

# **UNIVERSITÄT DUISBURG-ESSEN**

## **Dokumentation und Ressourcenplanung für effizientes Software Engineering**

Seminararbeit

Vorgelegt dem Fachbereich Wirtschaftswissenschaften  
der Universität Duisburg-Essen

von: Torsten Kraemer  
Richard-Wagner-Str. 54  
45128 Essen  
Matrikelnummer: 1465071

Wintersemester 2007/2008, 11. Studiensemester  
voraussichtlicher Studienabschluss: Sommersemester 2008

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b> .....	<b>II</b>
<b>Tabellenverzeichnis</b> .....	<b>III</b>
<b>Abkürzungsverzeichnis</b> .....	<b>IV</b>
<b>1 Einführung</b> .....	<b>1</b>
<b>2 Anforderungen an die Dokumentation</b> .....	<b>3</b>
2.1 Status quo des Projekts erfassen .....	3
2.1.1 Erweiterter Projektstatusbericht .....	3
2.1.2 Projektkontrollberichte .....	4
2.1.3 Meilenstein-Trendanalyse .....	5
2.2 Erfassung der eingesetzten Ressourcen .....	6
2.2.1 Projekttagebuch und Aufwandserfassung .....	6
2.2.2 LOC - Methode .....	7
2.2.3 Sonstige Aufwände .....	7
2.3 Kennzahlen für neue Projekte gewinnen .....	8
2.3.1 Projektabschlussbericht .....	8
2.3.2 Erfasste Ressourcen archivieren .....	9
2.3.3 Ausblick auf Aufwandsschätzung .....	9
<b>3 Methoden der Aufwandsschätzung</b> .....	<b>10</b>
3.1 Analogiemethode .....	10
3.2 Expertenschätzung durch die Delphi-Methode .....	11
3.3 Function Point - Methode .....	11
3.4 CoCoMo / CoCoMo II .....	13
3.4.1 Formel zur Berechnung des Aufwands .....	13
3.4.2 Formel zur Berechnung der Entwicklungszeit .....	13
3.4.3 Weiterentwicklung .....	14
3.5 UML und Nutzung im RUP .....	14
3.6 Vergleich .....	16
<b>4 Visualisierung der Dokumentation</b> .....	<b>18</b>
4.1 Gantt - Diagramm .....	18
4.2 PERT - Diagramm .....	18
4.3 Intranet (CMS, Wiki, Foren) .....	19
<b>5 Fazit</b> .....	<b>21</b>
<b>Anhang A: Legende der Vergleichstabelle</b> .....	<b>22</b>
<b>Literaturverzeichnis</b> .....	<b>23</b>

## **Abbildungsverzeichnis**

Abbildung 1.1: Messbare Kriterien im Software Engineering .....	1
Abbildung 2.1: Beispielhafte Darstellung der Meilenstein-Trendanalyse.....	5
Abbildung 4.1: Beispiel eines Gantt-Diagramms .....	18
Abbildung 4.2: Beispiel eines PERT-Diagramms .....	19
Abbildung 5.1: Der Unsicherheitsfaktor Mensch .....	21

## **Tabellenverzeichnis**

Tabelle 3.1: Vergleich der Methoden zur Aufwandsschätzung (Teil 1) .....	16
Tabelle 3.2: Vergleich der Methoden zur Aufwandsschätzung (Teil 2) .....	17

## **Abkürzungsverzeichnis**

CMS ..... Content Management System

CoCoMo .... Constructive Cost Model

HTTP ..... Hypertext Transfer Protocol

IT ..... Informationstechnik

LOC ..... Lines of code

PERT ..... Project Evaluation and Review Technique

RUP ..... Rational Unified Process

UML ..... Unified Modelling Language

Wiki ..... Kurzform des hawaiischen *Wiki Wiki* (deutsch: sehr schnell)

# 1 Einführung

**„Was man nicht misst,  
das kann man nicht steuern.“<sup>1</sup>**

**- Tom de Marco**

Die Dokumentation von Softwareprojekten ist längst keine Belanglosigkeit mehr, die nebenbei und unkoordiniert gepflegt werden kann. Sie entwickelt sich immer mehr zu einer Herausforderung an die Projektleitung und auch an die einzelnen Teammitglieder, die mit und an ihr arbeiten müssen.

Zunächst ist es notwendig, den Begriff der Dokumentation zu definieren. Wenn hier von Dokumentation die Rede ist, ist die Beobachtung und Aufzeichnung der Prozesse zur Softwareerstellung und der dazu eingesetzten Ressourcen gemeint. Die Dokumentation der Quellcodes oder sogar die Dokumentation für den Endbenutzer ist nicht Gegenstand dieser Arbeit.

Diese Ausarbeitung betrachtet die verschiedenen Anforderungen, die an eine umfassende Dokumentation gestellt werden. Sie stellt Modelle vor, wie diese Dokumentation realisiert werden kann, und erläutert, unter welchen Voraussetzungen ein Mehrwert von ihr ausgehen kann, so dass von einem effizientem Software Engineering gesprochen werden kann.



**Abbildung 1.1: Messbare Kriterien im Software Engineering**

---

<sup>1</sup> Original: "You can't control what you can't measure"  
(aus: *Controlling Software Projects: Management, Measurement & Estimation*)

Um den Umfang überschaubar zu halten, wird von den drei messbaren Kriterien des Software Engineering in dieser Arbeit nur der Aspekt „Komplexität und Umfang“ gemessen und abgeschätzt. Die Qualitätssicherung ist nur ein nebensächliches Thema und eine Berücksichtigung der verschiedenen Entwicklungsmethoden alleine könnte schon Thema einer weiteren Arbeit sein.

## Vorgehensweise

2 Das zweite Kapitel beschreibt zunächst die Anforderungen an eine Dokumentation für in der Umsetzung befindliche Projekte. Diese Anforderungen sind in drei Punkte unterteilbar, die jeweils kurz erläutert und durch verschiedene Vorgehensmodelle konkretisiert werden. Der letzte Teil des zweiten Kapitels stellt bereits eine Überleitung in zukünftige Projekte dar.

3 Im dritten Kapitel werden Methoden zur Komplexitäts- und Umfangsschätzung für in der Planung befindliche Projekte vorgestellt. Es folgt ein tabellarischer Vergleich dieser Methoden, aus dem sich eine Empfehlung ableiten lässt, welches Modell wann sinnvoll ist.

4 Wie die in den vorigen beiden Kapiteln erfassten Daten am besten darzustellen sind, erläutert das vierte Kapitel. Dazu gehören verschiedene Möglichkeiten zur Visualisierung unter Berücksichtigung einer technischen Realisierung.

5 Der letzte Punkt dieser Arbeit ist ein abschließendes Fazit.

## 2 Anforderungen an die Dokumentation

Ein Software Engineering - Projekt stellt viele Anforderungen an alle beteiligten Systeme und Personen innerhalb eines Unternehmens. Die Anforderungen an die entsprechende Projektdokumentation dieses Prozesses lassen sich prinzipiell in drei Bereiche aufteilen:

- Die Erfassung des aktuellen Projektstatus
- Die Erfassung der eingesetzten Ressourcen
- Die Gewinnung von Informationen

### 2.1 Status quo des Projekts erfassen

Ein wichtiger Begriff im Umfeld der IT-Projektentwicklung ist der des „Meilensteins“ (englisch: „Milestone“). Er steht für wichtige im Vorfeld definierte Zwischenziele während des operativen Softwareentwicklungsprojekts und kennzeichnet den Abschluss einer Phase / Periode. Die Verfehlung oder Verschiebung eines Meilensteins hat große Auswirkungen auf den gesamten Projektverlauf.

Nach Projektbeginn ist es notwendig, den aktuellen Status fortlaufend zu überwachen. So ist es möglich, gegebenenfalls lenkend in den Projektablauf eingreifen und Aussagen über den Projektfortschritt (beispielsweise für die Einhaltung gesetzter Fristen) einhalten zu können. Dazu benötigt der oder benötigen die Projektleiter ein Werkzeug, das den gesamten Fertigstellungsgrad einzelner Teilaufgaben und somit letztendlich auch des gesamten Systems untersuchen und darstellen kann. Die folgenden Unterkapitel befassen sich mit drei dieser Methoden.

#### 2.1.1 Erweiterter Projektstatusbericht

Die Erstellung eines erweiterten Projektstatusberichts nach [GrJS2003, 237f] erfordert, den zeitlichen Rahmen einzuhalten, die notwendigen Aktivitäten durchzuführen und alle benötigten Resultate wie Dokumente oder auch Programmcode zu liefern. In einem periodenorientierten<sup>2</sup> Berichtswesen sind so erstellte Status eine wichtige Entscheidungsgrundlage.

---

<sup>2</sup> Eine Periode entspricht einem im Vorfeld definierten Meilenstein

Ein fertiger Projektstatusbericht gliedert sich in:

- Aufgabenstellung und Zusammenfassung der Ergebnisse
- Ergebnisse der Situationsanalyse
- Detaillierung der im Projektauftrag vereinbarten Ziele
- Denkbare Lösungen, ihre Voraussetzungen und Konsequenzen
- Bewertung und Vorschläge von Alternativen
- Darlegung des weiteren Vorgehens

Unter den genannten Punkten sind die Ergebnisse der Situationsanalyse am wenigsten eindeutig und werden in [GrJS2003] nicht konkret erläutert. Diese Vorgehensweise ist eher als Berichtswesen zu verstehen und richtet sich an die operativen Teammitglieder, die die Situation einschätzen können und die Projektleitung informieren.

### **2.1.2 Projektkontrollberichte**

Eine andere Möglichkeit zur Erfassung des gegenwärtigen Projektstands stellt [ZuGK2004, 131f] dar. In den Projektkontrollberichten vermitteln die Projektmitarbeiter den Gruppenleitern den gegenwärtigen Status, die diesen wiederum an die Projektleitung weitergeben. Die Berichte können sowohl in festgelegten Intervallen, die sich nach der Gesamtdauer des Projekts und dem Projektfortschritt richten, als auch nach bestimmten Phasen<sup>3</sup> erstellt werden.

Die möglichst kurz zu haltenden Berichte enthalten folgende Punkte:

- Datum, Name des Autors, Projektbezeichnung, Projektphase
- Kurzzusammenfassung der in Arbeit befindlichen Tätigkeiten
- Auflistung aller Probleme und falls möglich der Problemlösungen<sup>4</sup>
- Festlegung der nächsten Schritte
- Anmerkungen

Ein solcher Projektkontrollbericht enthält eine subjektive Einschätzung der Projektmitarbeiter als unmittelbar Beteiligte über den aktuellen Status. Er orientiert sich also sehr nah an der Implementierung, ist aber auch anfällig für Fehlein-

---

<sup>3</sup> Eine Phase entspricht einem im Vorfeld definierten Meilenstein

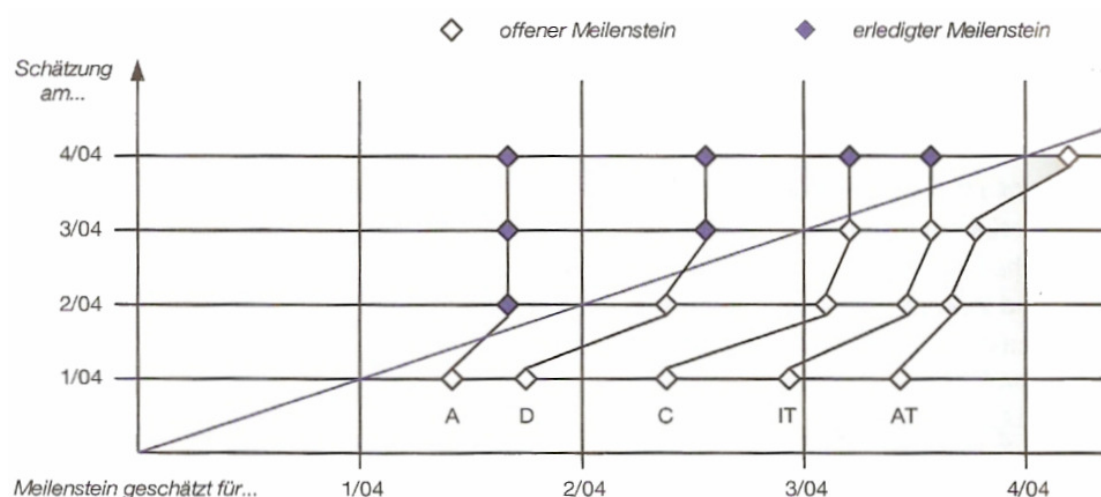
<sup>4</sup> Dieser Punkt ist im Zusammenhang mit dem Aufbau einer Wissensdatenbank für zukünftige Projekte sehr hilfreich, siehe Kapitel 2.3 - Kennzahlen für neue Projekte gewinnen

schätzungen. Es muss außerdem durch Gruppen- und Projektleiter bewertet und in das Gesamtprojekt eingeordnet werden.

### 2.1.3 Meilenstein-Trendanalyse

Ebenfalls in [ZuGK2004, 132ff] wird eine weitere Methode dargestellt, mit der der Fortschritt in einer Softwareentwicklung überwacht werden kann. Bei der Meilenstein-Trendanalyse wird der Projektverlauf über das Erreichen vorher gesetzter Meilensteine visualisiert.

Regelmäßige Neuschätzungen<sup>5</sup> über die Entwicklung der Zeitpunkte für die Fertigstellung noch nicht erreichter Meilensteine liefern einen Rückschluss darüber, wie genau die vorigen Schätzungen aussahen und an welchen Stellen Verzögerungen aufgetreten sind. Durch eine stetige Anpassung der Voraussagequalität können diese Verzögerungen minimiert und der weitere Projektverlauf genauer eingeschätzt werden.



**Abbildung 2.1: Beispielhafte Darstellung der Meilenstein-Trendanalyse**

Quelle: [ZuGK2004, 134]

Der Vorteil dieser Methode besteht neben einem schnellen Überblick über den Projektfortschritt darin, dass auch Aussagen über die Projektkultur getroffen werden können. So können möglicherweise wiederholte (leichte) Verschiebungen ein Indiz dafür sein, dass die Teamleiter die Projektleitung verträsten, um diese zufrieden zu stellen. Bestimmte Arten von Verzögerungen können schnell die Aufmerksamkeit der Projektleitung erregen, damit entsprechende Maßnahmen eingeleitet werden:

<sup>5</sup> Diese Neuschätzungen finden in zeitlich festgelegten Intervallen unabhängig von den Meilensteinen statt.

- *Eine starke Verschiebung* eines Meilensteins lässt darauf schließen, dass ein Problem aufgetreten ist, dessen Behebung mehr Zeit in Anspruch nimmt. Obwohl dies das Projekt verzögert, ist von einer realistischen Einschätzung auszugehen.
- *Eine wiederholte Verschiebung unterhalb oder parallel zur Datumslinie*<sup>6</sup> kann hingegen bedeuten, dass eine Aufgabe deutlich unterschätzt wurde oder immer neue Probleme auftreten. Solche Meilensteine benötigen die volle Aufmerksamkeit der Projektleitung.

## 2.2 Erfassung der eingesetzten Ressourcen

Die Erfassung der eingesetzten Ressourcen ist zwar primär zur nachträglichen Abrechnung der Projekte für externe Auftraggeber wichtig, hat aber auch für die Prüfung interner Aufwendungen eine Berechtigung. Es werden alle Posten und die jeweils zugehörigen eingesetzten Ressourcen erfasst, so dass auch ein Rückschluss auf die Effizienz möglich ist. Für zukünftige Projekte stellen diese Daten außerdem eine wertvolle Erfahrungsgrundlage dar, deren Ausführung in Kapitel 2.3.2 folgt.

### 2.2.1 Projekttagbuch und Aufwandserfassung

Die nächstliegende Möglichkeit zur Erfassung der aufgewendeten Ressourcen ist ein einfaches Projekttagbuch in Kombination mit der individuellen Arbeitszeiterfassung pro Mitarbeiter [ZuGK, 130].

Das Projekttagbuch umfasst alle projektrelevanten Ereignisse, da es eine Liste aller Besprechungen, Reviews, Integrations- und Testsitzungen enthält. Diese besteht pro Eintrag jeweils aus dem Datum, der Dauer, den Namen der beteiligten Mitarbeiter, einer kurzen Beschreibung sowie dem Verweis auf das ausführliche Dokument mit allen weiteren Details. Auf diese Weise werden auch die verwaltenden Nebentätigkeiten erfasst, die oftmals in Projekten unberücksichtigt bleiben.

Jeder Mitarbeiter erfasst seine Arbeitszeit unter Angabe der bearbeiteten Dokumente und Tätigkeiten. Wird dazu eine Software mit zentraler Datenbank eingesetzt, ist eine automatische Auswertung und Zuordnung der Arbeitszeiten zu bestimmten (Teil-)Projekten und / oder Kunden mit geringem Aufwand möglich. Damit die Akzeptanz der Mitarbeiter und die damit verbundene Qualität der er-

---

<sup>6</sup> Dies ist die Diagonale in Abbildung 2.1: Beispielhafte Darstellung der Meilenstein-Trendanalyse

hobenen Daten gewährleistet ist, empfiehlt sich eine simple Software, die ohne großen Aufwand zu bedienen ist und sich leicht in den Arbeitsalltag eingliedert.

### 2.2.2 LOC - Methode

Einer der ältesten Ansätze zur Bestimmung des eingesetzten Aufwands besteht darin, die Anzahl der Programmzeilen im Quellcode (LOC, „Lines of Code“) zu erfassen. Diese Methode stammt aus einer Zeit, *„als noch auf Lochkarten in Sprachen wie Assembler, FORTRAN oder COBOL programmiert wurden [sic!]“* [Dorl2004, 9].

Während damals noch jede Anweisung eine Lochkarte füllte und dementsprechende Aussagekraft über den Umfang eines Projekts besaß, ist das heute nicht mehr ohne weiteres möglich: Unterhalb und manchmal auch innerhalb der gängigen Programmiersprachen sind Algorithmen auf verschiedenen Wegen mit einer unterschiedlichen Anzahl an Programmzeilen zu bewältigen. Außerdem kann in modernen Programmiersprachen eine Reihe von Zeilen vorkommen, die keine Anweisungen sind (wie zum Beispiel Leerzeilen, Deklarationen, Kommentare oder auch zu Schleifen gehörende Steuerungszeichen). Für ein und dieselbe Anwendung können so in Abhängigkeit vom individuellen Programmierstil und der Erfahrung des Programmierers sehr große Differenzen bei der Anzahl der geschriebenen Zeilen die Folge sein.

Auch auf die Qualität eines Programms kann diese Methode unerwünschte Folgen haben:

„Wenn man die Produktivität der Mitarbeiter nach LOC bewertet, werden diese angehalten, die LOC zur Implementierung einer Funktionalität zu erhöhen. Auf die Qualität der Programme hat dies einen negativen Einfluß.“ [Dorl2004, 11]

Diese Methode ist also unter neuen Gesichtspunkten für die Abrechnung eines Projekts nicht zu empfehlen und dient lediglich als Werkzeug in komplexeren Modellen wie zum Beispiel CoCoMo (II). Da CoCoMo (II) eine Methode zur Kosten- bzw. Aufwandsschätzung ist, wird sie in Kapitel 3.4 behandelt.

### 2.2.3 Sonstige Aufwände

Obwohl der Faktor *„Zeit“* in einem Projekt sicherlich den größten Anteil an den zurechenbaren Aufwänden hat, sind laut [Ange2006] noch andere Posten ebenfalls zu erfassen. Zu diesen zählen:

- Verbrauchsmittel
- Abschreibung für Abnutzung oder getätigte Investitionen
- Sachgemeinkosten

- Personalgemeinkosten
- Honorare und andere Zahlungen an Dritte
- Gebühren (Patente, Lizenzen, Anträge an Behörden, usw.)

Ob die genannten Punkte jeweils für ein Projekt relevant sind, ist im Einzelfall zu entscheiden. Auch wenn sie in der Regel nicht anfallen, sollten sie zumindest in die Überlegungen zur abschließenden Erfassung der aufgewendeten Ressourcen einbezogen werden.

## 2.3 Kennzahlen für neue Projekte gewinnen

Der letzte wichtige Aspekt bei der Dokumentation einer Softwareentwicklung ist der Aufbau einer Wissensdatenbank für neue Projekte. Nicht nur, um aus vergangenen Fehlern zu lernen, sondern auch um positive Entwicklungen weiterhin nutzbar zu machen, ist es notwendig, die relevanten Aspekte in einer Form zu dokumentieren, die eine schnelle Wissensnutzung ermöglicht.

An diesem Punkt greifen also sowohl die Erfassung des Projektfortschritts<sup>7</sup> als auch der aufgewendeten Mittel<sup>8</sup> ineinander, um einen Mehrwert über die eigentliche Fertigstellung des oder der ursprünglichen Projekte hinaus zu generieren.

### 2.3.1 Projektabschlussbericht

Alle Erfahrungen, die im Laufe eines Projekts gesammelt wurden, werden in einem Projektabschlussbericht zusammengefasst. Dabei ist es wichtig, dass nicht nur Erfolge, sondern auch negative Erfahrungen und Probleme detailliert vermerkt werden.

In [GrJS2003, 238] werden folgende Informationen für einen solchen Bericht vorgeschlagen:

- Eine verständliche Zusammenfassung des Soll-Ist-Vergleichs mit besonderem Augenmerk auf den Zeitrahmen und das Budget
- Informationen für zukünftige Aufwandsschätzungen durch einen Vergleich der Aufwände aller Projektphasen
- Informationen für zukünftige Projektleiter ähnlicher Projekte hinsichtlich der eingesetzten Werkzeuge und Methoden, des gewonnenen Wissens

---

<sup>7</sup> Siehe Kapitel 2.1 - Status quo des Projekts erfassen

<sup>8</sup> Siehe Kapitel 2.2 - Erfassung der eingesetzten Ressourcen

und der Erfahrungen mit Dritten (zum Beispiel weitere Dienstleister oder Zulieferer)

Da nicht alle dieser Informationen für neue Projekte relevant sind, muss eine klare Gliederung oberste Prämisse sein. So ist gewährleistet, dass zukünftige Projektleiter schnell auf das gewonnene Wissen zugreifen und nützliche Informationen von unbrauchbaren trennen können.

Falls daraufhin genauere Informationen benötigt werden, kann auf die zuvor erstellten erweiterten Projektstatusberichte<sup>9</sup> zurückgegriffen werden.

### **2.3.2 Erfasste Ressourcen archivieren**

Unabhängig von der Methode der erfassten Ressourcen<sup>10</sup> ist es wichtig, die zur Verfügung stehenden Informationen dauerhaft zu sichern. Dabei steht der schnelle und direkte Zugriff auf die für ein neues Projekt relevanten Daten im Vordergrund.

Wenn diese Daten in einer Form archiviert sind, dass auf sie nur mit hohem Aufwand zugegriffen werden kann (ungünstigerweise zum Beispiel auf Datenbändern in einem verschlossenen Tresor), verlieren sie ihren Nutzen. Im Optimalfall können authentifizierte Benutzer (Projekt- oder Teamleiter) dezentral auf die Informationen zugreifen.

Neue (Teil-)Projekte können dann von zuvor gemachten Erfahrungswerten profitieren und hinsichtlich ihrer eigenen Ressourcennutzung zu diesen Erfahrungswerten in Beziehung gesetzt werden, um Probleme zu erkennen oder die Effizienz zu steigern.

### **2.3.3 Ausblick auf Aufwandsschätzung**

Wie Kapitel 2 gezeigt hat, dient die Dokumentation eines laufenden Projekts und die Erfassung von Kennzahlen nicht nur der Messung und Steuerung des eigentlichen Umsetzungsprozesses, sondern ist auch in die Zukunft gerichtet.

Nachdem der Status Quo eines Projekts also dokumentiert wurde, folgen in Kapitel 3 nun verschiedene Modelle zur Aufwandsschätzung in der Planung befindlicher Projekte.

---

<sup>9</sup> Siehe Kapitel 2.1.1 - Erweiterter Projektstatusbericht

<sup>10</sup> Siehe Kapitel 2.2 - Erfassung der eingesetzten Ressourcen

## 3 Methoden der Aufwandsschätzung

„Die Aufwandsschätzung ist die Basis, auf der Zeitplanung und Kostenschätzung überhaupt erst aufgebaut werden können. Daher nimmt die Aufwandsschätzung einen erheblichen Teil der Überlegungen zur Umsetzung ein.“ [ZuGK, 112]

Im dritten Kapitel geht es darum, zukünftige oder aktuell in der Planung befindliche Projekte einschätzen und bereits im Vorfeld Aussagen über die zu erwartenden Aufwände treffen zu können. Durch die im Unternehmen bekannten Personalkosten<sup>11</sup> gehen aus diesen Aufwänden unmittelbar die zu erwartenden Projektkosten hervor. In jeder Schätzung sollte jedoch die Tatsache, dass die Ressource „Mensch“ nicht unbegrenzt verfügbar ist<sup>12</sup>, stets Beachtung finden.

„Zwar kann es kein allgemeingültiges, für alle Projekte gleichermaßen passendes, Aufwandsschätzverfahren geben“ [GrJS2003, 195], doch gibt es einige gängige Methoden, die nun zunächst einzeln vorgestellt und anschließend tabellarisch verglichen werden.

### 3.1 Analogiemethode

Eine intuitive Methode zur Abschätzung des zu erwartenden Aufwands in einem neuen Projekt stellt die Analogiemethode dar [Dorl2004, 42ff]. Wenn eines oder mehrere der Kriterien

- Anwendungsgebiet
- Produktumfang
- Komplexitätsgrad
- Programmiersprache

im neuen Projekt einem bereits durchgeführten Projekt entsprechen, kann die Neuentwicklung analog zum bestehenden Produkt abgeschätzt werden.

Obwohl diese Vorgehensweise auf realen Daten basiert - und damit im Gegensatz zu theoretischen Modellen auch unerwarteten Ereignissen Sorge tragen kann - handelt es sich doch immer nur um eine individuelle Schätzung. Auf Grund ihrer Subjektivität und der fehlenden allgemeinen Vorgehensweise ist sie nicht wissenschaftlich nachvollziehbar.

---

<sup>11</sup> Gängigerweise werden die Personalkosten in Mannstunden oder Mannmonaten gemessen.

<sup>12</sup> Die Arbeitszeit einzelner Mitarbeiter kann zum Beispiel durch Urlaub, Weiterbildungsmaßnahmen oder auch Krankheit eingeschränkt werden

### 3.2 Expertenschätzung durch die Delphi-Methode

Die Expertenschätzung stellt eine weitere einfache Aufwandsschätzmethode dar. Ein Experte ermittelt einen optimistischen ( $A_o$ ), einen wahrscheinlichen ( $A_w$ ) und einen pessimistischen ( $A_p$ ) Aufwand für ein Projekt. Aus diesen berechnet sich der gewichtete Gesamtaufwand folgendermaßen:

$$A = \frac{1}{6} * (A_o + 4 * A_w + A_p)$$

Wenn mehr als ein Experte diese Methode durchführt, wird dies in der Regel durch die Delphi-Methode realisiert. Jeder Experte gibt eine anonyme Schätzung ab, die mit den anderen verglichen wird. Sollten sich deren Ergebnisse stark unterscheiden, erfolgen so lange neue Schätzungen, bis sich diese ähneln und ein Moderator einen Durchschnittswert bilden kann.

Während bei der klassischen Delphi-Methode keine Gruppendiskussion stattfindet, diskutieren im Wideband Delphi - Verfahren die Teilnehmer die Ergebnisse. Die Auswahl der verwendeten Variante hängt von der verfügbaren Zeit (Gruppendiskussionen sind in der Regel sehr zeitaufwendig) oder der Zusammensetzung der Gruppe ab: Wenn zu befürchten ist, dass einzelne Teilnehmer großen Einfluss auf die anderen Experten haben und ihre eigene Einschätzung durchsetzen, empfiehlt sich die klassische Methode. Als Vorteil dieses Verfahrens ist auf jeden Fall die große Dynamik und die Tatsache, dass Besonderheiten des Projekts stärker berücksichtigt werden können, zu nennen [Dorl2004, 40f; GrJS2003, 194ff].

### 3.3 Function Point - Methode

In den späten 1970er Jahren entwickelte Allen Albrecht bei IBM eine Methode, mit der der Umfang einer Implementierung gemessen werden kann. Sie ist bereits auf Basis eines vorhandenen Lastenhefts und somit sehr früh im Entwicklungsprozess einsetzbar. Die Analyse der Funktionspunkte („*Function Points*“) ist sprachenunabhängig und wird weder durch die Programmiermethode noch die Fähigkeiten des Projektteams beeinflusst. Dadurch ist sie im Gegensatz zur LOC-Methode<sup>13</sup> auch als Aufwandsschätzmethode geeignet.

„Function Points sind ein Maß für den Umfang (...) – aus der Sicht des Nutzers – eines IT-Produktes und des Projektes, welches es entwickelt.“ [GrJS2003, 160]

---

<sup>13</sup> Vgl. Kapitel 2.2.2 - LOC - Methode

Die Vorgehensweise bei diesem Schätzverfahren ist nach [Dorl2004, 13ff]:

1. *Bestimmung der Anzahl der Elemente pro Funktionstyp*

Die verfügbaren Funktionstypen sind externe Eingaben, externe Ausgaben, externe Abfragen, interne Datenbestände und Referenzdaten<sup>14</sup>.

2. *Komplexitätsstufen jedes Funktionstypen definieren*

Die möglichen Stufen sind einfach, mittel und hoch.

3. *Gewichtung jedes Typs nach seiner Komplexität*

Über die Gewichtung der Komplexitätsstufen kann pro Funktionstyp gesteuert werden, wie relevant dieser Aspekt für das fertige Softwareprodukt ist.

4. *Berechnung des unjustierten Funktionszählers*

Der unjustierte Funktionszähler ergibt sich aus der Summe aller Elemente eines Typs (aus 1.) multipliziert mit dem jeweiligen Gewicht (aus 3.).

5. *Bestimmung des Faktors zur technischen Komplexität*

Dieser Faktor setzt sich daraus zusammen, ob das Produkt zum Beispiel an mehreren Orten einsetzbar sein soll, Wiederverwendbarkeit gefordert wird oder welcher Art die Benutzeroberfläche sein soll<sup>15</sup>. Das Ergebnis ist ein Wert zwischen 0,65 und 1,35.

6. *Berechnung der justierten Funktionspunkte*

Die endgültige Anzahl an Funktionspunkten ist das Produkt des unjustierten Funktionszählers (aus 4.) und dem Faktor der technischen Komplexität (aus 5.).

Die Function Point – Methode basiert also auf den Produkthanforderungen und ist an viele Anwendungsbereiche, Techniken und Unternehmensverhältnisse anpassbar. Während der Entwicklung kann die Abschätzung iterativ verfeinert werden, sobald sich der Kenntnisstand ändert. Die Methode wird von mehr als 500 großen Unternehmen eingesetzt und wird – wahrscheinlich auch wegen der hohen Anzahl an verfügbaren Werkzeugen und der leichten Durchführbarkeit – als derzeit beste Methode zur Schätzung von kommerziellen Anwendungssystemen gesehen.

---

<sup>14</sup> In [Schä2002, 9] ist statt interner Datenbestände und Referenzdaten die Rede von internen und externen logischen Daten

<sup>15</sup> Genaue Definition siehe [Dorl2004, 15]

Da mit dieser Methode jedoch nur eine Abschätzung des Gesamtumfangs ohne Berücksichtigung einzelner Phasen möglich ist, Qualitätsanforderungen nicht berücksichtigt werden können und eine sehr starke Fokussierung auf die Funktionen stattfindet, ist auch dieses Modell nicht uneingeschränkt einsetzbar.

Eine Umrechnung von Funktionspunkten in Entwicklungsdauer oder -kosten basiert in der Regel auf Erfahrungswerten.

### 3.4 CoCoMo / CoCoMo II

1981 entwickelte Barry Boehm bei Boeing das Constructive Cost Modeling als algorithmisches Schätzmodell, das Aussagen über den zu erwartenden Aufwand und auch die Projektdauer treffen kann. Die Datengrundlage können zum Beispiel die geschätzten LOC oder Funktionspunkte sein.

Die folgenden Formeln basieren auf der Annahme einer Schätzung in LOC<sup>16</sup>.

#### 3.4.1 Formel zur Berechnung des Aufwands

$$A = m * KLOC^n$$

$A$  bezeichnet den Aufwand in Personenmonaten.

$KLOC$  bezeichnet den zu erwartenden Umfang in Tausend LOC.

Die Belegung der Faktoren  $m$  und  $n$  ist abhängig von der Systemklasse:

- *ORGANIC* (Vertrautes Problem, erfahrenes Team):
  - $A = 2,4 * KLOC^{1,05}$
- *SEMI-DETACHED* (Zwischenstufe):
  - $A = 3,0 * KLOC^{1,12}$
- *EMBEDDED* (Schwere Randbedingungen, neues Problem, wenig Erfahrung):
  - $A = 3,6 * KLOC^{1,2}$

#### 3.4.2 Formel zur Berechnung der Entwicklungszeit

$$TDEV = 2,5 * PM^D$$

$TDEV$  ist die zu erwartende Entwicklungszeit

---

<sup>16</sup> Dieses Verfahren ist in der Literatur gängiger und leichter nachzuvollziehen. Sollen Funktionspunkte als Grundlage dienen, müssen die unjustierten Werte verwendet werden.

$PM$  sind die zuvor errechneten Personenmonate.

$D$  ist wiederum abhängig von der Systemklasse:

- *ORGANIC*:
  - $TDEV = 2,5 * PM^{0,38}$
- *SEMI-DETACHED*:
  - $TDEV = 2,5 * PM^{0,35}$
- *EMBEDDED*:
  - $TDEV = 2,5 * PM^{0,32}$

Die Personenmonate können nicht beliebig durch die Anzahl der eingesetzten Mitarbeiter dividiert werden, da ein größeres Team unter anderem auch mehr Kommunikationsaufwand erfordert. Außerdem sind einige Aufgaben nicht parallel zu bearbeiten, so dass eine Erhöhung der Mitarbeiterzahl sich nicht linear auf die Bearbeitungsdauer auswirkt.

### 3.4.3 Weiterentwicklung

„Das COCOMO-Basismodell ist gut geeignet für eine grobe Abschätzung der Größenordnung der Softwarekosten. Die Genauigkeit des Modells ist notwendigerweise eingeschränkt, weil es ihm an Faktoren für Unterschiede bei der verwendeten Hardware, der Personalqualität und -erfahrung, dem Einsatz moderner Werkzeuge und Technologien und anderen Merkmalen fehlt, die bekanntermaßen einen signifikanten Einfluss auf die Kosten haben.“ [Wiki01]

Eine Weiterentwicklung des ursprünglichen Modells ist seit 2000 als CoCoMo II bekannt, in dem auch die Möglichkeit besteht, das Modell auf konkrete Rahmenbedingungen hin zu kalibrieren und so spezifische Projektbesonderheiten zu berücksichtigen<sup>17</sup>.

## 3.5 UML und Nutzung im RUP

UML ist keine Methode im klassischen Sinne der Aufwandsschätzung, sondern die Definition einer einheitlichen Notation, „um (...) Geschäftsprozesse, Anforderungen und die daraus resultierenden (...) Entwürfe (...) visuell beschreiben zu können“ [ZuGK2004, 195]. Wird diese Definition nun auf den Anwendungsfall übertragen, den Umfang einer Software zu messen, ergibt sich folgerichtig auch die

---

<sup>17</sup> An dieser Stelle sei auf weiterführende Literatur via <http://sunset.usc.edu/research/COCOMOII/> verwiesen.

Möglichkeit, diese Software bereits dann abschätzen zu können, wenn noch keine Funktionalität implementiert ist. Für eine grundsätzliche Einführung in UML sei dabei an dieser Stelle auf entsprechende Literatur verwiesen<sup>18</sup>.

Um den Umfang eines neuen Softwareprojekts abzuschätzen, sind mehrere der durch UML beschriebenen Diagramme denkbar:

- *Anwendungsfalldiagramm*  
Darstellung des von einem (Teil-)System erwarteten Verhaltens in dem gängigerweise Akteure und Anwendungsfälle dargestellt werden
- *Klassendiagramm*  
Darstellung der statischen Struktur eines (Teil-)Systems
- *Zustandsdiagramm*  
Darstellung der Zustände von Objekten und deren Übergänge ineinander
- *Aktivitätsdiagramm*  
Darstellung einzelner Aktivitäten und der zeitlichen Zusammenhänge

Die Auswahl des zu erstellenden Diagramms hängt in erster Linie davon ab, welche Art des Umfangs abgeschätzt werden soll. Während zum Beispiel ein Klassendiagramm am ehesten den klassischen Codeumfang widerspiegelt, ist ein Anwendungsfalldiagramm für den Überblick über die zu implementierenden Funktionalitäten und deren Komplexität hilfreich.

Mit UML eng verknüpft ist der *Rational Unified Process*, bei dem es sich um ein objektorientiertes Vorgehensmodell zur Softwareentwicklung handelt. RUP definiert neun Hauptarbeitsschritte, in denen UML als Notationssprache benutzt wird. Aus diesem Grund bietet sich die zuvor geschilderte Aufwandsschätzung speziell dann an, wenn der RUP als Technik eingesetzt wird und / oder die Informationen über das zu erstellende Projekt bereits in UML-Notation vorliegen. Die Überführung von nicht in UML geplanten Projekten in diese Notation nimmt erhebliche Zeit in Anspruch und ist deswegen nicht zu empfehlen.

Ein im RUP erstellter Phasenplan „kann bei entsprechender Ausarbeitung auch für weitere, ähnliche Projekte herangezogen werden“ [ZuGK2004, 93]. Deswegen ist diese Vorgehensweise sowohl in der Perspektive „Aufwandsschätzung“ als auch „Gewinnung von Kennzahlen für neue Projekte“ hilfreich.

---

<sup>18</sup> z.B. Christoph Kecher: UML 2.0. Das umfassende Handbuch. Galileo Press, Bonn 2006.

### 3.6 Vergleich

Die in diesem Kapitel vorgestellten Methoden können im Rahmen dieser Arbeit natürlich nur einen Teil der Möglichkeiten abbilden, mit denen Aufwandsschätzungen möglich sind. Die folgende Tabelle stellt diese Modelle gegenüber und nimmt eine subjektive Bewertung vor.

Die Erläuterung der Kriterien und eine Legende der Bewertung findet sich im Anhang A: Legende der Vergleichstabelle

<b>Modell</b> <b>Kriterium</b>	<b>Analogie-</b> <b>methode</b>	<b>Experten-</b> <b>schätzung</b>	<b>Function Point</b> <b>- Methode</b>
<b>Sprachenabhängigkeit</b>	→	↗	↑
<b>Genauigkeit</b>	↘	→	↗
<b>Frühzeitigkeit</b>	→	→	↑
<b>Detaillierbarkeit</b>	→	↘	↘
<b>Eindeutigkeit</b>	↘	↘	↗
<b>Objektivität</b>	↓	↘	↗
<b>Transparenz</b>	→	↓	↑
<b>Flexibilität</b>	↘	↑	→
<b>Benutzerfreundlichkeit</b>	↑	↗	↗

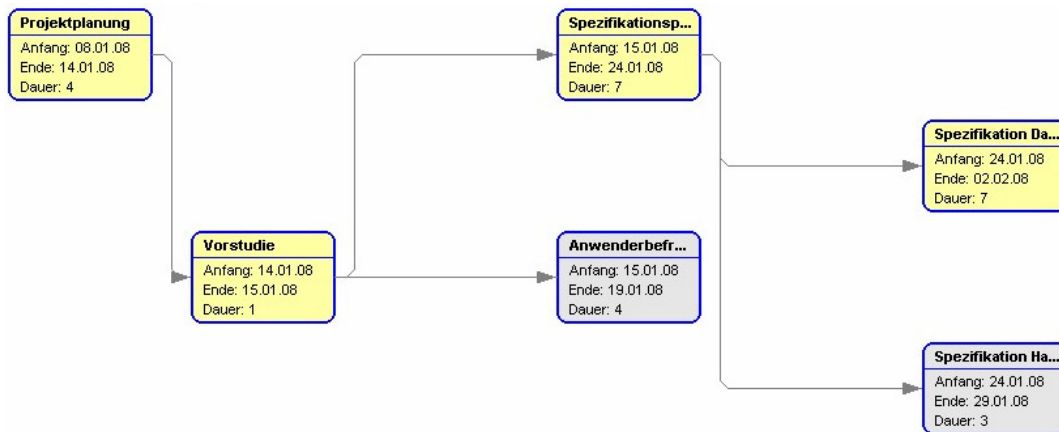
**Tabelle 3.1: Vergleich der Methoden zur Aufwandsschätzung (Teil 1)**

<b>Modell</b> <b>Kriterium</b>	<b>CoCoMo / CoCoMo II</b>	<b>UML</b>
<b>Sprachenabhängigkeit</b>	↑	↑
<b>Genauigkeit</b>	↑	→
<b>Frühzeitigkeit</b>	↑	↑
<b>Detaillierbarkeit</b>	↘	↗
<b>Eindeutigkeit</b>	↑	→
<b>Objektivität</b>	↑	→
<b>Transparenz</b>	↑	→
<b>Flexibilität</b>	→	↗
<b>Benutzerfreundlichkeit</b>	↗	↘

**Tabelle 3.2: Vergleich der Methoden zur Aufwandsschätzung (Teil 2)**



gerichtete Verknüpfungen der Aktivitäten können diese hierarchisch zueinander in Bezug gesetzt werden, so dass aus dem resultierenden Diagramm Abhängigkeiten hervorgehen.



**Abbildung 4.2: Beispiel eines PERT-Diagramms**

Ein PERT-Diagramm reagiert dynamisch auf Verzögerungen oder Neuordnungen einzelner Aktivitäten und ermöglicht so während eines laufenden Projekts einen aktuellen Überblick über den Projektstatus und die nächsten einzuleitenden Schritte.

### 4.3 Intranet (CMS, Wiki, Foren)

Durch Einsatz einer guten Software lassen sich aus einer einheitlichen Datenbasis die je nach Anforderung benötigten Diagramme erzeugen<sup>19</sup>. Um diese - und viele weiteren Daten - allen Beteiligten zur Verfügung zu stellen, wird eine Plattform benötigt, die zentral arbeitet und möglichst an jedem Arbeitsplatz verfügbar ist. Für solche Fälle bietet sich eine Webanwendung über das HTTP-Protokoll an, da es auf jedem PC verfügbar und auch in restriktiven Firmennetzen nutzbar ist. Der Anwender benötigt lediglich einen Webbrowser, um auf die Informationen zugreifen zu können.

Je nach Umfang der bereitgestellten Informationen und der gewünschten Komplexität können hierfür verschiedene Systeme auf dem Server eingerichtet werden:

<sup>19</sup> Beispiel: GanttProject, <http://ganttproject.biz/>

- Für die einfache Bereitstellung von Protokollen und Diagrammen bieten sich *Content Management Systeme* an. Diese können ohne großen Aufwand auch von technisch unversierten Benutzern gepflegt werden. Sie bieten in der Regel auch die technischen Voraussetzungen, um eine Zugriffsbeschränkung auf bestimmte Personen(-gruppen) zu realisieren oder eigene Erweiterungen zu implementieren, über die beispielsweise der Anwender die Darstellung eines Diagramms an seine Bedürfnisse anpassen kann.
- Wenn die bereitgestellten Informationen textbasiert sind und darüber hinaus für die gemeinsame Bearbeitung vorgesehen sind (Beispiele: Ideensammlungen, Team-To-Do-Listen, Wissensdatenbanken), empfehlen sich *Wikis*.
- Die öffentliche Diskussion eines Dokuments oder einer Information kann in einem *Forum* stattfinden. Diese Software ist – je nach Anwendungsgebiet – entweder anonym nutzbar (Beispiel: virtueller „Kummerkasten“) oder lässt einen Rückschluss auf die Identität eines Benutzers zu, um die interne Projektkommunikation zu fördern.

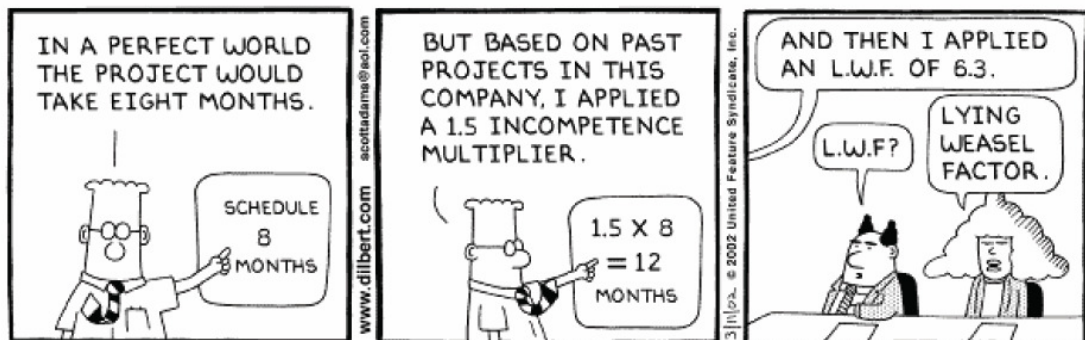
Es stehen für jeden Anwendungsfall mittlerweile mehrere ausgereifte Softwares zur Verfügung, die kostenlos eingesetzt und schnell eingerichtet werden können. Die Anforderungen an den Webserver sind relativ gering, müssen aber bei einer sehr hohen Benutzerzahl oder einem starken Datenaufkommen in der Planung entsprechend berücksichtigt werden.

## 5 Fazit

Wenn eine Dokumentation sorgfältig geplant und gepflegt wird, stellt sie ein mächtiges Werkzeug dar, das für viele Bereiche des Software Engineering genutzt werden und auch über ein aktuell durchgeführtes Projekt hinaus einen echten Mehrwert für das Unternehmen schaffen kann.

Damit eine Dokumentation erfolgreich sein kann, muss neben der formalen Korrektheit auch eine hohe Aufmerksamkeit auf die beteiligten Personen gelegt werden. Software wird nicht von Maschinen, sondern von Menschen erstellt. Projekte leben von den an ihnen beteiligten Personen. Bei der trügerischen Sicherheit, die manche Schätzungen im Vorfeld erzeugen, können diese Aspekte leicht in Vergessenheit geraten.

Die Durchführung eines Softwareprojekts hängt also nicht nur von der Softwarequalität, sondern vor allem auch von den an ihr beteiligten Menschen ab. Der Mensch als Unsicherheitsfaktor kann entweder aus Unachtsamkeit oder auch absichtlich aus politischen Gründen ein Projekt verzögern:



Copyright © 2002 United Feature Syndicate, Inc.

**Abbildung 5.1: Der Unsicherheitsfaktor Mensch**

Wenn die Projektleitung das stets berücksichtigt und die in der vorliegenden Seminararbeit vorgestellten Methoden angemessen einsetzt, ist der Grundstein für ein effizientes Software Engineering gelegt.

## Anhang A: Legende der Vergleichstabelle

### Erläuterung der Kriterien<sup>20</sup>

Kriterium	Erläuterung
Sprachenabhängigkeit	Die verwendete Methode muss unabhängig von der verwendeten Programmiersprache und der Erfahrung des Projektteams besitzen
Genauigkeit	Als größtmögliche Übereinstimmung der berechneten Soll-Werte aus der Aufwandsschätzung mit den sich im Projektverlauf ergebenden Ist-Werten
Frühzeitigkeit	Die Ausprägungen der relevanten Einflussfaktoren in Aufwandschätzungen sollen in möglichst frühen Phasen des Entwicklungsprozesses für das Projektmanagement erkennbar sein
Detaillierbarkeit	In der Schätzmethode muss die Berechnung bis auf die Ebene kleiner überschaubarer Einzelaktivitäten untergliedert werden können
Eindeutigkeit	Als Eigenschaft der Schätzmethode, dass bei gleichartiger Anwendung der Schätzverfahren durch unterschiedliche Personen das gleiche Ergebnis für den Aufwand ermittelt wird
Objektivität	Keine Einbeziehung von Faktoren in die Aufwandsschätzung, die nur durch subjektive Einschätzung bestimmt werden können
Transparenz	Die Darstellung und die Kalkulation im Schätzverfahren muss für Dritte inhaltlich und rechnerisch interpretationsfrei nachvollziehbar sein
Flexibilität	Das Projektmanagement-Verfahren der Aufwandsschätzung muss zu unterschiedlichen Zeitpunkten bei unterschiedlichem Kenntnisstand innerhalb des Entwicklungsprozesses eingesetzt werden können
Benutzerfreundlichkeit	Das Verfahren für die Schätzung muss einfach anwendbar sein, standardisiert werden können und nur mit den verfügbaren Parametern auskommen, die auch nachweislich einen relevanten Einfluss auf die Kalkulationswerte der Schätzung haben

### Bewertungslegende

- ↑ sehr gut
- ↗ gut
- mäßig
- ↘ schlecht
- ↓ sehr schlecht

<sup>20</sup> Übernahme und Ergänzung der Kriterien von [http://www.infforum.de/themen/projektmanagement/thema\\_PM\\_aufwandsschaetzung.htm](http://www.infforum.de/themen/projektmanagement/thema_PM_aufwandsschaetzung.htm)

## Literaturverzeichnis

- Ange2006 Angermeier, Georg: Aufwandserfassung, Definition im Projektmanagement-Glossar.  
<http://www.projektmagazin.de/glossar/gl-0636.html> 2006, Abruf am 2008-01-03. (Abruf kostenpflichtig)
- Dorl2004 Dorloff, Frank-Dieter: Grundlagen rechnergestützter betrieblicher Informationssysteme, Kapitel 3: Aufwandsschätzung (Skript zur Vorlesung im Sommersemester 2004), Essen 2004.
- GrJS2003 Gruner, Katrin; Jost, Christian; Spiegel, Frank: Controlling von Softwareprojekten. Friedr. Vieweg & Sohn Verlag / GWV Fachverlage GmbH, Wiesbaden 2003.
- Schä2002 Schätz, Bernhard: Softwaresystemmaße am Beispiel von CoCoMo II. <https://www4.in.tum.de/misc/perlen/perlen-folien/CoCoMo-Schaetz.pdf>, 2002-11-05, Abruf am 2007-12-28.
- Wiki01 Wikipedia: COCOMO. <http://de.wikipedia.org/wiki/COCOMO>, 2007-11-21, Abruf am 2007-01-02.
- Wiki02 Wikipedia: Gant Diagramm.  
[http://de.wikipedia.org/wiki/Bild:Gantt\\_diagramm.png](http://de.wikipedia.org/wiki/Bild:Gantt_diagramm.png), 2004-07-02, Abruf am 2007-01-03
- ZuGK2004 Wolfgang Zuser; Thomas Grechenig; Monika Köhle: Software Engineering mit UML und dem Unified Process. Pearson Education Deutschland GmbH, München 2004.

## Eidesstattliche Erklärung

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

---

Ort, Datum

---

Unterschrift