Das Schätzen von Softwareprojekten - Evaluierung und Anwendung von Schätzverfahren

Diplomarbeit

Zur Erlangung des akademischen Grades

Magister für Projektmanagement und Informationstechnik (FH)

Eingereicht von:

Manfred Hofbauer

Personenkennzeichen:
0110119020

im Fachbereich: Projektmanagement
am Fachhochschulstudiengang „Projektmanagement und Informationstechnik“

Kennzahl des Fachhochschul-Studienganges: 00119V

Erhalter : Fachhochschule des bfi Wien GesmbH

1020 Wien, Wohlmutstrasse 22

Betreuer: Prof. Dr. Wolfgang Ernst Katzenberger

April 2005

Ehrenwörtliche Erklärung

Ich versichere:

1. Dass ich die Diplomarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe,

2. Dass ich diese Diplomarbeit weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit, gleich welcher Prüfung, vorgelegt habe.

_____ _____

Datum Unterschrift

Vorwort

Ich habe dieses Thema aus Interesse an der Materie gewählt, da ich schon oft mit der Notwendigkeit von Schätzverfahren in Berührung gekommen bin.

In meiner beruflichen Laufbahn, die ich zu einem sehr großen Teil in der Softwareentwicklung verbracht habe, bin ich immer wieder vor dem Problem gestanden eine Schätzung für ein Projekt oder einen Projektteil abgeben zu müssen. Ich habe das jeweils mit den in den Betrieben üblichen Methoden getan, die nie strukturierte Aufwandschätzverfahren waren, sondern meist nur schlecht durchgeführte Expertenschätzungen.

Ich konnte auch immer wieder Bekanntschaft schließen mit den unangenehmen Auswirkungen von fehlenden oder fehlgeschlagenen Schätzungen. In vielen Fällen wurden auch keine Methoden des Projektmanagements eingesetzt.

Ich hoffe, dass diese Arbeit eine Hilfe für all jene ist, die in den Genuss kommen, Schätzungen abgeben zu dürfen.

Inhaltsverzeichnis

1 Abstract ..... - 14 -

1.1 Objectives ..... - 14 -

1.2 Structure ..... - 14 -

2 Einleitung ..... - 15 -

2.1 Ziele der Arbeit ..... - 15 -

2.2 Gliederung der Arbeit ..... - 15 -

2.3 Verwendete Software ..... - 16 -

2.3.1 Prozessmodellierung ..... - 16 -

2.3.2 Graphen und Tabellen ..... - 16 -

2.3.3 Andere Abbildungen ..... - 16 -

3 Derzeitige Situation ..... - 17 -

3.1 Persönliche Erfahrungen ..... - 17 -

3.2 Fakten ..... - 17 -

3.2.1 Der Chaos Report ..... - 18 -

3.2.2 Gründe für die Fehlschläge ..... - 19 -

3.2.3 Kritische Betrachtung ..... - 19 -

3.2.4 Untersuchung Weltz und Ortmann ..... - 19 -

3.2.5 Untersuchungen Capers Jones ..... - 20 -

4 Prozess Aufwandsschätzung ..... - 21 -

4.1 Definieren des Schätzobjektes ..... - 21 -

4.1.1 Was versteht man unter einem Schätzobjekt? ..... - 22 -

4.1.2 Definition von Inhalten und Nichtinhalten ..... - 22 -

4.1.3 Sammeln von Informationen ..... - 23 -

4.2 Messen ..... - 24 -

4.3 Schätzen ..... - 25 -

4.3.1 Begriff Einflussfaktoren ..... - 26 -

4.3.2 Gewichtung der Einflussfaktoren – Berechnen des Schätzergebnisses ..... - 27 -

4.3.3 Berechnen des Aufwandes ..... - 27 -

4.4 Dokumentation der Aufwandsschätzung ..... - 28 -

4.4.1 Welche Unterlagen müssen erstellt werden? ..... - 28 -

4.4.2 Einsatz von unterstützender Software ..... - 29 -

4.4.2.1 Schätztools ..... - 29 -

4.4.2.2 Versionsverwaltungstools ..... - 29 -

4.4.3 Unternehmenskapital ..... - 31 -

5 Berechnen des Aufwandes ..... - 32 -

5.1 Allgemeines ..... - 32 -

5.2 Möglichkeiten zur Ermittlung des Aufwandes ..... - 32 -

5.2.1 Ablesen aus einer Erfahrungskurve ..... - 32 -

5.2.1.1 Beispiel für die Erstellung einer eigenen Erfahrungskurve ..... - 34 -

5.2.1.2 Zu beachten ..... - 36 -

5.2.1.3 Zusammenfassung ..... - 36 -

5.2.1.4 Anmerkung ..... - 37 -

5.2.2 Anwenden einer Schätzgleichung ..... - 37 -

5.2.2.1 COCOMO II ..... - 37 -

5.2.2.2 SLIM ..... - 37 -

5.2.2.3 Rules of Thumb ..... - 38 -

5.2.2.4 Determining software schedules ..... - 39 -

5.2.2.5 Schätzgleichungen aus Benchmarking-Datenbanken ..... - 40 -

5.2.2.6 Zusammenfassung ..... - 41 -

5.2.3 Umrechnen mittels Spezialsoftware ..... - 41 -

5.2.3.1 Direkte und indirekte Kalibrierung von Einflussgrößen ..... - 42 -

5.2.3.2 Zusammenfassung ..... - 42 -

6 Aufwandsschätzung im Projektmanagement ..... - 43 -

6.1 Allgemeines ..... - 43 -

6.2 Stellung der Aufwandsschätzung im Projektmanagement ..... - 44 -

6.2.1 Startphase ..... - 45 -

6.2.2 Abwicklungsphasen ..... - 45 -

6.2.3 Koordinations- und Änderungsphasen ..... - 45 -

6.2.4 Abschlussphase ..... - 46 -

6.3 Umlegen des Aufwandes auf den Projektplan ..... - 46 -

6.3.1 Einsatz von Spezialtools ..... - 46 -

6.3.1.1 Kritische Betrachtung ..... - 46 -

6.3.2 Einsatz der Prozentsatzmethode ..... - 47 -

6.3.2.1 Kritische Betrachtung ..... - 47 -

6.3.2.2 Anmerkung ..... - 47 -

6.3.3 Andere Möglichkeiten ..... - 48 -

6.4 Wiederholen der Aufwandsschätzung ..... - 48 -

6.5 Aufwand für die Aufwandsschätzung ..... - 49 -

7 Aufwandsschätzung im Unternehmen ..... - 50 -

7.1 Maßnahmen ..... - 50 -

7.1.1 Einbeziehung der Mitarbeiter ..... - 50 -

7.1.2 Erweiterte Ausbildung für die Projektleiter ..... - 50 -

7.1.3 Gründung eines Competence-Centers ..... - 51 -

7.1.4 Erstellung von Richtlinien zur Aufwandsschätzung ..... - 51 -

7.2 Vorteile aus der Aufwandsschätzung für das Unternehmen ..... - 52 -

7.2.1 Projektübergreifende Mitarbeiterplanung ..... - 52 -

7.2.2 Besseres Multiprojektmanagement ..... - 52 -

7.2.3 Wissensmanagement ..... - 52 -

8 Messverfahren und Metriken ..... - 53 -

8.1 Definition Metrik (Softwaremetrik) – Messverfahren ..... - 53 -

8.2 Arten von Messverfahren und Metriken ..... - 53 -

8.2.1 Messung der Größe - Lines of Code Metrik ..... - 54 -

8.2.1.1 Keine einheitlichen Standards ..... - 54 -

8.2.1.2 Verschiedene Programmiersprachen – verschiedene Anzahl von Lines of Code ..... - 55 -

8.2.1.3 Umrechnung LOC/FP ..... - 57 -

8.2.1.3.1 Umrechnungstabelle SPR ..... - 57 -

8.2.1.3.2 Umrechnungstabelle QSM ..... - 57 -

8.2.1.3.3 Umrechnungstabelle David Consulting Group ..... - 58 -

8.2.1.3.4 Zusammenfassung - Umrechnung LOC/FP ..... - 58 -

8.2.1.4 Gemischte Programmiersprachen ..... - 58 -

8.2.1.5 Anzahl der Codezeilen abhängig vom Entwickler und von Standards ..... - 58 -

8.2.1.6 Schätzung im Vorhinein, Überprüfbarkeit im Nachhinein ..... - 59 -

8.2.1.7 Objektivität des Messverfahrens ..... - 59 -

8.2.1.8 Unabhängigkeit von der Art der Anwendung ..... - 59 -

8.2.1.9 Objektorientierte Anwendungen nicht messbar ..... - 60 -

8.2.1.10 Zusammenfassung - Lines of Code Metrik ..... - 60 -

8.2.2 Messung der Funktion - Function-Points Metrik ..... - 61 -

8.2.2.1 Einheitliche Standards ..... - 61 -

8.2.2.2 Überprüfbarkeit jederzeit möglich ..... - 62 -

8.2.2.3 Subjektivität des Messverfahrens ..... - 62 -

8.2.2.4 Verschiedene Varianten ..... - 62 -

8.2.2.5 Anpassbar an die Art der Anwendung ..... - 62 -

8.2.2.6 Objektorientierte Anwendungen sind messbar ..... - 63 -

8.2.2.7 Zusammenfassung – Function-Points Metrik ..... - 63 -

8.2.3 Traditionelle Metriken ..... - 63 -

8.2.3.1 Halstead Metrik ..... - 64 -

8.2.3.2 McCabe Metrik ..... - 64 -

8.2.4 Objektorientierte Metriken ..... - 65 -

8.2.4.1 MOOSE ..... - 66 -

8.2.4.2 Kritische Betrachtung ..... - 66 -

9 Schätzverfahren im Überblick ..... - 67 -

9.1 COCOMO ..... - 67 -

9.1.1 Modelle ..... - 67 -

9.1.2 Messung ..... - 68 -

9.1.3 Schätzung ..... - 69 -

9.1.4 Kritische Betrachtung ..... - 69 -

9.1.4.1 Function-Point - Lines of Code ..... - 69 -

9.1.4.2 Komplexe Logik ..... - 70 -

9.1.5 Zusammenfassung ..... - 70 -

9.2 Expertenschätzung ..... - 70 -

9.2.1 Ablauf ..... - 71 -

9.2.2 Kritische Betrachtung ..... - 71 -

9.2.3 Zusammenfassung ..... - 71 -
9.3 Delphi-Methode ..... - 72 -
9.3.1 Ablauf ..... - 72 -
9.3.2 Kritische Betrachtung ..... - 72 -
9.3.3 Zusammenfassung ..... - 72 -
9.4 PERT - Dreizeitenmethode ..... - 72 -
9.4.1 Ablauf ..... - 73 -
9.4.2 Kritische Betrachtung ..... - 74 -
9.4.2.1 Ungenaue Schätzung ..... - 74 -
9.4.2.2 Überschätzung ..... - 75 -
9.4.2.3 Unterschätzung ..... - 76 -
9.4.3 Zusammenfassung ..... - 76 -
9.4.4 Anmerkung ..... - 76 -
9.5 Function-Point-Verfahren ..... - 76 -
9.5.1 Ablauf ..... - 78 -
9.5.1.1 Zähltyp festlegen ..... - 79 -
9.5.1.2 Umfang und Systemgrenze festlegen ..... - 79 -
9.5.1.3 Messung ..... - 80 -
9.5.1.4 Schätzung ..... - 80 -
9.5.2 Kritische Betrachtung ..... - 81 -
9.5.2.1 Viele Erfahrungen und Verfahren ..... - 81 -

9.5.2.2 Wenig frei verfügbare Informationen ..... - 82 -
9.5.2.3 Praktischer Einsatz ..... - 82 -
9.5.3 Zusammenfassung ..... - 82 -
9.6 Prozentsatzmethode ..... - 83 -
9.6.1 Ablauf ..... - 83 -
9.6.2 Kritische Betrachtung ..... - 84 -
9.6.3 Zusammenfassung ..... - 84 -
10 Andere Techniken, Eckdaten, Begriffe ..... - 85 -
10.1 Backfiring ..... - 85 -
10.1.1 Beispiel für die Anwendung ..... - 86 -
10.1.2 Ungenauigkeit ..... - 87 -
10.2 Benchmarking ..... - 87 -
10.2.1 Kennzahlen ..... - 87 -
10.2.1.1 Aufwand in den Projektphasen ..... - 87 -
10.2.1.2 Entwicklungsproduktivität ..... - 88 -
10.2.1.3 Weitere Kennzahlen ..... - 88 -
10.2.2 Ziele und mögliche Probleme der Benchmarking-Methode ..... - 88 -
10.3 Schleichender Funktionszuwachs ..... - 89 -
10.4 Schätzkultur ..... - 89 -
10.5 200x20x6 ..... - 90 -
11 Conclusio ..... - 91 -

11.1 Messverfahren und Metriken ..... - 91 -

11.2 Errechnung des Aufwandes ..... - 91 -

11.3 Welches Schätzverfahren soll gewählt werden ..... - 91 -

12 Abbildungsverzeichnis ..... - 93 -

13 Tabellenverzeichnis ..... - 95 -

14 Abkürzungsverzeichnis ..... - 96 -

15 Literaturverzeichnis ..... - 97 -

15.1 Studien & Papers ..... - 97 -

15.1.1 An Empirical Validation of Software Cost Estimation Models ..... - 97 -

15.1.2 Comparative Evaluation of Functional Size Measurement Methods ..... - 97 -

15.1.3 Determining software schedules ..... - 97 -

15.1.4 Functional Size Measurement for OO ..... - 97 -

15.1.5 Object-Oriented Metrics –A Survey ..... - 98 -

15.1.6 Software estimating rules of thumb ..... - 98 -

15.1.7 The Standish Group Chaos Report ..... - 98 -

15.2 Bücher ..... - 98 -

15.2.1 Applied Software Measurement ..... - 98 -

15.2.2 Aufwandschätzung von DV-Projekten ..... - 98 -

15.2.3 Aufwandschätzung von IT-Projekten ..... - 99 -

15.2.4 Das Software-Projekt ..... - 99 -

15.2.5 Der Termin ..... - 99 -

15.2.6 Projektmanagement ..... - 99 -

15.2.7 Rapid Development – Taming wild Software Schedules ..... - 99 -

15.2.8 Software Cost Estimation with COCOMO II ..... - 99 -

15.2.9 Spielräume ..... - 99 -

15.2.10 Vom Mythos des Mann-Monats ..... - 99 -

15.2.11 Was man nicht messen kann, kann man nicht kontrollieren ..... - 100 -

15.3 Webseiten: ..... - 100 -

15.3.1 Better-SCM-Initiative ..... - 100 -

15.3.2 Charismatek ..... - 100 -

15.3.3 DASMA ..... - 100 -

15.3.4 David Consulting Group ..... - 100 -

15.3.5 IEEE ..... - 100 -

15.3.6 IFPUG ..... - 101 -

15.3.7 ISBSG ..... - 101 -

15.3.8 ISO ..... - 101 -

15.3.9 QSM ..... - 101 -

15.3.10 Softstarsystems ..... - 101 -

15.3.11 SPR ..... - 101 -

15.3.12 Wikipedia ..... - 102 -

16 Danksagung ..... - 103 -

17 Anhang ..... - 104 -

17.1 Legende für die Darstellung der Prozesse ..... - 104 -

17.2 Übersicht über Metriken ..... - 105 -

17.2.1 Traditionelle Metriken ..... - 105 -

17.2.2 OO-Klassenmetriken ..... - 106 -

17.2.3 OO-Methodenmetriken ..... - 107 -

17.2.4 OO-Verbindungsmetriken ..... - 107 -

17.2.5 OO-Vererbungsmetriken ..... - 108 -

17.2.6 OO-Systemmetriken ..... - 109 -

1 Abstract

1.1 Objectives

The principal topic of this thesis are methods for estimating software-projects, as one of the various tools in project management.

I intend to

- impart knowledge about the basics of estimation
- question the meaning of estimation
- observe estimation as a process in a project and within the company
- give an overview about estimation methods
- evaluate and analyse common methods, as well as problems in practical use

1.2 Structure

- Chapter 3 covers my personal experience, the implementation and the quality of estimation in the IT-business.
- The following chapters will give some basic principles about estimation methods and estimation in general by viewing estimation as a process, and will also show the possibilities to calculate the effort.
- Afterwards the estimation will be examined from a project management and also corporate point-of view.
- The next chapter will evaluate measurement methods and metrics.
- The chapter „Schätzverfahren im Überblick“ will give an overview and evaluate common estimation methods.
- The last chapter will be a compilation of important methods, techniques and terms from the area of estimation.

2 Einleitung

2.1 Ziele der Arbeit

Das Kernthema dieser Diplomarbeit sind Schätzverfahren im Sinne von Aufwandsschätzungen als eines von vielen Werkzeugen des Projektmanagements.

Es soll(en)

- Grundlagen über Aufwandsschätzungen vermittelt werden.
- Der Sinn hinterfragt werden.
- Die Aufwandsschätzung als Prozess im Projekt als auch im Unternehmen betrachtet werden.
- Ein Überblick über Aufwandschätzverfahren gegeben werden.
- Einzelne bekannte Verfahren analysiert und evaluiert, sowie Probleme bei der Anwendung erörtert werden.

2.2 Gliederung der Arbeit

- Das folgende Kapitel 3 beschäftigt sich mit meinen persönlichen Erfahrungen, der Verbreitung sowie der Qualität von Aufwandsschätzungen in der IT-Branche.
- Die nächsten Kapitel versuchen Grundlagen über Schätzverfahren und die Aufwandsschätzung im Allgemeinen zu vermitteln, indem die Aufwandsschätzung als Prozess betrachtet wird, und Möglichkeiten aufgezeigt werden um den Aufwand zu berechnen.
- Anschließend wird die Aufwandsschätzung vom Blickpunkt des Projektmanagements und des Unternehmens betrachtet.
- Im nächsten Kapitel erfolgt eine Evaluierung möglicher Messverfahren und Metriken.
- Das Kapitel „Schätzverfahren im Überblick“ gibt einen Überblick über gebräuchliche Schätzverfahren, und beleuchtet diese kritisch.
- Der letzte Abschnitt ist eine Sammlung wichtiger Methoden, Techniken und Begriffe aus dem Bereich der Schätzverfahren.

2.3 Verwendete Software

2.3.1 Prozessmodellierung

Es war für diese Arbeit des Öfteren die Darstellung von Prozessen notwendig. Ich habe mich entschieden dazu die Syntax des Produkt ADONIS der BOC GmbH zu verwenden, da ich die Darstellung für einfach und zweckmäßig halte.

Eine Legende für die Symbole ist 17.1 „Legende für die Darstellung der Prozesse“ zu entnehmen.

2.3.2 Graphen und Tabellen

Graphen und Tabellen wurden, soweit erforderlich, mit Microsoft Excel erstellt.

2.3.3 Andere Abbildungen

Andere Abbildungen wurden je nach Bedarf mit ADONIS oder Microsoft Visio erstellt.

3 Derzeitige Situation

Dieses Kapitel beschäftigt sich mit der derzeitigen Situation der Verbreitung der Aufwandschätzverfahren sowie deren Qualität. Es werden persönliche Erfahrungen als auch Fakten aus Studien dargelegt.

3.1 Persönliche Erfahrungen

Ich habe die Erfahrung gemacht, dass Aufwandsschätzungen nur unzureichend und schlampig erfolgen. Die Ergebnisse, die mit solchen Aufwandsschätzungen erzielt werden, sind meist mangelhaft. Dabei habe ich den Eindruck gewonnen, je kleiner die Projekte waren und je mehr Projekte im ähnlichen Umfeld schon durchgeführt worden sind, umso bessere Ergebnisse konnten auch mit unzureichender Aufwandsschätzung erreicht werden.

Im Falle, dass ein Projekt überzogen wurde oder sogar gescheitert ist, wurde im Normalfall begonnen, Schuldige zu suchen. Wenig Mühe wurde darin investiert, die Aufwandsschätzung näher zu beleuchten und aus dem Fehlschlag zu lernen. Sehr treffend formuliert hat es Tom de Marco als er meinte, dass der Plan falsch ist, wenn Termine nicht gehalten oder Inhalte nicht abgedeckt werden[1].

Eine weit verbreitete Sichtweise ist auch, dass die Aufwandsschätzung zu viel Zeit kostet. Meiner Meinung nach ist dies mit der Vorgehensweise vergleichbar, bei einer Autofahrt aus Zeitmangel nicht zu tanken.

Ich bin überzeugt davon, dass mit einer guten Aufwandsschätzung der Grundstein für ein erfolgreiches Projekt und erfolgreiches Projektmanagement gelegt wird.

3.2 Fakten

Ich habe einige Untersuchungen herangezogen mit deren Hilfe es möglich sein sollte, objektiv eine Aussage über die Verbreitung und über die Qualität der Aufwandsschätzungen zu treffen. Diese Untersuchungen werden in den nachfolgenden Abschnitten näher behandelt.

---

[1] Vgl. 15.2.9 Spielräume S58

3.2.1 Der Chaos Report

Der Chaos-Report[2] der Standish-Group hat im Erscheinungsjahr 1995 reges Interesse in der IT-Branche ausgelöst. Zu diesem Report wurden 8.380 Anwendungen untersucht und die Projekte in folgende Kategorien eingeteilt:

Success	Ressourcen, Termine und Inhalte im Plan
Challenged	Projekt fertig gestellt und funktioniert, Termine und Ressourcen überschritten, Inhalte im geringeren Umfang als zu Beginn veranschlagt
Impaired	Projekt wurde abgebrochen

Tabelle 1: Chaos Report Kategorien

Die besorgniserregenden Ergebnisse nach Jahren geordnet:

Kategorie/Jahr	**1994**	**1996**	**1998**	**2000**	**2004**	**Gesamt**
Success in %	16%	27%	26%	28%	34%	26,20%
Challenged in %	53%	33%	46%	49%	51%	46,40%
Impaired in %	31%	40%	28%	23%	15%	27,40%
Summe	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

Tabelle 2: Chaos Report Ergebnisse

Wie diese Tabelle zeigt, wurde nur bei rund 26% der Projekte der geplante Aufwand nicht überschritten, was auf eine mangelnde beziehungsweise fehlende Anwendung von Aufwandschätzverfahren hinweist.

[2] Vgl. 15.1.7 The Standish Group Chaos Report

3.2.2 Gründe für die Fehlschläge

Ein weiteres Indiz dafür liefern die Gründe die für den Erfolg beziehungsweise den Misserfolg der Projekte genannt wurden, denn viele der Gründe sind Faktoren die im Rahmen der Aufwandsschätzung als so genannte „Einflussfaktoren“ kalkuliert werden sollten.

Zum Beispiel beeinflusst die Erfahrung der Mitarbeiter den Erfolg des Projektes. Diese Erfahrung muss natürlich bei der Aufwandsschätzung schon miteinbezogen werden, man kann nicht von einem Anfänger verlangen die gleiche Leistung wie ein Spezialist zu erbringen.

Ein weiteres Beispiel sind „klare Zielvorgaben“. Bei unvollständig definierten oder unklaren Zielen muss eine Schätzung auf Grund einer Annahme getroffen werden. Damit steigt natürlich das Projektrisiko und man könnte zum Beispiel mit Risikozuschlägen gegensteuern.

3.2.3 Kritische Betrachtung

„Nur schlechte Nachrichten sind gute Nachrichten“ lautet ein Leitsatz der Medien, somit wäre ein Chaos-Report 20xx mit 99% an erfolgreichen Projekten völlig uninteressant und auch unprofitabel.

Ob dieser Report in Europa auch seine Gültigkeit hat ist nicht wirklich restlos geklärt, da für diese Studie Manager aus US-Amerikanischen Unternehmen befragt wurden. Auch die Größe der Projekte kann vielleicht gar nicht auf europäische Verhältnisse umgelegt werden.

Genaue Analysen sind jedoch nicht möglich, da die Details, die zu diesem Report geführt haben, wie Staatsgeheimnisse behandelt werden.

3.2.4 Untersuchung Weltz und Ortmann

Eine weitere interessante Untersuchung haben Friedrich Weltz und Rolf Ortmann[3] durchgeführt. Es wurde der Einsatz von Methoden bei der Kalkulation in Anwenderunternehmen und in Softwarehäusern untersucht. Das Ergebnis sieht folgendermaßen aus:

---

[3] Vgl. 15.2.4 Das Software-Projekt S40 Tabelle 2

Kalkulationsart/Unternehmensart	Anwenderunternehmen	Softwarehäuser	Gesamt
Kalkulation methodengestützt	8%	14%	11%
Kalkulation methodengestützt und erfahrungsbezogen	8%	24%	15%
Kalkulation erfahrungsbezogen	60%	52%	57%
Kalkulation unterblieb	20%	5%	13%
Keine Angabe	4%	5%	4%
Summe	**100%**	**100%**	**100%**

Tabelle 3: Untersuchung zum Einsatz methodengestützter Kalkulation

Großteils wird, wenn überhaupt, eine „Expertenschätzung“ durchgeführt. Obwohl auch eine Expertenschätzung, im Falle sie gut und gründlich durchgeführt wird, eine gute Möglichkeit zur Aufwandsschätzung ist (siehe auch 9.2 - Expertenschätzung).

3.2.5 Untersuchungen Capers Jones

Auch Capers Jones, ein sehr angesehener Mann im Bereich der Softwaremetriken, hat in einer seiner Veröffentlichungen[4] festgestellt, dass manuell, also rein erfahrungsbezogen durchgeführte Schätzungen in 75% der Fälle falsch waren. In den meisten dieser Fälle wurde sowohl der Aufwand als auch die Durchlaufzeit signifikant unterschätzt. Die methodengestützten Aufwandsschätzungen wiesen leichte Überschätzungen auf.

In einer etwas früheren Veröffentlichung[5] spricht er davon, dass 50% der kapitalen Software-Desaster auf Zeitpläne zurückzuführen sind, die „inkompetent“ erstellt wurden („being incompetently established“).

[4] Vgl 15.1.6 Software estimating rules of thumb
[5] Vgl 15.1.3 Determining software schedules

4 Prozess Aufwandsschätzung

In diesem Kapitel wird versucht, anhand eines Prozessmodells die Teilprozesse der Aufwandsschätzung darzustellen und die einzelnen Aktivitäten, die dabei durchzuführen sind, näher zu erläutern.

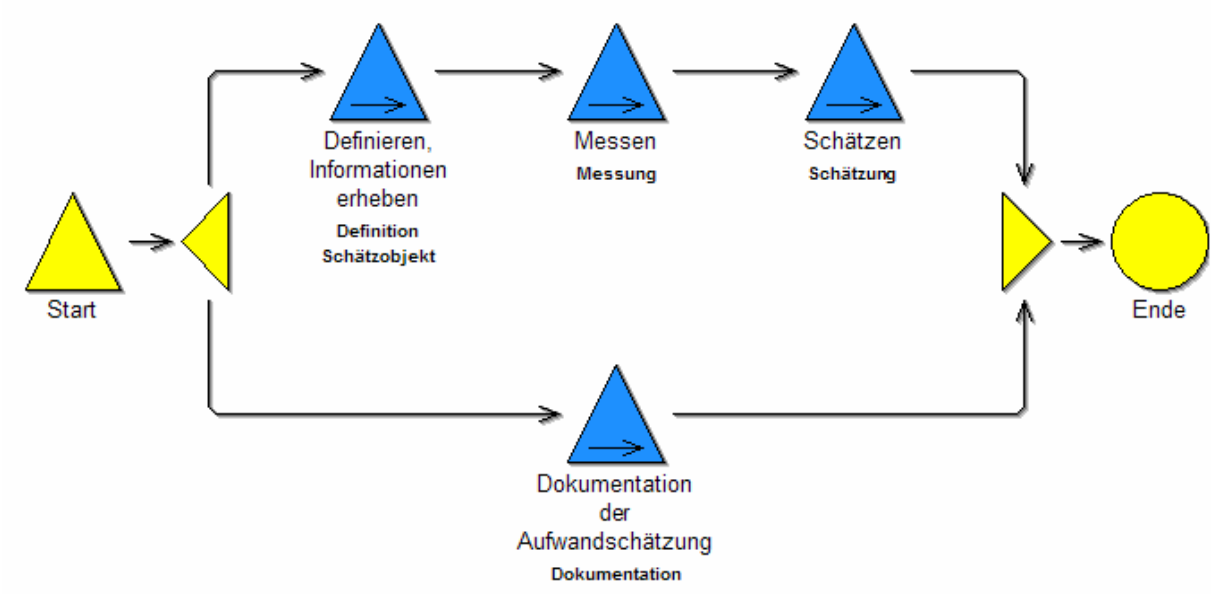

Abbildung 1: Prozess Aufwandsschätzung

Die Aufwandsschätzung besteht, wie oft in der Literatur zu lesen ist, aus der Messung und der Schätzung. Ich habe mich dazu entschlossen, zu der üblichen Darstellung dieser beiden Teile zwei weitere hinzuzufügen, nämlich die Definition des Schätzobjektes und die Dokumentation der Aufwandsschätzung. Ich halte diese beiden Subprozesse für so essenziell, dass ich sie separat behandeln möchte. Zu beachten ist auch, dass in Abbildung 1 der Subprozess „Dokumentation“ parallel zu den anderen Subprozessen abläuft.

4.1 Definieren des Schätzobjektes

Bevor mit dem Messen und Schätzen begonnen werden kann muss Klarheit herrschen, was überhaupt geschätzt werden soll. Aus diesem Grund soll zu Beginn der Aufwandsschätzung versucht werden, das Schätzobjekt so genau wie möglich zu definieren und soviel Informationen wie möglich über das Schätzobjekt in Erfahrung zu bringen. Dieser Schritt ist

in meinen Augen sehr wichtig, da er die Grundlagen für die anschließende Messung und Schätzung schafft.

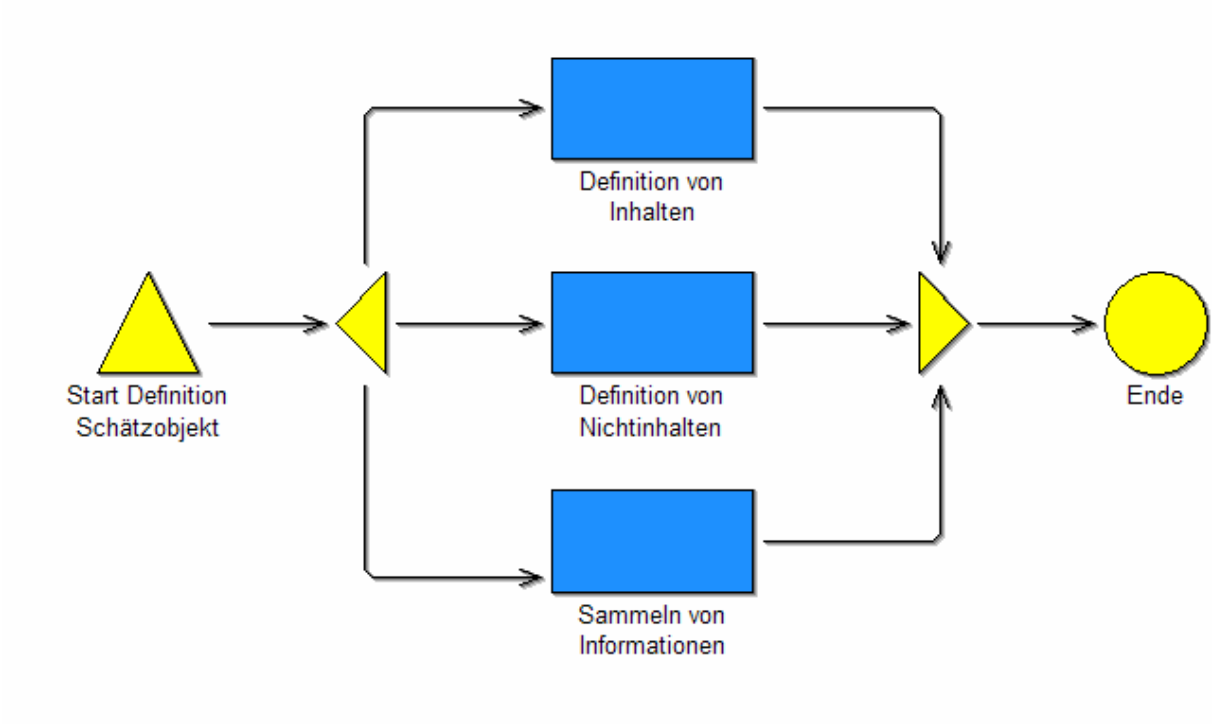

Abbildung 2: Definition des Schätzobjektes als erster Schritt zur Aufwandsschätzung

Diese Abbildung zeigt, dass die Abfolge der einzelnen Tätigkeiten parallel erfolgt. Ich meine, dass dies parallel erfolgen muss, da man die einzelnen Schritte nicht sauber voneinander trennen kann. Zum Beispiel ergeben sich bei der Definition von Inhalten automatisch Informationen über die Nichtinhalte und umgekehrt.

4.1.1 Was versteht man unter einem Schätzobjekt?

Als Schätzobjekt wird das zu schätzende Vorhaben oder Projekt bezeichnet. In der Literatur können aber auch immer wieder kleine Teile des Projektes, bis hin zu Arbeitspaketen oder aber ganze Phasen gemeint sein. Ich konnte keine aussagekräftige Definition dazu finden, vielleicht ist der Name selbst also „das Objekt, das geschätzt wird“ die beste Beschreibung.

4.1.2 Definition von Inhalten und Nichtinhalten

Ähnlich wie in der Startphase eines Projektes ist die Abgrenzung der Inhalte sehr wichtig. Die Abgrenzung der Inhalte wird mit der Abgrenzung des Projektes in der Praxis wahrscheinlich

sogar Hand in Hand gehen. Die Definition von Nichtinhalten erlaubt in der Regel eine einfachere Durchführung der Abgrenzung. Besonders bei der Erstellung von Angeboten, die meiner Meinung nach immer eine strukturierte Aufwandsschätzung enthalten sollten, hilft die Definition von Nichtinhalten zur Vermeidung von Missverständnissen.

Beispiel:
Ein Schätzobjekt ist die Erstellung einer Schnittstelle. Es werden dazu Daten zur Verfügung gestellt (Inhalt), es werden diese Daten jedoch nicht weiter aufbereitet oder ausgewertet und auch nicht in ein Fremdsystem überspielt (Nichtinhalte).

4.1.3 Sammeln von Informationen

Um eine Schätzung durchzuführen braucht man klarerweise Informationen über das Schätzobjekt. Ohne ausreichende Informationen über das Schätzobjekt gerät die Aufwandsschätzung zu einem Glücksspiel, und das ist sicher nicht im Sinne des Erfinders.

Je mehr Informationen man hat und je höher die Qualität dieser ist, desto genauer kann die Messung beziehungsweise die Schätzung ausfallen.

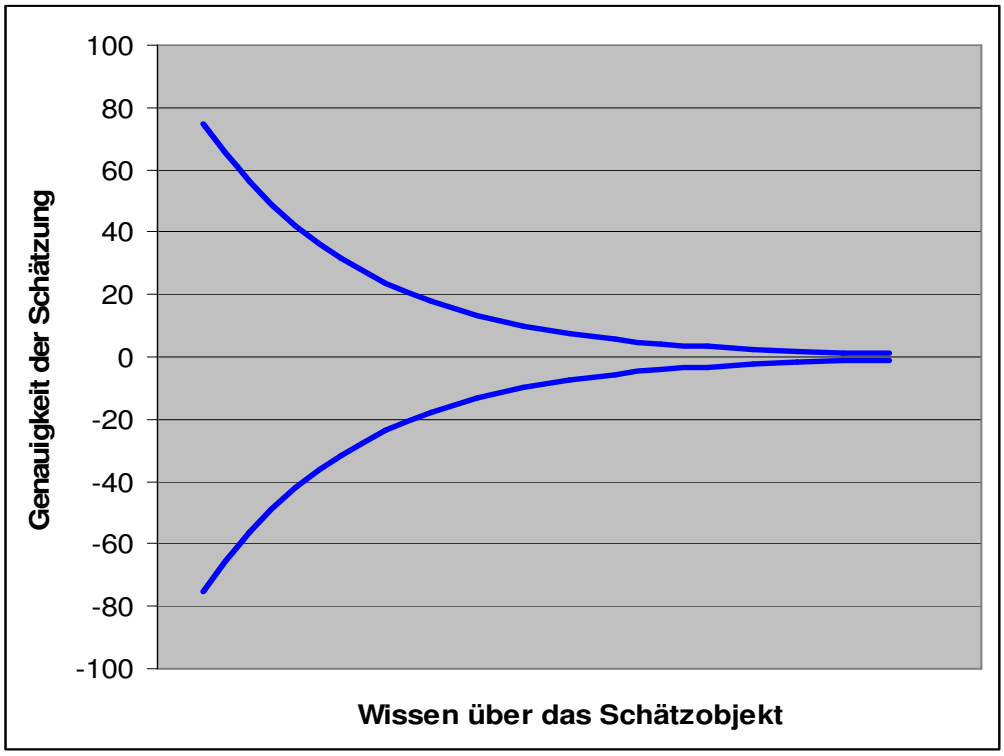

Abbildung 3: Schätzgenauigkeit steigt mit dem Wissen über das Schätzobjekt

Wie diese Abbildung zeigt, steigt die Genauigkeit der Aufwandsschätzung mit dem Wissen über das Schätzobjekt. Aus diesem Grund sind Schätzungen, die im Projektverlauf später durchgeführt werden in der Regel viel genauer als Aufwandsschätzungen, die bei der Erstellung eines Angebotes gemacht werden, da später im Projektverlauf auch mehr und genauere Informationen zur Verfügung stehen.

4.2 Messen

Messen bedeutet laut Definition die „quantitative Bestimmung des Wertes einer Messgröße“[6]. Es können verschiedenste Messgrößen gemessen werden im Bereich des Projektmanagements wie zum Beispiel Kosten, Dauer, Aufwand, Umfang, Qualität und Ähnliches. Die Messung erfolgt nach dem Definieren des Schätzobjektes (siehe auch Abbildung 1).

Bei der Messung im Sinne der Aufwandsschätzung wird versucht, das Schätzobjekt größenmäßig zu bestimmen um mit Hilfe der ermittelten Größe die eigentliche Schätzung

[6] 15.3.12 Wikipedia – Begriff „Messung“

durchführen zu können. Die mit Hilfe der Schätzung gewonnenen Informationen können dann in den Aufwand, der die Grundlage für die Projektplanung bildet, umgerechnet werden. Gemessen wird bei fast allen Schätzverfahren. Eine Ausnahme bildet laut Literatur[7] die Expertenschätzung, bei der nicht gemessen, sondern nur geschätzt wird.

Ich stelle aber in Frage, ob bei der Expertenschätzung nicht eigentlich auch eine Messung vorgenommen wird. Denn wenn ein Experte schätzen soll, und das auch gewissenhaft erledigt, muss er ja auch die ungefähre Größe des Schätzobjektes oder der Schätzobjekte bestimmen. Also wird dabei ja auch eine Messung durchgeführt.

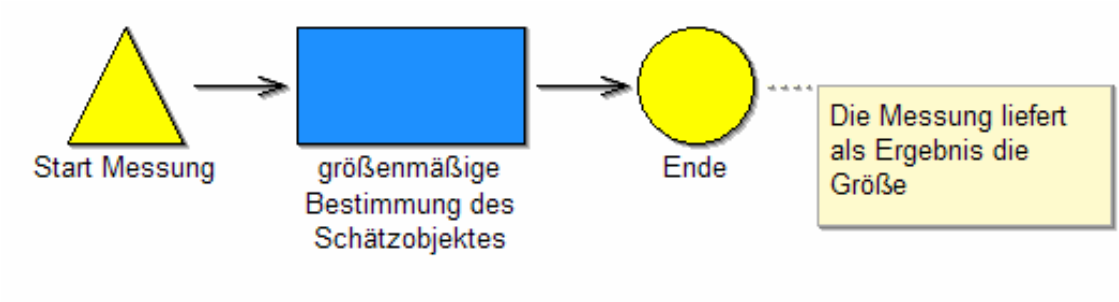

Abbildung 4: Messen mittels Anwendung eines Messverfahren

Wie aus dieser Abbildung ersichtlich, wird ein Messverfahren angewandt, um die Größe des Schätzobjektes zu ermitteln. Als Ergebnis wird die Größe, in einem dem Messverfahren eigenen Zählmaß, geliefert.

Im Kapitel 8 „Messverfahren und Metriken“ werden einzelne Messverfahren beziehungsweise Metriken näher betrachtet sowie analysiert und die Anwendungsmöglichkeit im Rahmen der Aufwandsschätzung untersucht.

4.3 Schätzen

Nachdem das Schätzobjekt größenmäßig bestimmt wurde, erfolgt die eigentliche Schätzung (siehe auch Abbildung 1). Die Definition der Schätzung lässt sich als „genäherte Bestimmung

[7] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten – S28

von Zahlenwerten, Größen oder Parametern durch Augenschein, Erfahrung oder statistisch-mathematische Methoden“[8] beschreiben.

Bei der Schätzung als Teil der Aufwandsschätzung werden in das Ergebnis der Messung noch zusätzliche Einflussfaktoren wie zum Beispiel Komplexität, Anzahl der Schnittstellen und Ähnliches eingerechnet.

Daher muss bei der Schätzung eine Ermittlung und Gewichtung dieser Einflussfaktoren vorgenommen werden.

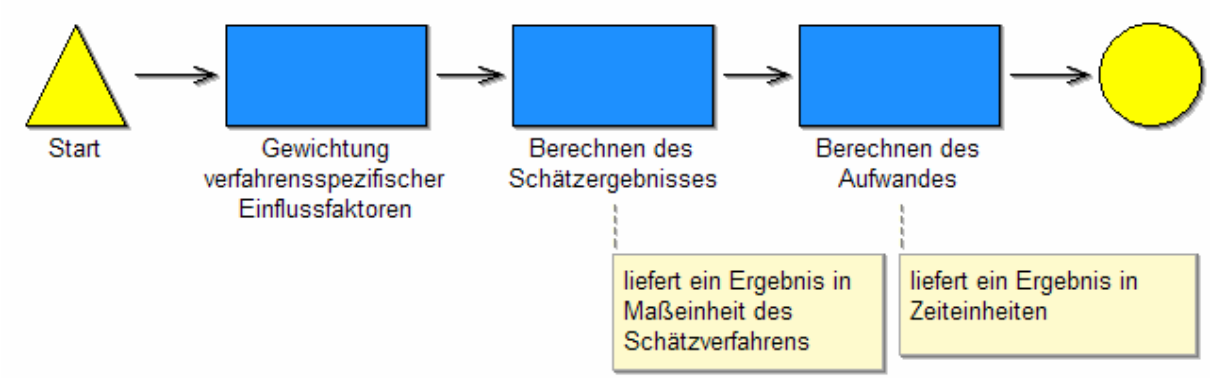

Abbildung 5: Berechnen des Schätzergebnisses

In dieser Abbildung ist der Teilprozess „Schätzen“ dargestellt, bei dem zuerst die Einflussfaktoren ermittelt und gewichtet werden. Anschließend erfolgt die Berechnung des Schätzergebnisses und des Aufwandes.

4.3.1 Begriff Einflussfaktoren

Die „Einflussfaktoren“ sind jene Umweltfaktoren, die auf ein Schätzobjekt einwirken aber nicht gemessen wurden. Diese können den Aufwand unter Umständen erhöhen oder vermindern. Dieser Begriff war schon in sehr frühen Versionen des Function-Point-Verfahrens gebräuchlich. Die Einbeziehung solcher Einflussfaktoren sollte zusätzlich noch die Komplexität des Schätzobjektes beurteilen. Beim Schätzverfahren COCOMO werden sie „Kostentreiber“ genannt. Solche Faktoren, wie auch immer sie genannt werden, lassen sich in

[8] 15.3.12 Wikipedia – Begriff „Schätzung“

fast jedem Schätzverfahren wieder finden. Ich werde bei dieser Arbeit den Begriff „Einflussfaktor“ weiter verwenden.

Beispiele für Einflussfaktoren:

- Die Qualität der für die Schätzung zur Verfügung stehenden Informationen
- Erfahrung der Mitarbeiter
- Komplexität der Anwendung
- Erfahrung des Unternehmens mit einer bestimmten Technik
- Anzahl der Schnittstellen
- Größe des Projektes
- Umfeld oder Branche
- …

Je nach Schätzverfahren gibt es verschiedene Arten von Einflussfaktoren, auch die Anzahl dieser Einflussfaktoren variiert stark. Beim Function-Point-Verfahren nach IFPUG existieren zum Beispiel 14 Einflussfaktoren, bei den Mark II Function-Points 20.

4.3.2 Gewichtung der Einflussfaktoren – Berechnen des Schätzergebnisses

Bei der Gewichtung der Einflussfaktoren werden die Einflussfaktoren nach Vorgaben des Schätzverfahrens gewichtet. Diese Gewichtung hat direkten Einfluss auf das Schätzergebnis. Nach der Gewichtung wird, wiederum abhängig von den Vorgaben des Schätzverfahrens, das Schätzergebnis errechnet. Die Maßeinheit dieses Schätzergebnis kann je nach Schätzverfahren unterschiedlich sein, zum Beispiel Function-Points oder Source-Lines of Code (SLOC). Ein Beispiel für eine Berechnung des Schätzergebnisses wird bei der Analyse des Function-Point-Verfahrens unter 9.5.1.4 „Schätzung“ gegeben.

4.3.3 Berechnen des Aufwandes

Die Berechnung des Aufwandes selbst, die typischerweise in Personenmonaten gemessen wird, ist nicht Teil eines jeden Schätzverfahrens. Zum Beispiel liefert das Function-Point-Verfahren als Schätzergebnis gewichtete Function-Points, und stellt nur eine Basis für die Ermittlung des Aufwandes zur Verfügung. Bei COCOMO II hingegen gibt es im

Schätzverfahren selbst Formeln zur Umrechnung des ermittelten Schätzergebnisses in Personenmonate.

Um die Schätzung im Rahmen des Projektmanagements verwenden zu können, muss eine Umrechnung in Zeiteinheiten, also in den Aufwand, erfolgen, ansonsten kann keine Planung vorgenommen werden.

Somit ist die Berechnung des Aufwandes einerseits nicht immer Teil der Schätzung sondern als Aufgabe im Projektmanagement-Prozess angesiedelt.

Ich habe mich deshalb entschlossen die goldene Mitte zu wählen und die Berechnung des Aufwandes als Tätigkeit beim Schätzen zu belassen (siehe Abbildung 5), werde diesem besonderen Thema aber ein eigenes Unterkapitel (5 „Berechnen des Aufwandes“) widmen.

4.4 Dokumentation der Aufwandsschätzung

Die Dokumentation der Aufwandsschätzung läuft parallel zu den anderen Subprozessen (Informationserhebung, Messung, Schätzung) ab, und versucht Unterlagen zu erstellen, die in erster Linie helfen sollen die Aufwandsschätzung nachvollziehbar zu machen.

Erst durch eine Nachvollziehbarkeit ist es möglich, Fehler in der Aufwandsschätzung zu erkennen und zu korrigieren. Eine weitere Aufgabe der Dokumentation ist die Wissenssicherung. Die Dokumentation ermöglicht somit schon erworbenes Wissen einfacherer zu teilen.

4.4.1 Welche Unterlagen müssen erstellt werden?

Es müssen alle Fakten dokumentiert werden die notwendig sind, um die Aufwandsschätzung nachvollziehbar zu machen. Dazu zählen auch Annahmen, die in Ermangelung an Informationen getroffen wurden.

In welcher Form die Dokumentation erfolgt und welche Art von Unterlagen erstellt werden ist meiner Meinung nach abhängig von der Gegebenheit des Projektes, und sollte aus diesem Grund an dieses angepasst werden. Auch der Leitspruch „So grob wie möglich, so fein wie nötig“ sollte Beachtung finden.

Ein Beispiel für mögliche Dokumente habe ich dem Buch „Aufwandschätzung von IT-Projekten“[9] entnommen. Hier ein Auszug aus den Unterlagen, die für eine vollständige Dokumentation als notwendig erachte werden:

- Logbuch (Angaben über Teilnehmer, Zeitpunkte, vorhandene Informationen, Unterlagen, Absprachen, Prämissen, Besonderheiten)
- Übersicht über die Systemgrenzen (Architekturdiagramm, logische Datenmodelle, Schnittstellen, etc.)
- Zählungen aus einem Schätztool
- Ergebnisse der Schätzung des Schätztools
- Kopien von Unterlagen aus dem IT-Projekt (Dialogstrukturen, Layouts, etc.)

4.4.2 Einsatz von unterstützender Software

Um vieles vereinfacht wird die Dokumentation im Falle Anwendungen eingesetzt werden, die es erlauben, den Dokumentationsprozess zu unterstützen.

4.4.2.1 Schätztools

Einige Produkte, die im Umfeld der Aufwandsschätzung anzutreffen sind, bieten die Möglichkeit, die Dokumentation der Schätzung in der Anwendung selbst durchzuführen. Ebenfalls kann auf externe Dokumente verwiesen werden. Somit sind Annahmen, Erklärungen, etc. direkt bei dem jeweiligen Teil der Aufwandsschätzung gespeichert oder externe Dokumente direkt einem Teil der Aufwandsschätzung zuordenbar.

4.4.2.2 Versionsverwaltungstools

Eine große Hilfe, um bei einer großen Anzahl von Dokumenten nicht den Überblick zu verlieren, sind Versionsverwaltungstools. Ich denke, dass ein Einsatz solcher Tools gerade bei wiederholter Durchführung der Aufwandsschätzung (siehe auch 6.4) hilft, die Dokumentation

[9] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S98

ohne großen Aufwand in jedem Schritt nachvollziehbar zu halten und damit viel Zeit zu sparen.

Es gibt verschiedene Versionsverwaltungstools, wobei ein Großteil davon sogar gratis bezogen werden kann (Open-Source). Einen guten Überblick über die verfügbaren Tools gibt die Webseite http://better-scm.berlios.de/comparison/, die auch detailliert die Vor- und Nachteile der jeweiligen Anwendung beschreibt.

Hier ein kleines Beispiel, wie die Dokumentation mit einem solchen Versionsverwaltungstool aussehen könnte:

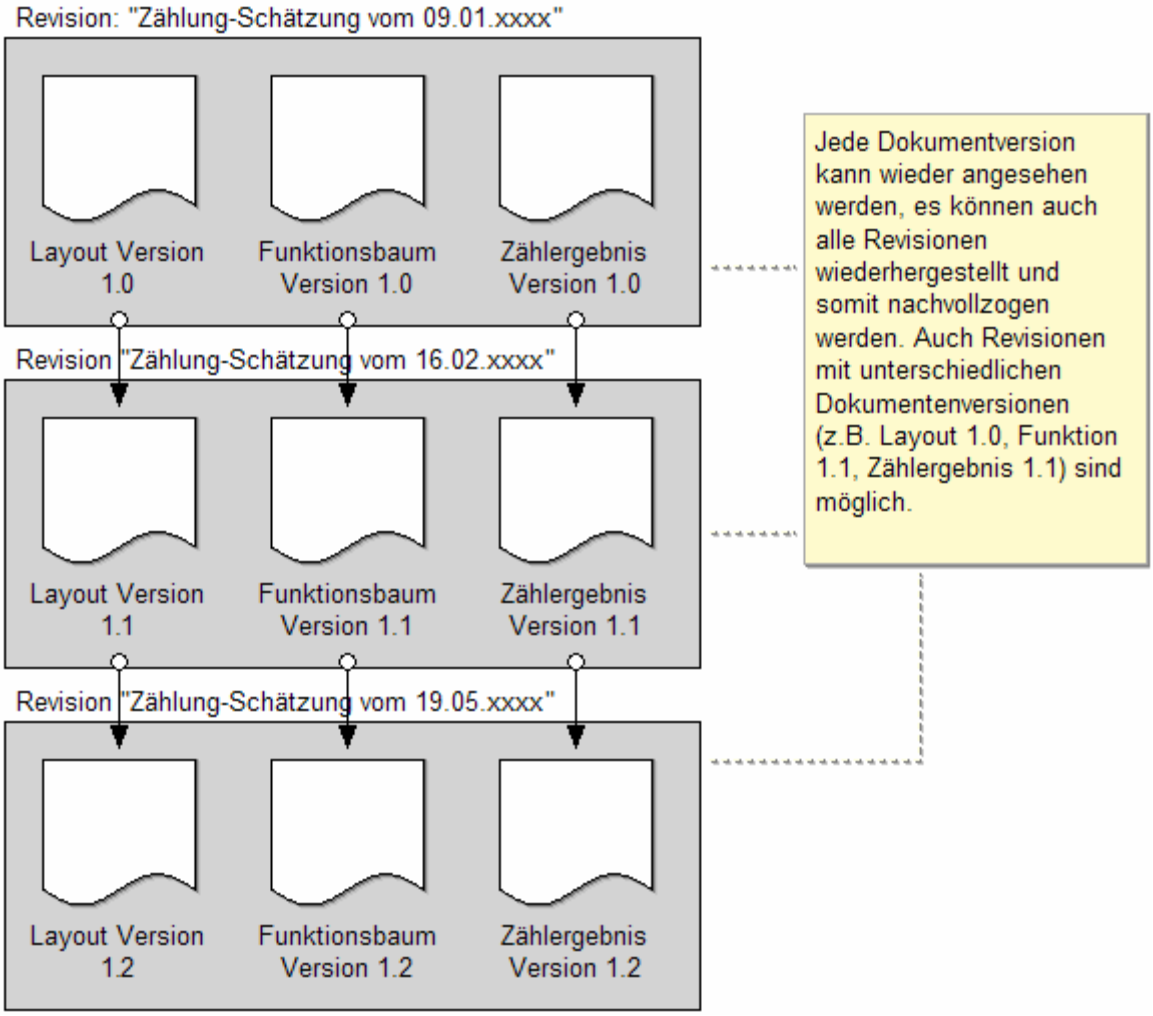

Abbildung 6: Einsatz von Versionsverwaltungstools für die Dokumentation

Abbildung 6 zeigt, dass es möglich ist die Dokumente in verschiedenen Versionen zu verwalten. Es können auch einzelne Versionen der Dokumente anschließend zu Revisionen zusammengefasst werden, die den einzelnen Schätzungen entsprechen. Dieses Beispiel zeigt aber nur eine von mehreren Möglichkeiten des Einsatzes auf.

4.4.3 Unternehmenskapital

Das Kapital eines Unternehmens selbst besteht nicht nur aus den in der Buchhaltung erfassten Geldwerten, sondern auch aus dem Humankapital[10], also den Mitarbeitern und dem Wissen, das diese Mitarbeiter besitzen. Somit steigt mit dem Wissen auch der Wert des Unternehmens. Man sollte sich auch vor Augen führen, dass Wissen die einzige Ressource ist, die durch Teilen wächst, und somit das Unternehmenskapital mehrt.

Eine saubere und strukturierte Dokumentation ermöglicht es Wissen zu erzeugen, indem aus durchgeführten Projekten gelernt werden kann, und ist daher auch von diesem Standpunkt aus von großer Wichtigkeit.

[10] Vgl. 15.2.9 Spielräume S35ff

5 Berechnen des Aufwandes

Erst mit dem Berechnen des Aufwandes wird eine Aufwandsschätzung für die Projektplanung verwendbar. Dieses Kapitel widmet sich den Möglichkeiten die zur Verfügung stehen, um aus einem ermittelten Schätzergebnis den Aufwand zu ermitteln.

5.1 Allgemeines

Bei der Berechnung des Aufwandes wird das Ergebnis der Schätzung in Zeiteinheiten, typischerweise Personenmonate (PM) oder Personenjahre (PJ), umgerechnet. In älteren Büchern lässt sich auch der Begriff des Mannmonats (MM) oder Mannjahres (MJ) finden.

Man sollte meinen, dass die Berechnung des Aufwandes, da sie ja das eigentliche Ziel einer Aufwandsschätzung ist, ein in der Literatur ausführlich behandeltes Thema darstellt. Leider ist dem nicht so. Informationen darüber sind sehr dünn gesät, sowohl in der Fachliteratur als auch im Internet.

5.2 Möglichkeiten zur Ermittlung des Aufwandes

Ich habe folgende Möglichkeiten gefunden, um das ermittelte Schätzergebnis in Zeiteinheiten umzulegen.

5.2.1 Ablesen aus einer Erfahrungskurve

Die erste Möglichkeit besteht aus dem Ablesen aus einer so genannten „Erfahrungskurve“. Die erste Erfahrungskurve, die im Rahmen der Aufwandsschätzung verwendet wurde, hängt stark mit den ersten Veröffentlichungen des Function-Point-Verfahrens, von dem IBM-Mitarbeiter Albrecht im Jahre 1979 entwickelt, zusammen. Aus diesem Grund liest man auch öfters von der „IBM-Erfahrungskurve“.

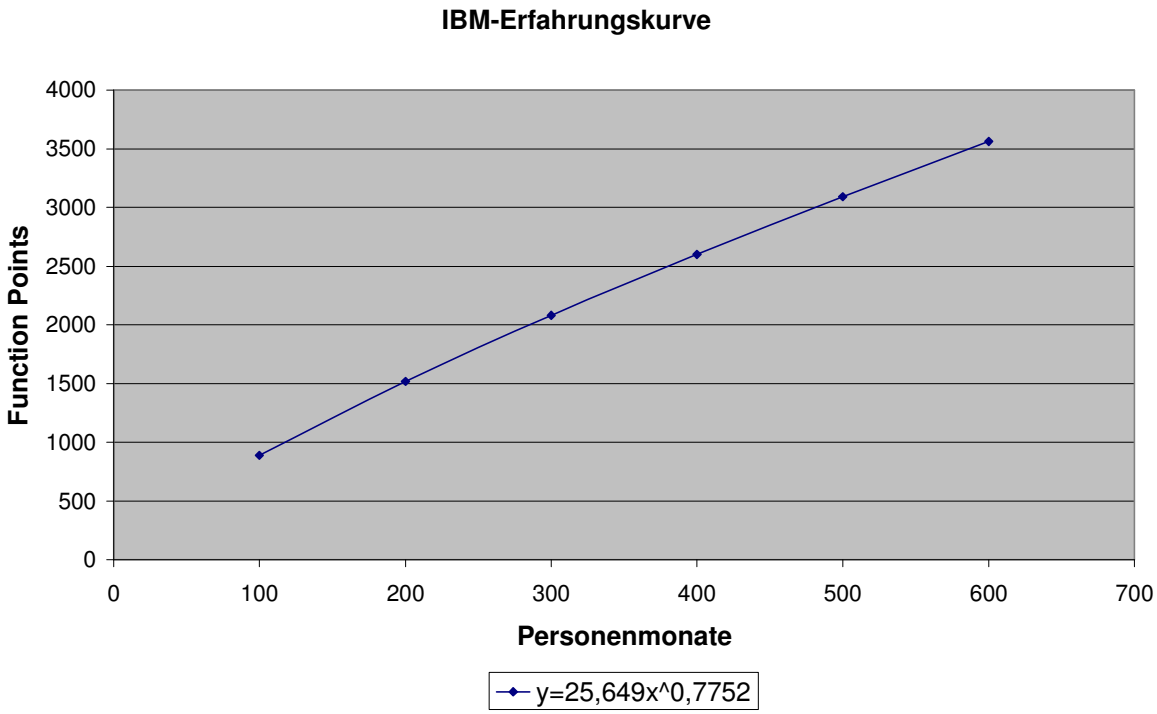

Abbildung 7: IBM-Erfahrungskurve

Mit Hilfe solcher Kurven können anhand der ermittelten Function-Points die Personenmonate abgelesen werden, indem der jeweilige Punkt am Graphen gesucht wird.

Die Daten, die der IBM-Kurve zugrunde liegen, sind das letzte Mal 1991 angepasst worden[11], also zu veraltet, um sie heute als Grundlage für eine Errechnung des Aufwandes verwenden zu können. Auch Erfahrungskurven aus anderen Unternehmen sind nur bedingt verwendbar, da jeweils andere Einflussfaktoren (Kunden, Firmenkultur, Technologie, Branche etc.) auf die zu Grunde liegenden Daten wirken. Abgesehen davon werden solche Kurven selten veröffentlicht, weil dadurch Rückschlüsse auf die Produktivität des Unternehmens gezogen werden können. Mir selbst sind nur eben diese IBM-Erfahrungskurve und die so genannte „Badewannenkurve“ von VW bekannt.

[11] Vgl. Eintrag Bundschuh im Forum der 15.3.3 DASMA

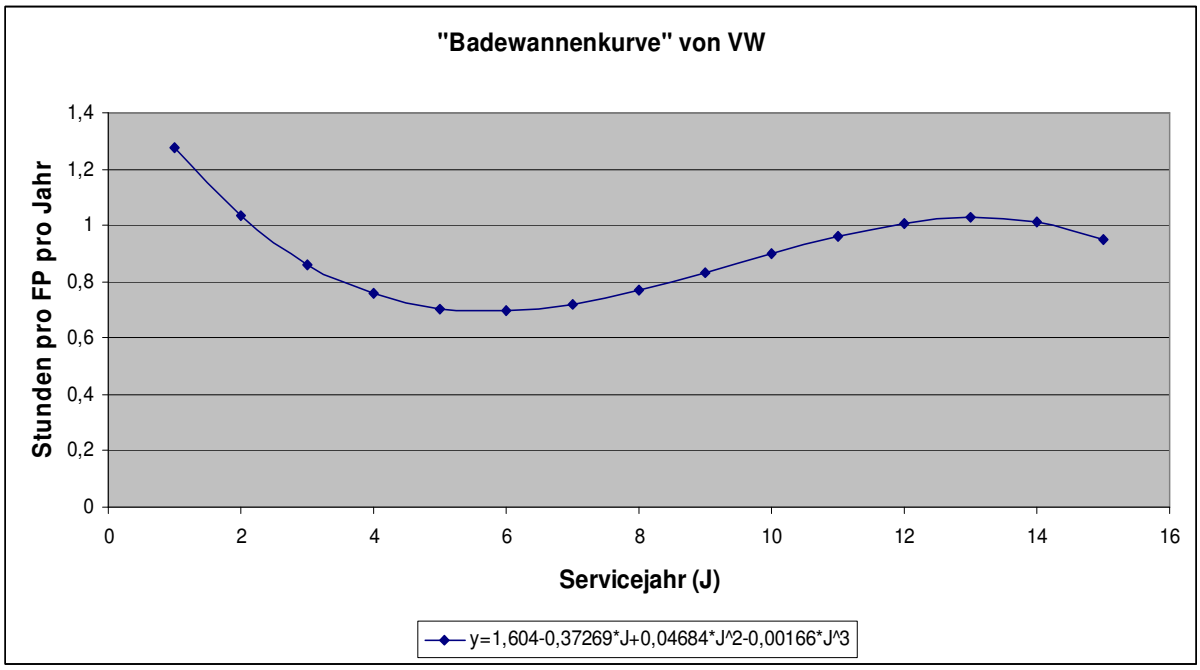

Abbildung 8: Badwannenkurve von VW

Die Badewannenkurve stellt jedoch nicht den Aufwand für die Softwareentwicklung, sondern den Wartungsaufwand eines IT-Systems im Laufe der Lebenszeit dar. Sie fällt von ca. 1,4 Stunden pro Function-Point im ersten Jahr auf ca. 0,7 Stunden im sechsten Jahr, und steigt dann wieder auf über 1 Stunde im dreizehnten Jahr.

Da es wie erwähnt schwierig ist, Erfahrungskurven von anderen Unternehmen zu verwenden, muss man, im Falle man diese Möglichkeit ins Auge fasst, auf die Erstellung einer eigenen unternehmensspezifischen Erfahrungskurve zurückgreifen, die, wie im nachfolgenden Beispiel erklärt, abläuft.

5.2.1.1 Beispiel für die Erstellung einer eigenen Erfahrungskurve

Folgendes Beispiel soll demonstrieren wie eine eigene Erfahrungskurve erstellt werden kann. Es sollten laut Bundschuh[12] mindestens 3 große, 3 mittlere und 3 kleine Projekte, als Grundlage für eine solche Kurve dienen. Man benötigt von diesen Projekten den Aufwand in Personenmonaten und die Anzahl der Function-Points. Man überträgt nun die Function-Point/Personenmonate Wertepaare in ein Koordinatensystem.

[12] Vgl. Eintrag Bundschuh im Forum der 15.3.3 DASMA

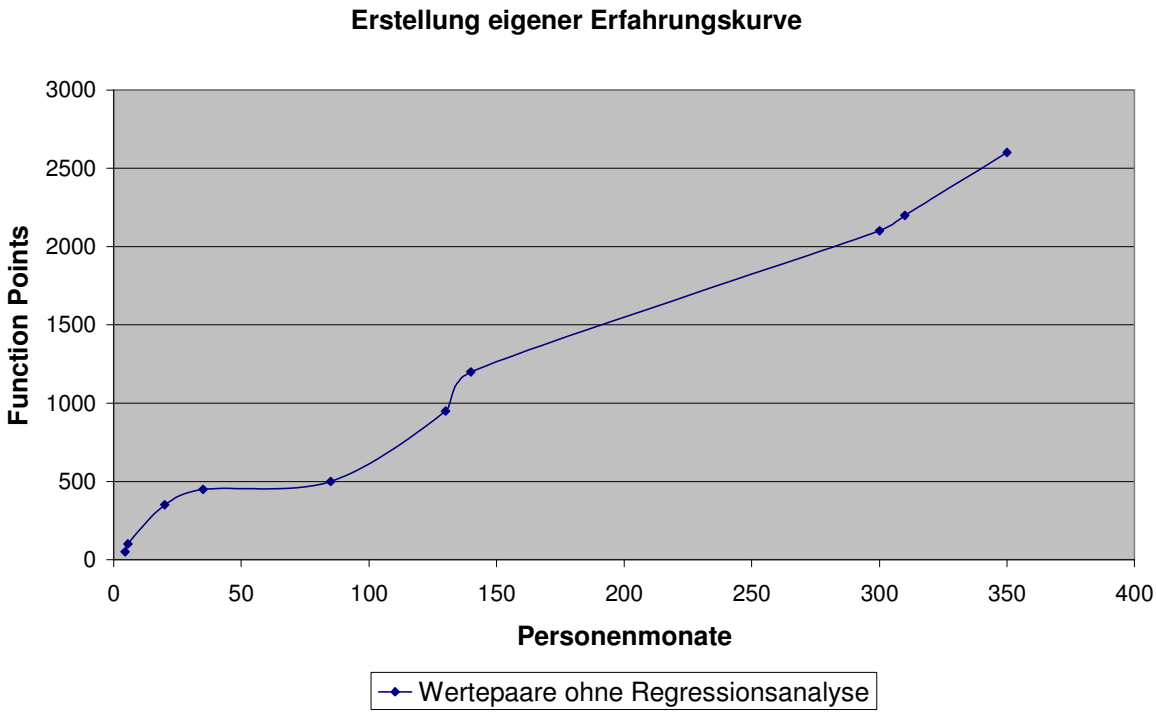

Abbildung 9: Erstellung einer Erfahrungskurve - Eintragen der Wertepaare

Anschließend kann mittels Regressionsanalyse eine Erfahrungskurve ermittelt werden. Durchzuführen ist das ganze zum Beispiel mittels Microsoft Excel. Dazu muss ein „Add-in“ aktiviert werden, dass standardmäßig im Lieferumfang enthalten ist. In der Version 2003 geschieht das im Menü Extras -> Add-Ins -> Analyse-Funktion. Ab diesem Zeitpunkt steht ein neuer Menüeintrag im Menü Extras „Analysefunktionen“ zur Verfügung. Nach Aufrufen dieses Menüpunkts wählt man die Regressionsanalyse und gibt die erfassten Wertepaare als Datenquelle an. Als Ergebnis erhält man eine Erfahrungskurve.

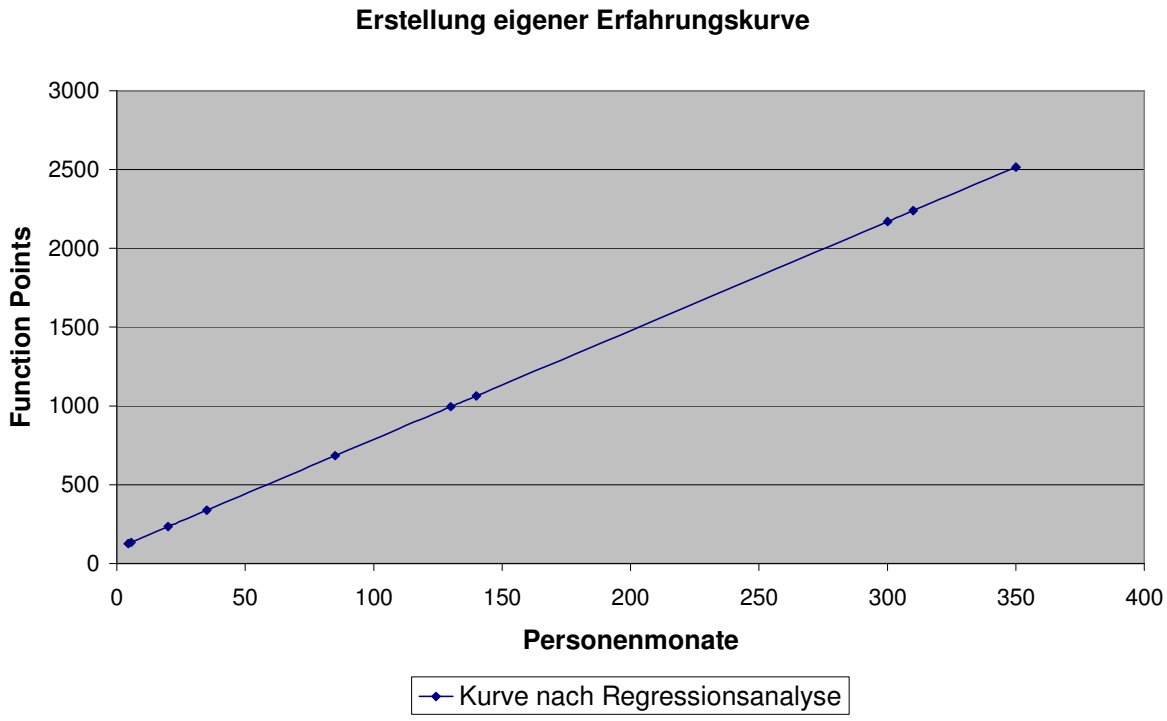

Abbildung 10: Erstellung einer Erfahrungskurve - Ergebnis

5.2.1.2 Zu beachten

Im Falle man sich im Unternehmen für den Einsatz einer solchen Erfahrungskurve entschließt, sollte man für die Regressionsanalyse auf jeden Fall auf den Expertenrat eines Statistikers zurückgreifen, und sich nicht alleinig auf Excel verlassen, da es mehrere Möglichkeiten gibt, um über Regressionsanalyse zu so einer Kurve zu gelangen, Excel jedoch nicht alle abdeckt.

5.2.1.3 Zusammenfassung

Ein Ablesen aus einer Erfahrungskurve beruht auf der Annahme, dass der Aufwand von nur einer variablen Größe, in diesem Beispiel Function-Points abhängt. In Wirklichkeit gibt es jedoch eine Vielzahl von Größen die zu berücksichtigen sind, um zu einem exakten Ergebnis zu kommen. Es wäre zum Beispiel denkbar für eine jede Art von Technologie, die im Unternehmen eingesetzt wird, eine eigene Erfahrungskurve zu erstellen. Man kann jedoch nicht für alle Einflussgrößen eine eigene Erfahrungskurve basteln, da es zu viele Einflussfaktoren gibt, die teilweise auch voneinander abhängen. Im Falle man aber

gleichartige Projekte in einem ähnlichem Umfeld betrachtet ist eine Erfahrungskurve eine ausgezeichnete Möglichkeit. Dadurch ist eine Erfahrungskurve auch für einen Einsatz als frühe Schätzung möglich, um zum Beispiel die Machbarkeit eines Projektes aufgrund beschränkter Ressourcen evaluieren zu können.

5.2.1.4 Anmerkung

Erfahrungskurven werden immer wieder im Umfeld der Function-Point-Analyse gebraucht. Ich bin jedoch der Meinung, dass eine Verwendung auch bei allen anderen Schätzverfahren möglich sein muss. Zum Beispiel könnte man Lines of Code oder Anzahl der Module dazu verwenden, um eine solche Erfahrungskurve zu erstellen.

5.2.2 Anwenden einer Schätzgleichung

Die zweite Möglichkeit ist die Anwendung einer Schätzgleichung. Im Gegensatz zum Ablesen aus einer Erfahrungskurve können mehrere Parameter in die Umrechnung einfließen. Nachfolgend eine Zusammenfassung der bekanntesten Schätzgleichungen.

5.2.2.1 COCOMO II

Eine solche Schätzgleichung ist tief in das Schätzverfahren selbst integriert. Die nähere Erläuterung würde jedoch einer genauen Erklärung und Analyse des gesamten Schätzverfahrens bedürfen und damit den Umfang dieser Arbeit sprengen. Zusammengefasst kann man sagen, dass anhand von Gleichungen und Graphen der Aufwand abhängig von der Projektphase berechnet wird.

5.2.2.2 SLIM

Auch im Umfang des Software-Lifecycle-Management (SLIM) existiert eine Formel mit der der Aufwand errechnet werden kann. Sie ist ebenfalls fixer Bestandteil des Schätzverfahrens. SLIM ist ungefähr zeitgleich mit der Function-Point Methode (1979) entwickelt worden.

5.2.2.3 Rules of Thumb

Die „Rules of Thumb“ sind eine Veröffentlichung von Capers Jones[13] in einer Zeitschrift aus dem Jahre 1996, und bestehen aus einer Zusammenfassung von Formeln zur Ermittlung verschiedener Kenngrößen. Unter anderem enthalten sie auch eine Möglichkeit zur Berechnung des Aufwandes („Rule 10“). Um den Aufwand berechnen zu können müssen zuvor zwei weitere „Rules“ ausgerechnet werden, nämlich die Durchlaufzeit („Rule 7“) und die Größe des Teams („Rule 8“).

Rule 7	Durchlaufzeit=FunctionPt ^{0,4}
Rule 8	Teamgröße=FunctionPt/150
Rule 10	Aufwand in PM=Durchlaufzeit * Teamgröße

Tabelle 4: Rules of Thumb - Berechnung des Aufwandes

Wenig beachtet wird bei der Anwendung dieser Schätzgleichung, dass Function-Points der Version IFPUG 3.0 verwendet wurden. Um diese Formeln mit IFPUG 4.x anwenden zu können muss daher eine Umrechnung der Function-Points durchgeführt werden. Ich habe dazu nur eine Quelle[14] gefunden, die besagt dass bei einer Zählung nach IFPUG 4.x 20-30% weniger Function-Points gezählt werden. Ebenfalls wenig beachtet wird die Aussage des Autors zu Beginn des Artikel, dass solche Regeln

- ungenau und bekannt für eine hohe Fehlerrate sind
- nicht für Angebote oder andere „serious purposes“ verwendet werden sollten.
- einfach verwendet werden können und eine grobe Überprüfung anderer drastischer Methoden gewährleisten.

[13] Vgl. 15.1.6 Software estimating rules of thumb
[14] Vgl. 15.2.1 Applied Software Measurement S118ff

5.2.2.4 Determining software schedules

Ein wenig früher als die „Rules of Thumb“ veröffentlichte Capers Jones in einer Zeitschrift[15] der IEEE[16] eine Formel zur Berechnung der Durchlaufzeit. Eine Berechnungsmöglichkeit des Aufwandes wird dabei nicht geboten. Die Formel beinhaltet einen „power level“ der über folgende Tabelle abgelesen wird:

Software domain	Best in Class	Average	Worst in Class
Commercial	0,39	0,42	0,45
MIS	0,41	0,43	0,46
Systems	0,43	0,45	0,48
Military	0,45	0,46	0,50

Tabelle 5: Ermitteln des „power level“ für die Berechnung

Wie aus dieser Tabelle ersichtlich, spielt die Art der Software als auch die Erfahrung des Unternehmens auf diesem Gebiet eine Rolle.

Software domain	Bedeutung
Commercial	Kommerzielle Software für die Vermarktung
MIS	Software die von einem Unternehmen für die eigene Verwendung erstellt wird
Systems	Software für physische Geräte wie Telefonanlagen, Flugkontrollsysteme, Betriebssysteme
Military	Militärische Projekte

Tabelle 6: Beschreibung der "Software domain"

[15] Vgl. 15.1.3 Determining software schedules
[16] 15.3.5 IEEE

In dieser Tabelle sind die verschiedenen Bedeutungen der „Software domain“ aufgeführt. Bei militärischen Projekten ist meiner Meinung nach aber in Frage zu stellen ob solche US-Projekte mit europäischen Militärprojekten verglichen werden können.

Erfahrungslevel	Bedeutung
Best in class	Firmen mit mehr als 50% wieder verwendbaren Komponenten sowie viel Erfahrung in diesem Bereich
Average	Durchschnittliche Erfahrung in diesem Bereich
Worst in Class	Wenig oder keine Erfahrung in diesem Bereich

Tabelle 7: Beschreibung des „Erfahrungslevel“

In dieser Tabelle werden die möglichen „Erfahrungslevel“ näher beschrieben. Nachdem der „power level“ bestimmt werden konnte, kann nach der Formel

$$\text{„Durchlaufzeit}=\text{FunctionPt}^{\text{power level}}\text{“}$$

die Durchlaufzeit berechnet werden.

So wäre zum Beispiel für ein 500 Function-Point großes MIS Projekt, bei dem wenig Erfahrung vorhanden ist, eine Durchlaufzeit von rund 17,5 Monaten ($500^{0,46}$) zu erwarten.

Eine leicht modifizierte Version dieser Formel wird auch bei „Rapid Development“[17], einem bekannten Buch mit Ratschlägen und Vorschlägen zur Softwareentwicklung, vorgestellt.

Meiner Meinung nach ist eine Anwendung für eine erste Schätzung durchaus möglich, ich halte aber die Ratschläge aus 5.2.2.3 genauso für diese Formel für gültig.

5.2.2.5 Schätzgleichungen aus Benchmarking-Datenbanken

Es existieren einige Benchmarking-Datenbanken die eine Vielzahl von Projekten beinhalten, die in verschiedensten Branchen und von verschiedensten Unternehmen durchgeführt wurden.

[17] 15.2.7 Rapid Development – Taming wild Software Schedules

Die wohl bekannteste Benchmarking-Datenbank ist die der ISBSG[18] (International Software Benchmarking Standards Group). Bei diesen Studien wird unter anderem auch mittels Regressionsanalyse (siehe auch 5.2.1 - Ablesen aus einer Erfahrungskurve) eine Formel zur Berechnung des Aufwandes ermittelt. Wie beim Ablesen aus einer Erfahrungskurve wird bei solchen Formeln aber nur eine Variable berücksichtigt. Es können sich beim Einsatz von Benchmarking-Datenbanken aber auch andere Probleme ergeben (siehe 10.2.2 „Ziele und mögliche Probleme der Benchmarking“).

5.2.2.6 Zusammenfassung

Schätzgleichungen von Schätzverfahren dienen dazu, den Aufwand im Zusammenspiel mit dem betreffenden Schätzverfahren zu ermitteln, können aber nicht außerhalb dieses Schätzverfahrens verwendet werden.
Universell einsetzbare Schätzgleichungen sind sehr einfach und schnell zu handhaben und können dadurch gut für die Überprüfung einer Schätzung oder für eine sehr frühe Schätzung herangezogen werden. Sie sind aber auch mit großer Ungenauigkeit behaftet und sollten daher nur mit großer Sorgfalt und Vorsicht eingesetzt werden. Da sich jedoch die Grundlagen der Daten auf der diese universellen Gleichungen beruhen, laufend ändern denke ich, dass entweder eine aktuelle Schätzgleichung aus einer Benchmarking-Datenbank oder eine gewartete Erfahrungskurve die besten Möglichkeiten zur raschen Aufwandsschätzung darstellen.

5.2.3 Umrechnen mittels Spezialsoftware

Es gibt Softwareprodukte die unter anderem darauf spezialisiert sind, eine Umrechnung in Personenmonate von einer gemessenen Größe entweder in Lines of Code oder Function-Points vorzunehmen. Solche Programme arbeiten gewöhnlich mit einer Mischung der vorher genannten Möglichkeiten. Das heißt es werden intern Algorithmen eingesetzt, um den Einfluss verschiedener Parameter (Einflussfaktoren) auf den Aufwand zu ermitteln. Zusätzlich gibt es eine meist große Datenbasis aus verschiedensten Projekten, die dazu verwendet wird, um Einflussfaktoren exakter zu bestimmen. So muss zum Beispiel bei dem

[18] 15.3.7 ISBSG

Produkt KnowledgePlan der Firma SPR[19], bei der Erfassung eines Projektes die Branche und die Systemart angeben werden. Dadurch kann der Aufwand und auch die Durchlaufzeit des Projektes unter Rücksichtnahme der Branche und der Systemart berechnet werden.

5.2.3.1 Direkte und indirekte Kalibrierung von Einflussgrößen

Da solche Produkte Hunderte von Parametern zur Bestimmung der Schätzgröße einsetzen, können diese nicht alle vom Anwender geändert werden. Einflussfaktoren wie zum Beispiel die verwendete Technologie oder Erfahrung des Teams können, müssen aber nicht eingegeben werden. Es gibt jedoch auch Faktoren die implizit Auswirkungen auf den zu erwartenden Aufwand haben.

Die Teamgröße hat zum Beispiel Auswirkung auf die Produktivität. Je mehr Mitarbeiter desto geringer ist die Produktivität. Dieses Phänomen lässt sich mit einem „Interaktionsverlust" begründen, man liest auch manchmal von „Reibungsverlusten"[20].

Das Zusammenspiel von so vielen möglichen Einflussfaktoren wird schnell so komplex, dass eine Berechnung ohne eine solche Spezialsoftware so gut wie nicht möglich ist.

5.2.3.2 Zusammenfassung

Um mit spezieller Software brauchbare und nachvollziehbare Ergebnisse erzielen zu können, ist es notwendig, sich genau zu informieren, welche Einflussfaktoren berücksichtigt werden können und welche nicht. Weiters empfehle ich, dass man sich ein Bild über die zu Grunde liegende Datenbasis macht. Wenn die Datenbasis zum Beispiel nur aus Projekten im US-amerikanischen Raum besteht, ist nicht unbedingt gewährleistet, dass die Ergebnisse repräsentativ für Europa sind. In so einem Fall ist es notwendig eine Kalibrierung der Anwendung vorzunehmen.

Im Rahmen der Recherchen zu meiner Diplomarbeit habe ich durchaus positive Erfahrung mit der Auskunftsbereitschaft der Hersteller gemacht.

---

[19] 15.3.11 SPR
[20] Vgl. 15.2.5 Der Termin S94ff

6 Aufwandsschätzung im Projektmanagement

Dieses Kapitel widmet sich der Aufwandsschätzung im Rahmen des Projektmanagements.

6.1 Allgemeines

Durch die Aufwandsschätzung im Rahmen des Projektmanagements wird versucht, den Aufwand (sowohl Kosten als auch Ressourcen), der in einem Projekt anfällt, möglichst genau zu bestimmen. Nur so kann gewährleistet werden, dass Termine als auch Ressourcen geplant beziehungsweise verplant werden können. Der ermittelte Aufwand ist die Grundlage, um einen Projektplan erstellen zu können und somit die Grundlage, um Inhalte, Termine und Ressourcen im Rahmen des klassischen Projektdreiecks kontrollieren zu können.

Es ist daher für die Planung und Kontrolle unerlässlich eine Abschätzung des Aufwandes vorzunehmen.

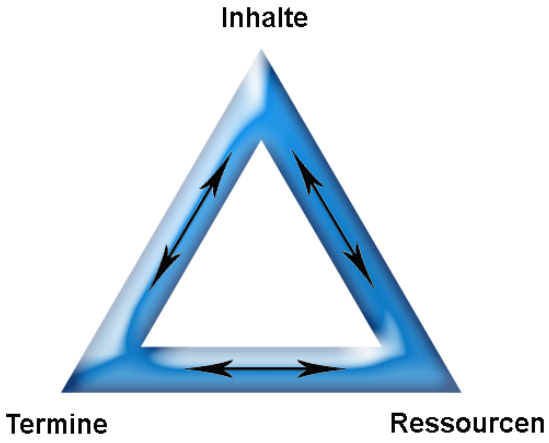

Abbildung 11: Klassisches Projektmanagementdreieck

Wie in Abbildung 11 ersichtlich, bilden die Begriffe Inhalte, Termine und Ressourcen die Eckpunkte des Projektmanagementdreiecks, und stehen in Wechselwirkung zueinander (durch schwarze Pfeile symbolisiert). Wenn sich nun zum Beispiel die Inhalte erhöhen würden, würden sich entweder Termine oder Ressourcen oder sogar beide erhöhen. Ähnliche Folgen würde eine Reduktion der Ressourcen nach sich ziehen.

Es gibt noch andere, besonders in deutschen Projektmanagementforen, verbreitete Varianten dieses Dreiecks, bei denen auch die Kanten und auch die Mitte des Dreiecks eine Bedeutung erhalten. Ich möchte mich aber an diese einfache, aussagekräftige Variante halten.

6.2 Stellung der Aufwandsschätzung im Projektmanagement

Ich denke die Aufwandsschätzung und zugehörigen Aufwandschätzverfahren sind kein Prozess innerhalb des Projektmanagements, sondern vielmehr ein Werkzeug, das vielen Teilen des Projektmanagements dienen kann, wie auch aus Abbildung 12 ersichtlich ist.

Die Literatur tendiert meiner Meinung nach dazu, die Aufwandsschätzung als Mittelpunkt des Projektmanagements zu sehen, vergisst jedoch darauf, dass zu einem erfolgreichen Projektmanagement auch noch andere Faktoren gehören. Natürlich ist die Aussage von Tom de Marco richtig „Was man nicht messen kann, kann man nicht kontrollieren“[21], jedoch bedeutet das nicht zwangsläufig, dass das Messen immer eine Kontrolle nach sich zieht.

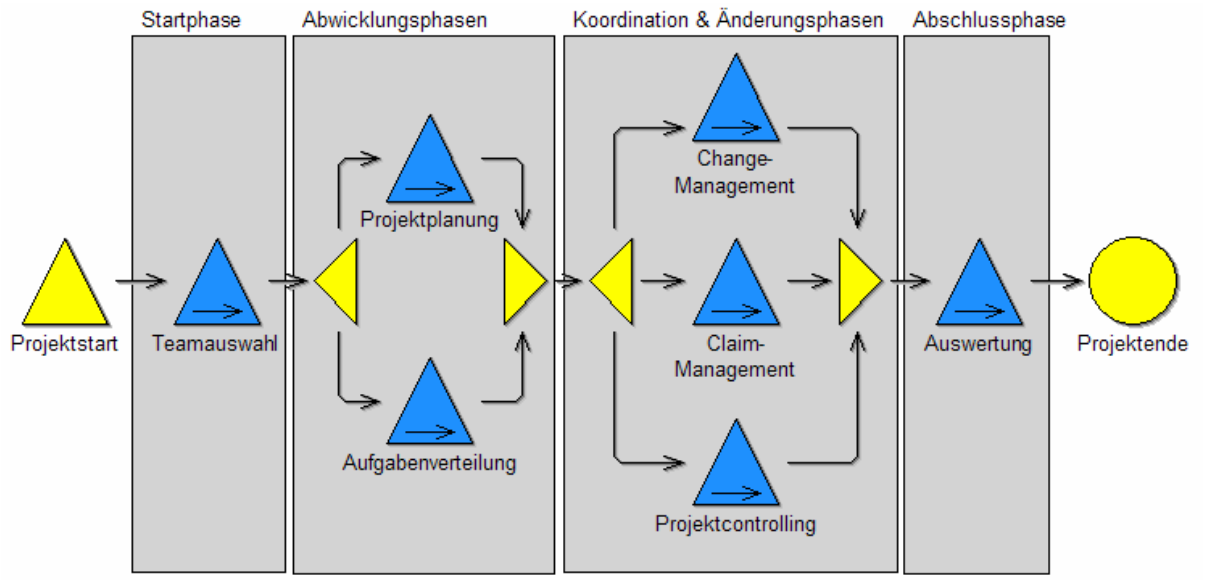

Abbildung 12: Verwendung der Aufwandsschätzung im Projektmanagement

[21] Vgl. 15.2.11 Was man nicht messen kann, kann man nicht kontrollieren, Cover

Diese Abbildung zeigt Prozesse beziehungsweise Aufgaben des Projektmanagements[22] denen das Werkzeug „Aufwandsschätzung“ dienen kann. Eingeteilt sind diese in die entsprechenden Projektphasen.

6.2.1 Startphase

Schon in dieser frühen Phase muss eine Aufwandsschätzung vorgenommen werden, um eine grobe Ressourcenplanung durchführen zu können. Für den Fall, dass eine Ermangelung von Informationen herrscht, und noch keine strukturierte Schätzung mit Hilfe eines Aufwandschätzverfahrens möglich ist, sollte ein Projekt von ähnlichem Umfang gesucht werden, um mittels Analogie Schlüsse auf die benötigte Ressourcen und Termine ziehen zu können.

6.2.2 Abwicklungsphasen

In diesen Phasen wird die eigentliche Planung durchgeführt. Um Durchlaufzeit und Kosten berechnen zu können muss zuvor der Aufwand mittels Aufwandschätzverfahren ermittelt werden (siehe auch 6.1 Allgemeines).

6.2.3 Koordinations- und Änderungsphasen

Beim Änderungsmanagement (Changemanagement) wird die Aufwandsschätzung benötigt, um Vorhersagen über die Kosten und zusätzliche Dauer treffen zu können, die Änderungen im Verlauf des Projektes bewirken. Beim Claim-Management dient die Aufwandsschätzung dazu, die Kosten die ein „Fremd-Claim“ (also ein Anspruch von anderer Seite) verursacht, zu beurteilen. Somit kann eine Entscheidung getroffen werden, ob es sich lohnt um diesen Claim zu kämpfen oder ob eine Lösung im Wege der Kulanz besser ist.

In 5.2 wurde ausgeführt welche Möglichkeiten es gibt das Schätzergebnis in Aufwand umzurechnen. Der Abgleich der Soll mit den Ist-Daten ist eine wesentliche Aufgabe des Projektcontrollings. Erst durch Schätzverfahren werden sowohl Soll-Daten, als auch Ist-Daten messbar und bilden somit die Grundlage um diese Aufgabe erfüllen zu können

[22] Vgl. 15.2.6 Projektmanagement S63

6.2.4 Abschlussphase

In der Abschlussphase ist es möglich aus dem abgeschlossenen Projekt heraus Informationen zu gewinnen, die als Grundlage für neues Wissen dienen können. Dazu ist es notwendig herauszufinden, wo die Aufwandsschätzung gute Ergebnisse geliefert hat, und wo eventuell eine Kalibrierung vorgenommen werden muss.

6.3 Umlegen des Aufwandes auf den Projektplan

Eine sicherlich wichtige, wenn nicht überhaupt die wichtigste Frage die sich für einen Projektleiter stellt, der mit dem aus der Schätzung hervorgegangenen Aufwand konfrontiert wird ist, wie jetzt dieser Aufwand auf den Projektplan umgelegt werden kann.

Dieser Teil der Aufwandsschätzung wird, wie auch schon die Ermittlung des Aufwandes selbst (siehe 5 Berechnen des Aufwandes), in der Literatur leider nur sehr dürftig behandelt. Nachfolgende Methoden wurden von mir ausfindig gemacht und werden hier vorgestellt, anschließend werde ich versuchen eine Anregung für die Ermittlung einer eigenen Möglichkeit zu geben.

6.3.1 Einsatz von Spezialtools

Wiederum ist der Einsatz von Spezialsoftware möglich. Als Beispiel möchte ich KnowledgePlan der Firma SPR[23] nennen. Diese, die Schätzung begleitende Software (Messung wird nicht abgedeckt), erstellt automatisch einen Projektplan. Der Projektplan ist in Phasen aufgeteilt, die dem klassischen Wasserfall-Modell ähneln. Es wird aber auch in kleinere Projektphasen bis hin zu einzelnen Arbeitspaketen verzweigt.

6.3.1.1 Kritische Betrachtung

Beim Einsatz von Spezialtools wird ein Projektplan erstellt, der meist auf einen bestimmten Entwicklungsansatz abzielt. Somit muss man sich mit der Abwicklung des Projektes auf einen solchen Entwicklungsansatz zurückziehen, das muss wiederum aber nicht der beste Ansatz für dieses Projekt sein.

---

[23] 15.3.11 SPR

Auch als sehr negativ zu bemängeln ist die Vielzahl von Spezialtools die als „helle Sterne“ erscheinen, jedoch nach kurzer Zeit wieder verschwinden, oder nicht weiterentwickelt werden.

6.3.2 Einsatz der Prozentsatzmethode

Diese sehr häufig eingesetzte Methode ist selbst auch ein Schätzverfahren und wird noch in 9.6 - Prozentsatzmethode näher behandelt. Diese Methode kann aber auch verwendet werden, um den ermittelten Aufwand auf die einzelnen Projektphasen aufzuteilen. Dieser Einsatz ist üblich beim Function-Point-Verfahren und sogar fixer Bestandteil im COCOMO II-Verfahren, bei dem die Verteilung des Aufwandes abhängig von der Art der Anwendung aus einem Graphen abgelesen wird.

6.3.2.1 Kritische Betrachtung

Der Einsatz dieser Methode ist zwar die gebräuchlichste, meiner Meinung nach jedoch von sehr zweifelhafter Qualität. Ich möchte meine Meinung anhand des folgenden Beispiels mit Function-Points erklären, wobei das Gleiche für COCOMO II und andere Schätzverfahren gilt:

Für ein Software-Produkt, das aus 30 verschiedenen Modulen besteht werden zur Realisierung 500 gewichtete Function-Points ermittelt. Dazu wurde für ein jedes Modul jede einzelne Funktion analysiert und gemessen. Der Aufwand wurde anschließend aus einer Erfahrungskurve abgelesen, und liegt bei 85 Personenmonaten.

Beim Umlegen wird jetzt aber nur der Aufwand auf die Projektphasen aufgeteilt, die Daten für die einzelnen Module und Funktionen werden dabei nicht berücksichtigt und gehen somit verloren!

6.3.2.2 Anmerkung

Im Falle man sich für die Prozentsatzmethode entschließt sollte man sich auf jeden Fall Gedanken machen, welcher Aufteilungsschlüssel verwendet werden kann, um nicht schon ermittelte Informationen wieder zu verlieren.

6.3.3 Andere Möglichkeiten

Meiner Meinung nach ist es möglich nach Anwenden der Prozentsatzmethode den ermittelten Aufwand entsprechend der gemessenen Größe pro Modul weiter zu unterteilen, und somit keine Information zu verlieren. Wenn zum Beispiel in der Realisierungsphase 40% von 100 Personenmonaten anfallen, ist ein Teil der 40 Personenmonate für allgemeine Aufgaben wie Projektmanagement oder Administration zurückzubehalten (nehmen wir an 20%), die verbleibenden 32 Personenmonate werden dann je nach Größe der Module auf diese aufgeteilt.

Dieses Beispiel soll nur eine Möglichkeit aufzeigen. Es ist natürlich notwendig Anpassungen vorzunehmen, abhängig von dem gewählten Phasenmodell oder der Art des Moduls. Wie schon eingangs erwähnt existiert dazu sehr wenig Literatur.

6.4 Wiederholen der Aufwandsschätzung

Eine Aufwandsschätzung muss im Projektverlauf mehrmals wiederholt werden, um

- exakter zu werden
- mögliche Fehler aufzuzeigen
- angefallene Änderungen beurteilen zu können

Erst durch eine Wiederholung der Aufwandsschätzung wird es möglich den Projektverlauf zu kontrollieren.

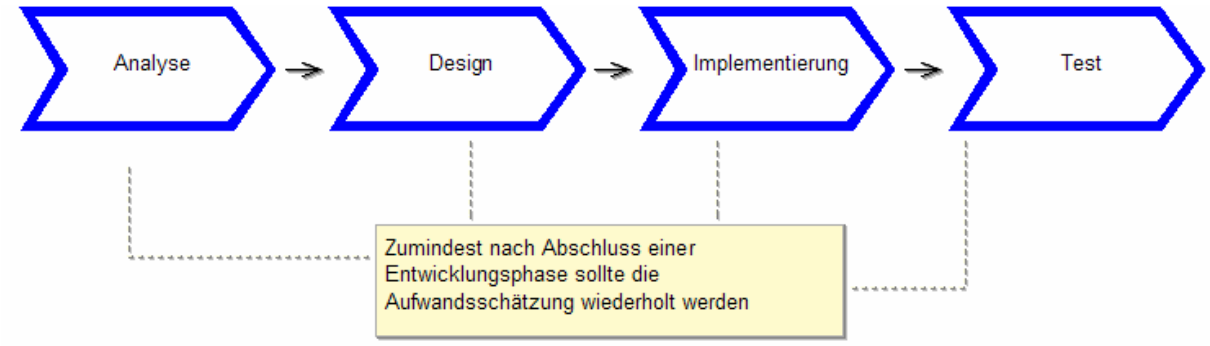

Abbildung 13: Wiederholung der Aufwandsschätzung

Wie diese Abbildung anhand eines sehr stark vereinfachten Phasenmodells zeigt, ist eine Wiederholung der Aufwandsschätzung zumindest nach jeder Phase zu empfehlen.
Ebenfalls ist es wichtig nach jeder Änderung eine Aufwandsschätzung durchzuführen um die angefallenen Änderungen zu beurteilen und die Auswirkungen auf das Projekt planen zu können. Da man bei einer solchen Anzahl von Schätzungen rasch den Überblick verliert, empfehle ich auf jeden Fall den Einsatz von elektronischen Hilfsmitteln, wie sie auch bei der Dokumentation (siehe auch 4.4 Einsatz von unterstützender Software) verwendet werden können.

6.5 Aufwand für die Aufwandsschätzung

Die Frage nach dem Aufwand der für die Aufwandsschätzung gebraucht wird, wird immer wieder gestellt und diskutiert[24]. Ich selbst meine, dass die Aufwandsschätzung ein Werkzeug des Projektmanagements ist (siehe auch 6.2), und sehe keinen Bedarf dafür, extra Kosten auszuweisen, da die Aufwandsschätzung das Projektmanagement, Projektcontrolling, Changemanagement etc. bei der bestmöglichen Durchführung der Aufgabe unterstützt. Man würde auch nicht fragen welcher Aufwand bei der Bedienung von Microsoft Projects auftritt.

Für den Fall, dass unbedingt die Kosten für die Aufwandsschätzung ausgewiesen werden müssen, sollte man aber auch eine Berechnung der Einsparungen, die dadurch erwirtschaftet werden, vornehmen.

[24] Vgl. Diskussionsbeitrag im Forum der 15.3.3 DASMA

7 Aufwandsschätzung im Unternehmen

In diesem Kapitel wird betrachtet, welche Maßnahmen bei der Einführung von Aufwandschätzverfahren im Unternehmen helfen und diese unterstützen können. Weiters wird analysiert welche Vorteile die Aufwandsschätzung dem Unternehmen bietet.

7.1 Maßnahmen

Folgende Maßnahmen können die Aufwandsschätzung selbst oder die Akzeptanz dieses Instrumentes im Unternehmen wesentlich verbessern.

7.1.1 Einbeziehung der Mitarbeiter

Ich betrachte die Mitarbeiter als wesentlichen Erfolgsfaktor bei der Einführung der Aufwandsschätzung. Alle betroffenen Mitarbeiter müssen auf diesem Gebiet nicht nur gut ausgebildet sein, sondern auch über den Sinn und die Vorteile einer Aufwandsschätzung für das Unternehmen und für sich selbst aufgeklärt werden.

Es müssen auch Ängste abgebaut werden die eine Akzeptanz gefährden könnten. Auch das Management darf keine falschen Erwartungen in Bezug auf die Aufwandschätzverfahren haben. Man muss sich im Klaren sein, dass die Verwendung eines Werkzeuges nicht grundlegende oder strukturelle Probleme im Unternehmen lösen kann.

7.1.2 Erweiterte Ausbildung für die Projektleiter

Einer speziellen Ausbildung bedürfen die Projektleiter, die Aufwandsschätzungen durchführen und die Verantwortung dafür übernehmen müssen. Ein großer Fehler wäre es den Projektleitern bei der Einführung zu wenig Hilfestellung zu bieten und dadurch zu wenig Akzeptanz in diesem wichtigen Personenkreis zu finden.

Die Grundlage für die Ausbildung einer Aufwandsschätzung muss aber eine fundierte Projektmanagementausbildung bilden, die es überhaupt erst erlaubt, das Werkzeug Aufwandsschätzung sinnvoll im Projekt einzusetzen.

7.1.3 Gründung eines Competence-Centers

In manchen Büchern wird die Gründung eines Competence-Centers vorgeschlagen, welches Aufgaben rund um die Aufwandsschätzung wahrnimmt. Dazu zählen sowohl die Hilfestellung und Mitarbeit bei der Aufwandsschätzung selbst als auch laufende Verbesserungen und Anpassungen des Verfahrens an das Unternehmen. Ich bin der Meinung, dass nicht unbedingt ein eigenes Competence-Center dafür gegründet werden muss. Ich halte eine Einbettung als Werkzeug in die unternehmensinterne Projektmanagementprozesse ohne Competence-Center durchaus für möglich. Im Falle der Notwendigkeit einer Hilfestellung, kann diese auch von externen Anbietern zugekauft werden. Man sollte sich bei der Entscheidung, ob eigenes Competence-Center notwendig ist oder nicht, auch die Frage stellen, ob ein Competence-Center nicht auch mehr Aufgaben im Rahmen des Projektmanagements wahrnehmen könne, als alleinig die Unterstützung der Anwendung von Aufwandschätzverfahren.

7.1.4 Erstellung von Richtlinien zur Aufwandsschätzung

Ich habe die Erfahrung gemacht, dass gerade bei der Aufwandsschätzung sehr oft nicht klar ist, in welcher Art und Weise die Schätzungen durchzuführen sind. Es werden von einem Mitarbeiter „Testing“ und „Dokumentation“ eingerechnet, ein anderer lässt sie weg im festen Glauben, diese Tätigkeiten würden vom Projektleiter kalkuliert. Es herrscht auch oft Unklarheit, ob Ausbildungszeiten oder Einarbeitungszeiten, die in einem Projekt anfallen, in das Projekt eingerechnet werden. „Welche Risiken sind schon im Stundensatz enthalten, welche müssen vom Projektleiter kalkuliert werden“, ist eine andere Frage die nicht immer klar ist. Beispiele für solche Missverständnisse, die zu massiven Verzögerungen und Verlusten führen können, gibt es viele.

Aus diesem Grund muss mit Richtlinien und Vorgaben Klarheit geschaffen werden. Ich meine auch, dass solche Richtlinien einem steten Wandel sowie Anpassung im Rahmen des Sinnvollen unterliegen müssen.

7.2 Vorteile aus der Aufwandsschätzung für das Unternehmen

Für das Unternehmen selbst ergeben sich einige Vorteile, die direkt aber auch indirekt mit der Einführung der Aufwandsschätzung zu tun haben. Es folgt hier eine Zusammenfassung der wichtigsten Punkte.

7.2.1 Projektübergreifende Mitarbeiterplanung

Durch eine strukturierte und in das Projektmanagement eingebettete Aufwandsschätzung können Mitarbeiter verplant werden, ohne Gefahr zu laufen bei einer Verzögerung eines Projektes ein anderes Projekt in Ressourcenengpässe zu führen. Es wird auch besser möglich, Urlaub, Fortbildung und Administrationszeiten zu planen, was wiederum hilft Ressourcenengpässe zu vermeiden.

7.2.2 Besseres Multiprojektmanagement

Der Geschäftsführung verhilft die Aufwandsschätzung über die Mitarbeiterplanung zu der Möglichkeit, sich auf Grund von Fakten für oder gegen die Durchführung von Projekten zu entscheiden, oder sich auf die Suche nach anderen Alternativen zu machen.

Durch die Aufzeichnungen, die im Laufe der Zeit durch die Aufwandsschätzung gewonnen werden, können auch die Risiken und Möglichkeiten der Projekte besser abgeschätzt werden, somit werden weitere Entscheidungsgrundlagen für das Multiprojektmanagement geschaffen.

7.2.3 Wissensmanagement

Eine Vielzahl von Informationen wird durch die Dokumentation der Aufwandsschätzung verfügbar, womit die Grundlage geschaffen wird, Wissen zu schaffen. Durch weitere Verfeinerung dieser Informationen, können unternehmensspezifische Wissenslandkarten erstellt werden. Diese Wissenslandkarten können der Personalplanung helfen rechtzeitig Mitarbeiter mit der benötigten Qualifikation einzustellen, können aber auch dazu verwendet werden, bei Wissensbedarf anzuzeigen, in welchen Unternehmensbereichen bestimmtes Wissen vorhanden ist.

8 Messverfahren und Metriken

Dieses Kapitel gibt einen Überblick über die vorhandenen Metriken und Messverfahren im Umfeld der Softwareentwicklung geben. Es wird untersucht werden welche Messverfahren aus welchem Grund für die Aufwandsschätzung geeignet sind.

8.1 Definition Metrik (Softwaremetrik) – Messverfahren

Es gibt einige Definitionen für den Begriff „Metrik“. So hat zum Beispiel Fenton schon im Jahre 1991 vier verschiedene Definitionen für den Begriff „Metrik“ gefunden[25]. Ich denke im Umfeld dieser Arbeit ist folgende Definition die am besten zutreffende:

„Eine Softwaremetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet“[26].

Wenn ich nun den Begriff Messverfahren mit einem „Verfahren zur Messung“ definiere, umfasst das Messverfahren alle Tätigkeiten inklusive der Anwendung der Metrik, die notwendig sind, um eine Messung durchzuführen.

8.2 Arten von Messverfahren und Metriken

Im Zusammenhang mit der Aufwandsschätzung gibt es zwei grundlegende Arten von Messverfahren. Diese bestimmen entweder

- die Größe der Software
- oder die Funktion der Software

Es existiert auch noch eine Vielzahl von anderen Messverfahren die zum Beispiel die Komplexität der Software messen. Diese sind für Aufwandschätzverfahren jedoch wenig geeignet. In anderen Bereichen ist ein Einsatz solcher Metriken aber durchaus möglich. Aus diesem Grund werde ich versuchen unter 8.2.3 „Traditionelle Metriken“ einen Überblick über diese zu geben.

---

[25] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S183
[26] Vgl. 15.3.12 Wikipedia – Begriff „Metrik (Software)“

8.2.1 Messung der Größe - Lines of Code Metrik

Software im Allgemeinen besteht aus dem so genannten „Sourcecode“. Um die Größe eines Programms zu messen wird dieser „Sourcecode“ hergenommen. Das Zählmaß ist in diesem Falle die einzelne Code-Zeile. Man spricht von Lines of Code (LOC) oder Kilolines of Code (KLOC).

Im Zusammenhang mit Lines of Code kommt auch immer wieder das bekannte Schätzverfahren, COCOMO (zur Unterscheidung zu COCOMO II öfters auch als COCOMO81 bezeichnet), in der Literatur vor. Lines of Code ist ein wichtiger Bestandteil dieses Schätzverfahrens. Im Zusammenhang mit COCOMO II, einer komplett überarbeiteten Variante des ursprünglichen Verfahrens, wird von SLOC (Source-Lines of Code) gesprochen.

Software in Lines of Code zu messen ist eine sehr umstrittene Technik. Sehr viele Experten im Bereich von Aufwandschätzverfahren wie zum Beispiel Capers Jones oder Tom DeMarco, lassen kein gutes Haar an dieser Methode. Ich werde hier die meiner Meinung nach wichtigsten Argumente dafür und dagegen zusammenfassen.

8.2.1.1 Keine einheitlichen Standards

Beim Versuch der händischen Zählung stößt man sehr bald auf Sonderfälle, wie zum Beispiel:

- mehrere Programmbefehle in einer Zeile
- Kommentarzeilen
- Leerzeilen

Damit stellt sich die Frage nach einer Richtlinie zur Zählung, da man ja schon bestehende Standards wieder verwenden möchte. Solche Richtlinien zur Zählung oder internationale Standards hat es paradoxerweise sehr lange überhaupt nicht gegeben, obwohl Programme schon seit den Anfängen der Programmierung in Lines of Code gemessen werden.

Auch heute sind die vorhandenen Standards wenig verbreitet und auch wenig bekannt. Einige Ansätze gibt es, wie zum Beispiel von der IEEE (Institute of Electrical & Electronics

Engineers[27]), die die Richtlinie 1045 für die Zählung von Lines of Code entwickelt hat. Diese Richtlinie ist leider nur Mitgliedern der IEEE zugänglich, was wiederum eine Verbreitung dieser Richtlinie nicht sonderlich fördert. Die Bekanntheit der Richtlinie 1045 ist ebenfalls sehr gering. So beklagt in ein und demselben Buch ein Autor das Fehlen einer solchen[28], der zweite Autor verweist hingegen auf eben diese[29].

Es scheint, dass das Fehlen einer solchen Richtlinie schon lange ein Problem ist. Eine ausführliche Behandlung der Problematik hat Capers Jones in einem seiner Bücher geliefert[30]. Demnach wurde ein Standard schon im Jahre 1978 gefordert.

Im Zuge der Zählung von Lines of Code liest man auch manchmal von Non-Commented-Source-Statements (NCSS), mit dem Befehlszeilen ohne Kommentar oder Leerzeilen bezeichnet werden.

8.2.1.2 Verschiedene Programmiersprachen – verschiedene Anzahl von Lines of Code

Eine große Fehlerquelle beim Einsatz der LOC-Metrik, ist die Tatsache, dass sich die Anzahl der Codezeilen mit der Programmiersprache ändert. Somit muss beim Vergleich verschiedener Projekte auf die Programmiersprache geachtet werden. Bei höheren Programmiersprachen werden üblicherweise weniger Codezeilen produziert.

In der Literatur wird in diesem Zusammenhang auch öfters von einem mathematischen Paradoxon gesprochen. Aus diesem Grund hier eine kurze Erläuterung anhand eines Beispiels. Bei diesem Beispiel wird der Personenmonat mit 5000 Geldeinheiten gerechnet.

---

[27] 15.3.5 IEEE
[28] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S202
[29] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S134
[30] Vgl. 15.2.1 Applied Software Measurement S53ff

Programmiersprache	Assembler	Ada
Lines of Code	100.000	25.000
Aufwand in Personenmonaten	200	100
Kosten	1.000.000	500.000
Kosten pro Zeile	10	20
Lines of Code pro Personenmonat (LOC/PM)	500	250

Tabelle 8: Mathematisches Paradoxon bei Lines of Code[31]

In dieser Tabelle ist ein und dasselbe Programm in Assembler und Ada gegenübergestellt. Aufgelistet sind die Anzahl der Lines of Code, der Entwicklungsaufwand und die damit verbundenen Kosten. Die letzten beiden Reihen enthalten Kennzahlen, nämlich Kosten pro Zeile und Lines of Code pro Personenmonat.

Das Ergebnis ist offensichtlich falsch, da die Kennzahlen für Assembler besser zu sein scheinen. Begründen lässt sich das ganz einfach, denn es müssen ja zwangsläufig mehr Codezeilen für Assembler erzeugt werden, da es eine maschinennahe Programmiersprache ist. Somit fallen beide Kennzahlen für Assembler besser aus.

Im Gegensatz zu der Literatur bin ich aber nicht der Meinung, dass dieses „Paradoxon“ ein Hinweis auf die Unzulänglichkeit dieser Metrik ist, vielmehr wurde die Produktivitätskennzahl LOC/PM falsch berechnet. Um zu einem richtigen Ergebnis zu kommen hätte man eine Umrechnung auf assembleräquivalente Codezeilen vornehmen müssen.

Ich nehme an, dass in der Vergangenheit sehr oft Lines of Code, egal welcher Programmiersprache, miteinander verglichen wurden, und dadurch hat dieses „mathematische Paradoxon“ Eingang in die Literatur gefunden.

[31] Tabelle vereinfacht Literatur (Vgl. 15.2.1 Applied Software Measurement) S58 entnommen

8.2.1.3 Umrechnung LOC/FP

Um zu einem vergleichbaren Maß an Lines of Code zu kommen müssen, wie im vorigen Absatz erwähnt, die Lines of Code sozusagen „auf den gleichen Nenner“ gebracht werden. Zu diesem Zweck existieren Umrechnungstabellen. Mit diesen Umrechnungstabellen ist eine Umrechnung der Programmiersprache in assembleräquivalente Codezeilen oder in Function-Points möglich. Es existieren nachfolgende Umrechnungstabellen die meiner Meinung nach plausibel und gewartet erscheinen. Alle anderen, immer wieder im Internet anzutreffenden Tabellen, sind meist alte Versionen der SPR-Tabellen oder von zweifelhafter Qualität.

8.2.1.3.1 Umrechnungstabelle SPR

Diese Umrechnungstabelle, die als Ergebnis gewichtete Function-Points liefert, stammt von der Firma SPR. Seit einiger Zeit ist der Bezug auch nur mehr kostenpflichtig möglich, auch wenn im COCOMO II Buch[32] noch ein Downloadlink abgedruckt wurde, der aber leider nicht mehr existiert. Es gibt jetzt nur noch einen Hinweis auf die Möglichkeit des käuflichen Erwerbs.

Diese Umrechnungstabelle wurde dafür erstellt im Kontext des „Backfiring“, einer von der Firma SPR erfundenen Methode, verwendet zu werden. Eine Verwendung außerhalb der „Backfiring“ Methode ist eigentlich nicht vorgesehen. Praktisch wird dafür aber so gut wie nur diese Tabelle verwendet. Die Verwendung dieser Tabelle ohne Wissen über die zu Grunde liegende Methode und die damit verbundenen Risiken kann zu erheblichen Unschärfen im Messergebnis führen (siehe auch 10.1 „Backfiring“).

8.2.1.3.2 Umrechnungstabelle QSM

Die zweite Umrechnungstabelle ist von der Firma QSM[33] und unter http://www.qsm.com/FPGearing.html abrufbar. Das Ergebnis besteht aus ungewichteten Function-Points. Die Grundlage für diese Tabelle liefern mit Hilfe des Tools der Firma QSM abgeschlossene Projekte.

---

[32] Vgl. 15.2.8 Software Cost Estimation with COCOMO II S19
[33] 15.3.9 QSM

8.2.1.3.3 Umrechnungstabelle David Consulting Group

Auf diese Tabelle bin ich durch ein veröffentlichtes Interview mit Barry Boehm, dem Erfinder von COCOMO gestoßen. Diese wird von der David Consulting Group[34] unter http://www.davidconsultinggroup.com/indata.htm veröffentlicht, das Ergebnis verwendet gewichtete Function-Points als Zählmaß. Diese Tabelle weicht sehr von den Zahlen der Umrechnungstabelle von SPR ab. Am augenscheinlichsten ist, dass die Basisgröße „Basic Assembly“, die verwendet wird, um den Umrechnungsfaktor zu verwenden massiv, verändert wurde.

8.2.1.3.4 Zusammenfassung - Umrechnung LOC/FP

Bei einer jeden Umrechnung von Lines of Code zu Function-Points wird eine Unschärfe erzeugt, da bis heute keine verlässliche Beziehung zwischen Lines of Code und Function-Points gefunden werden konnte. Dadurch wird das Ergebnis negativ beeinflusst.

8.2.1.4 Gemischte Programmiersprachen

Es kommt in Softwareprojekten auch vor, dass Programmiersprachen gemischt verwendet werden. So besteht zum Beispiel die Möglichkeit aus C-Programmen heraus SQL-Anweisungen abzusetzen. Ein solcher Code lässt sich sehr schwer auf ein vergleichbares Niveau bringen, dabei sind auch Umrechnungstabellen nur begrenzt verwendbar. Ähnlich schwierig gestaltet sich die Bewertung von Programmen die teilweise oder ganz von Programmgeneratoren erstellt wurden.

8.2.1.5 Anzahl der Codezeilen abhängig vom Entwickler und von Standards

Die Anzahl der Codezeilen mit der ein Programm geschrieben wird ist auch abhängig von Erfahrung und Stil des Entwicklers. Auch stärker modularisierte Programme haben tendenziell eine höhere Anzahl von Codezeilen, da für ein jedes Unterprogramm ein eigenes Sprachkonstrukt erstellt werden muss. Auch ein besser lesbares Programm kann eine höhere Anzahl von Codezeilen verursachen.

---

[34] 15.3.4 David Consulting Group

Ebenfalls besteht die Möglichkeit, dass vorhandene Unternehmens- oder Projektstandards mehr Codezeilen verursachen.

All diese Faktoren tragen zu einer Unschärfe bei der Messung aber auch zu Problemen bei der Kontrolle des Messergebnisses bei, und weisen auf eine Abhängigkeit vom Umfeld, Vorgaben, Unternehmen und Projekt hin.

8.2.1.6 Schätzung im Vorhinein, Überprüfbarkeit im Nachhinein

Die Schätzung der Anzahl der Lines of Code muss zum Ziel der Aufwandsschätzung im Vorhinein erfolgen. Das halte ich für sehr problematisch, da ich der Meinung bin, dass kein Entwickler im Vorhinein Auskunft darüber geben kann, aus wie vielen Programmzeilen sein Programm bestehen wird.

Die Überprüfung des Ergebnisses kann jedoch erst nach Fertigstellung der Software (das heißt nach der Codierung) erfolgen. Damit können Fehler die bei der Messung unterlaufen sind erst recht spät im Projektverlauf aufgedeckt werden.

8.2.1.7 Objektivität des Messverfahrens

Lines of Code gelten als objektiv. Objektiv in dem Sinne, dass sie ohne Einflussnahme eines Menschen gezählt werden können.

Bei der Anwendung von Aufwandschätzverfahren misst jedoch wiederum der Mensch die Lines of Code bevor der Code überhaupt existiert. Dementsprechend ist eine solche Objektivität im Kontext der Aufwandsschätzung nicht gegeben.

8.2.1.8 Unabhängigkeit von der Art der Anwendung

LOC sind unabhängig von der Art der Anwendung, es ist also gleichgültig, ob die zu entwickelnde Software eine Systementwicklung, Datenbankentwicklung oder ein Echtzeitsystem ist.

8.2.1.9 Objektorientierte Anwendungen nicht messbar

Die Messung von Anwendungen, die mittels des OO-Paradigmas entwickelt wurden ist mit der LOC-Metrik nicht möglich, da der Nutzen des OO-Paradigmas in erhöhter Qualität sowie Wiederverwendung von Programmteilen liegt. Beides ist mit der LOC-Metrik nicht messbar.

8.2.1.10 Zusammenfassung - Lines of Code Metrik

Ich bin der Meinung, dass Messverfahren die mit der LOC-Metrik arbeiten für Aufwandsschätzungen aus den aufgelisteten Gründen ungeeignet sind.

- \+ unabhängig von der Art der Anwendung
- ± Objektivität
- – keine verbreiteten einheitlichen Standards zur Zählung
- – LOC abhängig von Programmiersprache, dadurch ist eine Umrechnung notwendig, was zu Unschärfen führt
- – bei gemischten Programmiersprachen ergeben sich Unschärfen
- – Anzahl der LOC abhängig vom Entwickler und von Standards
- – Überprüfung nur im Nachhinein möglich
- – objektorientierte Anwendungen nicht messbar

Zu erwähnen wäre auch, dass sogar in COCOMO II empfohlen wird Messverfahren zu verwenden, die die Funktion des Programms messen. Auch bei diesem Schätzverfahren, in dem Lines of Code einen festen Bestandteil haben, wird für die Messung von Lines of Code Abstand genommen.

Für die Ermittlung von anderen Kennzahlen oder zum Vergleichen mit anderen Projekten kann ich mir eine Anwendung von Lines of Code aber durchaus vorstellen. Zum Beispiel kann man mit der Zählung von Lines of Code auf Unix-Ebene (über den Befehl „CAT“) innerhalb von Sekunden eine ungefähre Größe der Anwendung ermitteln, und somit wichtige, wenn auch nicht besonders genaue Eckdaten gewinnen.

8.2.2 Messung der Funktion - Function-Points Metrik

Wenn ein Programm anhand von Funktionen gemessen wird, wird nicht der Sourcecode, sondern die Funktionalität als Zählmaß herangezogen. Das im Zusammenhang mit der Aufwandsschätzung verwendete Zählmaß heißt „Function-Point“ und ist ein Teil des gleichnamigen Schätzverfahrens. Function-Points werden auch des Öfteren als „Funktionspunkte“ bezeichnet. Die gemessenen Function-Points werden als ungewichtete Function-Points (**U**nadjusted **F**unction-**P**oints - UFP) bezeichnet, die durch eine anschließende Schätzung zu gewichteten Function-Points (**A**djusted **F**unction-**P**oints – AFP) werden.

Mittlerweile gibt es verschiedene Varianten von „Function-Point“ wie zum Beispiel „COSMIC-Full-Function-Points“ oder „Data-Points“. In einigen Varianten unterscheidet sich auch die Messung. Ein Überblick über diese Varianten mit einer Kurzbeschreibung kann Tabelle 12 entnommen werden.

Wie bei der LOC-Metrik erfolgt auch hier abschließend eine Zusammenfassung der Argumente, die für das Messverfahren, und Argumente, die gegen das Messverfahren sprechen.

8.2.2.1 Einheitliche Standards

Function-Points sind von der International Function-Point-User-Group (IFPUG)[35] standardisiert. Für eine Function-Point Zählung steht das aktuelle „Counting-Practice-Manual“, kurz CPM genannt in der Version 4.2 zur Verfügung, das sich nur wenig von der Version 4.1 unterscheidet.

Ungewichtete Function-Points nach Version 4.1 sind sogar als Standard der International Organization for Standardization (ISO)[36] im Standard ISO/IEC 20926:2003 anerkannt. Es sind auch weitere Varianten der Function-Points wie zum Beispiel Mark II (ISO/IEC 20968) oder NESMA (ISO/IED 24570) standardisiert und ISO-zertifiziert.

---

[35] 15.3.6 IFPUG
[36] 15.3.8 ISO

8.2.2.2 Überprüfbarkeit jederzeit möglich

Die ursprüngliche Messung der Function-Points ist jederzeit im Projektverlauf überprüfbar und sogar zwingend gefordert (siehe auch 6.4 „Wiederholen der Aufwandsschätzung“). Festgestellten Abweichungen kann aus diesem Grund rechtzeitig entgegengesteuert werden.

8.2.2.3 Subjektivität des Messverfahrens

Function-Point-Zählungen können nicht maschinell durchgeführt werden, da es dem Grundsatz der Function-Point Methode widerspricht, demnach die Zählung aus Benutzersicht erfolgen muss[37].

Aber auch wenn die Messung von Menschen vorgenommen wird, kann das nicht automatisch mit einer subjektiven Beeinflussung der Messung gleichgesetzt werden. Eine ausführliche Dokumentation der Fakten und Annahmen, die zu einer Messung geführt haben, hält sicherlich auch von einer, warum auch immer absichtlich getätigten, Beeinflussung ab.

8.2.2.4 Verschiedene Varianten

Mittlerweile gibt es zirka zehn verschiedene Arten von Function-Point-Verfahren, bei denen auch die Messungen teilweise unterschiedlich vorgenommen werden. Es ist nicht einfach, einen Überblick über diese Varianten und die Vor- und Nachteile zu bekommen. Eine Mischung verschiedener Varianten im gleichen Unternehmen sollte jedoch vermieden werden, da die Umrechnung von der einen in die andere Variante nur bedingt möglich ist. Einige dieser Varianten sind überdies so gut wie nicht mehr in Verwendung, und haben eher historischen Charakter als praktischen Nutzen.

8.2.2.5 Anpassbar an die Art der Anwendung

Die in 8.2.2.4 erwähnte Vielzahl an Varianten erlaubt aber andererseits eine gute Anpassung an die Art der Anwendung. Es gibt Varianten, die auf eine oder mehrere Arten von Anwendungen, wie zum Beispiel Echtzeitsysteme, objektorientierten Anwendungen oder

[37] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S204

wissenschaftliche Software, spezialisiert sind. Es gibt dazu einige Studien, sehr viele davon findet man im universitären Umfeld.[38]

8.2.2.6 Objektorientierte Anwendungen sind messbar

Es existieren einige Ansätze, um Anwendungen, die mit dem OO-Paradigma entwickelt wurden, mittels Function-Points zu messen. Es gibt zum Beispiel Object-Points, eine Variante der Function-Points, aber auch Vorgehensmodelle, wie bei der Messung von objektorientierten Anwendungen vorzugehen ist[39], bestehen bereits.

8.2.2.7 Zusammenfassung – Function-Points Metrik

Die Function-Point Metrik hat sich im Rahmen der Aufwandsschätzung durchgesetzt und ist praktisch zum Standard geworden.

- \+ einheitliche Standards (auch ISO-zertifiziert)
- \+ Überprüfung jederzeit möglich
- \+ Anpassung an die Art der Anwendung möglich
- \+ objektorientierte Anwendungen messbar
- ± subjektiv
- − verschiedene Varianten

8.2.3 Traditionelle Metriken

Es gibt abgesehen von der LOC-Metrik und der Function-Point Metrik, die in 8.2.1 und 8.2.2 behandelt wurden, auch noch eine große Anzahl von anderen Metriken, die ich als „traditionelle“ Metriken bezeichnen möchte. Diese Metriken tauchen immer wieder im Kontext der Aufwandsschätzung auf.

Mit Halstead und McCabe möchte ich zwei davon herausheben, da sie immer wieder in der Literatur zu finden sind. Der Halstead Metrik wie auch der McCabe Metrik liegt der Sourcecode als Ausgangsgröße zugrunde. Ich meine, dass diese deshalb nicht für frühe Messungen, wie sie bei der Aufwandsschätzung auftreten, geeignet sind.

---

[38] z.B. 15.1.4 oder 15.1.2
[39] z.B. „Fallstudie III“ der IFPUG kann auf der Homepage bestellt werden

8.2.3.1 Halstead Metrik

Es wird hier durch eine Bewertung von Operatoren und Operanden gemessen. Als Operatoren werden Aktionen wie arithmetische Operatoren (+,-,* etc.), Schlüsselwörter (WHILE, FOR etc.), Zuweisungen und Klammern gesehen. Als Operanden werden Daten wie zum Beispiel Konstanten oder Variablen gesehen.

Diese Metrik wird stark kritisiert, unter anderem von Capers Jones[40], da bei dem Versuch, die veröffentlichten Zahlen nachzuvollziehen große Anomalien aufgetreten sind.

8.2.3.2 McCabe Metrik

Die McCabe Metrik ist ein interessanter Ansatz, bei dem versucht wird, die strukturelle Komplexität zu berücksichtigen, indem die Entscheidungen im Code gemessen werden. Hier ein grafisches Beispiel:

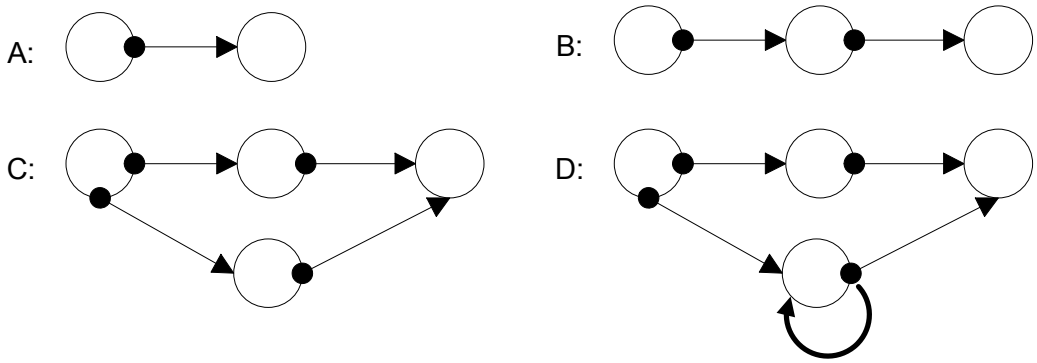

Abbildung 14: Beispiel für die Messung der Komplexität nach McCabe[41]

Bei diesem Beispiel stellen die Kreise (Schnittpunkte) die Codeabschnitte dar, die aufeinander folgen müssen, die Linien sind Kontrollwege die diese Abschnitte verbinden.

Nach folgender Formel kann der McCabsche Indikator, der auch zyklomatisches Maß oder zyklomatische Komplexität genannt wird, errechnet werden:

Anzahl der Verbindungen – Anzahl der Schnittpunkte +2 = zyklomatische Komplexität.

---

[40] Vgl. 15.2.1 Applied Software Measurement S106ff
[41] Beispiel Vgl. 15.2.11 - Was man nicht messen kann, kann man nicht kontrollieren S207

Für das Beispiel aus **Fehler! Verweisquelle konnte nicht gefunden werden.** ergibt sich somit folgendes Ergebnis:

Programm	Anzahl Verbindungen	Anzahl Schnittpunkte	McCabscher Indikator
A	1	2	1-2+2=1
B	2	3	2-3+2=1
C	4	4	4-4+2=2
D	5	4	5-4+2=3

Tabelle 9: Lösung Messung der Komplexität nach McCabe

In diesem Beispiel haben die Programme A und B die gleiche Komplexität, C ist komplexer als B und D komplexer als C. Ein McCabscher Indikator von beispielsweise 5 würde ein einfaches und leicht zu verstehendes Programm bedeuten, alles über 50 ist praktisch untestbar. Diese Metrik ist ebenfalls sehr umstritten, da die ursprüngliche Veröffentlichung fast ausschließlich empirischer Natur ist[42]. Diese Metrik wird oft im Zusammenhang mit Kennzahlen für die Qualität der Anwendung (Fehlerhäufigkeit etc.) verwendet.

8.2.4 Objektorientierte Metriken

Es existiert eine sehr große Anzahl von Metriken, Bundschuh spricht von mehr als 200 verschiedenen[43]. Eine Liste, großteils einer Studie[44] entnommen, ist im Anhang 17.2 „Übersicht über Metriken“ zu finden. Diese Metriken beziehen sich meistens auf Teile der objektorientierten Analyse, wie zum Beispiel Klassen, Methoden, Vererbungen. Sie werden aus diesem Grund auch in solche Kategorien unterteilt.

[42] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S213
[43] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S225
[44] Vgl. 15.1.5 Object-Oriented Metrics –A Survey

8.2.4.1 MOOSE

Dabei handelt es sich um eine Metriksuite für objektorientierte Sprachen, die sehr oft in der Literatur auftaucht. Diese Suite umfasst verschiedene Metriken und wurde zu Beginn der Neunziger von Chidamber und Kemerer verfasst. Bei der Suche nach Literatur zu OO-Metriken stößt man immer wieder auf diese Metriksuite.

8.2.4.2 Kritische Betrachtung

Ich habe den Eindruck gewonnen, dass ununterbrochen neue OO-Metriken veröffentlicht werden. Man kann es überzeichnet auch eine „Inflation“ von neuen OO-Metriken nennen. Leider gibt es wenige Studien und Untersuchungen zu den einzelnen Metriken, es ist dementsprechend nicht bekannt, ob diese neuen Metriken überhaupt etwas Sinnvolles messen.

Ich bin deshalb der Meinung, dass der Einsatz solcher Metriken nur unter genauester Analyse der gemessenen Ergebnisse erfolgen sollte.

9 Schätzverfahren im Überblick

Dieses Kapitel soll die wichtigsten Aufwandschätzverfahren zusammenfassen und einen Überblick über vorhandenen Schätzverfahren geben.

Es werden bewusst nur die gebräuchlichsten Aufwandschätzverfahren in komprimierter Form abgehandelt, da ansonsten der Umfang dieser Arbeit bei weitem gesprengt würde. Einen guten Überblick. über andere, heutzutage weniger gebräuchliche Schätzverfahren geben die Bücher „Applied Software Measurement“[45] und „Aufwandschätzung von DV-Projekten“[46].

Letztgenanntes enthält einen der wenigen Veröffentlichungen zum Vergleich verschiedener Schätzverfahren. Leider ist dieses Buch schon älter und vergleicht alte Versionen der Aufwandschätzverfahren miteinander. Eine weitere Quelle für solche Vergleiche ist ebenfalls etwas älter und stammt vom Kemerer, dem Autor der MOOSE Metriksuite (siehe auch 8.2.4.1) und lautet „An Empirical Validation of Software Cost Estimation Models“[47].

9.1 COCOMO

Das ursprüngliche Aufwandschätzverfahren COCOMO, das auch COCOMO81 genannt wird, besitzt heute nur mehr historischen Wert. Ich möchte aus diesem Grund hier schwerpunktmäßig die überarbeitete und heutzutage wichtigere Variante COCOMO II näher betrachten.

9.1.1 Modelle

Dieses Aufwandschätzverfahren besteht im Kern aus drei verschiedenen Modellen, nämlich

- Application-Composition-Model
- Early-Design-Model
- Post-Architecture-Model

[45] 15.2.1 Applied Software Measurement
[46] 15.2.2 Aufwandschätzung von DV-Projekten
[47] 15.1.1 An Empirical Validation of Software Cost Estimation Models

Das Application-Composition-Model ist bei der Veröffentlichung von COCOMO II noch als „emerging extension“ aufgeführt[48], ich zähle dieses Modell, das für sehr frühe Phasen der Entwicklung gedacht ist, deshalb nicht zum etablierten Standard und möchte es aus diesem Grund auch nicht näher analysieren. Das Early-Design-Model setzt später im Software-Entwicklungszyklus auf, kann jedoch noch immer als frühe Schätzung angesehen werden. Das Post-Architecture-Model wird für die Entwicklungsphase und die Wartungsphase eingesetzt.

9.1.2 Messung

Die Messung in COCOMO81 wurde in Lines of Code durchgeführt. Die Nachteile, die sich durch den Einsatz einer solchen Metrik ergeben, wurden ausführlich in 8.2.1 besprochen. Aus diesem Grund wird bei COCOMO II die funktionsbezogene Messung angewandt[49].

Modell	**Maßeinheit**
Application-Composition-Model	Object-Points
Early-Design-Model	Function-Points (ungewichtet)
Post-Architecture-Model	Function-Points (ungewichtet), Lines of Code

Tabelle 10: COCOMOII Kernmodelle

In dieser Tabelle wird die Maßeinheit für die Kernmodelle dargestellt. Lines of Code können ausnahmsweise im Post-Architecture-Model verwendet werden, da zu diesem Zeitpunkt die Entwicklung schon zu einem großen Teil fortgeschritten, oder sogar abgeschlossen ist.

Im Normalfall werden Umrechnungstabellen verwendet, um aus Function-Points Lines of Code zu ermitteln (siehe auch 8.2.1.3), da die Formeln, die bei der Schätzung verwendet werden, auf LOC ausgerichtet sind. COCOMO II verwendet per Definition ungewichtete Function-Points.

[48] Vgl. 15.2.8 Software Cost Estimation with COCOMO II S192
[49] Vgl. 15.2.8 Software Cost Estimation with COCOMO II S14ff

9.1.3 Schätzung

Die Schätzung wird je nach Modell mit einer großen Anzahl von Formeln, Graphen und Einflussfaktoren durchgeführt. Es besteht eine extrem hohe Anzahl an Einflussfaktoren und Parametern, die in Formeln eingesetzt werden. Diese Formeln wurden mittels Regressionsanalyse erstellt.

9.1.4 Kritische Betrachtung

Die nachfolgenden Betrachtungen fassen die Erfahrungen zusammen, die ich im Zuge meiner Recherchen für diese Arbeit erworben habe.

9.1.4.1 Function-Point - Lines of Code

Der für mich eklatanteste Nachteil bei COCOMOII ist die Umrechnung von Function-Points in Lines of Code. Man verwendet ein Verfahren, das die Funktionen misst und rechnet das Ergebnis dann in Lines of Code um, was alleine genommen schon zu einer erheblichen Unschärfe führen muss.

Eine große Überraschung war für mich die Erkenntnis, dass in der einzigen großen Veröffentlichung von COCOMO II[50], SPR-Backfiring-Tabellen verwendet werden (siehe auch 8.2.1.3). Diese basieren jedoch auf gewichteten Function-Points, und nicht wie bei COCOMO II vorgeschriebenen auf ungewichteten Function-Points. Somit werden, wenn man nach diesem Buch vorgeht, zusätzlich zu der ohnehin nicht unbeträchtlichen Unschärfe, die durch die Umrechnung entsteht, weitere +/-35% (maximale Anpassung bei der Umrechnung von ungewichteten in gewichtete Function-Points) dazu addiert. Ich habe mir daraufhin ein paar Spezialprogramme für COCOMO II angesehen, und bin zu der Erkenntnis gelangt, dass alle Tools als Umrechnungsfaktoren die gewichteten Function-Points der SPR-Backfiring-Tabellen verwenden. Getestet wurden von mir das Spezialtool Costar[51] und CostExpert. Bei Letzterem ist keine Homepage mehr verfügbar, es kann auch kein Kontakt zum Hersteller aufgenommen werden.

---

[50] Vgl. 15.2.8 Software Cost Estimation with COCOMO II S20
[51] 15.3.10 Softstarsystems

9.1.4.2 Komplexe Logik

Die Anwendung von COCOMO II ist sehr komplex. Durch die verschiedenen Modelle und der Möglichkeit, sehr viele Parameter zu verändern und zu adjustieren, ist eine Anwendung ohne unterstützende Software so gut wie nicht möglich. Positiv in diesem Hinblick ist die Tatsache, dass in dem schon öfters erwähnten Buch „Software Cost Estimation with COCOMO II“ ein Datenträger mit zwar sehr spartanisch gehaltenen, aber durchaus brauchbaren Programmen zur Verfügung gestellt wird.

Auf der anderen Seite erlauben diese komplexen Formeln aber eine Berechnung der für das Projektmanagement notwendigen Informationen, wie Aufwand, Durchlaufzeit, Teamgröße und ähnlichem.

9.1.5 Zusammenfassung

Der große und immer wieder kritisierte Nachteil von COCOMO81 war die LOC-Metrik. In COCOMO II wird jetzt zu Beginn mit Function-Points gerechnet, anschließend erfolgt eine Umrechnung in Lines of Code, die zusätzliche Unsicherheit in das Aufwandschätzverfahren bringt. Für die Umrechnung werden dazu noch Tabellen von externen Anbietern verwendet, die völlig aus dem Kontext des ursprünglichen Zwecks, nämlich der Methode „Backfiring“ (siehe auch 10.1 - Backfiring) gerissen sind.

Die hohe Komplexität des Verfahrens wirft für mich auch die Frage auf, ob bei Fehlschätzungen die Ursache überhaupt herausgefunden werden kann, da ein Fehler ja in einem der vielen oder sogar in mehreren der einzustellenden Parameter liegen kann.

Die mit der Literatur mitgelieferte Software und die Tatsache, dass auch Aufwand, Durchlaufzeit und Größe des Teams berechnet werden kann, sind eindeutig Vorteile dieses Verfahrens.

9.2 Expertenschätzung

Die Expertenschätzung ist das älteste und am häufigsten anzutreffende Aufwandschätzverfahren. Es gibt keine wirkliche Trennung der Messung und Schätzung, die Messung erfolgt meist implizit durch den Experten der die Schätzung vornimmt.

9.2.1 Ablauf

Bei der Expertenschätzung muss ein für dieses Schätzobjekt bestimmter Experte den Aufwand auf Grund seiner Erfahrung abschätzen.

9.2.2 Kritische Betrachtung

Der am meisten geäußerte Kritikpunkt ist, dass eine Expertenschätzung nicht nachvollziehbar ist. Ich meine aber, dass eine Expertenschätzung, die sauber durchgeführt wird auch nachvollziehbar sein kann. Dazu gehört natürlich eine saubere und vollständige Dokumentation des gesamten Vorganges (siehe auch 4.4 - Dokumentation der Aufwandsschätzung).

Fraglich ist, ob eine solche saubere und vollständige Dokumentation nicht mehr Aufwand als ein strukturiertes Aufwandschätzverfahren verursacht.

Sehr positiv herausgestrichen werden oft die exakten Ergebnisse, die mit Expertenschätzungen erzielt werden können, vorausgesetzt, dass es gleichartige Vergleichsprojekte gibt. Ich denke, dass dies aber nicht nur bei Expertenschätzungen der Fall ist. Bei jedem Aufwandschätzverfahren können bei gleichartigen Projekten gute und exakte Ergebnisse erzielt werden. Eine Herausforderung tritt erst bei neuartigen und anders gelagerten Projekten auf. Bei solchen Projekten halte ich persönlich die Expertenschätzung für sehr unzuverlässig, da nicht immer ein Experte für solche neuen Projekte verfügbar sein kann.

Im Allgemeinen ist es überhaupt sehr schwer zu beurteilen wer überhaupt ein Experte ist.

9.2.3 Zusammenfassung

Die Expertenschätzung wird sehr oft durchgeführt, und ist sehr oft nur ein Deckname für eine schlechte und nicht nachvollziehbare Aufwandsschätzung. Im Falle eine Expertenschätzung ordentlich und sauber durchgeführt wird ist diese durchaus eine Alternative für ein Unternehmen, das großteils mit ähnlichen Projekten arbeitet, oder immer in der gleichen Branche tätig ist.

9.3 Delphi-Methode

Die Delphi-Methode ist eine populäre Erweiterung der Expertenschätzung.

9.3.1 Ablauf

Bei der Delphi-Methode werden für den gleichen Teil der Schätzung mehrere Experten herangezogen, die eine eigenständige Schätzung durchführen. Anschließend werden die ermittelten Ergebnisse bekannt gegeben und die eigenen Ergebnisse überarbeitet. Diese Wiederholung passiert so oft bis eine Annäherung der Ergebnisse stattgefunden hat.

Es ist auch möglich, eine solche Befragung anonym durchzuführen oder die Experten über die Ergebnisse diskutieren zu lassen.

9.3.2 Kritische Betrachtung

Da die Delphi-Methode eigentlich eine Erweiterung der Expertenschätzung ist, gelten die gesamten Kritikpunkte aus 9.2.2.

Positiv herauszustreichen ist, dass die Schätzung nachvollziehbar und gut dokumentiert ist. Eine Delphi-Schätzung muss nämlich von einem jeden Experten so zu Papier gebracht werden, dass ein anderer Experte sie mit seiner eigenen abgleichen kann. Durch diesen Abgleich mit anderen Experten wird überdies auch noch das Risiko der Fehlschätzung (Ausreißer) vermindert.

9.3.3 Zusammenfassung

Ich bin der Meinung, dass die Delphi-Methode eine wesentliche Verbesserung der Expertenschätzung darstellt, und dieser auf jeden Fall vorzuziehen ist. Der Mehraufwand der dadurch entsteht sollte die bessere Ergebnisqualität bei weitem aufwiegen.

9.4 PERT - Dreizeitenmethode

Die Dreizeitenmethode wird auch „Pi mal Daumen“ oder Beta-Methode genannt, und ist Bestandteil der PERT-Technik und bedeutet „Program Evaluation and Review Technique“. Die PERT-Technik wurde ursprünglich für die Abwicklung des Polaris-Raketen Projektes der

US-Navy entwickelt. Mit dieser Technik werden PERT-Netzpläne erstellt. Innerhalb dieser PERT-Technik wird die Dreizeitenmethode verwendet um Schätzungen vorzunehmen. Beim ersten Einsatz dieser Technik konnte das Projekt 18 Monate vor dem Zeitplan abgeschlossen werden.

9.4.1 Ablauf

Für ein Arbeitspaket werden drei verschiedene Zeitpunkte geschätzt:

- Optimistische Dauer (OD)
- Häufigste Dauer (HD)
- Pessimistische Dauer (PD)

Anschließend wird die mittlere Dauer (MD) mit Hilfe der Formel MD = (OD + 4HD + PD) / 6 errechnet. Dieser Formel liegt eine Linearisierung der Betaverteilung zu Grunde.

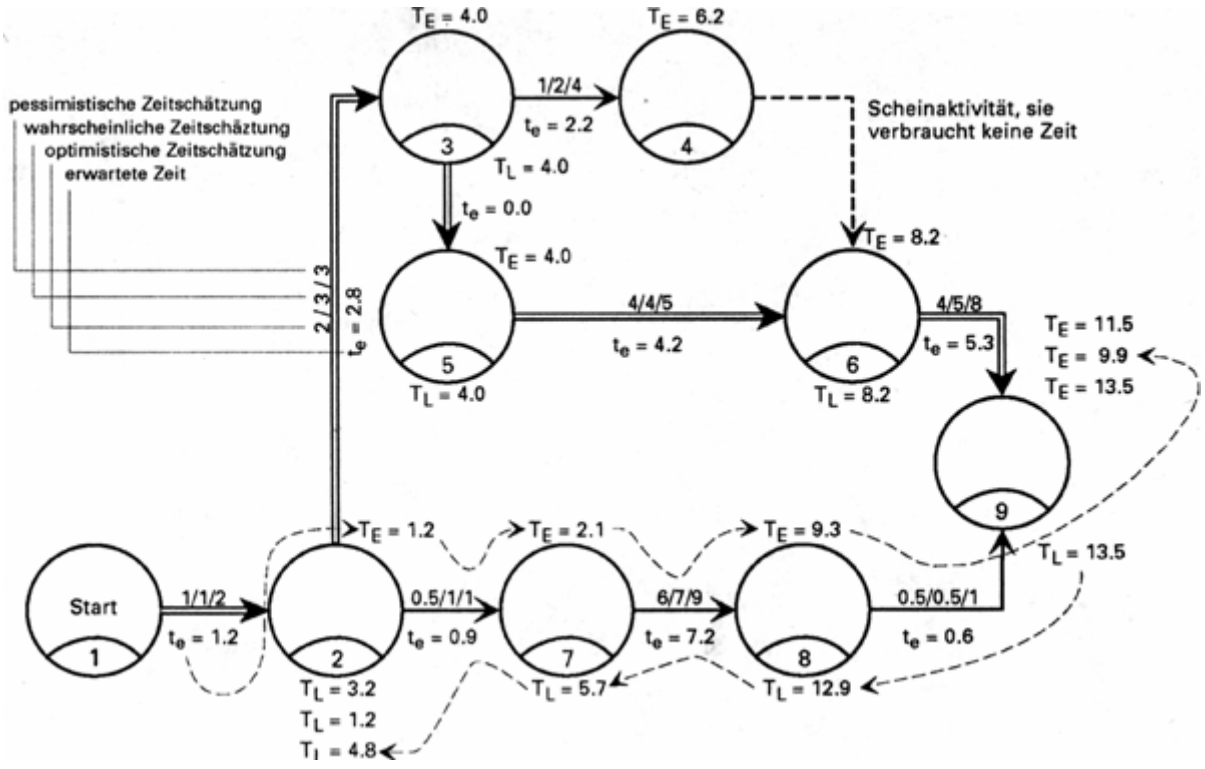

Abbildung 15: Lehrbeispiel PERT-Netzplan der Uni-Kassel[52]

[52] http://www.uni-kassel.de/~dehlwww/Grundlagen1/Downloads/Pert/Pert.html

Diese Abbildung zeigt einen klassischen PERT-Netzplan, schon eingetragen wurden die errechneten Werte („te“ = „time expected“). Für den Knoten 7 wurde zum Beispiel über (6+7*4+9)/6 der Wert 7,2 ermittelt.

9.4.2 Kritische Betrachtung

Wenn man Literatur über PERT liest, wird man rasch auf den „unglaublichen“ Erfolg dieser Technik hingewiesen, da ja das Parade-Projekt 18 Monate vor dem vorhergesagten Zeitpunkt fertig gestellt wurde. Hier folgt eine kritische Betrachtung dieser und anderer Umstände.

9.4.2.1 Ungenaue Schätzung

Das ungeheure Lob, das PERT genießt, da es 18 Monate vor dem eigentlichen Endtermin fertig gestellt wurde ist für mich nicht verständlich, da man von einem Erfolg oder Misserfolg des Projektes nur sehr bedingt auf die Güte eines Aufwandschätzverfahrens schließen kann. Um eine objektivere Beurteilung dieses Lobes durchzuführen, habe ich folgende Überschlagsrechnung erstellt, die auf leider nur sehr spärlichen Angaben über das Polaris-Projekt beruht.

Durchführungszeitraum	1957 – Juli 1960
Durchlaufzeit in Monaten	43
Geplante Durchlaufzeit in Monaten	43 + 18 = 61
Unterschreitung in Monaten	61 – 43 = 18
Unterschreitung gerundet auf ganze Prozent	30%

Tabelle 11: Überschlagsrechnung Projekterfolg PERT

Wie diese Überschlagsrechnung zeigt, weist das Projekt eine erhebliche Abweichung von 30% auf, die meinem Wissen nach nie näher untersucht wurde.

9.4.2.2 Überschätzung[53]

Ein weiterer erheblicher Kritikpunkt bezieht sich auf die verwendete Beta-Verteilung. Diese mit der Dreizeitenmethode erstellte Schätzung liefert tendenziell ein Ergebnis welches hinter der häufigsten Dauer liegt (linkssteile Verteilung), da der Abstand von PD zu HD meist größer geschätzt wird als der von OD zu HD. Sehr gut ersichtlich ist dieses Phänomen auch an der Abbildung 15. Bei diesem Beispiel ist dieser Abstand immer größer.

Die Überschätzung liegt meiner Meinung nach in der Vorsicht des Experten, der diese abgeben muss, begründet. Durch die Abgabe eines möglichst hohen pessimistischen Wertes kann das Risiko einer Fehlschätzung vermieden werden.

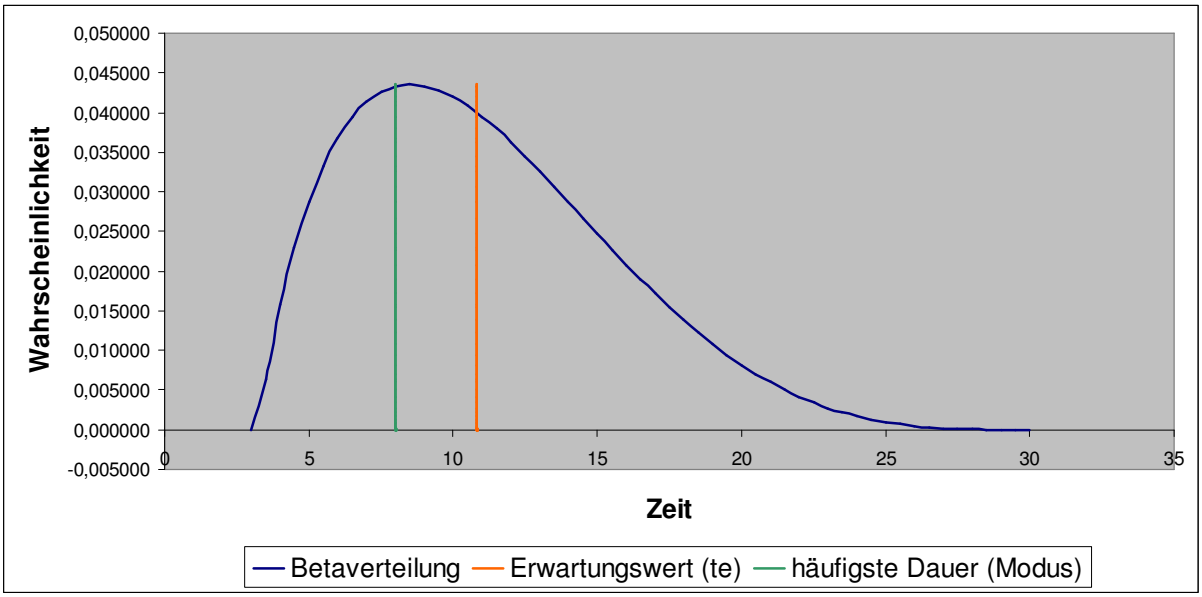

Abbildung 16: Beispiel für die Überschätzung mit PERT

Diese Abbildung zeigt, dass die häufigste Dauer vor dem errechneten Erwartungswert liegt und somit Überschätzungen auftreten.

[53] unveröffentlichte Untersuchungen von Wolfgang E. Katzenberger und Gerold Patzak

9.4.2.3 Unterschätzung

Bei der Berechnung des Ende-Zeitpunktes einer Aufwandsschätzung wird bei der PERT-Technik immer nur der Ende-Zeitpunkt des kritischen Pfades verwendet. Es bleibt dabei völlig unberücksichtigt, dass der kritische Pfad sich verschieben kann.

Eine solche Verschiebung tritt speziell bei Projektplänen mit Abhängigkeiten zu Sammelvorgängen auf.

9.4.3 Zusammenfassung

Bei der PERT-Technik tritt häufig eine Überschätzung auf, da die häufigste Dauer vor dem Erwartungswert liegt. Auch die fehlerhafte Vorhersage der Projektdauer, sobald sich der kritische Pfad verschiebt, lässt auf keine zuverlässigen Ergebnisse hoffen.

Für den Fall, dass die PERT-Methodik eingesetzt wird, empfehle ich eine andere Gewichtung der drei Zeiten und eine Simulation der wahrscheinlichen Projektdauer mittels Statistiksoftware.

9.4.4 Anmerkung

Eine Anekdote[54], die mir im Zuge meiner Recherchen vermittelt wurde besagt dass PERT nicht erfunden wurde, um ein Projekt zu managen sondern als Werkzeug, um die Aufmerksamkeit vom Projekt abzulenken, indem man den Bürokraten große Diagramme mit vielen Kästchen bot, die zeitaktuell gehalten werden mussten. Somit waren die Bürokraten mit anderen Dingen beschäftigt und die Ingenieure konnten in Ruhe arbeiten.

9.5 Function-Point-Verfahren

Function-Points als Maßeinheit und das Aufwandschätzverfahren selbst sind heute das am weitesten verbreitete strukturierte Aufwandschätzverfahren. Dieses Verfahren wurde 1979 von Albrecht veröffentlicht. Es gibt eine Reihe von Neuerungen und auch anderen Varianten die dieser Veröffentlichung folgten. Aufgrund des hohen Alters dieses Verfahrens gibt es für fast eine jede Anwendungsmöglichkeit Vorgehensmodelle oder Erfahrungsberichte. Ein

[54] Vgl. http://c2.com/cgi/wiki?PertChart

Beispiel dafür wären Aufwandsschätzungen im Data-Warehouse Bereich, bei objektorientierter Entwicklung oder bei Webanwendungen. Eine Besonderheit dieses Aufwandschätzverfahrens ist auch die Tatsache, dass die Zählung immer aus der Sicht des Anwenders durchgeführt wird.

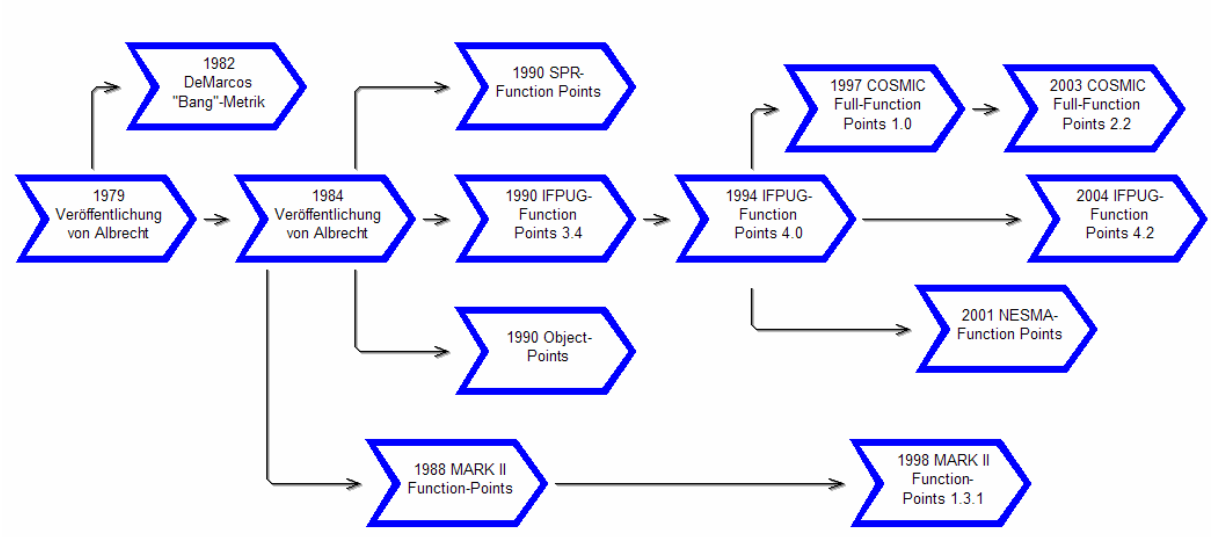

Abbildung 17: Historie und Function-Point-Varianten

Diese Abbildung zeigt die wichtigsten Entwicklungsschritte und Varianten der Function-Point Methode. Es folgt hier eine kurze Zusammenfassung der wichtigsten Varianten:

Variante	Beschreibung
Bang-Metrik	Verwendet andere Systemmerkmale zur Gewichtung und ist daher auch für wissenschaftliche Projekte (Algorithmen etc.) verwendbar.
COSMIC-Full Function-Points	Ermöglicht verschiedene Blickwinkel und berücksichtigt ein Software-Layer-Konzept
IFPUG Function-Points	Standard, der sich aus Albrechts Veröffentlichung entwickelt hat. Die am meisten verwendete Variante.
MARKII Function-Points	Eine hauptsächlich in Großbritannien verbreitete Variante mit einfacheren Regeln als die der IFPUG. Hat 5 zusätzliche Systemmerkmale.
NESMA-Function-Points	Eine Variante aus den Niederlanden, speziell für Weiterentwicklungsprojekte entwickelt
Object-Points	Ansatz für die objektorientierte Entwicklung (siehe auch 8.2.4 - Objektorientierte Metriken), sie ist nicht besonders bekannt und wird seit kurzem in COCOMOII verwendet
SPR Function-Points	Vereinfachung der IFPUG-FP 4.x, wurde für einen extrem frühen Projekteinsatz mit einem geringen Informationsstand entwickelt, daher ist sie ungenauer aber bestens für eine erste Schätzung geeignet

Tabelle 12: Überblick Function-Point Varianten

9.5.1 Ablauf

In diesem Unterkapitel wird der Ablauf nach der IFPUG 4.x-Variante dargestellt. Im Detail weichen manche Anderen davon ab.

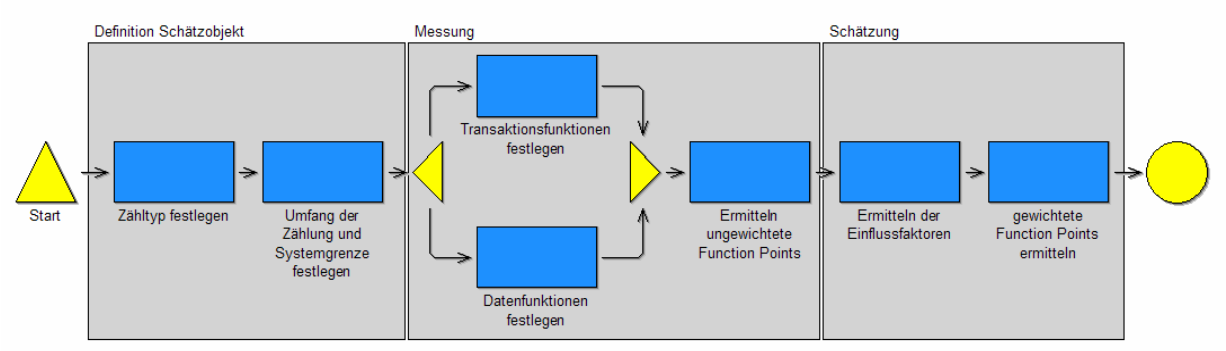

Abbildung 18: Ablauf Function-Point-Verfahren nach IFPUG

Wie aus dieser Abbildung ersichtlich, folgt dieser Ablauf dem schon in Kapitel 4 „Prozess Aufwandsschätzung“ erläuterten Prinzip „Definition – Messung – Schätzung“.

9.5.1.1 Zähltyp festlegen

Bei dieser Tätigkeit wird festgelegt ob es sich um

- eine Neuentwicklung
- eine Weiterentwicklung einer schon gezählten Anwendung
- ein „Anwendungssystem“

handelt.

Zu verstehen sind diese drei Zähltypen folgendermaßen:
Bei einem Neuentwicklungsprojekt ist noch kein Anwendungssystem erstellt, ein solches wird aber durch dieses Projekt initialisiert. Bei einem Weiterentwicklungsprojekt muss das Anwendungssystem (die Zählung des schon bestehenden Systems) aktualisiert werden, da Funktionen hinzugefügt oder weggefallen sein könnten.

9.5.1.2 Umfang und Systemgrenze festlegen

Hier wird versucht, aus Benutzersicht zwischen verschiedenen Anwendungssystemen zu unterscheiden, und eine Abgrenzung des Schätzobjektes zu anderen Systemen festzulegen.

9.5.1.3 Messung

Für die größenmäßige Bestimmung des Schätzobjektes werden Datenbestände und Transaktionen gezählt und deren Komplexität bewertet.

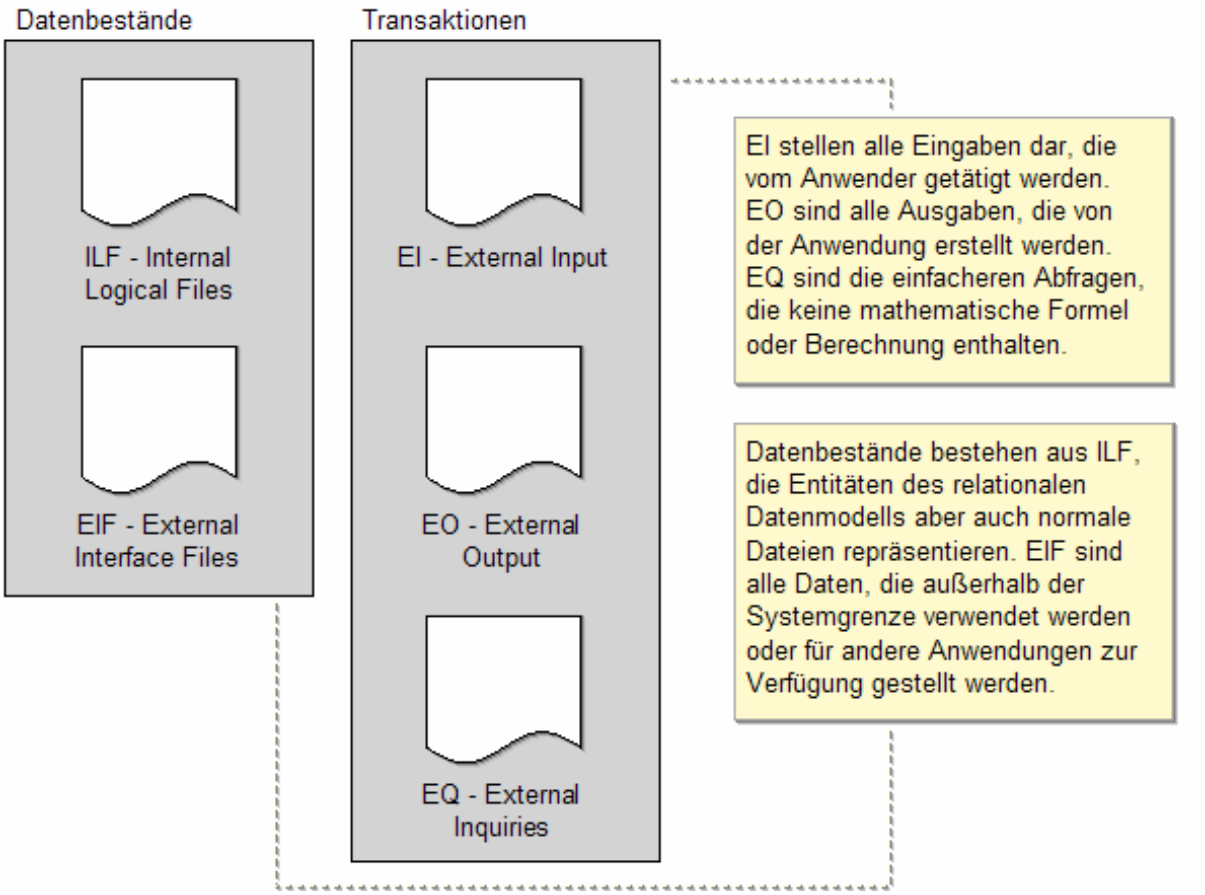

Abbildung 19: Messung erfolgt in Datenbeständen und Transaktionen

Diese Abbildung zeigt mit Hilfe welcher Elemente der Anwendung die Messung durchgeführt wird. Den Abschluss einer Messung bildet die Bestimmung des Komplexitätsgrades, dieser Elemente.

9.5.1.4 Schätzung

Bei der Schätzung werden 14 verschiedene Systemmerkmale von 0 bis 5 gewichtet und anschließend summiert. Diese Gewichtung und Summierung ist nicht zu verwechseln mit der Bestimmung der Komplexität bei der Messung!

Mit folgender Formel lässt sich dann der Faktor berechnen der benötigt wird, um ungewichtete Function-Points in gewichtete Function-Points umzurechnen:

Umrechnungsfaktor = (Summe der Gewichtungen *0,01) +0,65

Eine Umrechnung könnte folgendermaßen aussehen:

Ergebnis der Messung	12 ungewichtete Function-Points
Einflussfaktor	**Gewichtung 0-5**
Datenkommunikation	1
Verteilte Verarbeitung	0
Leistungsfähigkeit	0
Begrenzte Kapazität	0
Transaktionsrate	0
Interaktive Dateneingabe	0
Benutzerfreundlichkeit	0
Interaktive Änderung	1
Komplexe Verarbeitung	1
Wiederverwendbarkeit	5
Installationshilfen	1
Betriebshilfen	1
Mehrfachinstallation	1
Änderungsfreundlichkeit	0
Summe Gewicht Einflussfaktoren	11
Gewichtung nach Formel = 0,65+ (Summe*0,01)	0,76
Ergebnis ungerundet = 12 * 0,76	9,12
Gewichtete Function-Points	10

Tabelle 13: Beispiel Gewichtung Einflussfaktoren

Diese Tabelle zeigt ein einfaches Beispiel nach dem IFPUG4.1 Standard. Bei diesem Standard wird auf ganze gewichtete Function-Points aufgerundet.

9.5.2 Kritische Betrachtung

Diese Betrachtungen fassen die Eindrücke zusammen, die ich bei meinen Recherchen über Function-Points gewonnen habe.

9.5.2.1 Viele Erfahrungen und Verfahren

Das Schätzverfahren Function-Points ist eines der Ältesten und hat sich schon sehr bewährt. Seine Grundstruktur ist jedoch in allen Weiterentwicklungen bis heute gleich geblieben. Aus

diesem Grund gibt es viele Erfahrungswerte und Vergleichsmöglichkeiten für bestimmte Problemstellungen. Es gibt für sehr viele Anwendungsmöglichkeiten auch eigene Varianten, spezielle Verfahren oder Erfahrungsberichte.

9.5.2.2 Wenig frei verfügbare Informationen

Der Umgang mit Informationen rund um dieses Verfahren ist jedoch sehr restriktiv, was oftmals mit sensiblen Firmendaten zusammenhängt. Man könnte nämlich aus veröffentlichten Projekten oder Kennzahlen die Produktivität des Unternehmens ablesen. Aber auch der Umgang mit nichtsensiblen Informationen wie dem Counting-Practice-Manual (CPM), dem Handbuch der wichtigsten Organisation auf diesem Gebiet, der IFPUG[55], ist kostenpflichtig. Zertifizierungsprüfungen, Tutorien, Workshops sind ebenfalls oft mit nicht unerheblichem Aufwand oder Kosten verbunden. Das führt zu der Ansicht, dass die Erlernung dieses Verfahrens an sich schwierig ist.

9.5.2.3 Praktischer Einsatz

Eine sehr nebensächliche Rolle beim Function-Point-Verfahren bilden für die Praxis wichtige Fragen wie zum Beispiel:

- Welche Durchlaufzeit ist zu erwarten?
- Welcher Aufwand ist zu erwarten?
- Wie groß muss das Team sein?

Erfahrungsberichte sind auch sehr oft auf einer theoretischen Ebene verfasst, und mit einer unglaublichen Anzahl an Abkürzungen versehen. Jemand der sich nicht intensiv mit Function-Points auseinandersetzt, kann sehr oft nichts mit solchen Berichten anfangen.

9.5.3 Zusammenfassung

Dieses Verfahren ist sehr hoch entwickelt. Die Vielzahl von Vorgehensmodellen, Varianten und Erfahrungsberichten erlaubt den branchenübergreifenden Einsatz auch bei neuer

[55] 15.3.6 IFPUG

Technologie. Dieser durchaus positive Eindruck wird durch mangelnde Informationen über den praktischen Einsatz getrübt.

9.6 Prozentsatzmethode

Auch die Prozentsatzmethode selbst ist ein Aufwandschätzverfahren, dient aber auch oft als Hilfsmittel innerhalb eines anderen Aufwandschätzverfahrens.

9.6.1 Ablauf

Zuerst wird ein Aufteilungsschlüssel für jede Projektphase festgelegt. Hier ein Beispiel für einen solchen Schlüssel:

Projektphase	**Prozentsatz**
Planung	**33%**
Codierung	**17%**
Prüfen der Komponenten und erste Probeläufe des Systems	**25%**
Prüfen des Systems mit allen Komponenten	**25%**
Summe	100,00%

Tabelle 14: Beispiel Prozentsätze Prozentsatzmethode[56]

Die Wahl des Phasenmodells sollte sich nach den eigenen Bedürfnissen richten. Nun wird entweder eine Phase abgeschlossen, oder detailliert geschätzt. Mit diesem Ergebnis kann dann auf den Gesamtaufwand geschlossen werden. Für den Fall, dass eine detaillierte Schätzung gefordert ist, muss ein anderes Aufwandschätzverfahren, wie zum Beispiel Delphi oder die Expertenschätzung, zusätzlich verwendet werden.

[56] Entnommen aus 15.2.10 Vom Mythos des Mann-Monats S20

9.6.2 Kritische Betrachtung

Bei ähnlichem Projektumfeld und ähnlichen Projekten liefert diese Methode sehr genaue Ergebnisse. Der Aufwand ist minimal, da nur eine Phase geschätzt werden muss. Bei neuartigen Projekten mit wenig Erfahrung ist der Einsatz aber mehr als fragwürdig. Der Einsatz bleibt auch auf ein Phasenmodell beschränkt, Spiralmodelle oder iterative Entwicklungsansätze können mit diesem Verfahren nicht abgebildet werden.

9.6.3 Zusammenfassung

„Mit minimalem Einsatz optimale Ergebnisse erzielen“, könnte dieses Verfahren am besten beschreiben. Leider ist die Anwendung bei neuartigen Projekten oder innovativen Vorgehensmodellen nicht möglich.

10 Andere Techniken, Eckdaten, Begriffe

Dieses Kapitel dient als Sammlung sehr wichtiger Techniken, Eckdaten und Begriffe im Rahmen der Aufwandsschätzung, die unbedingt erwähnt werden sollten.

10.1 Backfiring

Diese Methode wurde von der Firma SPR[57] erfunden, und bietet im Rahmen der Aufwandsschätzung mit Function-Points die Möglichkeit, Altsysteme, deren Dokumentation zu dürftig ist, um eine Zählung durchführen zu können, zu erfassen. Dazu wurde ein Umrechnungsfaktor erstellt, mit dem Rückschlüsse von der Programmzeilenanzahl auf Function-Points gezogen werden können.

Programmiersprache	**level**	**Min**	**Modus**	**max**
Basic Assembly	1,0		320	
COBOL	3,0	65	106,7	150
FORTRAN	3,0	75	107	160
C	2,5	60	128	170

Tabelle 15: Auszug aus der SPR-Backfiring-Tabelle[58]

Diese Abbildung ist ein Auszug der SPR-Backfiring-Tabelle. Es sind die einzelnen Programmiersprachen ablesbar, dazu gibt die Spalte „level“ an, wie viele Assembler-Statements einem Statement dieser Programmiersprache entsprechen. Die Spalte „Modus“ zeigt, wie viele Statements notwendig sind, um einen gewichteten Function-Point mit dieser Programmiersprache zu erstellen. Die Spalten „min“ und „max“ liefern die Spannweite.

[57] 15.3.11 SPR
[58] Zahlen entnommen aus 15.2.1- Applied Software Measurement

10.1.1 Beispiel für die Anwendung

Ein C Programm hat keine verwertbare Dokumentation mehr. Nun wird die Anzahl der Codezeilen mit 5000 Lines of Code ermittelt. Anschließend erfolgt die Umrechnung mittels Tabelle (5000/128=39) was eine Function-Point Anzahl von durchschnittlich 39 gewichteten Function-Points ergibt. Da ein komplexer Code mehr Zeilen pro Function-Point aufweist, wurde folgender „code size adjustment multiplier“ eingeführt:

Summe von Problem-, Code und Datenkomplexität	Anpassungsfaktor
3	0,70
4	0,75
5	0,80
6	0,85
7	0,90
8	0,95
9	1,00
10	1,05
11	1,10
12	1,15
13	1,20
14	1,25
15	1,30

Tabelle 16: Anpassungsfaktor für die Backfiring Technik

Mit diesem „multiplier“ kann das Ergebnis weiter angepasst werden. Somit ergibt sich bei einer Komplexität von 10 ein Anpassungsfaktor von 1,05 und damit ein Wert von (39*1,05) 41 Function-Points.

10.1.2 Ungenauigkeit

Diese Technik ist sehr umstritten, da sie teilweise erhebliche Abweichungen aufweisen kann. Darauf deutet auch die „min“ und „max“ Spalte in der Umrechnungstabelle hin. Auch SPR selbst rät von einer Verwendung, wenn es irgendwie möglich ist, ab[59].

Da diese Technik aber manchmal die einzige Möglichkeit darstellt, um ein Altsystem messen zu können, muss man sich vor Augen führen, dass eine Schätzung, die eine größenordnungsmäßige Zuordnung eines Altsystems erlaubt, besser ist, als keine Zuordnung.

10.2 Benchmarking

Strukturierte Aufwandschätzverfahren und insbesondere Function-Points bieten eine hervorragende Möglichkeit, Software und Projekte zu messen. Damit wird es möglich Projekte, Teile der Projekte aber auch ganze Branchen mit den ermittelten Kennzahlen zu beurteilen und diese zu vergleichen. Solche Kennzahlen können entweder selbst ermittelt werden oder durch Zukauf einer Benchmarking-Datenbank erfolgen. Die bekanntesten Vertreter sind die ISBSG-Studie und die ESA/INSEAD Projekt-Datenbank.

10.2.1 Kennzahlen

Es gibt viele Kennzahlen die ermittelt werden können. Die wichtigsten möchte ich hier nennen.

10.2.1.1 Aufwand in den Projektphasen

Welcher Aufwand in den Projektphasen angefallen ist, ist für den Fall, dass man die Prozentsatzmethode als Aufwandschätzverfahren, oder als Teil eines anderen Aufwandschätzverfahren einsetzt (siehe auch 9.6), wichtig, da Angaben aus der Literatur sehr stark schwanken können.

---

[59] Vgl. http://www.spr.com/products/programming.shtm “we strongly discourage its use in virtually every conceivable circumstance”

10.2.1.2 Entwicklungsproduktivität

Die eigene Produktivität zu messen ist sicherlich der größte Wunsch eines Unternehmens. Dazu gibt es die Kennzahl PDR (Project Delivery Rate). Sie drückt aus, wie viele Stunden benötigt werden, um einen Function-Point zu realisieren, was bedeutet, je höher die PDR, desto niederer die Produktivität.

Damit können Aussagen getroffen werden wie zum Beispiel, dass die Produktivität von 1989 an bis zum Jahr 2002 kontinuierlich gesunken ist[60].

10.2.1.3 Weitere Kennzahlen

Auch die Betrachtung der Teamgröße kann sehr aufschlussreich sein. So kann zum Beispiel die Produktivität von großen und kleinen Teams verglichen werden. Daraus lässt sich wieder errechnen, wie groß ein Team maximal sein darf, um produktiv zu sein.

Aber auch die Frage, welche Programmiersprache oder Technik eingesetzt wird, kann mit der Produktivität verglichen werden.

10.2.2 Ziele und mögliche Probleme der Benchmarking-Methode

Nach der Literatur ist es das Ziel von Benchmarking, eine Standortbestimmung und gezielte Maßnahmen zur Optimierung der IT-Projekte und zur Weiterentwicklung des Projektmanagements zu finden[61]. Jedoch ist es schwierig bei mehreren hundert möglichen Einflussfaktoren korrekte Aussagen zu treffen. Sehr leicht kann es vorkommen, dass Faktoren, die gar nicht als Benchmarking-Daten erfasst wurden, großen Einfluss auf das Ergebnis haben.

Aber auch die Inhalte von Benchmarking-Datenbanken sind kritisch zu hinterfragen, da es sehr gut möglich ist, dass Unternehmen nur positiv abgeschlossene Projekte für diese Datenbanken einreichen. Sehr oft wird auch bei der Ist-Zeit-Erfassung manipuliert. Es werden

---

[60] Vgl. Eintrag Bundschuh ins Forum der 15.3.3 DASMA
[61] Vgl. 15.2.3 Aufwandschätzung von IT-Projekten S 271

zum Beispiel keine Überstunden oder Mehrarbeitsstunden in das Projekt gerechnet oder wenn die Zeit knapp wird, die Ist-Zeiten auf andere Projekte verbucht.

10.3 Schleichender Funktionszuwachs

Der schleichende Funktionszuwachs oder auch „Requirements Creep“ genannt, ist ein von Capers Jones geprägter Begriff. Er stellt die sich ändernden Anforderungen in einem Projekt oder neue Erkenntnisse über das Schätzobjekt während der Projektlaufzeit dar. Nach Messungen ist ein Funktionszuwachs von 3,5% der Projektlaufzeit pro Monat keine Seltenheit, und sollte auf jeden Fall in eine Aufwandsschätzung mit einfließen.

Es gibt verschiedene Möglichkeiten diesem Funktionszuwachs entgegenzuwirken. Einerseits ist eine saubere Definition von Anforderungen und die Definition von Inhalten und Nichtinhalten des Projektes eine gute Grundlage, um gestellten Zusatzanforderungen zu begegnen. In weiterer Folge hilft funktionierendes und gelebtes Claim Management, einen Überblick über Forderungen, die gestellt werden, zu behalten. Change Management kann helfen Änderungen aufzunehmen und Projektpläne in einer realistischen Weise darauf abzustimmen. Sowohl bei der genauen Definition des Schätzobjektes, beim Claim Management sowie beim Change Management spielt das Werkzeug Aufwandsschätzung eine wichtige Rolle.

10.4 Schätzkultur

Dieser Begriff wird oft im Zusammenhang mit Aufwandschätzverfahren genannt und beschreibt, wie bei einer Schätzung vorgegangen werden sollte und welche Maßnahmen zu unterlassen sind. Es werden dabei aber im Großen und Ganzen Maßnahmen und Vorgänge beschrieben, die genauso einem Projekt (Projektkultur), oder dem Unternehmen (Unternehmenskultur) zugerechnet werden könnten.

10.5 200x20x6

Eine für mich sehr einleuchtende Faustregel ist die 200x20x6-Regel. Sie besagt, dass, wenn 200% der Ressourcen eingesetzt werden, die Projektlaufzeit um maximal 20% verkürzt werden kann, die Fehler aber um das 6-fache ansteigen[62].

[62] 15.2.3 Aufwandschätzung von IT-Projekten S275

11 Conclusio

Im Verlauf dieser Arbeit habe ich einige für mich überraschende Erkenntnisse und Antworten gewonnen. Hier eine Zusammenfassung meiner Arbeit.

11.1 Messverfahren und Metriken

Es gibt in der Softwareentwicklung sehr viele Metriken, die immer wieder im Zusammenhang mit Aufwandschätzverfahren genannt werden, es eignen sich jedoch nur sehr wenige für die Aufwandsschätzung selbst, obwohl der Einsatz in anderen Bereichen durchaus möglich ist. Ich empfehle daher ein Messverfahren zu wählen, das die Funktion der Software misst.

Sehr wenig erforscht und belegt erscheinen mir Metriken für die objektorientierte Softwareentwicklung. Es gibt sehr viele Ansätze, doch kann bei einer solchen Flut an Metriken nie erforscht werden, ob diese etwas Sinnvolles messen.

Als am meisten etablierte Möglichkeit kann man bei objektorientierten Metriken die MOOSE Metrik-Suite betrachten.

11.2 Errechnung des Aufwandes

Die Messung als Vorgang in der Aufwandsschätzung mit anschließender Schätzung ist ein in der Literatur sehr ausführlich behandeltes Thema. Umso überraschender war die Erkenntnis, dass es sehr wenig Literatur gibt wie aus einer solchen Schätzung der Aufwand an sich errechnet und der Aufwand auf den Projektplan umgelegte werden kann. Ich habe aus diesem Grund dieses Thema ausführlich behandelt.

11.3 Welches Schätzverfahren soll gewählt werden

Es sind bis heute verschiedene Schätzverfahren im Gebrauch. Obwohl einige davon Mängel aufweisen ist allen gemein, dass bei ähnlichem Projektumfeld und konsequenter Anwendung ein erfolgreicher Einsatz möglich ist. Auf keinen Fall sollte auf eine gründliche und durchgehende Dokumentation der Aufwandsschätzung verzichtet werden.

Im Falle dass Projektart oder das Projektumfeld häufig wechseln, empfehle ich das Function-Point Verfahren, da es am ehesten eine objektive Beurteilung erlaubt.

Bei der Ermittlung des Aufwandes sowie der Erstellung des Projektplanes sollten die Einflussfaktoren mittels Spezialsoftware oder mittels zugekauften Benchmarking-Daten angepasst werden.

12 Abbildungsverzeichnis

Abbildung 1: Prozess Aufwandsschätzung ..... - 21 -

Abbildung 2: Definition des Schätzobjektes als erster Schritt zur Aufwandsschätzung..... - 22 -

Abbildung 3: Schätzgenauigkeit steigt mit dem Wissen über das Schätzobjekt..... - 24 -

Abbildung 4: Messen mittels Anwendung eines Messverfahren ..... - 25 -

Abbildung 5: Berechnen des Schätzergebnisses ..... - 26 -

Abbildung 6: Einsatz von Versionsverwaltungstools für die Dokumentation ..... - 30 -

Abbildung 7: IBM-Erfahrungskurve ..... - 33 -

Abbildung 8: Badwannenkurve von VW ..... - 34 -

Abbildung 9: Erstellung einer Erfahrungskurve - Eintragen der Wertepaare ..... - 35 -

Abbildung 10: Erstellung einer Erfahrungskurve - Ergebnis ..... - 36 -

Abbildung 11: Klassisches Projektmanagementdreieck..... - 43 -

Abbildung 12: Verwendung der Aufwandsschätzung im Projektmanagement ..... - 44 -

Abbildung 13: Wiederholung der Aufwandsschätzung..... - 48 -

Abbildung 14: Beispiel für die Messung der Komplexität nach McCabe..... - 64 -

Abbildung 15: Lehrbeispiel PERT-Netzplan der Uni-Kassel ..... - 73 -

Abbildung 16: Beispiel für die Überschätzung mit PERT ..... - 75 -

Abbildung 17: Historie und Function-Point-Varianten ..... - 77 -

Abbildung 18: Ablauf Function-Point-Verfahren nach IFPUG ..... - 79 -

Abbildung 19: Messung erfolgt in Datenbeständen und Transaktionen ..... - 80 -

Abbildung 20: Legende für die Darstellung der Prozesse ..... - 104 -

13 Tabellenverzeichnis

Tabelle 1: Chaos Report Kategorien ..... - 18 -

Tabelle 2: Chaos Report Ergebnisse ..... - 18 -

Tabelle 3: Untersuchung zum Einsatz methodengestützter Kalkulation ..... - 20 -

Tabelle 4: Rules of Thumb - Berechnung des Aufwandes ..... - 38 -

Tabelle 5: Ermitteln des „power level“ für die Berechnung ..... - 39 -

Tabelle 6: Beschreibung der "Software domain" ..... - 39 -

Tabelle 7: Beschreibung des „Erfahrungslevel“ ..... - 40 -

Tabelle 8: Mathematisches Paradoxon bei Lines of Code ..... - 56 -

Tabelle 9: Lösung Messung der Komplexität nach McCabe ..... - 65 -

Tabelle 10: COCOMOII Kernmodelle ..... - 68 -

Tabelle 11: Überschlagsrechnung Projekterfolg PERT ..... - 74 -

Tabelle 12: Überblick Function-Point Varianten ..... - 78 -

Tabelle 13: Beispiel Gewichtung Einflussfaktoren ..... - 81 -

Tabelle 14: Beispiel Prozentsätze Prozentsatzmethode ..... - 83 -

Tabelle 15: Auszug aus der SPR-Backfiring-Tabelle ..... - 85 -

Tabelle 16: Anpassungsfaktor für die Backfiring Technik ..... - 86 -

Tabelle 17: Abkürzungsverzeichnis ..... - 96 -

14 Abkürzungsverzeichnis

LOC	Lines of Code
SLOC	Source Lines of Code
KLOC	Kilo Lines of Code
FP	Function-Point
AFP	Adjusted Function-Point
UFP	Unadjusted Function-Point
NCSS	Non Commented Source Statements
PDR	Project Delivery Rate
PERT	Program Evaluation and Review Technique

Tabelle 17: Abkürzungsverzeichnis

15 Literaturverzeichnis

15.1 Studien & Papers

15.1.1 An Empirical Validation of Software Cost Estimation Models

Chris F. Kemerer, Mai 1987, Artikel der käuflich auf http://www.acm.org/ erworben werden kann

15.1.2 Comparative Evaluation of Functional Size Measurement Methods

Universiteit Gent –Comparative Evaluation of Functional Size Measurement Methods: An experimental analysis

http://www.feb.ugent.be/fac/research/WP/Papers/wp_04_234.pdf

15.1.3 Determining software schedules

Capers Jones, Februar 1995, Artikel der käuflich auf http://www.computer.org/ erworben werden kann

15.1.4 Functional Size Measurement for OO

Universiteit Gent – Functional Size Measurement for Object-Oriented Conceptual Schemas: Design and Evaluation Issues

http://www.feb.ugent.be/fac/research/WP/Papers/wp_04_233.pdf

15.1.5 Object-Oriented Metrics –A Survey

M.Xenox, D.Stavrinoudis, K.Zikouli and D.Christodoulakis – Preceedings of the FESMA2000, Federation of European Software Measurement Associations, Madrid, Spain 2000

http://www.swqlab.de/course/literature/Xenos_OOMetrics-ASurvey.pdf

15.1.6 Software estimating rules of thumb

Capers Jones, März 1996, Artikel der käuflich auf http://www.computer.org/ erworben werden kann

15.1.7 The Standish Group Chaos Report

The Standish Group, Eigenverlag der Standish-Group, verschiedene Ausführungen von 1994-2004, Kostenlage zwischen 2000 und 2500 USD – Stand November 2004 Link: http://www.standishgroup.com/

Kostenlose Downloads:

http://www.standishgroup.com/sample_research/chaos_1994_1.php

http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf

http://www.standishgroup.com/press/article.php?id=2

15.2 Bücher

15.2.1 Applied Software Measurement

Capers Jones, Second Edition 1995, McGraw-Hill, ISBN 0-07-032826-9

15.2.2 Aufwandschätzung von DV-Projekten

Thomas Noth, Mathias Kretzschmar, 2.Auflage, Springer-Verlag, ISBN 3-540-16069-8 und ISBN 0-387-16069-8

15.2.3 Aufwandschätzung von IT-Projekten

Bundschuh Manfred, Fabry Axel, 2.Auflage 2004, mitp-Verlag, ISBN 3-8266-0864

15.2.4 Das Software-Projekt

Friedrich Weltz, Rolf G. Ortmann, Campus Verlag, ISBN 3593347393

15.2.5 Der Termin

Tom DeMarco, Carl Hanser Verlag München Wien, ISBN 3-446-19432-0

15.2.6 Projektmanagement

Patzak Gerold, Günter Ratay, Linde Verlag Wien Ges.m.b.H, 3. Auflage 1998, ISBN 385122757-3

15.2.7 Rapid Development – Taming wild Software Schedules

Steve McConnell, Microsoft Press, 1996, ISBN 1-5561-900-5

15.2.8 Software Cost Estimation with COCOMO II

Barry W. Boehm, Chris Abts, A. Winsor Browsn, Sunita Chulani, Bradford K. Clark Ellis Horowitz, Ray Madachy, Donald Reifer, Prentice Hall PTR-Verlag, 2000, ISBN 0-13-026692-2

15.2.9 Spielräume

Tom DeMarco, Carl Hanser Verlag München Wien, ISBN 3-446-21665-0

15.2.10 Vom Mythos des Mann-Monats

Frederick P. Brooks jun., 1.Auflage 2003, mitp-Verlag, ISBN 3-8266-1355-4

15.2.11 Was man nicht messen kann, kann man nicht kontrollieren

Tom DeMarco, Juni 2004 , mitp-Verlag ISBN 3826614887

15.3 Webseiten:

15.3.1 Better-SCM-Initiative

Homepage die verschiedene Versionskontrollsysteme vergleicht.

http://better-scm.berlios.de/

15.3.2 Charismatek

Homepage der Firma Charismatek

http://www.charismatek.com

15.3.3 DASMA

Homepage der Deutschen Anwendergruppe für Software-Metrik und Aufwandsschätzung

http://www.dasma.org

15.3.4 David Consulting Group

Homepage der David Consulting Group

http://www.davidconsultinggroup.com

15.3.5 IEEE

Homepage des Institute of Electrical & Electronics Engineers

http://www.computer.org

15.3.6 IFPUG

Homepage der International Function Point User Group

http://www.ifpug.org/

15.3.7 ISBSG

Homepage der International Software Benchmarking Standards Group, die jährlich eine Benchmarking-Datenbank veröffentlicht

http://www.isbsg.org

15.3.8 ISO

Homepage der International Organization for Standardization

http://www.iso.org/

15.3.9 QSM

Homepage der Firma QSM

http://www.qsm.com

15.3.10 Softstarsystems

Homepage der Firma Softstarsystems die das COCOMO-Tool „Costar“ herstellt.

http://www.softstarsystems.com/

15.3.11 SPR

Homepage der Firma SPR (Software Productivity Research)

http://www.spr.com

15.3.12 Wikipedia

http://www.wikipedia.com/

http://www.wikipedia.at/

Wikipedia die freie Enzyklopädie

16 Danksagung

An dieser Stelle möchte ich mich bei jenen Personen bedanken, die mich beim Projekt „Diplomarbeit“ unterstützt haben.

Mein besonderer Dank gilt meinem Diplomarbeitsbetreuer Prof. Dr. Wolfgang Katzenberger. Herr Prof. Dr. Katzenberger hat bei mir als Lektor das Interesse an diesem Thema geweckt und ist mir als Diplomarbeitsbetreuer mit seinem Wissen, und seinem Erfahrungsreichtum stets zur Seite gestanden.

17 Anhang

17.1 Legende für die Darstellung der Prozesse

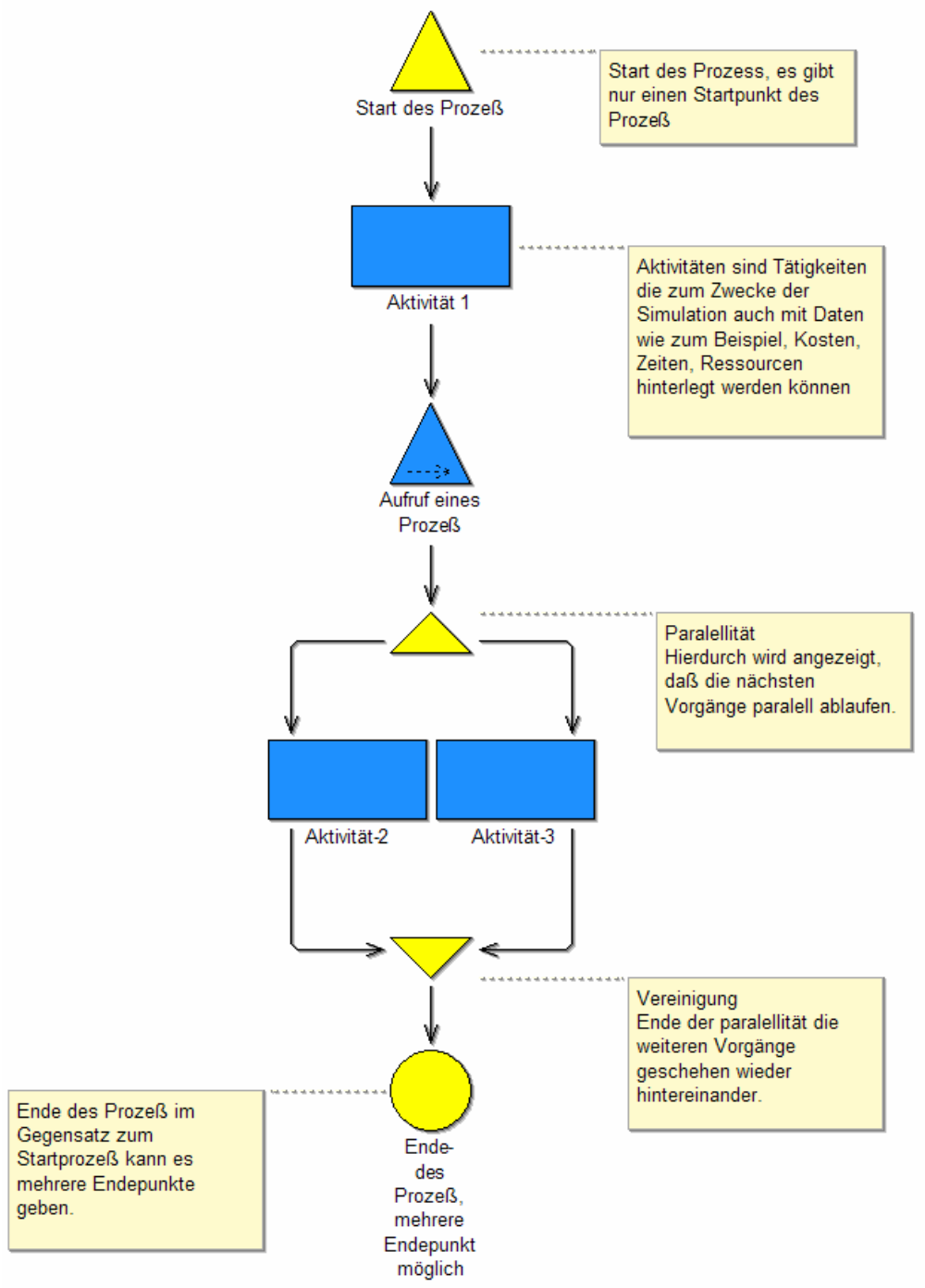

Abbildung 20: Legende für die Darstellung der Prozesse

17.2 Übersicht über Metriken

17.2.1 Traditionelle Metriken

Abkürzung	Beschreibung
AML	(Average Module Length) measures the average module size.
BAM	(Binding Among Modules) measures data sharing among modules.
CCN	(Cyclomatic Complexity Number) measures the number of decisions in the controlgraph.
CDF	(Control flow complexity and Data Flow complexity) is a combine metric based onvariab
COC	(Conditions and Operations Count) counts pairs of all conditions and loops within the
COP	(Complexity Pair) combines cyclomatic complexity with logic structure.
COR	(Coupling Relation) assigns a relation to every couple of modules according to the ki
CRM	(Cohesion Ratio Metrics) measure the number of modules having functional cohesiondivi
DEC	(Decision Count) offers a method to measure program complexity.
DSI	(Delivered Source Instructions) counts separate statements on the same physical linea
ERE	(Extent of Reuse) categorises a unit according the lever of reuse (modificationsrequi
ESM	(Equivalent Size Measure) measures the percentage of modifications on a reusedmodule.
EST	(Executable Statements) counts separate statements on the same physical line as distinct and ignores comment lines, data declarations and headings.
FCO	(Function Count) measures the number of functions and the source lines in every function.
FUP	(Function Points) measures the amount of functionality in a system.
GLM	(Global Modularity) describes global modularity in terms of several specific views of modularity.
IFL	(Information Flow) measures the total level of information flow between individual modules and the rest of a system
KNM	(Knot Measure) is the total number of crossing points on control flow lines.
LOC	(Lines Of Code) measures the size of a module.
LVA	(Live Variables) deals with the period each variable is used.
MNP	(Minimum Number of Paths) measures the minimum number of paths in a program and the reachability of any node.
MOR	(Morphology metrics) measure morphological characteristics of a module, such as size,depth, width and edge-to-node ratio.
NLE	(Nesting Levels) measures the complexity as depth of nesting.
SSC	(composite metric of Software Science and Cyclomatic complexity) combines software science metrics with McCabe's complexity measure.

SSM	(Software Science Metrics) are a set of composite size metrics.
SWM	(Specification Weight Metrics) measure the function primitives on a given data flow diagram.
TRI	(Tree Impurity) determines how far a graph deviates from being a tree.
TRU	(Transfer Usage) measures the logical structure of the program.

17.2.2 OO-Klassenmetriken

Abkürzung	Beschreibung
AHF	(Attribute Hiding Factor) is the ratio of the sum of inherited attributes in all system classes under consideration to the total number of available classes attributes.
CCO	(Class Cohesion) measures relations between classes.
CEC	(Class Entropy Complexity) measures the complexity of classes based on their information content.
CLM	(Comment Lines per Method) measures the percentage of comments in methods.
CRO	(Coupling between Object Classes) number of classes which a class is using
DAM	(Data Access Metric) is the ratio of the number of private attributes to the total numberof attributes declared in the class.
FOC	(Function Oriented Code) measures the percentage of non object–oriented code that is used in a program.
INP	(Internal Privacy) refers to the use of accessory functions even within a class.
LCM	(Lack of Cohesion between Methods) indicates the level of cohesion between the methods.
MAA	(Measure of Attribute Abstraction) is the ratio of the number of attributes inherited by aclass to the total number of attributes in the class.
MFA	(Measure of Functional Abstraction) is the ratio of the number of methods inherited by aclass to the total number of methods accessible by members in the class.
MHF	(Method Hiding Factor) is defined as the ratio of the sum of the invisibilities of allmethods defined in all classes to the total number of methods defined in the systemunder consideration.
NAD	(Number of Abstract Data types) is the number of user-defined objects used asattributes in a class that are necessary to instantiate an object instance of the class.
NCM	(Number of Class Methods in a class) measures the measures available in a class butnot in its instances.
NIV	(Number of Instance Variables in a class) measures relations of a class with otherobjects of the program.
NOA	(Number Of Ancestors) is the total number of ancestors of a class.
NPA	(Number of Public Attributes) counts the number of attributes declared as public in aclass.
NPM	(Number of Parameters per Method) is the average number of parameters per methodin a class.
NRA	(Number of Reference Attributes) counts the number of pointers and references usedas attributes in a class.
PCM	(Percentage of Commented Methods) is the percentage of commented methods.

PDA	(Public Data) counts the accesses of public and protected data of a class.
PMR	(Percent of Potential Method uses actually Reused) is the percentage of the actualmethod uses.
PPD	(Percentage of Public Data) is the percentage of the public data of a class.
RFC	(Response For a Class) is the number of methods in the set of all methods that can beinvoked in response to a message sent to an object of a class.
WCS	(Weighted Class Size) is the number of ancestors plus the total class method size.
WMC	(Weighted Methods per Class) is the sum of the weights of all the class methods.

17.2.3 OO-Methodenmetriken

Abkürzung	Beschreibung
AMC	(Average Method Complexity) is the sum of the cyclomatic complexity of all methodsdivided by the total number of methods.
AMS	(Average Method Size) measures the average size of program methods.
MAG	(MAX V(G)) is the maximum cyclomatic complexity of the methods of one class.
MCX	(Method Complexity) relates complexity with the number of messages.
RFC	(Response for a class) number of functions called directly through methods
WMC	(Weighted Methods per Class) number of defined methods per class

17.2.4 OO-Verbindungsmetriken

Abkürzung	Beschreibung
CBO	(Coupling Between Objects) counts the number of classes a class is coupled with.
CCP	(Class Coupling) measures connections between classes based on the messages they exchange.
CFA	(Coupling Factor) is the ratio of the maximum possible number of couplings in the system to the actual number of couplings not imputable to inheritance.

17.2.5 OO-Vererbungsmetriken

Abkürzung	Beschreibung
AIF	(Attribute Inheritance Factor) is the ratio of the sum of inherited attributes in all classesof the system under consideration to the total number of available attributes for all classes.
DIT	(Depth of Inheritance Tree) measures the number of ancestors of a class.
FEF	(Factoring Effectiveness) is the number of unique methods divided by the total number of methods.
FIN	(FAN-IN) is the number of classes from which a class is derived and high valuesindicate excessive use of multiple inheritance.
HNL	(Class Hierarchy Nesting Level) measures the depth in hierarchy that every class islocated.
MIF	(Method Inheritance Factor) is the ratio of the sum of the inherited methods in allclasses to the total number of available methods for all classes.
MRE	(Method Reuse metrics) indicate the level of methods reuse.
NMI	(Number of Methods Inherited) measures the number of methods a class inherits.
NMO	(Number of Methods Overridden) is the number of methods need to be re-declared bythe inheriting class.
NOC	(Number Of Children) is the total number of children of a class.
PFA	(Polymorphism Factor) is the ratio of the actual number of possible differentpolymorphic situations of a class to the maximum number of possible distinctpolymorphic situations for this class.
PMO	(Percent of Potential Method uses Overridden) is the percentage of the overriddenmethods.
RDB	(Ratio between Depth and Breadth) is the ratio between the depth and the width of thehierarchy of the classes.
RER	(Reuse Ratio) is the ratio of the number of superclasses divided by the total number ofclasses.
SIX	(Specialisation Index) measures the type of specialisation.
SPR	(Specialisation Ratio) is the ratio of the number of subclasses divided by the number ofsuperclasses.

17.2.6 OO-Systemmetriken

Abkürzung	Beschreibung
ADI	(Average Depth of Inheritance) is computed by dividing the sum of nesting levels of allclasses by the number of classes.
ANA	(Average Number of Ancestors) determines the average number of ancestors of all theclasses.
APG	(Application Granularity) is the total number of objects divided by the total number offunction points.
ASC	(Association Complexity) measures the complexity of the association structure of asystem.
CAN	(Category Naming) divides classes into semantically meaningful sets.
CRE	(Number of time a Class is Reused) measures the references in a class and the numberof the applications that reuse this class.
FDE	(Functional Density) is the ratio of LOC to the function points.
NCT	(Number of Classes Thrown away) measures the number of times a class is rejecteduntil it is finally accepted.
NOH	(Number Of Hierarchies) is the number of distinct hierarchies of the system.
OLE	(Object Library Effectiveness) is the ratio of the total number of object reuses divided bythe total number of library objects.
PRC	(Problem Reports per Class) measures defect reports on this class.
PRO	(Percent of Reused Objects Modified) declares the percentage of the reused objectsthat have been modified.
SRE	(System Reuse) declares the percentage of the reuse of classes.