FOM Fachhochschule für Oekonomie & Management
Leverkusen

Berufsbegleitender Studiengang zum

Bachelor of Science (B.Sc.) Wirtschaftinformatik

3. Semester

Hausarbeit im Schwerpunktfach General Studies I

Aufwandschätzung von Softwareprojekten

Betreuer(in): Dipl.-Inf. Ullrich Krause
Autor: Andreas Pelekies
Kempener Str. 203
51467 Bergisch Gladbach
Matrikelnummer.: 211572
Bergisch Gladbach, den 12.01.2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Das Geld und die Zeit sind zwei wichtige Ressourcen eines Unternehmens. Jedes wirtschaftlich arbeitende Unternehmen versucht diese begrenzten Mittel bestmöglich einzusetzen. In der heutigen Zeit wird neben der umfassenden Integration von Software in Produkten auch ein großer Anteil der unternehmensinternen Prozesse softwareseitig unterstützt. Die Bedeutung von Standard- und Spezialsoftware ist so groß, dass der Ausfall von Software oder das Scheitern von Softwareprojekten die wirtschaftliche Existenz von Unternehmen gefährden kann. Dies ist insbesondere in der geplatzten Blase der Dotcom Unternehmen zu Beginn dieses Jahrtausends deutlich geworden. Trotzdem beginnt die meiste Literatur zum Thema Projektmanagement bei Softwareprojekten mit Schlagworten wie „Chaos in der Softwarebranche ?“[1]. Viele Softwareprojekte werden nicht in der dafür vorgesehenen Zeit, im vorgesehenen Kostenrahmen oder mit dem vollen ursprünglich geplanten Leistungsumfang abgeschlossen. Als ein berühmtes Beispiel für ein Projekt, dass nur mit großer Verzögerung eingeführt wurde, sei die Einführung der LWK-Maut (Toll Collect) genannt. Die niedrigen Erfolgsraten von IT-Projekten belegen diverse Studien. Diese erstellt zum Beispiel das amerikanische Unternehmen Standish Group in regelmäßigen Abständen.

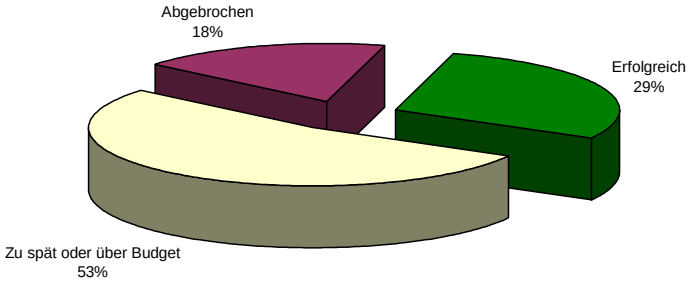

Abbildung 1: Erfolgsraten von IT-Projekten[2]

Nach der von der Standish Group im Jahr 2004 durchgeführten Studie überschreitet mehr als die Hälfte aller Projekte den geplanten Rahmen, ein Sechstel scheitert vollständig. Nur ein Drittel der Projekte wird erfolgreich abgeschlossen. Doch warum ist das so? Nach der Klärung begrifflicher Grundlagen geht diese Hausarbeit darauf ein, was Softwareprojekte von übrigen Projekten unterscheidet und versucht so darauf eine Antwort zu finden. Im Anschluss werden praxisübliche Methoden zur Aufwandschätzung vorgestellt. Abschließend erfolgt jeweils eine Bewertung der vorgestellten Methoden.

---

[1] Hindel, B. u.a. (2006), S. 1

[2] In Anlehnung an Standish Group 2004; Studie mit ca. 10.000 Projekten, 58% in USA, Rest außerhalb; vgl. Ebert, C. (2006), S. 2

2 Begriffliche Grundlagen

2.1 Projekt

Der Begriff des Projektes ist in den alltäglichen Sprachgebrauch übergegangen. Neben der Planung des Baus eines Gartenhäuschens wird oft schon der Anruf bei einer Service-Hotline als Projekt aufgefasst. Daher ist eine genaue Definition des Begriffes notwendig: Ein Projekt ist ein anspruchsvolles Vorhaben, das Limitationen der Zeit, der Finanzen und anderen Ressourcen unterliegt. Es besitzt einen definierten Startpunkt und einen definierten Endtermin. Es ist losgelöst von anderen Tätigkeiten zu betrachten und ist durch seine Einmaligkeit der Bedingungen unter denen es durchgeführt wird ausgezeichnet. Hierzu zählen beispielsweise die an der Umsetzung mitwirkenden Organisationen und Ressourcen.[3] Ein wichtiger Aspekt von Projekten ist der, dass Projekte fehlschlagbehaftet sind, wie es schon in der Einleitung beschrieben wurde. Auf Grund dieser Definition ist der Anruf bei einer Service-Hotline mit Sicherheit nicht als Projekt einzustufen.
Der Ablauf eines Softwareprojektes wird in mehrere Phasen aufgeteilt die nach dem Wasserfallmodell chronologisch durchlaufen werden.

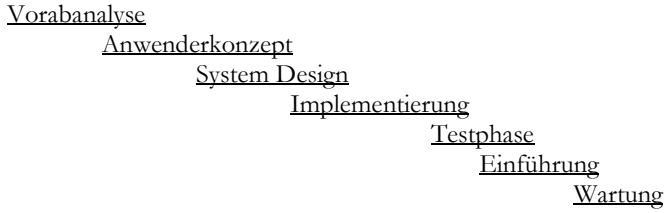

Abbildung 2: Wasserfallmodell

2.2 Was ist eine „Schätzung"?

Um in das Thema der Aufwandschätzung einsteigen zu können, bedarf es der Klärung des Begriffs Schätzung: Eine Schätzung ist ein „Verfahren zur annähernden Bestimmung unbekannter … Parameter … auf der Grundlage vorhandener Zahlenangaben …".[4] Bezogen auf ein Projekt handelt es sich also um eine Vorhersage über Dauer und Kosten eines Projektes[5]. Um zu dieser Vorhersage zu gelangen gibt es mehrere Methoden, von denen drei Kategorien kurz angesprochen werden sollen:

- Analogiemethoden - Schätzen durch Vergleichen/Expertenbefragung
  Das ausführende Unternehmen hat bereits vergleichbare Projekte durchgeführt und kann so aufgrund vorhandener Zahlenangaben das neue Projekt schätzen. Zusätz-

[3] Vgl. Hindel, B. u.a. (2006), S. 8
[4] o.V. (2008-1), o.S.
[5] Vgl. McConnell, S. (2006), S. 33

lich können externe Berater mit hinzugezogen werden, die vergleichbare Projekte bereits durchgeführt haben.

- Algorithmische Methoden

  Der zu erwartende Aufwand wird mittels einer mathematischen Formel ermittelt. Diese Formel ist im Vorfeld aufgrund empirischer Untersuchungen oder mathematischer Modelle entstanden. Sie ermittelt anhand von Gewichtungen oder Stichproben den zu erwartenden Aufwand.[6]

- Schemabasierte Methoden

  Neben einigen weiteren Methoden zählt die COCOMO-Methode (Constructive-Cost-Model) und die Function-Point-Analyse zu den verbreiteten Software- bzw. schemabasierten Methoden der Kostenschätzung. Hierbei kombiniert die Function-Point-Analyse sowohl den Bereich der algorithmischen als auch den der Kennzahlenmethoden. Auf die Anwendung der drei Kategorien wird ab Kapitel 3 genauer eingegangen.

Die Abschätzung von Aufwand und Kosten ist nur dann sinnvoll, wenn die Schätzung eine möglichst hohe Präzision hat, also möglichst gut ist. Als gute bis sehr gute Schätzung zählt eine Abweichung von ± 10%, die aber nur bei gut überwachten Projekten erreicht werden kann.[7] Jedoch stimmen Softwareprojekte selten in Ihrem Umfang des ersten Konzeptes und der später umgesetzten Version überein. Die Berücksichtigung dieser Dynamik ist entscheidend für die Qualität der jeweiligen Schätzung. Das Ziel einer möglichst präzisen Aufwandschätzung unter der Berücksichtigung praxistauglicher Gesichtspunkte gibt die folgende Definition wieder:

„Eine gute Aufwandschätzung ist eine Schätzung, die einen Blick auf das Projekt ermöglicht, der klar genug dafür ist, dass die Projektleitung gute Entscheidungen zur Steuerung des Projekts fällen kann, um so die Ziele des Projekts zu erreichen.“ [8]

2.3 Was Softwareprojekte von anderen Projekten unterscheidet

Verglichen mit dem Bau eines Gartenhäuschens gestaltet sich die Durchführung eines Softwareprojektes ungleich schwieriger. Dies liegt unter anderem daran, dass hier ein immaterielles Produkt erstellt wird, „dessen Fertigstellungsgrad und dessen Qualität nur sehr

---

[6] Vgl. Mertens, P.; Wieczorrek, H.W. (2007), S. 212f.
[7] Vgl. McConnell, S. (2006), S. 40
[8] Ders. (2006), S. 45

schwer visuell wahrgenommen werden können“ [9]. Insbesondere seien die folgenden sechs Punkte genannt: [10]

- Das Festlegen von Methoden zur akkuraten Aufwandschätzung ist schwierig.
  Bedingt durch den rasanten Technologiewechsel im Bereich der Softwareerstellung und der sich rasch ändernden Anforderungen werden für die Aufwands- und Kostenschätzung von Softwareprojekten spezielle Methoden benötigt, die diesem Umstand gerecht werden können.
- Schwere Einschätzbarkeit des bisherigen Projektfortschritts
  Eine mangelhafte Planung in einer frühen Phase des Softwareprojektes suggeriert häufig einen zu weiten Projektfortschritt. So galt ein Projekt zur Erstellung einer Datenschnittstelle zwischen zwei auf dem Markt befindlichen Standard-Softwareprodukten bereits nach vier Wochen als fast abgeschlossen. Jedoch zeigte sich dann in der Testphase, dass es im Zusammenbringen der beiden unterschiedlichen Architekturen einen derart schwerwiegenden Mangel gab, dass sich die Gesamtprojektzeit auf über neun Monate verlängerte.
- Qualitätssicherung während des Projekts ist schwierig
  Eine frühzeitige, projektbegleitende Qualitätssicherung ist somit sinnvoll. Dies ist aber gerade aufgrund der Immaterialität der Software visuell nur sehr schwer möglich und bedarf daher spezieller Techniken um die Qualität messbar zu machen (z.B. durch Metriken, Reviews, Inspektionen, Walk Throughs, Projektcontrolling u.a.). Auf diese soll in dieser Arbeit jedoch nicht eingegangen werden.
- Schwere Verständlichkeit der Zusammenhänge
  In der Regel handelt es sich zumindest bei den Auftraggebern der Projekte um IT-Laien. Genauso wichtig wie für diese Gruppe die Kenntnis des Projektfortschritts ist, so schwierig ist es die Zwischenergebnisse nachzuvollziehen. Dies kann zu Interessenkonflikten und Misstrauen im Ablauf des Projektes führen.
- Dynamische Änderung der Ziele während der Projektphase
  Eine Besonderheit bei Softwareprojekten ist, dass sich die Anforderungen an die zu erstellende Software häufig während des Projektes ändern. „[Die] .. Kunden wären nicht einmal erfreut, wenn .. [es] so verlaufen würde, weil die Kunden zu dem Zeitpunkt, an dem sie die Software bekommen, nicht mehr wollen, was geplant war,

[9] Henrich, A. (2004), KE 1 S. 13
[10] Vgl. ders. (2004), KE 1 S. 13ff.

sondern etwas anderes.“ [11] Der daraus resultierende Zusammenhang zwischen Anforderungen und Kosten ist dem Anwender nur schwer zu vermitteln. So kann z.B. die Ansprungreihenfolge von Feldern auf einer Maske für den Anwender eine immense Bedeutung haben, für den Programmierer jedoch mit geringem Aufwand verbunden sein. Andererseits stellt das zusätzliche Einblenden eines weiteren Datums auf der Maske für den Anwender ein *wäre schön*-Kriterium dar, welches für den Entwickler jedoch mit tagelanger Arbeit verbunden ist, weil diese Daten in der vom Anwender benötigten Form nicht zur Verfügung stehen. Die Behandlung solcher sich ändernden Anforderungen können durch ein Claim-/ und Changemanagement abgefangen werden.

- Die Arbeit ist nachträglich nur noch schwer teilbar
  Als sechster Punkt sei die häufige Unteilbarkeit der Arbeit bei der Softwareentwicklung genannt. Oftmals ist die Entwicklung einzelner Module eines Softwareprojektes sehr komplex. Gerät ein Projekt in Zeitverzug, führt die zur Verfügungsstellung neuer Mitarbeiter statt zu einer Beschleunigung zu weiterer Verzögerung. Dies liegt darin begründet, dass die bestehenden Mitarbeiter Anteile Ihrer Arbeitszeit für die Einarbeitung der neuen Mitarbeiter in die jeweiligen Themen verwenden müssen und der erhoffte Mehrwert erst mittel- oder langfristig eintreten kann.

3 Kosten- und Aufwandschätzung

Ein Hauptziel der Kosten- und Aufwandschätzung ist es den zeitlichen und den finanziellen Rahmen eines Projektes zu ermitteln. Jedoch geschieht dies nicht zum Selbstzweck. Erst auf Grund dieser Schätzungen kann die Planung des Gesamtprojektes durch eine sinnvolle Ressourcenplanung, zum Beispiel der Personalplanung für das Projekt, vervollständigt werden. Die Grundlage der Zeitplanung stellt der Personenmonat(PM) als Maßeinheit dar. Eine mögliche Definition erfolgt im Kapitel 3.2. Nach der Ermittlung des Projektumfangs in Personenmonaten stellt sich als nächstes die Frage nach der Projektdauer. Um zu dieser zu gelangen ist es jedoch nicht möglich die ermittelten Personenmonate einfach durch die zur Verfügung stehenden Personen zu teilen. Rein rechnerisch könnte so ein Projekt mit einem Umfang von 100 Personenmonaten durch den Einsatz von 50 Mitarbeitern in zwei Monaten abgearbeitet werden. Dieses Vorgehen ist bei der Erstellung von Software unrealistisch. Neben dem Projektumfang und den damit verbundenen Kosten liefern die folgenden Methoden auch die optimale Projektdauer als Ergebnis.

---

[11] Beck, K., Fowler, M. (2001), S. xiii

3.1 Die Schätzung „nach Erfahrung"

Eine naheliegende Methode zur Abschätzung eines Projektaufwandes besteht im Heranziehen vergleichbarer Projekte. Diese Projekte können bereits im projektdurchführenden Unternehmen abgeschlossen sein (Analogiemethode), oder es können Experten hinzugezogen werden, die auf entsprechende Erfahrung zurückblicken können.

Gerade in kleinen und mittleren Unternehmen (KMU) wird diese Methode häufig verwendet. Über die Vielzahl bereits abgeschlossener Projekte hinweg ist ein Wissens- und Erfahrungspool vergleichbarer Anforderungen entstanden. Um den Prozess der Projektkalkulation zu verdeutlichen sei die Anpassung kaufmännischer Software anhand von SAGE-Produkten aufgeführt. Diese Projekte haben selten eine Größenordnung von mehr als einem Personenmonat Aufwand. Folgende Anforderung lag zu Grunde:

Ein langjährig auf dem Markt befindliches Produkt wird um eine Zusatzleistung ergänzt. Diese Ergänzung wird durch eine andere Abteilung als die mit der Erstellung des Produktes beauftragte Abteilung erbracht. Der Kunde soll auf der Abrechnung keinen Unterschied erkennen können. Intern jedoch soll der Produktpreis nun anteilig auf die unterschiedlichen Kostenstellen aufgeteilt werden.

Zur Erstellung des Angebotes wurde folgendes Kalkulationsblatt herangezogen:

Kalkulation "Kostenstellenaufteilung"

Kunde:	**XY GmbH**
Projektnummer:	**2007-15233-3**
Zieltermin:	**27. Apr 07**
Tagessatz:	1.100,00 €

lfd Nr.	*Bezeichnung*	*Anzahl/MT*	*Wert*	*Bemerkung*
1	Erweiterung Artikelstamm			
1.1	Maske	0,5	550,00 €	
1.2	Datenbank	0,2	220,00 €	
2	Umbuchungsfunktion	2,5	2.750,00 €	**Updatesicher!**
3	Datenpflege	1	1.100,00 €	
4	Test	0,75	825,00 €	
5	Dokumentation	0,5	550,00 €	
6	Anwenderschulung	0,5	550,00 €	
	Summe	5,95	6.545,00 €	
	Sicherheit 20%	1,19	1.309,00 €	
	Errechneter Wert	**7,14**	**7.854,00 €**	
	Angebotswert:		**7.800,00 €**	

Abbildung 3: Kalkulationsblatt "Kostenstellenaufteilung"

Das Projekt wurde innerhalb einer Kalenderwoche durchgeführt und beim Anwender implementiert.

Die Qualität dieser Methode steigt und fällt mit dem Grad der Erfahrung der abschätzenden Personen. Bei entsprechender Expertise ist die Qualität in aller Regel sehr hoch. Die

Gefahr liegt jedoch darin begründet, dass insbesondere beim Einsatz neuer Technologien die bisherigen Erfahrungswerte nicht ohne weiteres übertragen werden können. So kann auch ein eingeräumter Puffer von 20% nicht genügen, um alle Unabwägbarkeiten abzudecken. Als Beispiel sei hier erneut die Schnittstellenentwicklung zwischen zwei Standard-Softwareprodukten angeführt. Das Softwarehaus hat selbst Expertise in der Anpassung jedes der beiden Softwareprodukte. Beide basieren jedoch auf unterschiedlichen Technologien und werden von unterschiedlichen Abteilungen betreut, so dass jeweils Experten vorhanden sind. Die entscheidende Neuerung war also die Verbindung zwischen diesen beiden Technologien. Die Unterschätzung der mangelnden Erfahrung in der Kombination der beiden Technologien führte zu immensen Zeit und Budgetüberschreitungen.

3.2 Das Constructive-Cost-Model (COCOMO)

Bereits in den 70er Jahren des vorherigen Jahrhunderts kam die Forderung nach Methoden und Modellen zur Aufwandschätzung von Softwareprojekten auf. Aus dieser Anfangszeit stammt die COCOMO-Methode, die Anfang der 80er Jahre von Barry Boehm begründet wurde und noch bis heute als eine der am besten dokumentierten Methoden gilt. Im Rahmen der Entwicklung der objektorientierten Programmierung wurde das Modell Anfang der 90er Jahre zu COCOMO2 erweitert.[12] Insgesamt ist das Modell derart komplex, dass eine umfassende Betrachtung den Rahmen dieser Hausarbeit sprengen würde. Aufgrund der hohen Bedeutung soll es dennoch in Auszügen vorgestellt werden.

Die Grundlage des COCOMO-Verfahrens ist der Programmumfang. Basierend auf der Anzahl der Programmzeilen wird dann der für die Erstellung des Programms benötigte Aufwand berechnet. Hier stellt sich jedoch die Frage, in wie weit die Ermittlung des Programmumfangs einfacher ist, als die Abschätzung des Gesamtkosten.

In der Praxis werden nur sehr selten Programme vollständig neu entwickelt. Häufig können vorhandene Funktionen verwendet, kombiniert oder angepasst werden. Auf diese Art lässt sich der Umfang des zu erstellenden Programms in der Regel relativ einfach abschätzen. Jedoch gibt es auch Verfahren, wie zum Beispiel die in Kapitel 3.3 beschriebene Function-Point-Analyse, die zur methodischen Ermittlung eines Programmumfangs herangezogen werden können.

---

[12] Vgl. o.V. (2008-2), o.S.

3.2.1 Die Rahmenbedingungen für COCOMO[13]

Unter Berücksichtigung von Krankheit und Urlaub umfasst nach Boehm ein Arbeitsmonat 152 Arbeitsstunden, die an 19 Arbeitstagen erbracht werden. Pro Jahr existieren 12 Arbeitsmonate.

Bei der Berücksichtigung der Anzahl der Programmzeilen werden Kommentare sowie Zeilen, die ausschließlich während der Entwicklung benötigt werden (z.B. zu Testzwecken) nicht beachtet. Die Aufwandschätzung umfasst alle Phasen ab dem Anwenderkonzept bis hin zur Einführung. Dabei werden auch die für das Projektmanagement benötigten Zeiten mit berücksichtigt.

COCOMO versucht zusätzlich viele Einflussfaktoren beginnend bei der Art der zu erstellenden Software bis hin zu der Qualifikation der beteiligten Personen zu beachten. Daher hat Boehm mehrere Modelltypen eingeführt:

- das COCOMO-Basismodell
- das COCOMO-Zwischenmodell
- das detaillierte COCOMO-Modell.

3.2.2 Das COCOMO-Basismodell[14]

Das Basismodell startet mit einer ersten Abschätzung der erforderlichen Personenmonate (PM) aufgrund der zu erstellenden Anzahl an Programmzeilen in Tausender-Einheiten (kilos of delivered source instructions).

$$PM = 2{,}4 * (KDSI)^{1{,}05}$$

Boehm hat diese Formel aus der Analyse von 63 abgeschlossenen Projekten abgeleitet. Sie besagt, dass bei steigender Anzahl an Programmzeilen der Aufwand überproportional wächst, da ein entsprechend höherer Bedarf an Abstimmungsaufwand mit berücksichtigt werden muss.

Aus dieser Formel hat Boehm eine Formel für die Projektdauer (time for development) abgeleitet. Dabei wird angenommen, dass ein Projekt eine optimale Dauer dadurch auch eine optimale Zusammensetzung des Projektteams erhält.

$$TDEV = 2{,}5 * (PM)^{0{,}38}$$

Die so ermittelte Gesamtdauer eines Projektes ist noch nicht sehr aussagekräftig. Um eine genauere Planung machen zu können, die mit Meilensteinen kontrolliert werden kann, benötigt man die Information, wie sich diese Zeit auf die einzelnen Phasen aufteilt. Dazu

---

[13] Vgl. Henrich, A. (2004), KE 3 S. 65ff.
[14] Vgl. ders. (2004), KE3 S. 67ff.

bietet Boehm Tabellen an, die abhängig von der zu erstellenden Produktgröße die Verteilung der Projektdauer bzw. des Projektaufwands auf die Phasen beschreibt.

Phase/Größe	*2 KDSI*	*8 KDSI*	*32 KDSI*	*128 KDSI*
Pläne und Anforderungen	10%	11%	12%	13%
Produktentwurf	19%	19%	19%	19%
Programmierung	63%	59%	55%	51%
Integration und Test	18%	22%	26%	30%

Tabelle 1: Verteilung der Projektdauer auf die Phasen[15]

Phase/Größe	*2 KDSI*	*8 KDSI*	*32 KDSI*	*128 KDSI*
Pläne und Anforderungen	6%	6%	6%	6%
Produktentwurf	16%	16%	16%	16%
Programmierung	68%	65%	62%	59%
Integration und Test	16%	19%	22%	25%

Tabelle 2: Verteilung des Aufwands auf die Phasen[16]

Zu beachten ist, dass der echte Aufwand höher ist, da COCOMO erst nach der Phase Pläne und Anforderungen einsteigt und daher die hier angegebenen Werte noch hinzuaddiert werden müssen. So liegt der Gesamtaufwand für ein Projekt also bei 106% und die Projektdauer liegt zwischen 110% und 113%.

Dieses noch sehr grobe Modell lässt sich durch drei Submodelle verfeinern, in denen die äußeren Einflussfaktoren auf das Projekt mitberücksichtigt werden[17]:

- Organische Projekte

  Dieses Modell umfasst relativ kleine Projekte (bis 50 KDSI), bei denen jeder Mitarbeiter das gesamte Projekt kennt.

- Teilintegrierte Projekte

  Mittelgroße Projekte (50 bis 300 KDSI), bei denen mehrere Entwickler mit Spezialwissen in einem inhomogenen Team zusammenarbeiten, werden von diesem Modell berücksichtigt.

- Eingebettete Projekte

  Für große Projekte (ab 300 KDSI) mit einer großen Anzahl an Entwicklern und nur einer geringen Anzahl an Analytikern ist dieses Modell gedacht.

---

[15] Vgl. Henrich, A. (2004), KE 3 S. 70
[16] Vgl. ebd.
[17] Vgl. Werdenich, K. (2002), S.18

Diese Unterscheidung führt zu einer Anpassung der Faktoren in der Grundformel, die den höheren Abstimm- und Kommunikationsbedarf in großen Projekten berücksichtigt.

Projektart	*Aufwand*	*Dauer*
Organisch	$PM = 2{,}4 * (KDSI)^{1{,}05}$	$TDEV = 2{,}5 * (PM)^{0{,}38}$
Teilintegriert	$PM = 3{,}0 * (KDSI)^{1{,}12}$	$TDEV = 2{,}5 * (PM)^{0{,}35}$
Eingebettet	$PM = 3{,}6 * (KDSI)^{1{,}20}$	$TDEV = 2{,}5 * (PM)^{0{,}32}$

Tabelle 3: Anpassung der Berechnung nach Projektart

Natürlich genügt das Basismodell nicht, um die Qualität der Schätzung an die veränderliche Realität besser anzupassen. Daher wurde das Modell verfeinert. Da die detaillierte Erläuterung dieser Verfeinerungen jedoch den Rahmen dieser Arbeit sprengen würde, seien sie hier nur kurz angerissen.

3.2.3 COCOMO-Zwischenmodell (Intermediate)[18]

Der erste Grad der Verfeinerung des Basismodells liegt in der Überführung zum Zwischenmodell. Die beiden Hauptunterschiede liegen darin, dass zum einen zusätzliche Einflussfaktoren mit berücksichtigt werden; zum anderen ein Projekt in Teilkomponenten unterteilt werden kann, die jeweils einzeln nach den unterschiedlichen Projektarten kalkuliert werden. Anschließend wird der Gesamtumfang Bottom-Up zusammengefasst, so dass sich wieder eine Kalkulationsbasis ergibt. Zu den fünfzehn neuen Einflussfaktoren zählen unter anderem

- ein Mindestmaß an Zuverlässigkeit der erstellten Software
- der Umfang der Datenbasis
- die Leistungsfähigkeit der Programmierer.

3.2.4 Das detaillierte COCOMO[19]

Die nächste Verfeinerungsstufe findet in der Form des detaillierten COCOMO statt. Hier kommt neben weiteren Einflussfaktoren auch die Möglichkeit hinzu, die einzelnen Phasen je Modul unterschiedlich zu gewichten. Als Beispiel der zusätzlichen Faktoren sei die Berücksichtigung der Anteile an aus anderen Projekten wieder verwendbarem Quellcode genannt. Ein weiterer Faktor stellt die Erfahrung der Entwickler mit der zu verwendenden Programmiersprache dar.[20]

---

[18] Vgl. Werdenich, K. (2002), S.26ff.
[19] Vgl. ders., S. 28f.
[20] Vgl. Henrich, A. (2004), KE 3 S. 83f.

3.2.5 COCOMO2

Die zweistufige Erweiterung des COCOMO-Basismodells musste mit den neuen Anforderungen der objektorientierten Programmierung angepasst werden. Der Unterschied liegt zum einen in der höheren Kapselungsmöglichkeit objektorientierter Programm. Zum anderen ermöglicht diese Sprache die Vererbung, also die Möglichkeit bestehende Programmmodule ohne Änderung des Programmtextes um weitere Eigenschaften zu erweitern. Im gleichen Rahmen wurde darauf geachtet, dass mit COCOMO2 jedes Projekt ein maßgeschneidertes Submodell erhalten kann, dass möglichst dicht an den spezifischen Einflussfaktoren orientiert ist.

3.2.6 Vom Aufwand zu den Kosten

Möchte man nun zu einer Kostenschätzung gelangen, kann zum Beispiel das Durchschnittsgehalt der beteiligten Personen als Verrechnungsfaktor herangezogen werden. In einer aktuellen Studie der Linux Foundation ist die COCOMO-Methode zur Nachkalkulation des Wertes von Linux-Kernels herangezogen worden.[21]

Gesamtzahl Programmzeilen (DSI)	6.772.902	
Entwicklungsaufwand (angepasstes teilintegriertes Modell) $PM = 4.64607 * (KDSI)^{1,12}$	90688,77 Monate	7557,4 Jahre
Projektdauer (COCOMO-Basismodell) $TDEV = 2,5 * (PM)^{0,38}$	191,34 Monate	15,95 Jahre
Durchschnittsanzahl Entwickler (geschätzt)	473,96	
Geschätzte Entwicklungskosten (Durchschnittsgehalt $75,662.08/Jahr, Modifizierer 2,40)	$1.372.340.206	~ 1.017.000.000 €

Tabelle 4: Der Wert des Linux-Kernels[22]

3.2.7 Zur Qualität von COCOMO

Die Hauptschwierigkeit beim COCOMO liegt in der Komplexität. Der dadurch verbundene Aufwand schreckt viele, vor allem kleine Firmen ab, dieses Modell einzusetzen. Auch die Notwendigkeit, das Modell jeweils je Projekt wieder an die entsprechenden Gegebenheiten anpassen zu müssen, ist ein Hemmnis in der Verbreitung. Richtig angewendet liefert es jedoch eine relativ genaue Prognose. Als Beispiel wurde ein Projekt mit einem Umfang von 41 KDSI nach der in Tabelle 4 beschriebenen Art nachkalkuliert. Nach COCOMO ergab sich eine errechnete Projektdauer von 22 Monaten. Die reale Projektdauer betrug 23 Monate, was einer Abweichung von etwa 5 % entspricht.

---

[21] Vgl. McPherson, A. u.a. (2008), o.S.
[22] Vgl. ebd.

3.3 Function-Point-Verfahren

COCOMO verlangt bereits in einer sehr frühen Phase die Berücksichtigung detaillierter technischer Kenntnisse. Die Schätzung des Programmumfangs fällt häufig schwer, wenn das System Design selbst noch nicht abgeschlossen ist. Das Function-Point-Verfahren setzt daher schon einen Schritt eher an. Es ermittelt die entscheidenden Punkte aus den zu erstellenden Funktionen in fünf Schritten[23]:

- Ermitteln der Roh-Function-Points
- Bewertung der Funktionskategorien
- Berücksichtigung von Einflussfaktoren
- Bestimmung der Gesamt-Function-Points
- Aufwandschätzung

3.3.1 Ermitteln der Roh-Function-Points[24]

Zur Ermittlung der Roh-Function-Points wird die Projektaufgabe zunächst fachlich strukturiert und anschließend die Anzahl der für die jeweilige Umsetzung benötigten Function-Points ermittelt. Die Grundkategorien lauten:

- (externe) Eingabedaten
- (externe) Ausgabedaten
- (externe) Abfragen
- (interne) Datenbestände und
- Referenzdaten

Zu den externen Eingabedaten gehören alle Dateneingaben, die das System verarbeiten können soll. Führen mehrere unterschiedliche Dateneingaben zu einer logisch gleichen Abarbeitung, werden sie nur einmal berücksichtigt.

Bei den externen Ausgabedaten kann es sich um Listenausgaben, Bildschirmausgaben oder auch Schnittstellen zu anderen Anwendungen handeln. Hierbei werden nur diejenigen Ausgaben berücksichtigt, die nach der erfolgten Eingabe durch das Modul verändert wurden. Ein reines Durchreichen der Daten führt nicht zu einem Function-Point.

Elementare Prozesse, die Anfragen an das System stellen, werden als Abfragen verstanden. Erfordert zum Beispiel die Suchanfrage durch einen Benutzer ebenfalls eine Ein- und eine Ausgabe, so sind diese getrennt zu berücksichtigen.

---

[23] Vgl. Mertens, P.; Wieczorrek, H.W. (2007), S. 215 ff.
[24] Vgl. ebd.

Alle nicht temporären Datenbanken und sonstige Anwenderdaten, die von dem zu erstellenden System gepflegt werden, zählen zu den Datenbeständen.
Schließlich umfassen die Referenzdaten alle Daten die lediglich gelesen, aber nicht geschrieben werden.

3.3.2 Bewertung der Funktionskategorien

Da nicht alle Projekte in jeder der fünf Kategorien gleich kompliziert sind, werden jede der fünf Kategorien in drei unterschiedliche Komplexitätsklassen eingeteilt. So macht es für den Umfang der Entwicklung einen Unterschied, ob zum Beispiel nur eine Art von Datenbank angesprochen werden soll, oder vier verschiedene Datenbanken.

Kategorie	*einfach*	*mittel*	*komplex*
Eingabedaten	3	4	6
Ausgabedaten	4	5	7
Abfragen	3	4	6
Datenbestände	7	10	15
Referenzdaten	5	7	10

Tabelle 5: Funktionspunkte nach Komplexität[25]

Nachdem die zu erstellenden Funktionen jeweils je Kategorie in eine der drei Klassen eingeteilt wurden, wird ihre Anzahl mit dem jeweiligen Function-Point aus Tabelle 5 multipliziert. Die Summe S1 der hierdurch entstehenden Einzelergebnisse ergibt das Resultat dieses Schrittes.

3.3.3 Berücksichtigung von Einflussfaktoren[26]

Im nächsten Schritt werden die Einflüsse weiterer Faktoren berücksichtigt und auf einer Skala von 0 bis 5 bewertet:

- 0 = kein Einfluss
- 1 = gelegentlicher Einfluss
- 2 = mäßiger Einfluss
- 3 = mittlerer Einfluss
- 4 = bedeutender Einfluss
- 5 = starker Einfluss

[25] Vgl. Mertens, P.; Wieczorrek, H.W. (2007), S. 218
[26] Vgl. ders. (2007), S. 218f.

Es werden insgesamt zehn Einflussfaktoren unterschieden, von denen die ersten beiden doppelt gezählt werden:

- Schwierigkeit und Komplexität der Rechenoperationen (doppelte Bewertung)
- Anzahl der Ausnahmeregelungen (doppelte Bewertung)
- Verflechtung mit anderen IT-Systemen
- Dezentrale Verarbeitung und Datenhaltung
- erforderliche Maßnahmen der IT-Sicherheit
- Performance des umzusetzenden IT-Systems
- Datenbestandskonvertierungen
- Benutzer- und Anwenderfreundlichkeit
- Komplexität und Schwierigkeit der Logik
- Wiederverwendbarkeit von einzelnen Komponenten

Durch die Aufsummierung der einzeln skalierten Bewertungen ergibt sich eine Summe S2 zwischen 0 und 50. Diese wird nun verwendet um die Summe S1 um bis zu 30% auf- bzw. abzuwerten. Als Ergebnis diesen Schrittes ergibt sich:

$$S3 = 0{,}7 + S2/100$$

3.3.4 Bestimmung der Gesamt-Function-Points[27]

Basierend auf den vorangegangenen Schritten kann nun die Gesamtpunktzahl errechnet werden:

Total Function Points: $TFP = S1 * S3$

3.3.5 Aufwandschätzung

Im letzten Schritt können die erforderlichen Personenmonate nach der Formel

$$PM = 0{,}0136 * (TFP)^{1{,}308}$$

errechnet werden.[28] Diese Formel berücksichtigt, genau wie COCOMO, dass der Aufwand größerer Projekte überproportional zu ihrer Größe steigt.

3.3.6 Bewertung

Die Function-Point-Methode ist aufgrund der Nähe zum Lastenheft relativ leicht anwendbar. Ebenso lassen sich kleinere Änderungen der Anforderungen im Projektverlauf relativ einfach integrieren, da sich die Struktur dadurch nicht ändert. Jedoch bedingen die wenigen

---

[27] Vgl. Mertens, P.; Wieczorrek, H.W. (2007), S. 220
[28] Vgl. Henrich, A. (2004), KE 3 S. 102

Korrekturfaktoren auch die Gefahr einer hohen Ungenauigkeit im Gegensatz zu COCOMO.

3.3.7 Ermittlung der KDSI

Allerdings kann die Function-Point-Methode gut zur Abschätzung des Programmumfanges als Basis der COCOMO-Methode verwendet werden. Um mit Hilfe der Function-Point-Methode an die KDSI zu gelangen, brauchen lediglich die beiden Formeln zur Berechnung der Personenmonate gleichgesetzt zu werden[29]:

$$3{,}0\ (\mathrm{KDSI})^{1{,}12} = 0{,}0136\ (\mathrm{TFP})^{1{,}308}$$

$$(\mathrm{KDSI})^{1{,}12} = 0{,}00453\ (\mathrm{TFP})^{1{,}308}$$

$$\mathrm{KDSI} = 0{,}008077\ (\mathrm{TFP})^{1{,}168}$$

$$\mathrm{DSI} = 8{,}077\ (\mathrm{TFP})^{1{,}168}$$

4 Theorie und Praxis – ein Vergleich

Die beschriebenen Szenarien verdeutlichen die Notwendigkeit einer fundierten Aufwandschätzung. Als Vision für eine ideale Projektkontrolle könnte ein Armaturenbrett für Produktmanager[30] dienen, wie es 1995 in einem Artikel der Zeitschrift American Programmer gefordert war.

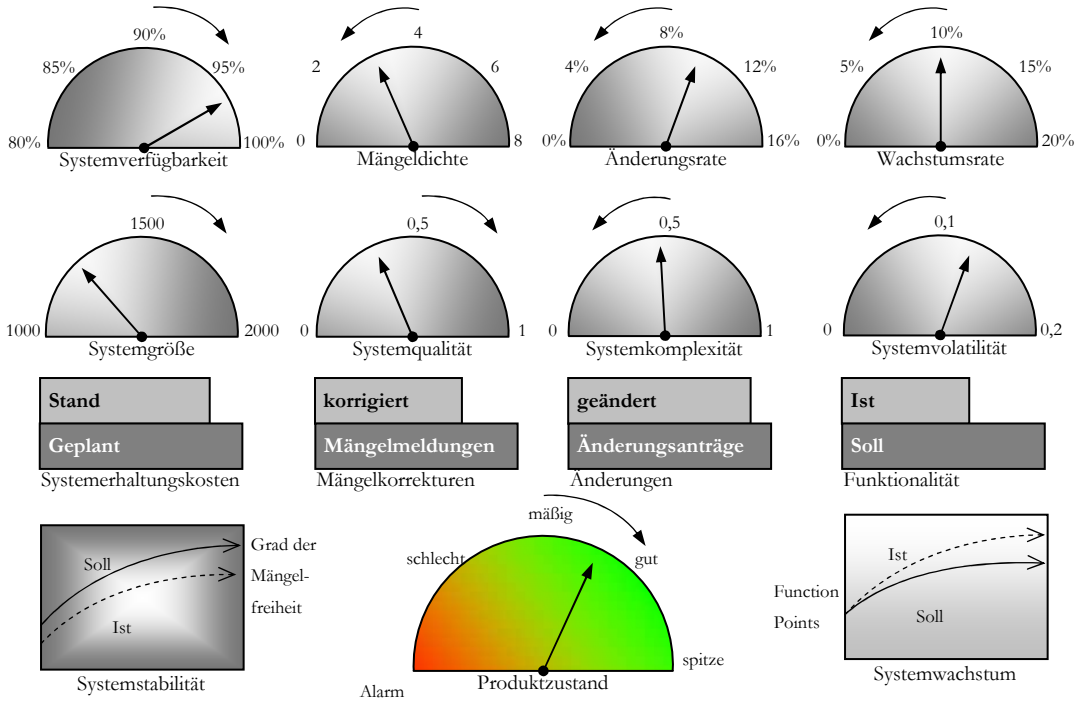

Abbildung 4: Ein Armaturenbrett für Produktmanager[31]

---

[29] Vgl. Henrich, A. (2004), KE 3 S. 111
[30] Vgl. Thomsett, R. (1995), o.S.
[31] Quelle: Sneed, H. M.; Hasitschka, M.; Teichmann, M-T. (2005), S. 63

Die Einführung eines solchen Instrumentes ist jedoch in der Praxis unrealistisch. Das Armaturenbrett kann nur die Daten anzeigen, die zuvor auch erhoben und verarbeitet wurden. Hierfür ist die Einbringung entsprechenden Personals erforderlich, was wiederum Kosten mit sich bringt. Die Softwareentwicklungsbetriebe sind nicht bereit, diese Kosten zu tragen. Zusätzlich ist das Verständnis des Nutzen und der Notwendigkeit solcher Metriken nicht ausreichend verbreitet, so dass eine entsprechende Investition durchgeführt würde.[32]

Die Theorie mit ihren unterschiedlichen Schätzmethoden liefert Ansatzpunkte für eine Abschätzung des Aufwandes. Die Qualität steht und fällt jedoch mit der Ihr zugrundeliegenden Erfahrung aus anderen Projekten. So wird ein neu gegründetes Unternehmen ohne eigene Projekterfahrung rein durch die Anwendung der zur Verfügung stehenden Techniken nur schwer zu realistischen Schätzungen größerer Projekte gelangen. Es fehlt hier die Erfahrung um die Basis zur Anwendung der Methoden zu schaffen. Ein solches Unternehmen wäre in der Anfangszeit auf externe Berater angewiesen.

Um die Qualität der gesamten Aufwandschätzung zu verbessern, bietet sich eine mehrstufige Kombination aller Methoden an[33]:

Zuerst wird eine Schätzung durch Vergleich oder durch einen Experten durchgeführt. Das Ergebnis hiervon sollte sowohl neben einer realistischen auch eine pessimistische sowie eine optimistische Schätzung umfassen.

Als nächstes können Methoden zur Ermittlung des zu erwartenden Quellcodeumfangs herangezogen werden, die dann im dritten Schritt zum Beispiel mit der COCOMO-Methode zu einem geschätzten Zeit- und Kostenrahmen führen. Das Ganze erfolgt erneut unter realistischen, pessimistischen sowie optimistischen Bedingungen.

Ein Vergleich aller Ergebnisse kann nun zur Gesamtabschätzung herangezogen werden. Ergeben die Ergebnisse der jeweiligen Methoden vergleichbare Werte, so kann von einer hohen Qualität des Ergebnisses ausgegangen werden. Weichen die einzelnen Ergebnisse jedoch stark voneinander ab, kann dies ein Hinweis auf Probleme im Konzept geben. In der Praxis ist die Anwendung dieser Kombination sicherlich nur bei größeren Projekten realistisch, weil die Mehrfachausführung der einzelnen Schätzungen mit entsprechend hohen Kosten verbunden ist.

Gerade die Besonderheit der sich dynamisch ändernden Anforderungen während des Projektverlaufes lässt ein solches Vorgehen unrealistisch erscheinen. Es müsste mit jeder auf-

---

[32] Vgl. Sneed, H. M.; Hasitschka, M.; Teichmann, M-T. (2005), S. 64

[33] Vgl. Henrich, A. (2004), KE 3 S. 110ff.

tretenden Änderung erneut durchlaufen werden, um die Konsequenzen für den Projektverlauf nicht aus den Augen zu verlieren. Der Einsatz eines durchdachten Change-/ bzw. Claim-Managements erscheint wesentlich realistischer umsetzbar.

Zusammenfassend lässt sich auf der einen Seite die Notwendigkeit akkurater Aufwandschätzung nicht verleugnen. Auf der anderen Seite ist die Bereitschaft entsprechend in die Aufwandschätzung zu investieren noch nicht vollständig in den Firmen angekommen. Die bestehenden Methoden müssen an die jeweiligen Bedürfnisse angepasst werden, was wiederum mit Kosten verbunden ist. Der Einsatz entsprechender Werkzeuge kann die Kostenschätzung vereinfachen, ersetzt jedoch nicht das benötigte Grundlagenwissung über vergleichbare Projekte. Nur dann können die Werkzeuge effektiv eingesetzt werden. Somit bleibt gerade für kleine Softwareentwicklungsbetriebe mit der Umsetzung kleiner Softwareprojekte mittelfristig die Expertenschätzung die optimale Methodenwahl.

Literaturverzeichnis

Beck, K.; Fowler, M. (2001): Extreme Programming planen, München 2001

Ebert, C. (2006): Risikomanagement kompakt, München 2006

Henrich, A. (2004): Management von Softwareprojekten, Hagen 2004

Hindel, B.; Hörmann, K.; Müller, M.; Schmied, J. (2006): Basiswissen Software-Projektmanagement, 2. Aufl., Heidelberg 2006

McConnell, S. (2006): Aufwandschätzung bei Softwareprojekten, Unterschleißheim 2006

Mertens, P.; Wieczorrek, H.W. (2007): Management von IT-Projekten, 2. Aufl., Heidelberg 2007

Sneed, H. M.; Hasitschka, M.; Teichmann, M-T. (2005): Software-Produktmanagement, Heidelberg 2005

Thomsett, R. (1995): Addes Value Metrics – Adding Meaning to Measurement, in: American Programmer, Bd. 8, Nr. 12, 1995

Internetquellen

o.V. (2008-1): http://lexikon.meyers.de/wissen/Schätzung+41394600,
Stand 16. November 2008

o.V. (2008-2): http://csse.usc.edu/csse/research/COCOMOII/cocomo81.htm,
Stand 28. Dezember 2008

McPherson, A.; Proffitt, B.; Hale-Evans, R. (2008): Estimating the Total Development Cost of a Linux Distribution,
http://www.linuxfoundation.org/publications/estimatinglinux.php,
Stand 30. Dezember 2008

Werdenich, K. (2002): COCOMO – COnstructive COst Model,
http://www.software-kompetenz.de/servlet/is/6818/
COCOMO_Werdenich_Studarbeit.pdf?
command=downloadContent&filename=COCOMO_Werdenich_Studarbeit.pdf,
Stand 02. Januar 2009