li> <li>- in beiden Fällen abschließende Justierung mit dem Qualitätsfaktor</li> </ul>	$\text{Systemtyp} * (\text{KDSI})^{**1.05}$ mit Systemtyp: 2.0 = Standalone-PC 2.4 = integriertes System 2.8 = verteiltes System 3.0 = web-basiertes System 4.0 = eingebettetes Echtzeitsystem
COCOMO II	<ul style="list-style-type: none"> <li>- Ermittlung aus KDSI, Function-Point, Data-Point, Object-Point, UseCase-Point, Test-Point (Backfiring)</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	$\text{Systemtyp} * (\text{Größe} / \text{Produktivität})^{\text{Skalierungsexponent}}$ mit: - Systemtyp s. COCOMO I - Skalierungsexponent ist

<sup>5</sup> KDSI = 1.000 (K) delivered source instructions

		Produkt der 5 Skalierungsfaktoren COCOMO II
Function-Point	<ul style="list-style-type: none"> <li>- Zählung der Function-Points in Objekt-, Schnittstellen- und Anwendungsfalltabellen unter Berücksichtigung der Änderungsrate</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	Suche anhand der ermittelten Function-Points identisches Projekt in Produktivitätstabelle (ggf. Interpolation)
Data-Point	<ul style="list-style-type: none"> <li>- Zählung Data-Points in Objekt- und Schnittstellentabelle unter Berücksichtigung Änderungsrate</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	Wie bei Function-Point
Object-Point	<ul style="list-style-type: none"> <li>- Zählung Object-Points in Objekt-, Schnittstellen- und Anwendungsfalltabellen unter Berücksichtigung der Wiederverwendungsrate</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	Wie bei Function-Point
UseCase-Point	<ul style="list-style-type: none"> <li>- Zählung UseCase-Points in Schnittstellen- und Anwendungsfalltabellen unter Berücksichtigung der Änderungsrate</li> <li>- Justierung mit techn. Einflussfaktor</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	Wie bei Function-Point
Test-Point	<ul style="list-style-type: none"> <li>- Zählung Test-Points in Testfalltabelle</li> <li>- Justierung mit dem Qualitätsfaktor</li> </ul>	Wie bei Function-Point
Error-Projection	<ul style="list-style-type: none"> <li>- Restfehler = (Überdeckungsrate + 1) * Anzahl gefundener Fehler - Anzahl gefundener Fehler * 1.2</li> <li>- Justierung mit Restfehler * (Komplexität / 0.5) * (0.5 / Qualität)</li> </ul>	Wie bei Function-Point

Tab. 17: Ermittlung der Größe des Software-Produkts und Ableitung des Aufwands

Die Tabelle verdeutlicht inwieweit die Aufwandschätzmethoden durch das Programm SoftCalc benutzt werden.

#### 4.7 Die Einflussfaktoren in SoftCalc

In SoftCalc werden methodeneigene Einflussfaktoren der jeweiligen Aufwandschätzmethoden verwendet. Einzige Ausnahme bildet die Methode Error-Projection, die über keine eigenen Einflussfaktoren verfügt.

Die in SoftCalc berücksichtigten Faktoren der Schätzmethoden und ihre empirische Bedeutung werden im Anhang C dieser Arbeit detaillierter beschrieben.

Tabelle 18 stellt hinsichtlich der Einflussfaktoren einen Bezug zwischen Aufwandschätzmethoden im Original und ihrer Verwendung in SoftCalc her:

Aufwand-schätzmethode	Anzahl Faktoren (EF)		Berechnung des Gesamteinflussfaktors	
	in Soft-Calc	lt. Original	in SoftCalc	lt. Original
Function-Point	14 <sup>6</sup>	14	$0.65 + (0.01 * \sum_{i:14}^{EF})$	$0.65 + (0.01 * \sum_{i:14}^{EF})$
COCOMO I	20	15	$\prod_{i:20}^{EF}$	$\prod_{i:20}^{EF}$
COCOMO II	5	5	$\frac{1}{5} \sum_{i=1}^5$	$\prod_{i:5}^{EF}$
Object-Point	10	10	$1.25 - (0.01 * \sum_{i:10}^{EF})$	$1 - (0.01 * \sum_{i:10}^{EF})$
Data-Point	10	10	$1.25 - (0.01 * \sum_{i:10}^{EF})$	$\frac{1}{10} \sum_{i=1}^{10}$
UseCase-Point	21	21	$1.24 - (0.01 * \sum_{i:21}^{EF})$	$0.65 + (0.01 * \sum_{i:8}^{TF} \text{ } ^7)$ $1.4 + (-0.03 * \sum_{i:13}^{ETF} \text{ } ^8)$
Test-Point	8	-	$1.20 - (0.01 * \sum_{i:8}^{EF})$	-
Error-Projection	-	-	-	-

Tab. 18: Einflussfaktoren im Original und in SoftCalc

Zur weiteren Optimierung der Schätzergebnisse werden im Programm SoftCalc zusätzlich zu den methodeneigenen Einflussfaktoren die in Kapitel 4.5 genannten spezifischen Faktoren verwendet:

### 1.) Qualitätsfaktor

Dieser Faktor ergibt sich aus den Daten, die in der Qualitätstabelle enthalten sind. Basis der Ermittlung sind 7 Qualitätsmerkmale und 3 Qualitätstypen des Softwaresystems.

Durch den Schätzer sind nachstehende Merkmale zu bewerten:

- Zuverlässigkeit (*Reliability*)

<sup>6</sup> Fünf Einflussfaktoren wurden durch Faktoren ersetzt, die nicht IFPUG 4.1 entsprechen.

<sup>7</sup> Environmental Total Factor

<sup>8</sup> Technical Total Factor

- Bedienbarkeit (*Usability*)
- Effizienz (*Efficiency*)
- Wartbarkeit (*Maintainability*)
- Portabilität (*Portability*)
- Wiederverwendbarkeit (*Reusability*)
- Sicherheit (*Security*).

Es werden drei Qualitätstypen unterschieden:

- Anwendung (*Usage*)
- Wartung (*Maintenance*) und
- Wiederverwendung (*Reuse*).

Die Höhe des Qualitätsfaktors ist abhängig von der geplanten Qualität und wird aus der Differenz zwischen geplantem Wert und Mittelwert berechnet. Der Faktor hat eine Bandbreite von 0.5 bis 1.5, das heißt er beeinflusst den Aufwand um bis zu  $\pm 50\%$ .

Wenn die geplante Qualität (*Planned\_Score*) ist als der Durchschnittswert (*Median\_Score*), berechnet sich der Faktor mit:

$$\text{Qualitätsfaktor} = 1 + \frac{(\text{Planned\_Score} - \text{Median\_Score})}{\text{Distance}} * 0.5$$

mit  $\text{Distance} = 1 - \text{Median\_Score}$

Sollte die geplante Qualität (*Planned\_Score*) hingegen kleiner sein als der Durchschnittswert (*Median\_Score*) ermittelt sich der Faktor folgendermaßen:

$$\text{Qualitätsfaktor} = 1 - \frac{(\text{Median\_Score} - \text{Planned\_Score})}{\text{Distance}} * 0.5$$

mit  $\text{Distance} = \text{Median\_Score} - 0.5$

Wenn die geplante Qualität (*Planned\_Score*) dem Durchschnittswert (*Median\_Score*) entspricht, wird der Qualitätsfaktor mit folgender Vorschrift ermittelt:

$$\text{Qualitätsfaktor} = \frac{\text{Weigth}}{\frac{\text{Weigth\_Sum}}{\text{Nr\_Risks}}} * \text{Adjustement\_Factor}$$

mit Adjustment Factor = 1

## 2.) *Risikofaktor*

Der Faktor ergibt sich aus der Risikotabelle. Die möglichen Risiken eines Projekts müssen dabei vom Schätzer selbst identifiziert und klassifiziert werden. Grundsätzlich wird zwischen vier verschiedenen Risikogruppen unterschieden:

- organisatorischen Risiken
- technischen Risiken
- Anforderungsrisiken
- Personalrisiken

Der jeweilige Risikofaktor berechnet sich mit:

$$\text{Risikofaktor} = \text{Risikoauswirkung} * \text{Eintrittswahrscheinlichkeit} * (1 - \text{Risikoreduzierung})$$

Der gesamte Risikofaktor wird aus dem arithmetischen Mittelwert der einzelnen Risikofaktoren plus 1 ermittelt.

## 3.) *Produktivitätsfaktor*

Ohne diesen Einflussfaktor ist eine Schätzung mit dem Programm SoftCalc nicht möglich, da ohne Berücksichtigung der Produktivität keine zuverlässigen Werte ermittelt werden könnten. Für jedes bereits abgeschlossene Projekt müssen Aufwand, Kosten und Zeit erfasst werden. Zudem ist die Eingabe des verwendeten Software-Maßes erforderlich:

- Zeilen
- Statements
- Function-Points
- Data-Points
- Object-Points
- UseCase-Points
- Test-Points
- Errors

Es sollten zu jeder Projektart mindestens drei Projekte existieren, da andernfalls kein Intervall für die Ableitung des Aufwands gebildet werden kann.

## 4.) *Ressourcenfaktor*

Der Ressourcenfaktor ist wie der Risikofaktor optional und kann bei der Aufwandschätzung vernachlässigt werden, in dem eine entsprechende Option über die SoftCalc-Benutzeroberfläche eingegeben wird. Wie beim Risikofaktor obliegt es dem Schätzer zu bestimmen, inwieweit die folgenden Ressourcen einen speziellen Einfluss auf das Projekt haben:

- Person
- Hardware
- Software

Der einzelne Ressourcenfaktor wird berechnet mit:

$$\text{Ressourcenfaktor} = \text{Verlässlichkeit} * \text{Verfügbarkeit} * \text{relative Produktivität}$$

Dadurch wird im Ergebnis die durchschnittliche Produktivität, die sich aus bereits abgeschlossenen Projekten ergibt, an die aktuellen Projektgegebenheiten angepasst.

## 4.8 Die SoftCalc Benutzeroberfläche

Das folgende Kapitel widmet sich der Beschreibung der Benutzeroberfläche. Das Programm besitzt eine umfassende und anpassbare graphische Benutzeroberfläche, die sich in vier horizontale Abschnitte gliedert. Zudem werden am unteren Bildrand Fehlermeldungen als rot unterlegte und Hinweise als blau unterlegte Statuszeile ausgegeben.

Zur Veranschaulichung dient nachstehende Abbildung:

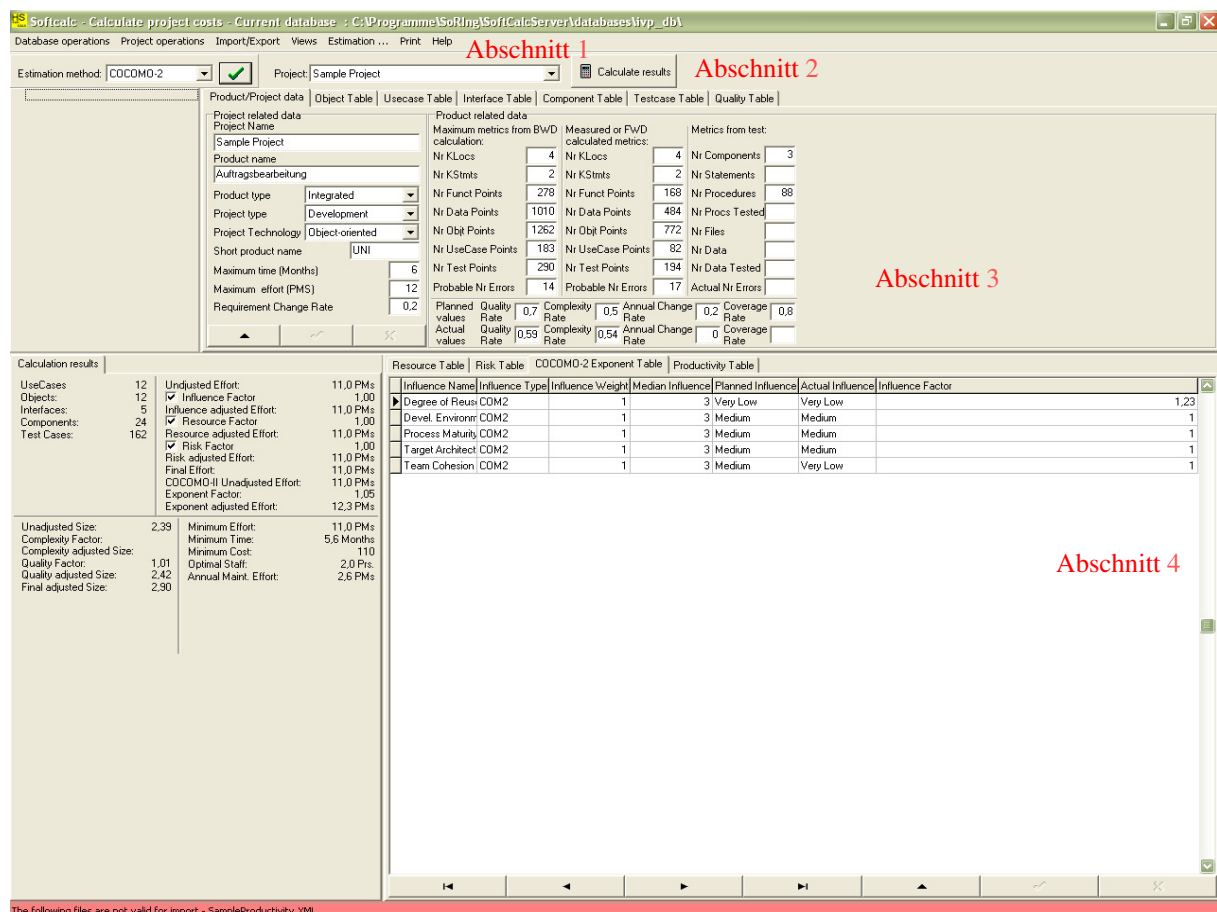


Abb. 10: Graphische Benutzeroberfläche SoftCalc

Abschnitt 1 besteht aus einer Menüzeile, die folgende Funktionen beinhaltet:

- *Database operations*, z.B. Erzeugen einer neuen Datenbank
- *Project operations*, z.B. Wechsel zu einem existierenden Projekt
- *Import/Export*, z.B. Export der Produktivitätsdaten in eine XML-Datei
- *Views*, z.B. Ansicht aller Tabellen
- *Estimation...*, z.B. Aktivierung oder Deaktivierung des Risikofaktors

- *Print*, z.B. Wahl der Darstellung der Druckergebnisse
- *Help*, z.B. Produktinfo SoftCalc

Im Abschnitt 2 befindet sich eine Statuszeile mit der ausgewählten Schätzmethode, dem Projekt- bzw. dem Produktnamen und einem Button zum Starten der Kalkulation.

Der Abschnitt 3 besteht aus einer Ansicht, welche, je nach der am oberen Rand ausgewählten Registerkarte, folgende Inhalte anzeigt:

- Produkt-/Projektdatei,
- Objektabelle,
- Anwendungsfalltabelle,
- Schnittstellentabelle,
- Komponententabelle,
- Testfalltabelle,
- Qualitätstabelle.

Die Ansicht Produkt- und Projektdatei unterteilt sich in zwei Bereiche:

Links werden die projektbezogenen Daten wie Name des Projekts und Projekttyp angezeigt. Die rechte Seite stellt die produktbezogenen Daten dar und besteht aus drei Spalten. Die erste Spalte enthält die Software-Maße (z.B. Anzahl der Function-Points) der Rückwärtskalkulation, die zweite Spalte die Maße der Vorwärtskalkulation und die dritte Spalte die Ergebnisse aus dynamischem Test und Analyse (z.B. Anzahl der Komponenten und Dateien).

Im Abschnitt 4 werden links die Ergebnisse der Kalkulation angezeigt. Rechts befindet sich eine Ansicht die je nach gewählter Registerkarte eine der folgenden Tabellen darstellt:

- Produkteinflusstabelle,
- Projekteinflusstabelle,
- Ressourcentabelle,
- Risikotabelle,
- Produktivitätstabelle

## **4.9 Erzeugte Ausgabedaten**

Die Ergebnisse der Aufwandschätzung werden in der Benutzeroberfläche angezeigt und für eine Druckausgabe aufbereitet. Zudem werden zwei XML-Dateien für den Export aus Soft-Calc erzeugt. Es handelt sich dabei um die Produktivitäts- und Projektdaten aus der Datenbank.

## 5 Die qualitative Bewertung des Programms SoftCalc

### 5.1 Messbeispiele

Das Programm SoftCalc in der Version 1.1 wurde inklusive Beispieldatenbank und Versionsbeschreibung von Harry M. Sneed zur Verfügung gestellt.

Mit dem Beispiel „Auftragsbearbeitung“ und den Fallstudien aus [Sneed 05] wurde die Funktionsweise des Programms gegen die Versionsbeschreibung SoftCalc (Version 1.1) geprüft.

Mit nachstehender Tabelle werden die Ergebnisse der Aufwandschätzung des Beispielprojekts „Auftragsbearbeitung“ mit dem Messtool SoftCalc veranschaulicht:

Schätzmethode	Ausgangsgröße	Aufwand in PM	Dauer in Monaten	Kosten	Teamgröße in Personen	Ø jährl. Wartungsaufwand in PM
COCOMO	4 KLOC	6,0	4,5	60	1,3	1,4
COCOMO II	4 KLOC	11,0	5,6	110	2,0	2,6
Data-Point	1010 DP	6,4	4,6	64	1,4	1,5
ErrorProjection	-	-	-	-	-	-
Function-Point	278 FP	9,6	5,3	96	1,8	2,3
Object-Point	772 OP	8,2	5,0	82	1,6	1,9
UseCase-Point	183 UCP	6,7	4,7	67	1,4	1,6
Test-Point	290 TP	9,4	5,3	94	1,8	2,2

Tab. 19: Ergebnisse der Aufwandschätzung des Beispielprojekts „Auftragsbearbeitung“

Bei den in Tabelle 19 dargestellten Aufwandschätzungen wurden Risiko-, Ressourcen- und Einflussfaktor jeweils mit 1.0 berücksichtigt. Der Qualitätsfaktor wurde mit einem Wert von 1.01 zugrunde gelegt und der Komplexitätsfaktor wurde nicht einbezogen.

Im Folgenden wird als repräsentatives Messbeispiel hinsichtlich der erfolgten Analyse des Programms SoftCalc eine Fallstudie aus [Sneed 05] detailliert dargestellt:

➤ *Fallstudie zur Schätzung einer Neuentwicklung*

Ziel der Schätzung war die Ermittlung des Aufwands und der sich daraus ergebenden Größen für die Neuentwicklung eines Datenerfassungssystems. Die Implementierung erfolgte in COBOL.

Aus der Projektspezifikation ergaben sich:

- 11 R2-Datenbanktabellen mit 216 Datenattributen und 34 Beziehungen
- 2 Masken mit 31 Feldern
- 32 Berichte mit 480 Feldern
- 63 Importdateien mit 1.179 Datenattributen
- 10 Batchprozesse
- 85 Batchvorgänge
- 2 Dialogvorgänge
- 1.335 Funktionen

Tatsächlich wurde das Projekt mit einem Aufwand von 128 Personenmonaten realisiert.

Laut den dokumentierten Daten der Fallstudie wurden zum Zeitpunkt der damaligen Schätzung folgende Aufwandskennzahlen ermittelt:

COCOMO I           = 94 Personenmonate

Function-Point     = 102 Personenmonate

Data-Point         = 105 Personenmonate

Das Schätzergebnis wich im Mittel um 28% vom tatsächlich benötigten Aufwand ab. Laut Harry M. Sneed ist diese Abweichung mit der Nichtberücksichtigung von Einarbeitungs- und Reisezeiten zu erklären.

An dieser Stelle werden die Eingabedaten der Fallstudie und ihre Verwendung im Programm SoftCalc hinsichtlich der Methode COCOMO I und der Function-Point-Methode ausführlich beschrieben:

## 1.) *COCOMO I*

Durch Analyse der verschiedenen Dokumentationen und Spezifikationen wurden 75.042 Anweisungen gezählt. Aufgrund günstiger technischer Bedingungen wurde der Einflussfaktor mit 0.88 kalibriert. Die Qualitätsziele wurden unterdurchschnittlich bewertet, so dass ein Qualitätsfaktor von 0.96 berücksichtigt wurde. Desweiteren wurde ein Systemschwierigkeitsgrad von 1.2 in die Berechnungen einbezogen. Der Schwierigkeitsgrad wurde unter Berücksichtigung eines geschätzten Produktivitätswertes ermittelt. Der Aufwand in Personenmonaten wurde mit folgender Vorschrift berechnet:

$$\text{Aufwand} = 1.2(75)^{1,05} * 0.96 * 0.88 = 94$$

Die Eingaben der Daten zur Fallstudie der Methode COCOMO I erfolgte manuell über die Benutzeroberfläche des Programms SoftCalc:

### ➤ Bestimmung des Produktumfangs:

Die Ermittlung des Umfangs bzw. der Größe basiert, wenn Quellcode vorhanden ist, auf der *Component Table*. Andernfalls werden die Anweisungen aus dem Inhalt der *Interface Table*, *Object Table* und *UseCase Table* ermittelt.

Es werden in den Tabellen die Einträge zu folgenden Attributen mit der in Klammern angegebenen Gewichtung berücksichtigt:

- Component Table: Statements (1)
- Interface Table: Arguments (2), Results (3)
- Object Table: #Keys (4), #Attributes (2)
- UseCase Table: Inputs (30, 40 oder 60<sup>9</sup>), Outputs (40, 60 oder 80)

Der Eintrag wird mit seiner Änderungsrate (*Change Rate*), die einen Wert zwischen 0 und 1 annehmen kann, multipliziert und mit diesem Ergebnis berücksichtigt.

An dieser Stelle ist zu erwähnen, dass bei manueller Erzeugung einer neuen Tabellenzeile die Änderungsrate mit einem Wert von 0 vorbelegt ist. Damit der Eintrag in die Zählung des Pro-

---

<sup>9</sup> Der Gewichtungsfaktor hängt von der Projekttechnologie ab.

duktumfangs einfließt, ist er entsprechend zu korrigieren (Standard = 1). Dieses Problem trat im manuellen Modus in jeder Tabelle auf.

➤ Berücksichtigung der Einflussfaktoren:

Das Produkt der in SoftCalc verwendeten 20 Einflussfaktoren stellt den Gesamteinflussfaktor der Methoden COCOMO I und II (*Influence Factor*) dar. Die einzelnen Werte variieren auf einer Bandbreite von 0.75 (very high), 0.90 (high), 1.00 (medium), 1.10 (low) zu 1.25 (very low) und entsprechen nicht den Werten der Originalvorschriften. Damit wird die empirische Bedeutung des Einzelfaktors im Gesamtprodukt aufgeweicht.

➤ Berücksichtigung der Qualitätsanforderungen:

Der Qualitätsfaktor stellt das arithmetische Mittel der 7 Qualitätskriterien der *Quality Table* dar. Mit diesem Faktor werden die Qualitätsanforderungen des zu entwickelnden Software-Produkts in der Schätzung berücksichtigt, z. B. mindert eine geringere Anforderung an die Wiederverwendbarkeit eines Software-Produkts den Aufwand entsprechend.

➤ Ergebnis der Schätzung mit dem Programm SoftCalc:

Durch das Programm wurden aufgrund der Eingabedaten der Fallstudie mit folgende Schätzergebnisse ermittelt:

Systemtyp	Aufwand in PM	justierter Aufwand in PM	Zeit in Monaten	Kosten in €	optimale Teamgröße in Personen	Ø jährl. Wartungs-aufwand in PM
Standalone-PC	186.3	163.7	15.4	15.846,-	10.7	0
Integrated	223.5	196.4	19.8	19.015,-	9.9	0
Distributed	260.8	229,2	29.0	22.184,-	7.9	0
Internet	279.4	245.5	32.0	23.769,-	7.7	0
Embedded	372.5	327.4	48.3	31.692,-	6.8	0

Tab. 20: Schätzergebnisse der Nachkalkulation zu COCOMO I

Nach der COCOMO 81-Originalvorschrift mit Aufwand =  $a * (KLOC)^b * c_i$  ergibt sich je Modell folgender *einflussjustierter Aufwand* in Personenmonaten (PM):

- Organic:  $3.2 * (75.042 * 0.001)^{1.05} * 0.88 = 262.24$
- Semi-detached  $3.0 * (75.042 * 0.001)^{1.12} * 0.88 = 332.62$
- Embedded  $2.8 * (75.042 * 0.001)^{1.20} * 0.88 = 438.54$

Die Berechnung der *Entwicklungszeit* in Monaten mit der Formel  $\text{TIME}_{\text{dev}}=2.5(\text{Aufwand})^d$  stellt sich wie folgt dar:

- Organic:  $2.5 * (262.24)^{0.38} = 20.75$
- Semi-detached  $2.5 * (332.62)^{0.35} = 19.08$
- Embedded  $2.5 * (438.54)^{0.32} = 17.51$

➤ Fazit:

Das Projekt wurde tatsächlich mit einem Aufwand von 128 Personenmonaten realisiert. Die hohe Schätzabweichung zu dem SoftCalc-Ergebnis ist Resultat der geringen Eingabedatenmenge. Mit SoftCalc werden die projekt- und produktspezifischen Gegebenheiten über eine größere Menge von Einflussfaktoren abgebildet. Das Ergebnis untermauert die Feststellung Harry M. Sneeds hinsichtlich der erforderlichen Schätzvoraussetzungen. Je ungenauer die Schätzgrundlagen umso größer die Schätzabweichungen.

Die Abweichung hinsichtlich der Originalvorschrift verdeutlicht, weshalb die Methode CO-COMO 81 durch COCOMO II abgelöst wurde. Sie entspricht in ihrer Urform nicht mehr den heutigen Anforderungen.

## 2.) *Function-Point-Methode*

Es wurden 1.739 ungewichtete Function-Points gezählt. Die Summe der Einflussfaktoren ergab 26. Der Wertkorrekturfaktor VAF betrug somit 0.91 ( $0.65 + (0.01 * 26)$ ). Durch die Anpassung der ungewichteten Function-Points an den Korrekturfaktor wurden 1.583 gewichtete Function-Points ermittelt. Der daraus resultierende Aufwand von 102 Personenmonaten wurde aus der IBM-Erfahrungskurve abgeleitet.

Die Eingaben der Fallstudie zur Function-Point-Methode wurden manuell über die Benutzeroberfläche des Programms SoftCalc erfasst:

➤ Bestimmung des Produktumfangs:

Dem Programm SoftCalc liegen die Zählregeln der IFPUG 4.1 zugrunde. Die Zählung der ungewichteten Function-Points erfolgt über *Object Table*, *Interface Table* und *UseCase Table*. In der *Object Table* werden die Elementzeilen zu den Objekttypen *Data Entity*, *Data View* und *Master File* anhand der Anzahl ihrer Attribute (*#Attributes*) gewichtet und gezählt.

In der *Interface Table* werden alle Einträge zum Objekttyp *File* hinsichtlich der Anzahl seiner Argumente (*Arguments*) betrachtet. In der *UseCase Table* werden die Attribute *Inputs* und *Outputs* abhängig von der Größe ihres Attributs *#Steps* berücksichtigt. Die einzelnen Gewichtungen werden in Tabelle 21 dargestellt:

Tabelle	Objekttyp	Anzahl der ...	Relation	Gewichtung
Object Table	Data Entity, Data View	Attribute	< 16	7
			16 – 30	10
			> 30	15
	Master File		< 16	5
			16 – 30	7
			> 30	10
Interface Table	File	Argumente	< 15	5
			16 – 30	7
			> 30	10
UseCase Table	Batch, Online, Client/server, Internet, System, Embedded	Schritte	≥ 3	Inputs = 3 Outputs = 4
			4 – 7	Inputs = 4 Outputs = 5
			> 7	Inputs = 6 Outputs = 7

Tab. 21: Gewichtung der Function-Points im Messtool SoftCalc

➤ Berücksichtigung der Einflussfaktoren:

Von den verwendeten 14 Einflussfaktoren stimmen 5 nicht mit den Einzelfaktoren der Function-Point-Methode gemäß IFPUG 4.1 überein.

Die Faktoren:

- verteilte Verarbeitung (*Distributed Data Processing*)
- begrenzte Kapazität (*Heavily Used Configuration*)
- Online-Dateneingabe (*Online Data Entry*)
- Betriebsanforderungen (*Operational Ease*)

wurden durch die Faktoren:

- Systemtyp (*System Type*)
- Mehrfachmandat (*Multi Mandate*)
- Produktzuverlässigkeit (*Product Reliability*)
- Geschäftskritikalität (*Business criticality*) und
- Systemsicherheit (*System Adatability*)

ersetzt, um nach Aussage Harry M. Sneeds veränderten Bedingungen gerecht zu werden.

Die Kalibrierung der Einflussfaktoren, erfolgt über die Benutzeroberfläche des Programms. An dieser Stelle fehlt eine detaillierte Beschreibung der empirischen Bedeutung der einzelnen Faktoren. Im Programm SoftCalc wurden rund 100 Einflussfaktoren verankert und ihr Sinn erschließt sich nicht intuitiv.

Der benötigte Wertkorrekturfaktor VAF wird entsprechend der Originalvorschrift berechnet.

➤ Ergebnis der Schätzung mit dem Programm SoftCalc:

Durch das Programm wurden aufgrund der Eingabedaten der Fallstudie mit folgende Schätzergebnisse ermittelt:

Systemtyp	Aufwand in PM	just.Aufwand in PM	Zeit in Monaten	Kosten in €	optimale Teamgröße in Personen	Ø jährl. Wartungs-aufwand in PM
Standalone-PC	107.1	97.4	11.4	51.582.531,-	8.6	24.4
Integrated	107.1	97.4	13.7	51.582.831,-	7.1	
Distributed	107.1	97.4	16.0	51.582.831,-	6.1	24.4
Internet	107.1	97.4	17.1	51.582.831,-	5.7	24.4
Embedded	107.1	97.4	22.8	51.582.831,-	4.3	24.4

Tab. 22: Schätzergebnisse der Nachkalkulation zur Function-Point-Methode

➤ Fazit:

Die Schätzungenauigkeit dieses Beispiel ist auf den ersten Blick nicht so gravierend wie in dem Beispiel zur Aufwandschätzmethode COCOMO II. Ursache sind ähnliche Produktivitätswerte. Bei der Ableitung des Aufwands aus der Summe der gewichteten Function-Points wurde jedoch die Produktivitätstabelle des Beispielprojekts „Auftragsbearbeitung“ verwendet. Insofern wird auf das Fazit zur Methode COCOMO I verwiesen.

## 5.2 Auswertung

### 5.2.1 Prüfung gegen die Qualitätsanforderungen an Software-Produkte

In diesem Kapitel erfolgt eine Grobprüfung gegen die allgemeinen Qualitätsanforderungen, die für alle Software-Produkte gelten. Detailliertere Prüfungen sind in Kapitel 5.2.2 beschrieben.

#### Funktionalität:

Das Programm SoftCalc erfüllt die generellen funktionalen Anforderungen vollständig hinsichtlich Aufwandschätzung und Ermittlung von Kosten, benötigten Ressourcen und erforderlicher Zeit. Ein Datenimport über die angebotene Systemschnittstelle ist problemlos und erfolgt korrekt.

#### Effizienz:

Systemverhalten und Ressourcenverbrauch liegen absolut im Normalbereich eines Programms dieser Größenordnung, das Kriterium ist voll erfüllt.

#### Wartbarkeit:

Eine Prüfung erfolgte ausschließlich aus Anwendersicht, so dass dieses Kriterium nicht beurteilt werden kann.

#### Übertragbarkeit:

SoftCalc lässt sich ohne Schwierigkeiten und fehlerfrei installieren. Da es sich um ein Standalone handelt, sind die Punkte Ersetzbarkeit und Kompatibilität nicht relevant. Damit ist dieses Kriterium voll erfüllt.

#### Zuverlässigkeit:

Das Programm läuft grundsätzlich stabil. Während des Tests ist es allerdings einmal zu einem Absturz gekommen. Dieser Fehler ließ sich nicht reproduzieren und wird daher bei der Beurteilung vernachlässigt. Die Qualitätsanforderung ist erfüllt.

#### Benutzbarkeit:

Der Lernaufwand ist relativ hoch. Das Programm erschließt sich nicht intuitiv und ist für einen unerfahrenen Anwender unverständlich. Die deutsche Dokumentation ist zum Teil widersprüchlich oder schwer verständlich, da Begrifflichkeiten nicht genau abgegrenzt werden. Abhängig vom Kontext werden für dieselben Sachverhalte unterschiedliche Begriffe benutzt bzw. werden unterschiedliche Sachverhalte mit denselben Begriffen umschrieben. Selbst ein schrittweises Vorgehen entsprechend der Versionsbeschreibung führt ohne Kenntnis von internen Programmzusammenhängen nicht zu den beschriebenen Ergebnissen. Die Erfüllung der Qualitätsanforderung wird als mangelhaft eingestuft und sollte verbessert werden.

## 5.2.2 Prüfung gegen die Anforderungen an ein Software-Messwerkzeug

SoftCalc ist Teil eines Software-Messungsarbeitsplatzes. Durch dazugehörige Analysewerkzeuge wie z.B. SoftAudit werden aus Dokumentationen und Codeanalysen XML-Dateien erzeugt, die als Eingabedateien über eine Importschnittstelle in die SoftCalc-Datenbanktabellen eingelesen werden. Ein Einsatz SoftCalcs unabhängig vom Software-Messungsarbeitsplatz ist nicht empfehlenswert, da die Benutzung der vorhandenen Schnittstelle für andere CASE-Werkzeuge nicht ohne Zusatzaufwand geeignet ist.

Sind alle Voraussetzungen hinsichtlich der erforderlichen Eingabedaten erfüllt, werden durch das Programm SoftCalc Aufwand, Kosten, Personal- und Zeitbedarf, sowie der jährliche Wartungsaufwand geschätzt. Die Ergebnisse werden textuell und etwas unübersichtlich in der Benutzeroberfläche angezeigt. Lediglich die Produktivität wird als Funktion aus Kosten und Produktivitätseinheiten in der Benutzeroberfläche dargestellt. Eine weitere Verwendung üblicher Diagrammformen erfolgt nicht, weder auf der Benutzeroberfläche noch in Form einer Druckausgabe. Es besteht die Möglichkeit, die Produktivitäts- und Projektdaten zu exportieren, allerdings ohne anschließende Toolunterstützung.

Das Programm SoftCalc bietet die Möglichkeit überdurchschnittlich genaue Schätzergebnisse zu erzielen, vorausgesetzt es wird die Empfehlung eingehalten mindestens 3 verschiedene Aufwandschätzmethoden in die jeweiligen Betrachtungen einzubeziehen. Dadurch kann ein Intervall bestimmt werden und gravierende Abweichungen zwischen den Methoden können analysiert und abgefangen werden. Die mit dieser Vorgehensweise erzielte breitgefächerte Schnittmenge von Einflussfaktoren führt zu einem genaueren Abbild der Projektrealität und einer gewissen Stabilität des Schätzergebnisses bei einzelnen Änderungen von Eingabeparametern.

Eine Verwendung des Programms in frühen Projektphasen bei Neuentwicklungsprojekten ist nur im manuellen Modus möglich. Damit ist die qualitative Anforderung der Frühzeitigkeit lediglich eingeschränkt erfüllt. Es sei denn, es sind genügend Daten aus vergleichbaren abgeschlossenen Projekten und Dokumentationen vorhanden (halbautomatischer Modus).

Ein Schätzverfahren sollte Eindeutigkeit gewährleisten, d.h. bei gleichartiger Anwendung durch unterschiedliche Personen muss ein annähernd gleiches Ergebnis für den Aufwand ermittelt werden. Dieser wichtige Ansatz konnte von mir als Einzelperson nicht untersucht wer-

den, aber die Eindeutigkeit ist für die Akzeptanz eines Software-Werkzeugs enorm wichtig und sollte weitergehend untersucht werden.

Die in SoftCalc berücksichtigten Einflussfaktoren lassen sich quantitativ und qualitativ beurteilen. Eine Schätzungenauigkeit kann sich durch den optionalen Gebrauch der Risiko- und Ressourcenfaktoren ergeben. Praxiserfahrungen zeigen deutlich, dass in jedem Projekt mehr oder wenig viele Risiken und individuelle Gegebenheiten besonders in Bezug auf die Ressource Personal vorhanden sind, die häufig Ursache für Terminverschiebungen oder Kostensteigerung sind. Nichtbetrachtung verfälscht das Schätzergebnis und nimmt dem Projektmanagement eine Chance präventiv zu handeln.

Laut Beschreibung ist mit SoftCalc eine Aufwandschätzung mit acht verschiedenen Methoden möglich. Allerdings erfolgte eine Modifizierung der Originalvorschriften. Die methodeneigenen Einflussfaktoren wurden unter Berücksichtigung der Praxiserfahrung Harry M. Sneeds variiert. Dies und die speziellen Einflussfaktoren von SoftCalc erhöhen die Komplexität der jeweiligen Schätzmethode.

Beispiel für eine Modifizierung ist die Aufwandschätzmethode COCOMO II. In SoftCalc wird bei der Ableitung des Aufwands aus dem Umfang des Software-Produkts der Systemtyp berücksichtigt. Im Original nach Dr. Boehm werden keine Systemtypen unterschieden. Zudem werden statt der bekannten 17 Kosten- und 5 Skalierungsfaktoren des Modells Post-Architecture 20 Kosten- und 5 Skalierungsfaktoren kalibriert. In der Summe sind max. 22 Einflussfaktoren möglich und nicht 25. Zudem hat jeder dieser Faktoren eine eigene Bandbreite, die empirisch ermittelt wurde. In SoftCalc wird jeweils eine einheitliche Bandbreite für alle Faktoren verwendet. Unterschieden nach Kosten- und Skalierungsfaktoren.

Die in der Dokumentation enthaltene Beschreibung zur Kalibrierung der Einflussfaktoren ist unvollständig und teilweise fehlerhaft. Die kurzen Stichpunkte genügen nicht für eine entsprechende Einordnung. Allein die Verwendung der Schlagwörter kann nicht zu einer ordnungsgemäßen Kalibrierung der Faktoren im Sinne des Autors führen. Durch Fehlinterpretationen sind nicht unerhebliche Schätzabweichungen möglich. Auch wenn unterstellt wird, dass das Programm SoftCalc nur durch erfahrene Anwender genutzt wird, sollte nicht unterschätzt werden, dass Begriffe durch verschiedene Sichtweisen eine andere Bedeutung bekommen und demzufolge kontextabhängig sind. Der Autor verwendet Einflussfaktoren, die entweder auf

seinen eigenen Erfahrungen beruhen oder ohne nähere Betrachtungen keine Entsprechung in den Originalvorschriften finden.

### **5.2.3 Fazit**

Der große Vorteil von SoftCalc besteht in der hohen Schätzgenauigkeit. Diese beruht zum Einen auf der Verwendung mehrerer Schätzansätze und zum Anderen auf Modifizierungen, die Resultat der langjährigen Praxiserfahrungen Harry M. Sneeds sind. Die hohe Schätzgenauigkeit geht jedoch zu Lasten der Benutzbarkeit aufgrund einer stark gesteigerten Komplexität. Das schränkt die allgemeine Akzeptanz und damit den Nutzerkreis ein. Einem unerfahrenen Anwender kann das Programm in der Form nicht empfohlen werden.

Hinsichtlich der benötigten Eingabedaten ist eine Schätzung mit dem Programm SoftCalc ohne umfassende Systemanalyse nicht möglich und kann nicht im Sinne des Autors eingesetzt werden. Ersatzweise können die Schätzungen auf den Produktivitätswerten der mitgelieferten Fallstudie „Auftragsbearbeitung“ basieren. Eine Berechnung kann ohne die erforderlichen Datenbanktabellen nicht erfolgen, da ohne Berücksichtigung der unternehmens-, produkt- und projektspezifischen Gegebenheiten die Schätzungen in jedem Fall falsch sind. Problematisch ist meines Erachtens, die Forderung nach drei ähnlich gelagerten Projekten, die bereits in der Vergangenheit umgesetzt wurden. Dies ist aus praktischer Sicht unrealistisch, da oftmals Neuland betreten wird, s. Reengineering- oder Migrationsprojekte und auf keine historischen Daten zurückgegriffen werden kann. Eine Betrachtung eines Praxisbeispiels meines Arbeitsumfeldes scheiterte am Umfang der Eingabedaten, die von SoftCalc vorausgesetzt werden. Mangels Schätzkultur und daraus resultierend fehlenden Daten aus bestehenden Projekten war eine Aufwandschätzung mit SoftCalc nicht möglich.

## 6 Zusammenfassung und Ausblick

Im Ergebnis werden mit dieser Diplomarbeit die besonderen Qualitätsanforderungen an ein Software-Messwerkzeug im Sinne der Aufwandschätzung herausgearbeitet. Eine Prüfung und Bewertung dieser Anforderungen wurde am Beispiel des Software-Messwerkzeugs SoftCalc dargestellt.

Vereinfacht betrachtet wird der Aufwand aus zwei Größen ermittelt, aus dem Produktumfang und der Produktivität. Beides variiert allein schon durch den Faktor Mensch stark. Jeder Entwickler setzt eine Aufgabe anders um, was zu unterschiedlichen Produktumfängen in Bezug auf LOC führt, ohne das sich aus dieser Tatsache allein die Qualität der entstandenen Software ableiten lässt. Die Produktivität lässt sich noch weniger absolut messen, denn entscheidend ist nicht die Menge des entwickelten Programmcodes, sondern die Menge in der geforderten Qualität.

Absolut genaue Zahlen wie bei einer maschinellen oder automatisierten Fertigung in der Industrie können nie ermittelt werden. Dies lässt sich auch deutlich am Grad der Standardisierung ablesen. Es gibt bis auf die Zählung der ungewichteten Function-Points keine Standardisierung im Bereich der Aufwandschätzung. Das Maßsystem unterliegt stärkerer Weiterentwicklung als in anderen Bereichen.

Es existiert eine beachtliche Anzahl an Methoden und Werkzeugen zur Aufwandsschätzung. Fast jede Methode federt die sogenannten „weichen Kriterien“ die der Faktor Mensch mit sich bringt, mit einer Vielzahl von Einflussfaktoren ab. Die Einflussfaktoren variieren abhängig von der realisierten Aufgabe und den Entwicklungsbedingungen. Die Notwendigkeit möglichst genaue Schätzungen als Planungsgrundlage zu erhalten, war eine Ursache für die Entstehung der unterschiedlichen Schätzmethode, denn jede berücksichtigt bestimmte Konstellationen der Softwareentwicklung. Im passenden Kontext benutzt können heute gute Schätzergebnisse erzielt werden. Die Schwierigkeit liegt mitunter darin im zur Verfügung stehenden Zeitrahmen die passende Methode nebst Werkzeug zu finden. Hierfür ist die Vorgehensweise von SoftCalc ein guter Ansatz. Mit einem Werkzeug bieten sich aufgrund der verschiedenen implementierten Schätzansätze und der Variationsmöglichkeiten vielschichtige Einsatzmöglichkeiten. Mit überarbeiteter Dokumentation und damit verbesserter Benutzerfreundlichkeit

des Tools sollten die beschriebenen Nachteile aufgrund der hohen Komplexität soweit abgedeckt werden, dass einer breiteren Akzeptanz nichts im Wege steht.

Aufgrund der in dieser Arbeit durchgeführten Analyse und Bewertung ist abschließend festzustellen, dass das Messtool SoftCalc die qualitativen Anforderungen an ein Software-Messwerkzeug im Sinne der Aufwandschätzung weitestgehend erfüllt.

## Abbildungsverzeichnis

Abb. 1: Niveaustufen einer Schätzung	10
Abb. 2: Festlegen der Systemgrenze	13
Abb. 3: Data-Point-Ableitung von Harry M. Sneed [Sneed 05]	28
Abb. 4: verkürzte Darstellung des CMM-Modells nach Arthur aus [Dumke 96]	37
Abb. 5: Qualitätskriterien nach ISO 9126 aus [Dumke 96]	39
Abb. 6: Allgemeinste Schritte einer Softwaresstrategie aus [Dumke 00]	41
Abb. 7: Teufelsquadrat nach Sneed [Sneed 05]	45
Abb. 8: Datenimporte in SoftCalc [Versionsbeschreibung S. 53]	47
Abb. 9: SoftCalc-Datenmodell [Versionsbeschreibung S. 12]	48
Abb. 10: Graphische Benutzeroberfläche SoftCalc	59

## Tabellenverzeichnis

Tab. 1: Prozentuale Kostenverteilung in konventionellen Software-Projekten	10
Tab 2: Funktionstypen und ihre Einordnung nach IFPUG 4.1	14
Tab. 3: Ermittlung der gewichteten Function-Points nach Zähltypen	16
Tab. 4: Kostenfaktoren des Modells COCOMO 81	18
Tab. 5: Kalibrierung der Einflussfaktoren des Modells COCOMO 81	19
Tab. 6: Beispiel einer Kalibrierung eines Produktattributs	21
Tab. 7: Skalierungs- und Kostenfaktoren des Modells COCOMO II	21
Tab. 8: Kalibrierung der Skalierungsfaktoren $SF_{1-5}$ mit COCOMO II.2000	22
Tab. 9: Kalibrierung der Kostenfaktoren $EM_{1-17}$ mit COCOMO II.2000	22
Tab. 10: Symbolbeschreibung COCOMO II-Aufwandschätzung	23
Tab. 11: Symbolbeschreibung COCOMO II-Entwicklungszeitschätzung	23
Tab. 12: Symbolbeschreibung COCOMO II-Umfangschätzung	24
Tab. 13: Einflussfaktoren der Data-Point-Methode [Sneed 05]	29
Tab. 14: Technische Komplexitätsfaktoren der UseCase-Point-Methode	32
Tab. 15: Umgebungsfaktoren der UseCase-Point-Methode	33
Tab. 16: Übersicht über die in den Datenbanktabellen abgebildeten Merkmale	49
Tab. 17: Ermittlung der Größe des Software-Produkts und Ableitung des Aufwands	54
Tab. 18: Einflussfaktoren im Original und in SoftCalc	55
Tab. 19: Ergebnisse der Aufwandschätzung des Beispielprojekts „Auftragsbearbeitung“	62
Tab. 20: Schätzergebnisse der Nachkalkulation zu COCOMO I	65
Tab. 21: Gewichtung der Function-Points im Messtool SoftCalc	67
Tab. 22: Schätzergebnisse der Nachkalkulation zur Function-Point-Methode	68

## Literaturverzeichnis

- [BuFab 00] Manfred Bundschuh, Axel Fabry: Aufwandschätzung von IT-Projekten, 1. Auflage, MITP-Verlag GmbH Bonn, 2000
- [Dumke 92] Reiner Dumke: Softwareentwicklung nach Maß, Schätzen, Messen, Bewerten, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH Braunschweig/Wiesbaden, 1992
- [Dumke 96] R. Dumke, E. Foltin, R. Koepe, A. Winkler: Softwarequalität durch Messtools – Assessment, Messung und instrumentierte ISO 9000, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1996
- [Dumke 00] Reiner Dumke, Franz Lehner (Hrsg.): Software-Metriken: Entwicklungen, Werkzeuge und Anwendungsverfahren, Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH Wiesbaden, 2000
- [Dumke 01] Reiner Dumke: Software Engineering, 3. Auflage, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH Braunschweig/Wiesbaden, 2001
- [EbDu 96] Christof Ebert, Reiner Dumke: Software-Metriken in der Praxis - Einführung und Anwendung von Software-Metriken in der industriellen Praxis, Springer-Verlag Berlin Heidelberg 1996
- [EbDu 05] C. Ebert, R. Dumke, M. Bundschuh, A. Schmietendorf: Best Practices in Software Measurement, Springer-Verlag Berlin Heidelberg, 2005
- [Knöll 91] Heinz-Dieter Knöll: Aufwandschätzung von Software-Projekten in der Praxis (Angewandte Informatik Bd. 8), Herausgeber Helmut Balzert, Bibliographisches Institut & F.A. Brockhaus AG Mannheim, 1991
- [KunDum 07] Martin Kunz, Reiner Dumke: Empirische Grundlagen zur COSMIC-FFP-Anwendung für die Aufwandschätzung, Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg, Preprint Nr. 7, 2007
- [Ligg 02] Peter Liggesmeyer: Software-Qualität, Testen, Analysieren und Verifizieren von Software, Spektrum Akademischer Verlag GmbH Heidelberg Berlin, 2002
- [Sneed 95.1] Harry M. Sneed: Schätzung der Entwicklungskosten von objektorientierter Software, Februar 1995

- [Sneed 95.2] Harry M. Sneed: Estimating the Development Costs of Object-Oriented Software, Februar 1995
- [Sneed 03] Harry M. Sneed: Testmetriken für die Kalkulation der Testkosten und die Bewertung der Testleistung, September 2003
- [Sneed 00] Harry M. Sneed: Aufwandsschätzung in der Softwarewartung, Proc. Of GI Software Management 2000, Österreichische Computer Gesellschaft, Marburg, November 2000
- [Sneed 05] Harry M. Sneed: Software-Projektkalkulation, Carl Hanser Verlag München Wien, 2005
- [Thaller 00] Georg Erwin Thaller: Software-Metriken einsetzen, bewerten, messen, 2. Auflage, HUSS-Medien GmbH, Verlag Technik Berlin, 2000
- Harry M. Sneed: Versionsbeschreibung SoftCalc Version 1.1, ANE CON Software Design und Beratung G.m.b.H., Mai 2007

## **Anhang A**

Tabellarische Beschreibung der XML-Eingabdateien des Messtools SoftCalc

XML-Datei	Grundlage	Inhalt	Import aus
<b>REntity</b>	Anforderungstexte	<ul style="list-style-type: none"> <li>▪ Anwendungsfälle</li> <li>▪ Objekte</li> <li>▪ Schnittstellen</li> <li>▪ Testfälle</li> </ul>	TextAnalyzer
<b>UEntity</b>	UML-Entwurfsdokumente	<ul style="list-style-type: none"> <li>▪ Anwendungsfälle</li> <li>▪ Objekte</li> <li>▪ Schnittstellen</li> </ul>	UMLAnalyzer
<b>DEntity</b>	Datenbankschema	<ul style="list-style-type: none"> <li>▪ Objektname</li> <li>▪ Objekttyp = Tabelle oder DB-Sicht</li> <li>▪ Anzahl Relationen</li> <li>▪ Anzahl Attribute</li> <li>▪ Anzahl Schlüssel</li> <li>▪ Anzahl Methoden</li> <li>▪ Änderungsrate</li> </ul>	DLI-Redoc, ADA-Redoc, SQL-Redoc, DLIAudit, ADAAudit oder SQLAudit
<b>CEntity</b>	Quellcode	<p><i>Modul- und Klassenentitäten:</i></p> <ul style="list-style-type: none"> <li>▪ Name</li> <li>▪ Typ</li> <li>▪ Anzahl Lines of Code</li> <li>▪ Anzahl Anweisungen</li> <li>▪ Anzahl Funktionen</li> <li>▪ Anzahl Variablen</li> <li>▪ Anzahl Attribute</li> </ul> <p><i>Schnittstellenentitäten:</i></p> <ul style="list-style-type: none"> <li>▪ Name</li> <li>▪ Typ</li> <li>▪ Anzahl Quellen, aus denen die Schnittstelle abgeleitet wurde</li> <li>▪ Anzahl Schnittstellenoperationen</li> <li>▪ Anzahl Argumente</li> <li>▪ Anzahl Ergebnisse</li> </ul>	SoftAudit oder SoftRedoc
<b>IEntity</b>	Schnittstellen	<ul style="list-style-type: none"> <li>▪ Quellen der Schnittstellendaten</li> <li>▪ Anzahl Schnittstellenoperationen</li> <li>▪ Anzahl Argumente</li> <li>▪ Anzahl Ergebnisse</li> <li>▪ Änderungsrate (default 0.0)</li> </ul>	SoftAudit
<b>TEntity</b>	Anforderungen bzw. Testfälle und Testobjekte	<ul style="list-style-type: none"> <li>▪ Testfallname</li> <li>▪ Testfalltyp</li> <li>▪ Anzahl Testfalleingabe (default 4)</li> <li>▪ Anzahl Testfallausgaben (default 2)</li> <li>▪ Anzahl getesteter Objekte (default 1)</li> <li>▪ Anzahl Testfallschritte (default 2)</li> <li>▪ Testfalländerungsrate (100 %)</li> </ul>	TestCase Analyzer

<b>XEntity</b>	Testdatengenerierung und -validierung	<ul style="list-style-type: none"> <li>▪ Anzahl verarbeiteter Tabellenzeilen</li> <li>▪ Anzahl inkorrektter Einträge</li> <li>▪ Anzahl Datenfelder definiert im Eintrag oder der Tabelle</li> <li>▪ Anzahl getesteter Datenfelder</li> <li>▪ Anzahl inkorrektter Datenfelder</li> <li>▪ Quotient der getesteten Felder (Datenüberdeckung)</li> </ul>	DataTest
<b>PEntity</b>	Testüberwachung	<ul style="list-style-type: none"> <li>▪ Systemname</li> <li>▪ Systemtyp</li> <li>▪ Anzahl getesteter Komponenten</li> <li>▪ Anzahl Code-Zeilen in den getesteten Komponenten</li> <li>▪ Anzahl Anweisungen in den getesteten Komponenten</li> <li>▪ Anzahl Methoden und Prozeduren in den getesteten Komponenten</li> <li>▪ Anzahl tatsächlich getesteter Methoden und Prozeduren</li> <li>▪ Anzahl entdeckter Fehler</li> <li>▪ Testüberdeckungsrate</li> </ul>	TestDocumentor

## **Anhang B**

Tabellarische Beschreibung der Datenbanktabellen des Messtools SoftCalc

Tabelle	Attribute	Inhalt
Projekttabelle	<ul style="list-style-type: none"> <li>▪ Projektname</li> <li>▪ Projekttyp</li> <li>▪ Projekttechnologie</li> <li>▪ maximale Zeit</li> <li>▪ maximaler Aufwand</li> <li>▪ Änderungsrate der Anforderungen</li> <li>▪ minimaler Aufwand</li> <li>▪ minimale Zeit</li> </ul>	Identifizierung und Beschreibung des zu schätzenden Projektes
Risikotabelle	<ul style="list-style-type: none"> <li>▪ Risikoname</li> <li>▪ Risikotyp</li> <li>▪ Gewicht des Risikos</li> <li>▪ Risikoeinwirkung</li> <li>▪ Risikowahrscheinlichkeit</li> <li>▪ Risikoreduzierung</li> <li>▪ Risikofaktor</li> </ul>	Projektrisiken
Objekttabelle	<ul style="list-style-type: none"> <li>▪ Objektname</li> <li>▪ Objekttyp</li> <li>▪ Anzahl Beziehungen zu anderen Objekttypen</li> <li>▪ Anzahl Methoden oder Operationen, die auf diesem Objekt ausgeführt werden</li> <li>▪ Anzahl von Schlüsseln zur Identifikation des Objekts</li> <li>▪ Anzahl von Attributen des Objekts</li> <li>▪ Objektveränderungsrate oder Wiederverwendbarkeit</li> </ul>	Beschreibung der Objekte des Zielsystems
Produkttable	<ul style="list-style-type: none"> <li>▪ Produktname</li> <li>▪ Produkttyp</li> <li>▪ jährliche Änderungsrate</li> <li>▪ Anzahl der Komponenten</li> <li>▪ Anzahl LOC</li> <li>▪ Anzahl Object-Points</li> <li>▪ Anzahl UseCase-Points</li> <li>▪ Anzahl Function-Points</li> <li>▪ Anzahl Fehler</li> <li>▪ Systemkomplexitätsrate</li> <li>▪ Systemqualitätsrate</li> <li>▪ Testüberdeckungsrate</li> </ul>	Identifizierung und Beschreibung der Produkte, die im Rahmen des Projektes entwickelt, migriert oder integriert werden sollen
Schnittstellentabelle	<ul style="list-style-type: none"> <li>▪ Schnittstellename</li> <li>▪ Schnittstellentyp</li> <li>▪ Anzahl Quellen, von denen die Schnittstelle bedient wird</li> <li>▪ Anzahl Operationen, die über die Schnittstelle laufen</li> <li>▪ Anzahl Argumente</li> <li>▪ Anzahl Ergebnisse</li> <li>▪ Schnittstellenänderungsrate bzw. Wiederverwendbarkeit</li> </ul>	Beschreibung der Schnittstellen des Zielsystems
Tabelle der Anwendungsfälle	<ul style="list-style-type: none"> <li>▪ Name des Anwendungsfalls</li> <li>▪ Typ des Anwendungsfalls</li> </ul>	Beschreibung der Geschäftstransaktionen im

	<ul style="list-style-type: none"> <li>▪ Anzahl alternativer Pfade durch den Anwendungsfall</li> <li>▪ Anzahl Schritte in allen Pfaden</li> <li>▪ Anzahl von Eingabedaten</li> <li>▪ Anzahl von Ausgabedaten</li> <li>▪ Änderungsrate der Transaktionen bzw. Wiederverwendbarkeit</li> </ul>	System
Komponententabelle	<ul style="list-style-type: none"> <li>▪ Komponentename</li> <li>▪ Komponententyp</li> <li>▪ Anzahl LOC</li> <li>▪ Anzahl Anweisungen</li> <li>▪ Anzahl Funktionen, Methoden und Prozeduren</li> <li>▪ Anzahl deklarierter Variablen</li> <li>▪ Änderungsrate der Komponenten bzw. Wiederverwendbarkeit</li> </ul>	Beschreibung der existierenden Software-Komponenten oder Source-Members des Zielsystems
Testfalltabelle	<ul style="list-style-type: none"> <li>▪ Testfallname</li> <li>▪ Testfallquelle</li> <li>▪ Anzahl Eingabevariablen</li> <li>▪ Anzahl Ausgabevariablen</li> <li>▪ Anzahl Objekte, die vom Testfall getestet werden</li> <li>▪ Anzahl Schritte in der Testfallausführung</li> <li>▪ Änderungsrate des Testfalls bzw. Wiederverwendbarkeit</li> </ul>	Beschreibung der Testfälle für das Zielsystem
Qualitätstabelle	<ul style="list-style-type: none"> <li>▪ Name des Qualitätsmerkmals</li> <li>▪ Qualitätstyp</li> <li>▪ Gewichtung dieses Merkmals</li> <li>▪ Mittelwert für dieses Merkmal</li> <li>▪ geplante Quelle für dieses Merkmal</li> <li>▪ berechneter Justierungsfaktor</li> </ul>	Qualitätscharakteristiken, die das Zielsystem erfüllen sollte
Produktivitätstabelle	<ul style="list-style-type: none"> <li>▪ Projektname</li> <li>▪ Projekttyp</li> <li>▪ Projekttechnologie</li> <li>▪ Größeneinheit</li> <li>▪ Anzahl Größeneinheiten</li> <li>▪ Projektaufwand in PM</li> <li>▪ Projektdauer in PM</li> <li>▪ Projektkosten in Währungseinheiten (Kilo)</li> </ul>	Daten der vergangenen Projekte
Ressourcentabelle	<ul style="list-style-type: none"> <li>▪ Ressourcenname</li> <li>▪ Ressourcentyp</li> <li>▪ Verlässlichkeit der Ressource</li> <li>▪ Verfügbarkeit der Ressource</li> <li>▪ relative Produktivität der Ressource</li> <li>▪ relative Kosten der Ressource</li> <li>▪ Justierungsfaktor für Ressourcen</li> </ul>	Daten bezüglich einzelner Projektquellen (Hardware, Software oder Menschen)
Tabelle der Einflussfaktoren	<ul style="list-style-type: none"> <li>▪ Name des Einflussfaktors</li> <li>▪ Typ des Einflussfaktors</li> <li>▪ Gewicht des Einflussfaktors</li> <li>▪ mittlerer Einfluss</li> </ul>	Einflussfaktoren der verschiedenen Aufwandsschätzmethoden

	<ul style="list-style-type: none"><li>▪ geplanter Einfluss</li><li>▪ aktueller Einfluss</li><li>▪ Einflussfaktor</li></ul>	
--	----------------------------------------------------------------------------------------------------------------------------	--

## **Anhang C**

Tabellarische Beschreibung der methodeneigenen Einflussfaktoren, die im Messtool SoftCalc benutzt werden

**C.1 Einflussfaktoren der Function-Point-Methode**

**C.2 Einflussfaktoren der COCOMO I und COCOMO II-Methode**

**C.3 Spezifische Einflussfaktoren COCOMO II-Methode**

**C.4 Einflussfaktoren der Test-Point-Methode**

**C.5 Einflussfaktoren der Data-Point-Methode**

**C.6 Einflussfaktoren der Object-Point-Methode**

**C.7 Einflussfaktoren der UseCase-Point-Methode**

## C.1 Einflussfaktoren der Function-Point-Methode

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
Data Communications	Datenkommunikation	0 = keine 1 = über ein Datenbanksystem 2 = über ein RPC 3 = über Middleware Framework 4 = über ein Message Queue 5 = über Internet
System Complexity	Zielsystemtypschwierigkeit	0 = Standalone-PC 1 = Netzwerk-PC 2 = standalone Mainframe-System 3 = ein Client/Server-System 4 = ein Intranet/Internet-System 5 = embedded Realtime-System
System Performance	Systemperfomanz	0 = keine Responsezeitgrenze 1 = Responsezeit unter 1 min. 2 = Responsezeit unter 10 sek. 3 = Responsezeit unter 4 sek. 4 = Responsezeit unter 2 sek. 5 = Responsezeit unter 1 sek.
Multi-Mandate	Mandantenfähigkeit	0 = ein Mandant mit einer Sprache 1 = mehrere Mandanten mit einer Sprache 2 = ein Mandant mit zwei Sprachen 3 = mehrere Mandanten mit zwei Sprachen 4 = ein Mandant in mehreren Sprachen 5 = mehrere Mandanten mit mehreren Sprachen
Transaction Rate	Transaktionsrate	0 = 0 Transaktionen pro Tag 1 = 1-1.000 Transaktionen pro Tag 2 = 1.000-10.000 Transaktionen pro Tag 3 = 10.000-100.000 Transaktionen pro Tag 4 = 100.000-1.000.000 Transaktionen pro Tag 5 = > 1.000.000 Transaktionen pro Tag
Reliability	Systemzuverlässigkeit	0 = > 1 Fehler pro Function-Point 1 = < 1 Fehler pro Function-Point 2 = < 0.1 Fehler pro Function-Point 3 = < 0.01 Fehler pro Function-Point 4 = < 0.001 Fehler pro Function-Point 5 = < 0.0001 Fehler pro Function-Point
Usability	Benutzerfreundlichkeit	0 = ein Anwender im Expertenmodus 1 = mehrere Anwender im Expertenmodus 2 = ein Anwender im Laienmodus 3 = mehrere Anwender im Laienmodus 4 = Anwender im Experten- und Laienmodus 5 = anwenderspezifische Oberflächen
Business Criticality	Geschäfts-	0 = Supportfunktion nicht geschäftswirksam

lity	relevanz	<p>1 = wichtige Supportfunktion</p> <p>2 = wichtige Sekundärfunktion</p> <p>3 = geschäftswirksame Sekundärfunktion</p> <p>4 = Geschäftsbereich-Hauptfunktion</p> <p>5 = Grundfunktion für mehrere Geschäftsbereiche</p>
Processing Complexity	Systemkomplexität	<p>0 = keine Komplexität (<math>\leq 0.2</math>)</p> <p>1 = geringe Komplexität (0.2–0.4)</p> <p>2 = niedrige Komplexität (0.4-0.5)</p> <p>3 = mittlere Komplexität (0.5-0.6)</p> <p>4 = hohe Komplexität (0.6-0.8)</p> <p>5 = sehr hohe Komplexität (<math>\geq 0.8</math>)</p>
Reusability	Wiederverwendbarkeit	<p>0 = keine Wiederverwendung (<math>\leq 0.2</math>)</p> <p>1 = geringer Wiederverwendungsgrad (0.2-0.4)</p> <p>2 = niedriger Wiederverwendungsgrad (0.4-0.5)</p> <p>3 = mittlerer Wiederverwendungsgrad (0.5-0.6)</p> <p>4 = hoher Wiederverwendungsgrad (0.6-0.8)</p> <p>5 = sehr hoher Wiederverwendungsgrad (<math>\geq 0.8</math>)</p>
Migration Difficulty	Migrationsnotwendigkeit	<p>0 = keine Migration erforderlich</p> <p>1 = Migration bestehender Dateien</p> <p>2 = Migration bestehender Datenbanken</p> <p>3 = Migration bestehender Programme</p> <p>4 = Migration bestehender Programme und Datenbanken</p> <p>5 = Migration bestehender Programme, Datenbanken und Kontrollprozeduren</p>
Security	Sicherheit	<p>0 = Datenschutzklasse 1 im Intranet</p> <p>1 = Datenschutzklasse 2 im Intranet</p> <p>2 = Datenschutzklasse 3 im Intranet</p> <p>3 = Datenschutzklasse 1 im Internet</p> <p>4 = Datenschutzklasse 2 im Internet</p> <p>5 = Datenschutzklasse 3 im Internet</p>
Installation Complexity	Installationskomplexität	<p>0 = eine Prototypversion</p> <p>1 = eine Version bei einem Anwender</p> <p>2 = eine Version bei mehreren Anwendern</p> <p>3 = mehrere Versionen bei einem Anwender</p> <p>4 = mehrere Versionen an mehreren Standorten</p> <p>5 = mehrere Versionen in mehreren Ländern</p>
Adaptability	Adaptierbarkeit	<p>0 = keine Änderung geplant (0%)</p> <p>1 = geringe Änderung erforderlich (&lt; 10%)</p> <p>2 = unterdurchschnittliche Änderung erforderlich (10-20 %)</p> <p>3 = durchschnittliche Änderung erforderlich (20-30%)</p> <p>4 = hohe Änderung erforderlich (30-50%)</p> <p>5 = komplette Reimplementierung erforderlich (&gt; 50%)</p>

## C.2 Einflussfaktoren der COCOMO I und COCOMO II-Methode

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
Complexity	Komplexität des Produkts	<p>Die Einflussfaktoren werden einheitlich auf folgender Skala gewichtet:</p> <p>0.75 = Einfluss sehr hoch            0.90 = Einfluss hoch            1.00 = Einfluss nominal            1.10 = Einfluss niedrig            1.25 = Einfluss sehr niedrig</p>
Data Volume	Datenvolumen des Produkts	
Documentation	Umfang der benötigten Produktdefinition	
Reusability	Wiederverwendbarkeit des benötigten Produkts	
Hardware	Anforderungen an die Hardware	
Software	Anforderungen an die Software	
Networking	Anforderungen an die Vernetzung	
Support	bereitzustellender Support	
Experience	Erfahrung der Projektmitarbeiter	
Knowledge	Ausbildungsstand der Projektmitarbeiter	
Skills	Fähigkeiten der Projektmitarbeiter	
Motivation	Motivation der Projektmitarbeiter	
Standard Process	verwendete Standardprozesse	
Unified Concept	einheitliches Produktkonzept	
Coding Standard	Programmierrichtlinien für das Produkt	
Test Automation	Grad der Testautomation	
Project Distribution	Grad der Projektverteilung	
Project Support	Managementunterstützung für das Projekt	
User Participation	Grad der Benutzereinbindung	
Project Migration	Qualität des Projektmanagements	

### C.3 Spezifische Einflussfaktoren der Methode COCOMO II

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
Degree of Reuse	Grad der Wiederverwendbarkeit	Die Einflussfaktoren werden einheitlich auf folgender Skala gewichtet:  0.91 = Einfluss sehr hoch 0.96 = Einfluss hoch 1.00 = Einfluss nominal 1.10 = Einfluss niedrig 1.23 = Einfluss sehr niedrig
Devel. Environment	Qualität der Entwicklungsumgebung	
Target Architecture	Komplexität der Zielumgebung	
Team Cohesion	Grad des Teamzusammenhalts	
Process Maturity	Beständigkeit des Entwicklungsprozesses	

## C.4 Einflussfaktoren der Test-Point-Methode

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
GUI-Testability	GUI-Testbarkeit	0 = ein Objekt pro GUI 1 = < 3 Objekte pro GUI 2 = 4-7 Objekte pro GUI 3 = > 7 Objekte pro GUI 4 = verlinkte GUI mit mehr als 7 Objekten
DB-Testability	Datenbanktestbarkeit	0 = 1 Attribut und 1 Schlüssel pro Datenbanktabelle 1 = 1 Schlüssel und mehrere Attribute pro Tabelle 2 = mehrere Attribute und mehrere Schlüssel pro Tabelle 3 = mehrere Attribute, Schlüssel und ein Link pro Tabelle 4 = mehrere Attribute, Schlüssel und Links pro Tabelle
Interface Testability	Schnittstellentestbarkeit	0 = 1 Parameter pro Schnittstelle 1 = mehrere Parameter pro Schnittstelle 2 = mehrere Parametergruppen pro Schnittstellen 3 = eine Nachricht pro Schnittstelle 4 = mehrere Nachrichten pro Schnittstelle
Process Testability	Prozesstestbarkeit	0 = Anwendungsfälle sind einzeln testbar 1 = Anwendungsfälle müssen in Reihe getestet werden 2 = Anwendungsfälle müssen parallel getestet werden 3 = Anwendungsfälle müssen asynchron getestet werden
Test Automation	Testautomationsgrad	0 = keine Testwerkzeuge 1 = einige Testaktivitäten automatisiert 2 = viele Testaktivitäten automatisiert 3 = alle Testaktivitäten 4 = alle automatisierten Testaktivitäten benutzen dieselbe Datenbank
Application Knowledge	Applikationswissen	0 = Tester haben kein Wissen über den Typ der Applikation 1 = Tester haben einiges Wissen über den Applikationstypen 2 = Tester sind mit dem Applikationstypen vertraut 3 = Tester sind mit dem Applikationstypen sehr vertraut
Test Team Experience	Testerfahrung	0 = Tester haben keine Testerfahrung 1 = Tester haben Testerfahrung 2 = Tester haben in vielen Projekten gearbeitet 3 = Tester haben über Jahre in vielen verschiedenen Projekten getestet
Test Process Maturity	Testprozessreife	0 = ad-hoc 1 = definiert 2 = wiederholbar 3 = automatisiert 4 = gemessen

## C.5 Einflussfaktoren der Data-Point-Methode

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
Project Concentration	Projektverteilung	0 = mehrere Länder 1 = mehrere Orte 2 = ein Ort 3 = mehrere Büros 4 = ein Büro
Team Experience	Erfahrung des Teams	0 = keine Erfahrung 1 = wenig Erfahrung 2 = durchschnittliche Erfahrung 3 = hohe Erfahrung 4 = exzellente Erfahrung
Team Knowledge	Expertise des Teams	0 = keine Erfahrung 1 = wenig Erfahrung 2 = durchschnittliche Erfahrung 3 = hohe Erfahrung 4 = exzellente Erfahrung
Tool Support	Werkzeugausstattung	0 = keine Werkzeuge 1 = wenige Werkzeuge 2 = durchschnittlich 3 = einige Werkzeuge 4 = viele Werkzeuge
User Support	Arbeitsbedingungen	0 = schlecht 1 = mäßig 2 = durchschnittlich 3 = gut 4 = sehr gut
Process Maturity	Grad der Prozessreife	0 = chaotisch 1 = wiederholbar 2 = definiert 3 = gemessen 4 = optimiert
Quality Assurance	Grad der Qualitätssicherung	0 = keine 1 = manuell und ungenügend 2 = manuell und genügend 3 = halbautomatisiert 4 = vollautomatisiert
Specification Formality	Formalisierungsgrad der Spezifikation	0 = informell 1 = strukturiert 2 = semiformal 3 = formal 4 = formal und graphisch
Program Language	Level der Programmiersprache	0 = 1. Sprachgeneration 1 = 2. Sprachgeneration 2 = 3. Sprachgeneration 3 = 4. Sprachgeneration

		4 = 5. Sprachgeneration
Test Automation	Level der Testautomatisierung	0 = 0% 1 = 0-25% 2 = 25-50% 3 = 50-75% 4 = 75-100%

## C.6 Einflussfaktoren der Object-Point-Methode

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
User Support	Grad der Benutzereinbindung	0 = keine 1 = etwas 2 = partiell 3 = durchschnittlich 4 = vollständig
Visual Programming	Nutzung visueller Programmierung	0 = keine 1 = etwas 2 = durchschnittlich 3 = hoch 4 = vollständig
Team Cohesion	Grad des Teamzusammenhalts	0 = keiner 1 = rudimentär 2 = durchschnittlich 3 = hoch 4 = vollständig
Project Network	Grad der Projektvernetzung	0 = keine 1 = schwach 2 = durchschnittlich 3 = stark 4 = total
Process Maturity	Grad der Prozessreife	0 = chaotisch 1 = wiederholbar 2 = definiert 3 = gemessen 4 = optimiert
OO-Methods	Nutzung von OO-Methoden	0 = keine 1 = partiell 2 = halb 3 = hauptsächlich 4 = vollständig
OO-Tools	Nutzung von OO-Tools	0 = keine 1 = Codierung 2 = Codierung und Design 3 = Codierung, Design und Test 4 = Codierung, Design, Test und Spezifikation
OO-Language	Stufe der OO-Programmiersprache	0 = prozedurale Sprache 1 = hybride Sprache 2 = OO-Sprache 3 = weiterentwickelte OO-Sprache
OO-Test	OO-Testautomation	0 = keine 1 = Unit-Test 2 = Unit- und Integrationstest 3 = Unit, Integrations- und Systemtest 4 = vollintegrierte Testumgebung für alle Phasen

Object Repository	Nutzung Object-Repository	0 = keine 1 = ausschließlich Code-Objekte 2 = Code- und Designobjekte 3 = Code-, Design- und Testobjekte 4 = alle Objekte inklusive Anforderungen und Änderungen
-------------------	---------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## C.7 Einflussfaktoren der UseCase-Point-Methode

Projekteinflussfaktoren:

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
Process Experience	Kenntnisse der Mitarbeiter	0 = keine 1 = etwas 2 = durchschnittlich 3 = überdurchschnittlich 4 = vollständig
Application Experience	Erfahrung mit der Anwendung	0 = keine 1 = Wochen 2 = Monate 3 = Jahre 4 = viele Jahre
Development Experience	Entwicklungserfahrung	0 = keine 1 = Wochen 2 = Monate 3 = Jahre 4 = viele Jahre
Analysis Capability	Analysefähigkeiten	0 = keine 1 = wenig 2 = durchschnittlich 3 = überdurchschnittlich 4 = super
Team Motivation	Teammotivation	0 = keine 1 = wenig 2 = durchschnittlich 3 = hoch 4 = maximal
Requirement Stability	Anforderungsqualität	0 = informell 1 = strukturiert 2 = semiformal 3 = formal 4 = formal und graphisch
Team Availability	Teamverfügbarkeit	0 = < 50% 1 = 50-75% 2 = 75-90% 3 = 90-99% 4 = 100%
Ease of Development	Qualität der Entwicklungsumgebung	0 = keine 1 = wenig 2 = durchschnittlich 3 = hoch 4 = maximal

Technische Einflussfaktoren:

Name des Einflussfaktors		Bedeutung und Gewichtung des Einflussfaktors
englische Bezeichnung	deutsche Entsprechung	
System Distribution	Systemverteilung	0 = keine 1 = etwas 2 = teilweise 3 = hauptsächlich 4 = voll
System Performance	Systemperformanz	0 = keine 1 = niedrig 2 = durchschnittlich 3 = hoch 4 = maximal
Comprehensibility	GUI-Effizienz	0 = keine 1 = niedrig 2 = durchschnittlich 3 = hoch 4 = sehr hoch
Complexity	Systemkomplexität	0 = keine 1 = niedrig (< 0.4) 2 = durchschnittlich (0.4-0.6) 3 = hoch (0.6-0.8) 4 = sehr hoch (> 0.8)
Reusability	Sytemwiederverwendbarkeit	0 = 0 1 = < 25% 2 = 25-50% 3 = 50-75% 4 = > 75%
Installability	Installations-einfachheit	0 = unvorhersehbar 1 = in Tagen 2 = in Stunden 3 = in Minuten 4 = in Sekunden
Usability	Code-Lesbarkeit	0 = Code ist unlesbar 1 = Code benötigt Tage, um verstanden zu werden 2 = Code benötigt Stunden, um verstanden zu werden 3 = Code benötigt Minuten, um verstanden zu werden
Portability	Systemportierbarkeit	0 = nicht portierbar 1 = portierbar mit >50% der Initialkosten 2 = portierbar mit 30-50% der Kosten 3 = portierbar mit 10-30% der Kosten 4 = portierbar mit < 10% der ursprünglichen Kosten
Adaptability	Systemadaptivität	0 = System kann nicht geändert werden 1 = Änderungsaufwand > Erstellungsaufwand 2 = Änderungsaufwand = Erstellungsaufwand 3 = Änderungsaufwand < Erstellungsaufwand 4 = Änderungsaufwand < 20% des Erstellungsaufwands
Concurrency	Mehrfachzugriff	0 = rein sequentiell 1 = teilweise synchron

		2 = voll synchron
Security	System-sicherheit	0 = total ungesichert 1 = teilweise ungesichert 2 = teilweise gesichert 3 = gesichert 4 = maximal gesichert
Interoperability	Systemin-teroperabilität	0 = unmöglich zu integrieren 1 = integrierbar mit einigen System derselben Umgebung 2 = integrierbar mit allen Systemen der gleichen Umgebung 3 = integrierbar mit einigen Systemen anderer Umgebungen 4 = integrierbar mit allen Systemen
User Efficiency	Bedienungs-freundlichkeit	0 = System erfordert tagelanges Training Systembenutzung kann in Stunden selbst erlernt werden System kann sofort ohne Vorbereitung benutzt werden



## **Selbständigkeitserklärung**

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, 27.11.2008

Corinna Benecke

