Studienarbeit

Methoden zur Aufwands-abschätzung bei Softwareprojekten im Vergleich

Andreas Seibt
Duvenacker 15

22523 Hamburg
eMail: A.Seibt@a-s-i.de

November 1999

Betreuung: Dr. Ralf Klischewski

Fachbereich Informatik
Arbeitsbereich Softwaretechnik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg

INHALT

Tabellen- und Abbildungsverzeichnis

ABBILDUNGEN:

TABELLEN:

1 Einleitung

1.1 Motivation

Jeder, der schon einmal an einem Software-Projekt teilgenommen hat, wird gefragt worden sein, wie lange er für eine bestimmte Sache brauchen würde.
Diese Frage zielt entweder auf den zeitlichen Rahmen, bis wann die erteilte Aufgabe erledigt sein wird, oder aber - und das ist bei kommerziellen Projekten meist vorrangig - auf den damit verbundenen finanziellen Aufwand.
Bei größeren Projekten kann es im Verlauf des Projektes zu Änderungen der Anforderungen bzw. der Randbedingungen kommen, welche die bisherige Abschätzung ungültig machen. Daher ist es in diesem Falle erforderlich, eine neue Abschätzung durchzuführen, um das Projektmanagement den neuen Gegebenheiten anzupassen.
Darüber hinaus können Aufwandsabschätzungen dazu dienen, zu überprüfen, ob sich ein Projekt noch auf dem richtigen Kurs befindet.

Menschen mit wenig Erfahrung werden die Frage nach dem zu erwartenden Aufwand nach Gutdünken irgendwie beantworten. Erfahrenere Personen vergleichen die aktuell gestellt Aufgabe mit bisherigen von ihnen gelösten Problemen und werden den Zeitrahmen ähnlich dem eines älteren Projekts wählen.
Aufgrund dieser Unsicherheiten wurde versucht, die Aufwandsabschätzungen durch Vorgehenssystematiken zu verbessern.

Da das Zusammenspiel von Menschen untereinander und mit der Technik bei Software-Projekten derart komplex ist, da es eine unüberschaubare Zahl von Faktoren gibt, die bei einer exakten Schätzung eine Rolle spielen würden, muß man aus diesen vielen Faktoren eine Auswahl treffen, die die Situation ausreichend beschreiben und somit die für den jeweils betrachteten Kontext akzeptable Abschätzung ermöglichen. Dabei muß ein Kompromiß zwischen Anzahl und Relevanz gefunden werden, damit die Methode noch anwendbar bleibt.

Ziel dieser Studienarbeit ist es, einen Überblick über die in der Literatur vorhandenen Methoden zur Aufwandsabschätzung bei Softwareprojekten zu schaffen und diese anhand von erarbeiteten Kriterien mit Erfahrungswerten, also relevanten Daten bereits durchgeführter Projekte zu überprüfen.

Die zentrale Frage dieser Arbeit lautet: Wie leistungsfähig sind diese Methoden bezüglich

- Schätzgenauigkeit
- unbedingte Voraussetzungen
- Nachvollziehbarkeit
- Berücksichtigung von Qualitätsmerkmalen

1.2 Vorgehen

Zuerst soll ein Überblick vermittelt werden, welche Methoden in der Literatur zu finden sind und welche Vorkenntnisse diese voraussetzen.
Bei meinen Recherchen in der Fachbereichsbibliothek Informatik und anderen Bibliotheken der Universität Hamburg, sowie Internet-Recherchen (allgemein und bei Online-Buchläden) habe ich leider nur wenige Publikationen zu diesem Thema gefunden (siehe Anhang D).
Die in diesen Büchern angegebenen Methoden werden zwar noch in ein oder zwei anderen Büchern zitiert, aber da sich hieraus keine neuen Erkenntnisse gewinnen ließen, wurden sie nicht weiter berücksichtigt.
Einige Teile der Primärliteratur, z.B. zur Function-Point-Methode (von IBM), zur Data-Point-Methode (von Sneed) und zur Object-Point-Methode (von der Software AG) war, trotz Nachfrage beim Urheber, mir nicht zugänglich.

Aus all diesen Gründen mußte ich mich bei meiner Arbeit auf die von mir verwendete Literatur beschränken. Deshalb sind viele der im Anhang vorgestellten Methoden stark veraltet.
Meine eigene Erfahrung scheint aber das Phänomen zu bestätigen, daß das Anwenden von systematischen Schätzverfahren eher die Ausnahme, denn die Regel in der Praxis darstellt.
In [Weltz92] wurde eine Untersuchung durchgeführt, die zeigen sollte, wie hoch der Anteil des Methodeneinsatzes bei der Aufwandsabschätzung in der Praxis ist. Folgendes Ergebnis wurde erzielt:

Methodeneinsatz bei der Kalkulation

	AU	*SwH*	*Gesamt*
Kalkulation erfolgte			
- methodengestützt	8 %	14 %	11 %
- sowohl methodengestützt als auch erfahrungsbezogen	8 %	24 %	15 %
- ganzheitlich, erfahrungsbezogen	60 %	52 %	57 %
Kalkulation unterblieb	20 %	5 %	13 %
Keine Angabe	4 %	5 %	4 %

AU: Anwenderunternehmen *SwH*: Softwarehaus

Tabelle 1-1 : Methodeneinsatz bei der Kalkulation

Dieses Ergebnis, daß im Durchschnitt nur 26% aller Projekte mit Methodenuntersützung kalkuliert wurden, verstärkt den Eindruck, daß entweder die bekannten Methoden zu aufwendig und schwer anwendbar sind, oder aber, daß dieses Thema bis jetzt eher stiefmütterlich behandelt worden ist.

Um diesem Phänomen auf den Grund zu gehen, habe ich die von mir recherchierten Methoden zur Aufwandsabschätzung verglichen. Hierzu habe ich einige Projektdaten zusammengetragen, die in einer ex-post Analyse als Datenmaterial dienen sollen.
Obwohl ich seit 1996 an verschiedenen Software-Projekten in wechselndem Umfang beteiligt war, konnte ich nur diejenigen Projekte verwenden, die annähernd komplett den Entwicklungszyklus (von der Analyse bis hin zur Realisierung) durchlaufen haben.

Kern dieser Arbeit ist der Vergleich einiger Methoden zur Aufwandsabschätzung. Dabei sollen die vier im vorherigen Abschnitt genannten Bewertungskriterien untersucht werden.
Bei meinen Recherchen habe ich in einer Publikation ([Noth84]) ebenfalls einen Vergleich finden können. Die dort zugrunde gelegten Bewertungskriterien sind vielfältig und interessant, lassen sich aber auf die vier von mir genannten ‘vergröbern’.
Dennoch hat diese Arbeit ihre Berechtigung, da der in [Noth84] vorgenommene Vergleich an sich für den Leser mangels Daten nicht nachvollziehbar ist; es werden meist nur die Ergebnisse präsentiert. Ferner hat sich die Softwareentwicklung in der Zeit seit 1984 dermaßen verändert, daß sowohl der damals angelegte (subjektive) Maßstab, als auch die berücksichtigten Projektdaten heute nur noch bedingt gültig sind.
Trotzdem dürfte es interessant sein, zu welchen Ergebnissen die Autoren von [Noth84] gekommen sind und zu welchen ich gekommen bin.

1.3 Überblick

Ausgangspunkt für Aufwandsabschätzungen sind Modelle zur Projektstrukturierung. Daher werden diese im ersten Abschnitt dieser Arbeit (Kapitel 2) kurz beleuchtet. Es wird gezeigt, welche Modelle es gibt und welchen Stellenwert sie in der heutigen Softwareentwicklung haben.
In Kapitel 2.2 wird der Themenbereich der Softwarequalität soweit aufgezeigt, wie es im Rahmen dieser Arbeit notwendig ist. Es soll geklärt werden, was Softwarequalität bedeutet und warum man diese sichern muß. Zu klären, welche Qualitätsmerkmale es gibt bzw. geben sollte und wie man diese messen kann, würde den Rahmen dieser Arbeit sprengen. Anhand der Zahl der Veröffentlichungen zu diesem Thema kann man erkennen, daß dieses Thema kontrovers diskutiert wird.
Um die Vergleiche zwischen den Methoden verstehen zu können, wird in Kapitel 2.3 gezeigt, welche Aufwandsbegriffe es gibt und wie diese zueinander stehen. Neben der einfachen Umrechnung der Ergebnisse wird auch versucht zu zeigen, warum es so schwierig ist, den Aufwand und die zeitliche Dauer zu unterscheiden.
Ferner wird, aufbauend auf diesen Grundlagen, ermittelt, welche Ansprüche man an Methoden zur Aufwandsabschätzung hat. Resultat davon sind vier Kriterien, die meiner Ansicht nach die wichtigsten Anforderungen darstellen.

Im zweiten Abschnitt (Kapitel 3) werden zuerst einige Methoden und Methodengruppen vorgestellt; eine komplette Auflistung aller im Rahmen dieser Arbeit recherchierten Methoden ist in Anhang B zu finden.
In Kapitel 3.3, der eigentlichen Bewertung, werden zuerst die Rahmenbedingungen dieses Vergleiches vorgestellt. Anschließend werden die drei von mir ausgewählten Methoden anhand der Projektdaten einzeln untersucht und die dadurch gewonnen Erkenntnisse aufgeführt. Kapitel 3.3.5 faßt dann noch einmal alle Ergebnisse dieser Bewertung zusammen.

Der dritte Abschnitt dieser Arbeit (Kapitel 4) stellt die Zusammenfassung dieser Arbeit dar und wirft noch einmal einen Blick auf die Zielsetzung und die gemachten Ergebnisse.

Im Anhang zu finden ist:
Eine Sammlung von Projektdaten (Anhang A), eine Auflistung aller im Rahmen dieser Arbeit recherchierten Methoden zur Aufwandsabschätzung (Anhang B) und die Anwendung von drei Methoden an den Daten dreier Projekte (Anhang C). Abschuß bildet Anhang D mit dem Literaturverzeichnis.

2 Grundlagen der Software-Projektsteuerung

In diesem Kapitel wird der praxisnahe Hintergrund beleuchtet, welche Vorgehensweisen beim Strukturieren und Steuern eines Projektes empfehlenswert sind. Hierbei wird auf drei verschiedene Modelle eingegangen.
Hieraus wird abgeleitet, daß Abschätzungen durchgeführt werden, um den aktuellen Stand des Projektes mit dem zu erwartenden Aufwand vergleichen und steuern zu können.
Anschließend wird untersucht, welchen Einfluß Software-Qualität und die S.-Qualitätssicherung auf den Aufwand haben. Es soll gezeigt werden, was unter Softwarequalität verstanden werden kann und warum eine Qualitätssicherung bei einem Softwareprojekt notwendig ist.
Ferner wird die Bedeutung des Abschätzergebnisses und der Zusammenhang zum zeitlichen Ablauf betrachtet, Begriffe werden geklärt und die Problematik dargestellt.
Zum Schluß werden aus diesen Erkenntnissen Kriterien für die spätere Bewertung der ausgewählten Methoden zur Aufwandsabschätzung abgeleitet.

2.1 Modelle zur Projektstrukturierung

Jeder, der strukturiert an eine Aufgabe herantritt, wird sich ab einer gewissen Größe des Problems kleinere Zwischenschritte überlegen, um diese dann Schritt für Schritt zu erledigen. Der nächste Schritt wird erst getan, wenn der vorherige erfolgreich abgeschlossen wurde.
Dieses ist im täglichen Leben (z.B. Hausbau, Organisation des Studiums) genauso üblich, wie bei der Software-Entwicklung.
In der Software-Entwicklung werden diese Zwischenschritte meist Meilensteine genannt. Aufgrund dessen werden folgende Anforderungen an Meilensteine gestellt:

- jeder Meilenstein muß ein klar definiertes Ziel haben
- das Erreichen dieses Zieles muß eindeutig meßbar sein
- erst, wenn das Ziel erreicht ist, wird zum nächsten Meilenstein übergegangen

Da Software-Projekte selten von nur einer oder zwei Personen durchgeführt werden, kann es sein, daß mehrere Teams mehrere Meilensteine zeitgleich bearbeiten. Daher muß nicht der nächste Meilenstein der sein, der als nächstes in einer seriellen Reihenfolge folgt, sondern der nächste, dessen Bearbeitung vom Fertigstellen des eben Erreichten abhängt.

Hieraus geht auch hervor, daß ein erfolgreiches Projektmanagement nicht allein von der Aufteilung und der Koordination von Meilensteinen abhängt, sondern daß zum einen die benötigten Ressourcen geplant werden müssen und, im Rahmen dieser Arbeit viel wichtiger, daß es zum anderen eine den Meilensteinen übergeordnete Struktur geben muß, die eine Gruppe von etwa gleichwertigen Meilensteinen zu Phasen zusammenfaßt.
Analog zu den Meilensteinen müssen die Phasen ebenfalls klar definierte und eindeutig überprüfbare Ziele besitzen, die die weitere Ausführung steuern.

Aus dieser Art des Vorgehens wurde das Phasenmodell bzw. Wasserfallmodell (dann in Treppenform dargestellt) entwickelt:

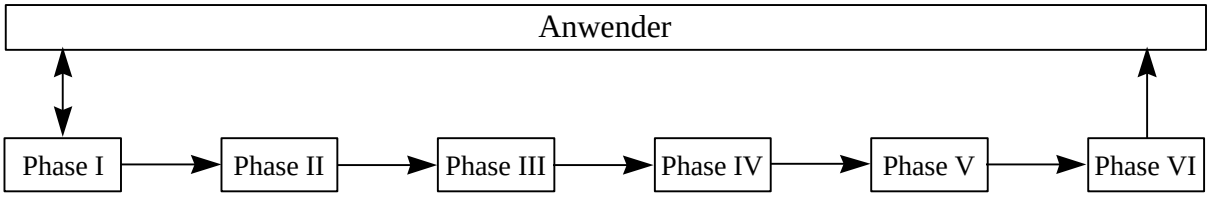

Abbildung 2-1 : Schaubild klassisches Phasenmodell

Der Anwender beauftragt die Softwareentwicklung, die dann mit Phase I beginnt. Die Entwicklung wird nun Schritt für Schritt der Reihe nach (ohne Rücksprung über die Phasengrenzen hinweg) durchgeführt und in der Phase VI wird das fertige Produkt dem Anwender übergeben.

Mit einer solchen Einteilung läßt sich jederzeit der aktuelle Stand des Projektes mit dem geplanten vergleichen (Soll-/Ist-Vergleich), um ggf. Probleme frühzeitig erkennen und lösen zu können.
Dieses zeigt auch die Notwendigkeit von Abschätzungen, denn womit sollte man den aktuellen Stand vergleichen, wenn man keine Aussagen darüber treffen kann, wie hoch der Aufwand zum Erreichen des Zieles ist? Dabei hängt es, wie später noch gezeigt wird, sehr stark von der Granularität der Methoden ab, ob der Aufwand nur für das komplette Projekt, für einzelne Phasen oder aber auch für einzelne Meilensteine abgeschätzt werden kann.
Aufbauend hierauf wird dann u.a. das Projektbudget auf die einzelnen Phasen verteilt. Daß es bei dieser Aufteilung nicht bleibt, zeigt die Praxis. Daher ist es notwendig, auch während des Projektes, z.B. nach Änderungen an den Zielsetzungen, erneute Aufwandsabschätzungen durchzuführen.
Da die Einteilung des Projektes in Meilensteine und in Phasen sehr stark vom Projekt selbst, aber auch vom Entwicklungskontext abhängen, benutzt jede Firma oder Abteilung in der Praxis ihre eigene Einteilung. So nimmt z.B. der Testaufwand, als Trendfolge der Software-Entwicklung in den USA ([vgl. Noth84]), immer mehr zu, so daß die Einteilung der Phasen verzerrt wird. Recht häufig müssen in Abhängigkeit von der Projektgröße Phasen, die bei kleineren Projekten gewählt wurden, in größeren aufgeteilt werden, um nicht zu unübersichtlich zu werden.

Nachteil dieses klassischen Phasenmodells (vgl. Abbildung 2-1) ist, daß hierbei davon ausgegangen wird, daß die Entwicklung ohne Beteiligung des Anwenders erfolgt. Dieses, in der Entwicklung der Softwaretechnik begründete Vorgehen ist heute nur noch die Ausnahme. In der Regel geschieht die Softwareentwicklung in (meist) direkter Partizipation des Anwenders. Da dieses mit diesem Modell nicht beschrieben werden kann, wurde versucht, diese Zyklizität aus Analyse-Entwurf-Test mit einem geeigneten Modell zu beschreiben. So entstand das Spiralmodell (Abbildung 2-2).

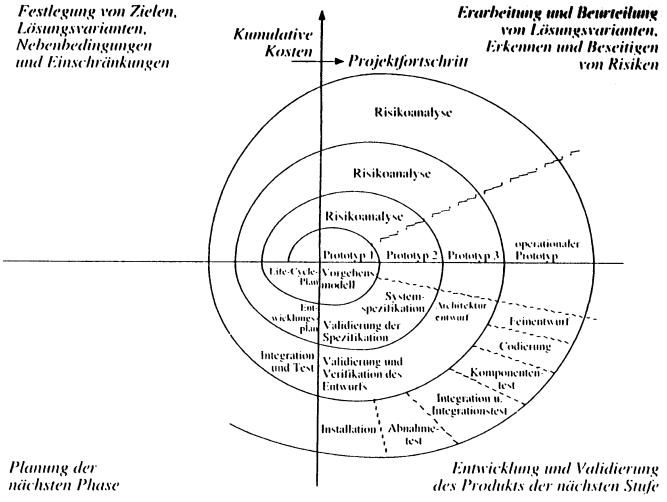

Abbildung 2-2 : Spiralmodell, Quelle: [Grycan99], S. D14-778

Vorgehensweise nach diesem Modell ist das Erstellen eines Prototyps nach erfolgter Anforderungsanalyse. Dieser wird dann vom Anwender bewertet und es werden Verbesserungsvorschläge gemacht. Anhand dieser Vorgehensweise wird dann die Anforderungsanalyse überarbeitet und ein neuer Prototyp entwickelt. Dieses geschieht solange, bis das gemeinsame Ziel erreicht wurde. Daher spricht man auch häufig vom 'Evolutionären Prototyping'.

Auch wenn dieses Modell bereits die Beteiligung des Anwenders im Prozeß der Software-Entwicklung berücksichtigt, so ist die Aufgabenverteilung unter allen Beteiligten nur ungenügend erfaßbar. So wurde die Methodik STEPS (SoftwareTechnik für Evolutionäre, Partizipative Systemgestaltung) entwickelt, die versucht, diesen evolutionären Prozeß genauer zu beschreiben (siehe Abbildung 2-3).
Nach der Projektetablierung erfolgt die Systemgestaltung in Zusammenarbeit mit dem Anwender in mehreren eigenen Zyklen. Danach wird das System gemeinsam in kleinen Arbeitszyklen spezifiziert. Die Software-Realisierung und die Umfeldverarbeitung wird dann vom Entwicklerteam übernommen. Die hierbei entstandene Systemversion wird dann mit dem Anwender abgestimmt, die der Anwender dann nutzt und ggf. pflegt. Aufgrund dieser Einsatzerkenntnisse wird dann eine neue Revision etabliert und der Zyklus erfolgt von neuem, so lange, bis das Projekt abgeschlossen werden kann.

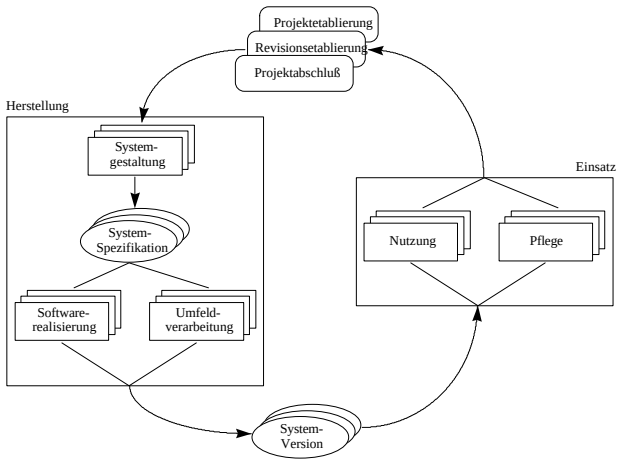

Abbildung 2-3 : Zyklisches Modell, vgl. STEPS [Floyd95]

Trotz des zyklischen Charakters der beiden zuletzt vorgestellten Modelle, ist dennoch ein phasenorientiertes Vorgehen zu erkennen. Und da für einen Vergleich verschiedener Methoden eine gemeinsame Basis gefunden werden muß, habe ich den Versuch unternommen, das Spiralmodell und das zyklische Modell durch ein modifiziertes Phasenmodell zu beschreiben.
Als Grundlage für Phasenmodelle, die nicht, wie hier im Beispiel in Abbildung 2-1 immer aus sechs Phasen bestehen müssen, eignet sich meines Erachtens nach eines aus [Kellner94], da dieses grob genug ist, um andere Phasenmodelle einzuschließen, und fein genug ist, um genaue Aussagen treffen zu können.

Dieses sog. "6+1" Phasenmodell gliedert sich in sechs Projektphasen und einer im Anschluß an das Projekt folgenden Extraphase, in der die Erfahrungen aus diesem Projekt für spätere Projekte aufgearbeitet und archiviert werden. Diese Phase ist zwar für die Aufwandsabschätzung an sich die wichtigste, da hier die Methodik aufgrund der nach Abschluß des Projektes gestiegenen Erfahrung verbessert werden kann (z.B. Erfahrungsdatenbank). Zum Verständnis der Phasenmodelle ist sie aber uninteressant und wird daher hier nicht weiter behandelt.
Die sechs Phasen sind folgende:

Phase I: Orientierung

> In dieser Phase soll sich im Umfeld des Projektes fachlich und inhaltlich orientiert werden mit den Fragen: „Worum geht es ?", „Was soll gemacht werden ?", „Wer will es haben ?" und „Wie ist die aktuelle Situation ?"
> Ergebnis dieser Phase ist eine ausgearbeitete Anforderungsanalyse.

Phase II: Untersuchung

Hier werden Lösungsmöglichkeiten untersucht und ermittelt, welche die besten Ergebnisse liefern und somit am besten die Bedürfnisse befriedigen kann.

Phase III: Funktionsbeschreibung

In dieser Phase werden die Lösungsvorschläge aus der Phase II aufgegriffen, um aus fachlicher Sicht der Anwender - unabhängig von der späteren technischen Realisierung - das Zielprodukt zu beschreiben (Systemvision).
Dabei sollen folgende Punkte definiert werden: Das wird es leisten, so wird es aussehen und so wird es benutzt. Meistens geschieht dieses in Form eines Pflichtenheftes, welches für beide Seiten bindend ist.
In dieser Phase wird die Zielsetzung der Funktionen häufig mit Oberflächenprototypen überprüft und ggf. korrigiert.

Phase IV: Design

Nun wird der technische Entwurf erstellt. Hierbei wird geklärt, wie die in Phase III beschriebenen Funktionen (im weiteren Verlauf auch Geschäftsvorfälle genannt) umgesetzt werden sollen. Eventuell stellt sich hierbei heraus, daß der Oberflächenprototyp überarbeitet werden muß.
In dieser Phase entsteht nun das fachliche und auch das technische Modell, aber auch alle anderen technischen Planungen, wie Rechnerausstattung, Backup-Verfahren und ähnliches werden beschrieben.

Phase V: Entwicklung

Diese Phase beschreibt die eigentliche Arbeit, die funktionalen Vorgaben aus Phase III und die technischen Vorgaben aus Phase IV zu realisieren, zu testen und zu dokumentieren.
Das Testen kann hierbei auf verschiedene Arten passieren:
Z. B. durch den Benutzer anhand einer teilfertigen Vorabversion, mit Demo-Daten oder durch unabhängige Tester.
Das Ziel dieser Phase ist erst dann erreicht, wenn das Endprodukt genau der Beschreibung aus Phase III entspricht.

Phase VI: Einführung

Hat das Projekt-Ergebnis die Phase V passiert, wird das fertige Produkt beim Auftraggeber in Betrieb genommen und ist, nach einer eventuell notwendigen Einweisung der Benutzer, abgeschlossen.

Während das klassische Phasenmodell nicht nur die logische, sondern vor allem die zeitliche Reihenfolge vorgibt und eine Rückkoppelung mit dem Anwender nur in den Phase I, sowie den Kontakt bei der Auslieferung in Phase VI vorsieht, können aber auch modernere Vorgehensweisen, die größere Zusammenarbeit zwischen Entwickler und Anwender vorsehen, mit kleineren Modifikationen in diesem Modell berücksichtigt werden.

Im folgenden sollen die drei verschiedene Entwicklungsmethoden gegenübergestellt werden unter der Frage, ob es möglich ist, diese durch ein Phasenmodell zu beschreiben.

Hierbei bedeutet ein einfacher Pfeil die Weitergabe von Ergebnissen oder auch (Teil-) Produkten nach Abschluß der Phase und ein Doppelpfeil symbolisiert die Zusammenarbeit mit dem Anwender durch Rückkoppelung.

Software-Entwicklung nach dem klassischen Phasenmodell, d.h. strikte Befolgung des Ablaufes der Phasen nach zeitlichen Vorgaben:

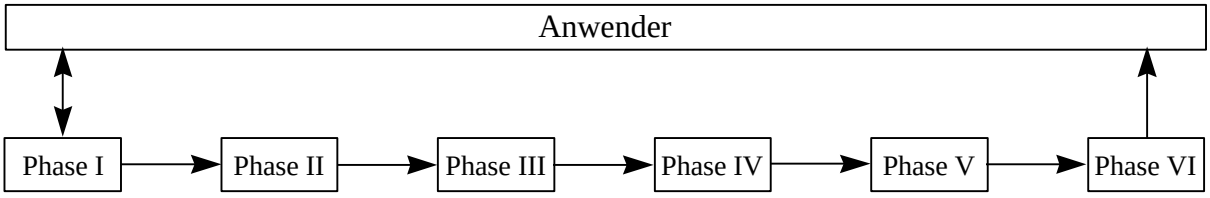

Abbildung 2-4 : Schaubild klassisches Phasenmodell

Software-Entwicklung im evolutionären Prototyping (Spiralmodell):

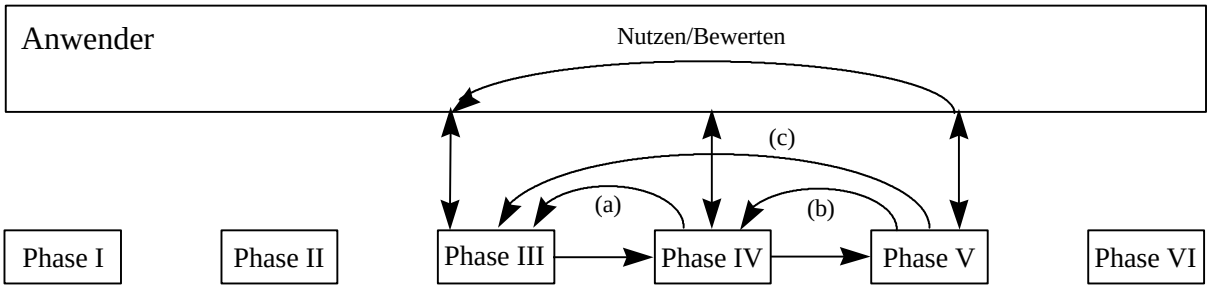

Abbildung 2-5 : Schaubild Phasenmodell zum evolutionären Prototyping

(a) Werden während des Designs z.B. am Prototypen zu korrigierende Merkmale gefunden, so werden diese Erkenntnisse wieder in die Modellierung und den Objektentwurf in Phase III übernommen.

(b) Werden während der Entwicklung oder beim Test Fehler gefunden, die auf Design-Fehler zurückzuführen sind, so wird das Design in Phase IV erneut überprüft.

(c) Zeigt sich bei der Entwicklung oder beim Test, daß Fehler bereits bei den Systemvisionen gemacht wurden, dann wird darauf reagiert, indem diese dann in Phase III ausgemerzt werden.

Auch wenn die Spirale einen Ursprung hat, so ist dieser für dieses Schaubild eher uninteressant, da hierbei der kontinuierliche Vorgang im Vordergrund steht, um ein Anwendungssystem an sich rasch verändernde Randbedingungen anzupassen.

Software-Entwicklung nach STEPS (SoftwareTechnik für Evolutionäre, Partizipative Systemgestaltung):

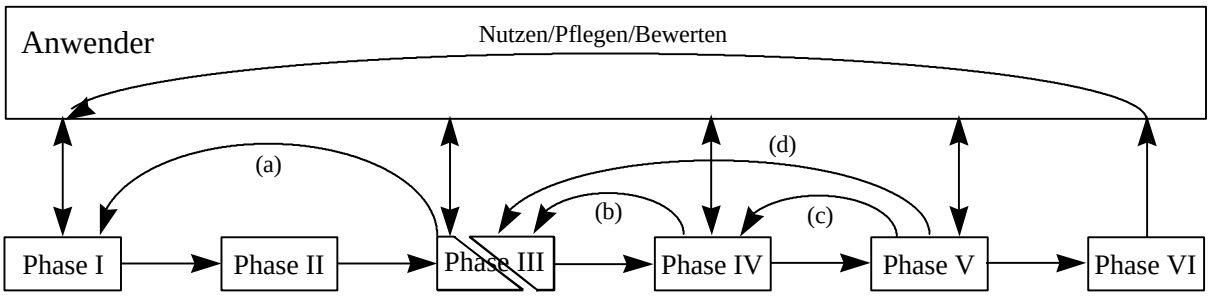

Abbildung 2-6 : Schaubild Phasenmodell zu STEPS

In allen Phasen, in denen Vorgaben vom Anwender verarbeitet werden (Phase I, II, IV und V), wird mit den Anwendern rückgekoppelt, um mögliche Abweichungen festzustellen.

Nach der Projektetablierung (Phase I) erfolgt die Systemgestaltung in Zusammenarbeit mit dem Anwender in mehreren eigenen Zyklen (a). Danach wird das System gemeinsam in kleinen Arbeitszyklen spezifiziert (Phase III). Die Software-Realisierung und die Umfeldverarbeitung wird dann vom Entwicklerteam übernommen (Phase IV und V). Die hierbei entstandene Systemversion wird dann mit dem Anwender abgestimmt (Phase VI), die der Anwender dann nutzt und ggf. pflegt. Aufgrund dieser Einsatzerkenntnisse wird dann eine neue Revision etabliert (a) und der Zyklus erfolgt von neuem, so lange, bis das Projekt abgeschlossen werden kann.
Darüber hinaus können noch folgende Situationen eintreten:

(a) Werden bei der Entwicklung von Systemvisionen Abweichungen zwischen fachlichem Verständnis des Anwenders und den Ergebnissen der Entwickler festgestellt, so werden diese durch eine neue Bewertung in Phase I mit einer neuen Lösung in Phase II überarbeitet.

(b) Werden während des Designs z.B. am Prototypen zu korrigierende Merkmale gefunden, so werden diese Erkenntnisse wieder in die Modellierung und den Objektentwurf in Phase III übernommen.

(c) Werden während der Entwicklung oder beim Test Fehler gefunden, die auf Design-Fehler zurückzuführen sind, so wird das Design in Phase IV erneut überprüft.

(d) Zeigt sich bei der Entwicklung oder beim Test, daß Fehler bereits bei den Systemvisionen gemacht wurden, dann wird darauf reagiert, indem diese dann in Phase III ausgemerzt werden.

Die Phase III ist in Abbildung 2-6 geteilt, um deutlich zu machen, daß eine Rücksprungkette von Phase V nach Phase I nicht möglich ist.
Wichtig hierbei um in endlicher Zeit ein fertiges Produkt zu erhalten ist, beim ersten Übergang von der linken Hälfte des Phase-III-Symbols in die rechte die Anforderungen an das zukünftige System und auch die in diesem Projekt enthaltenen Erweiterungsmöglichkeiten genau zu definieren (Pflichtenheft).
Da aber aufgrund der Rückkoppelungen und zyklischen Verhaltensweisen ursprüngliche Anforderungen z. T. Verworfen und neue aufgenommen werden, sollte eine Festlegung der Ziele in vier Kategorien erfolgen (vgl. [Kellner94]): Ziel muß (unbedingt erforderlich für das Projekt), soll (erforderlich für Projekt, aber kein K.O.-Kriterium), kann (positiver Nebeneffekt) und darf nicht (Abgrenzung nach unten hin, Erfüllung unbedingt erforderlich).

Eine Aufwandsabschätzung, die nach einer Phase *X* gemacht wird, gibt in der Regel den Restaufwand des Projektes am Phase *X*+1 wieder.
Dabei wird zwar ein Mindestmaß an Verzögerung, u.a. auch durch kleine Ergänzungen, berücksichtigt, aber keines der im Rahmen dieser Arbeit näher beleuchteten Verfahren berücksichtigt zyklische Vorgänge.
Daher ist es notwendig, bei jedem Rücksprung - (a) bis (d) - erneut eine Aufwandsabschätzung auf Basis der neuen Werte durchzuführen und diese mit der ursprünglichen Schätzung zu vergleichen. So kann auch ermittelt werden, ob eine Anforderungsänderung in den Zeitplan bzw. in das Budget paßt.

Zusammenfassend kann man sagen, daß das klassische Phasenmodell auf heutige Entwicklungen nicht mehr unverändert angewendet werden kann. Eine Transformation des Spiralmodells und des zyklischen Modells (STEPS) auf ein modifiziertes Phasenmodell sollte zeigen, daß eine gemeinsame Basis gefunden werden kann. Da diese Varianten des Phasenmodells aber schnell unübersichtlich werden können, geben die Abbildungen 2-2 und 2-3 den eigentlichen Charakter dieser Modelle am besten wieder.
Ferner ist ersichtlich, daß es schwierig ist, reale Projekte in theoretische Modelle einzufügen, so daß höchstwahrscheinlich in der Praxis eher Mischformen dieser drei Modelle vorkommen werden.

2.2 Software-Qualität / -Qualitätssicherung

Zunächst soll geklärt werden, was Software-Qualität bedeutet. Ferner soll aufgezeigt werden, welchen Voraussetzungen es bedarf, Softwarequalität zu sichern und welchen Einfluß dieses auf den zu erwartenden Aufwand hat.

In der Berufswelt wird der Slogan „Qualität ist, wenn der Kunde zurückkommt und nicht das Produkt“, recht häufig verwendet.
Dieses trifft den Kern der heutigen Auffassung des Begriffs ‘Qualität’, nämlich Qualität als Grad der Übereinstimmung zwischen dem, was der Auftraggeber möchte und dem, was letztendlich als Produkt herausgekommen ist.
Daher ist es auch verständlich, daß moderne Entwicklungsmethoden, wie z.B. STEPS, diese Übereinstimmung durch Partizipation des Endanwenders möglichst hoch halten sollen.

Klassischer Ansatz der Qualitätsmessung ist, Qualität als Quotient aus Anzahl der Fehler und Anzahl der Codezeilen (LoC, siehe auch Anhang B.2) zu betrachten. Dieses ist unzureichend, zeigt aber deutlich die Schwierigkeit, Qualität einfach meßbar zu machen. Gerade deswegen wird dieses Thema ausführlich in Publikationen behandelt, ohne ein allgemeingültiges und universal einsetzbares Konzept zu entwickeln.

Sneed (vgl. [Litke96]) beschrieb 1987 als erster, daß Qualität aus einer Reihe von Faktoren besteht:

- Zuverlässigkeit,
- Sicherheit,
- Zeiteffizienz,
- Speichereffizienz,
- Benutzerfreundlichkeit,
- Ausbaufähigkeit,
- Übertragbarkeit und
- Wartungsfreundlichkeit.

Hieraus wird ersichtlich, daß es bei dem Verständnis von Software-Qualität nicht nur um die Zufriedenheit des Kunden, sondern auch um entwicklungstechnische Eigenarten des Produktes geht.

Später folgten weitere Ansätze, wie z.B. von Boehm, Brown und Kaspar (vgl. [Noth84]), und auch [Kellner94] nennt einige interessante Ansätze.

In [Floyd95] ist ein Faktorenkatalog angegeben, der denen in der ISO 9000ff. normierten Faktoren (vgl. Erweiterung Data-Point-Verfahren, Kap. 3.2.3 (d)) ähnelt. Versucht man all diese Qualitätsfaktoren zu einem Katalog zu verbinden, so erhält man z.B. diesen Faktorenkatalog:

Einsatzqualität:

- Relevanz (inhaltliche Qualität)
  Wurden alle geforderten Merkmale realisiert ?
- Effektivität (Zeit- und Speichereffizienz)
  Läuft das Programm schnell und ressourcensparend ?
- Betriebsqualität (Toleranz und Robustheit, Sicherheit)
  Läuft es stabil oder bringen fehlerhafte Eingaben es zum Abstürzen ?
- Benutzungsqualität (Benutzerfreundlichkeit, Übersichtlichkeit, Erfüllung von Ergonomiekriterien, gute Dokumentation)
  Wie gut kann der Anwender das Produkt bedienen und verstehen ?

Produktqualität:

- Korrektheit
  Werden Fehler behandelt bzw. abgefangen oder sind welche bekannt ?
- Verständlichkeit (Dokumentation)
  Kann von projektfremden Dritten der Entwicklungsprozeß nachvollzogen und der Quellcode verstanden werden ?
- Änderbarkeit (Wartungsfreundlichkeit)
  Ist der Sourcecode so aufgebaut, daß er problemlos von projektfremden Dritten erweitert und verändert werden kann ?
- Wiederverwendbarkeit (Portabilität)
  Können Teile des Programmes wiederverwendet werden ?

Prozeßqualtität:

- Sicherung der Einsatz- und Produktqualität
  Wie gut sind die gesteckten Ziele bezüglich der Qualität erreicht worden ?
- Formale Qualität (Produkt im gesteckten Rahmen realisiert)
  Konnte das Projekt im zeitlichen und finanziellen Rahmen realisiert werden ?

Wie können nun diese Qualitätsmerkmale gesichert werden ?
Relevanz und *Benutzerqualität* können nur durch eine verstärkte Einbeziehung des Anwenders während des Entwicklungsprozesses erreicht werden.
Der Grundstein für *Effektivität* und *Betriebsqualität* muß in der Designphase gelegt werden, während die Voraussetzung zum Erreichen der *Produktqualitätsfaktoren* ein dementsprechender Programmierstil ist.
Einzige Ausnahme hierbei stellt der Faktor *Korrektheit* dar, da dieser, trotz umfangreicher Tests, in der Praxis nie absolut erreicht wird.

Um diese Faktoren zu erreichen bzw. das Erreichen zu sichern, ist die Projektorganisation gefragt, neben der kontinuierlichen Kontrolle dieser Ziele auch den damit verbundenen Mehraufwand im zeitlichen und finanziellen Rahmen zu halten (*Formale Qualität*).

Somit ist leicht ersichtlich, warum in der Praxis recht häufig Kompromisse eingegangen werden müssen, die dann von der optimalen Lösung eines Problems abweichen. Aus den o.g. Beispielen kann abgeleitet werden, daß eine Erhöhung der Qualität einen erhöhten Aufwand und somit eine Steigerung der Kosten, sowie eine Dehnung des zeitlichen Rahmens verursacht. Deutlich wird dieses an Abbildung 2-7, dem sogenannten Teufelsquadrat, denn bei gleichbleibender personeller Kapazität (innere Quadrate) steht ein Ziel dem anderen im Gegensatz:

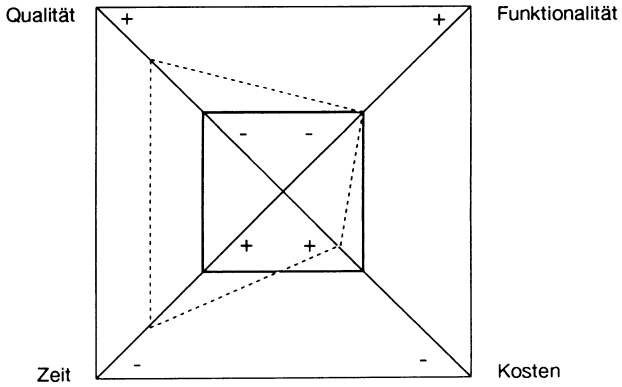

Abbildung 2-7 : Teufelsquadrat, Quelle: [Kellner94], S. 183

Steigt z.B. die Qualität, leidet die Funktionalität und die Kosten steigen. Gleiches geschieht, wenn weniger Zeit zur Verfügung steht.
Also können alle vier Eckpunkte des Quadrates niemals gleichzeitig optimal erreicht werden, auch wenn dieses sehr wünschenswert wäre.

Welches Qualitätsmerkmal wichtiger ist als andere, kann nicht generell entschieden werden, sondern muß in Abhängigkeit vom Einsatz- und Projektkontext entschieden werden.

2.3 Zeit- und Aufwandsmaße

Das Ergebnis einer Aufwandsabschätzung liegt meistens in der Einheit MM vor. Im amerikanischen bedeutet dieses ‘man-month’, also geschlechtsunabhängig Monate pro Mensch (vgl. ‘mankind’). In der Regel wird dieser Begriff (vermutlich auch wegen der Entwicklungsgeschichte der Softwaretechnik) mit Mannmonate wortwörtlich übersetzt.
Dabei wird dieser Begriff heutzutage meist wertungsfrei für männliche und auch weibliche Softwareentwickler gebraucht (wie z.B. bei Drei-Mann-Zelt). Um diese ‘Geschlechtslosigkeit’ der Abkürzung MM zu unterstreichen, wird auch häufig der Begriff ‘Mitarbeitermonate’ oder ‘Menschmonate’ gebraucht.
Es ist sicherlich eine Geschmacksfrage, welche Bedeutung man bevorzugt; ich werde aber im weiteren Verlauf entweder nur die Abkürzung MM oder aber ausgeschrieben ‘Mannmonate’ ohne sinngeschlechtliche Wertung verwenden, da dieses in der von mir erfahrenen Praxis am häufigsten gebräuchlich ist. Leserinnen dieser Arbeit möchten mir dieses bitte nachsehen.

Da aber nicht alle Projekte in dieser Einheit abgerechnet oder erfaßt werden, muß man wissen, wie man MM in MT (=Manntage) oder auch in MH (=Mannstunden) umrechnen kann. Bei diesen Abkürzungsdeutungen verhält es analog zu der o.g. Thematik.

Da es auch nicht bedeutet, daß eine Tätigkeit bei einem Aufwand von z.B. 5 MT in genau einer Arbeitswoche erledigt ist, werde ich diese Thematik in einem zum Verstehen dieser Arbeit ausreichenden Rahmen beleuchten. Kapitel 2.3.2 widmet sich dem Vergleich zwischen Aufwand und verstrichener Arbeitszeit.

2.3.1 Umrechnungen

In der Arbeitswelt hat ein durchschnittlicher Monat 21 Werktage und 168 Arbeitsstunden. Daher ergibt sich ein Schnitt von 8 Stunden pro Werktag.
Dieses ist zwar die Zeit, die ein normaler, vollzeitbeschäftigter Arbeitnehmer bei seinem Arbeitgeber anwesend ist, aber nicht der Umfang der Zeit, die er effektiv arbeitet.
Daher bedeutet die allgemeine Definition des Mannmonates (MM), als den Umfang der von einem vollzeitbeschäftigten Arbeitnehmer in einem Kalendermonat erbrachte Leistung, nicht, daß dieser 168 Arbeitsstunden volle Leistung bringt.

Während [Litke96] und [Noth84] auf das Modell COCOMO (vgl. Anhang B.3 (k)) verweisen, bei dem versucht wird, die Produktivität jedes einzelnen Mitarbeiters mittels mathematischer Verfahren zu ermitteln, liefert [Kellner94] den auf Erfahrungen basierenden Mittelwert, daß eine Arbeitswoche nur vier effektive Tage und ein Tag nur sechs effektive Stunden besitzt. Hierfür gibt die Autorin eine Reihe von Beispielen an, was mit der restlichen Zeit passiert: Z.B. einen Papierstau im Drucker beseitigen, Mißverständnisse mit anderen Mitarbeitern klären, dem Chef auf dem Flur begegnen und sich festreden oder einfach nur eine kleine Pause machen. Dieses ist natürlich nur ein kleiner Ausschnitt dessen, was noch denkbar wäre.

Daraus ergeben sich folgende Zusammenhänge:

Aufwand:			Zeit:		
1 MM ÷	17 MT =	102 MH	1 Monat =	21 Tage =	168 Stunden
	1 MT =	6 MH		1 Tag =	8 Stunden

Für einen einzelnen Mitarbeiter läßt sich der Aufwand 1 MM mit dem Kalendermonat gem. Definition gleichsetzen und somit einen Zusammenhang zwischen Zeit und Aufwand ermitteln.

2.3.2 Zusammenhang Aufwand [MM] und Kalendermonate

Wie bereits in Kapitel 2.3.1 gesehen, ist der Zusammenhang zwischen Aufwand und Zeit bereits bei einer Person nicht einfach ermittelbar und es stellt sich die Frage, wie sich dieses bei mehreren Mitarbeitern verhält.

Im optimalen Fall läßt sich eine Aufgabe mit dem Aufwand *A* mit *n* Mitarbeitern in der Zeit $\frac{A}{n}$ realisieren. Daß dieses in der Praxis nur bei sehr kleinen Gruppen zutrifft, liegt auf der Hand. Mit steigender Zahl von Mitarbeitern steigen, aufgrund der Unteilbarkeit einiger Vorgänge und Aufgaben, die Leerläufe einzelner Mitarbeiter. Ebenfalls nimmt der Aufwand für die Koordination der verschiedenen Tätigkeiten und Aufgaben mit der Zahl der Beteiligten zu (siehe Abbildung 2-8).

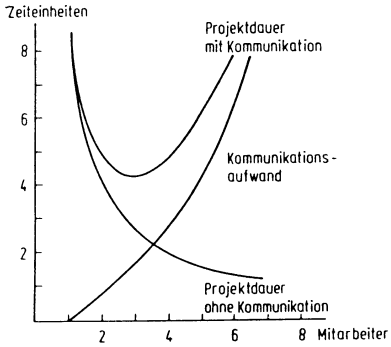

Abbildung 2-8 : Einfluß Kommunikationsaufwand, Quelle: [Noth84]

Daher stellt sich die Frage nach der optimalen Mitarbeiterzahl (MAZ) für ein Projekt, für das der Aufwand bereits geschätzt wurde.
Aufgrund der Parabelform der Projektdauer-Kurve mit Kommunikationsaufwand (siehe Abbildung 2-8) hat sich hierfür folgende Formel in der Literatur (vgl. z.B. [Noth84]) etabliert:

$$\text{opt. MAZ} = \sqrt{Aufwand} \text{ [MM]}$$

Dieses kann ich für kleinere Projekte mit einem Aufwand von bis zu 12 MM durchaus bestätigen.

Kennt man nun die optimale Zahl der Beteiligten und den Gesamtaufwand, bleibt nur noch die Frage zu klären, wie lange (in Kalenderwochen oder -monaten) dieses Projekt, unter Berücksichtigung von Arbeitsteilung und Kommunikationsbedarf, benötigt.

[Kellner94] liefert hierfür einen linearen Ansatz durch die Aussage, jeder Projektmitarbeiter würde den Projektleiter 15% seiner eigenen Arbeitskraft für Koordination und Kommunikation kosten. Die somit gewonnene Relation zwischen Projektdauer (*PD* in Monaten), Aufwand (*A* in MM) und Mitarbeiterzahl (*MAZ*, inkl. Projektleiter)

$$PD = \frac{A}{0{,}85 * MAZ + 0{,}15}$$

deckt die in Abbildung 2-8 gezeigte Kurve der Projektdauer mit Kommunikation nur als Näherung für kleine *MAZ* genau genug ab. Grenzwert sind acht Personen (sieben Mitarbeiter plus ein Projektleiter, der sich dann nur noch der Koordination widmet) und ein Gesamtaufwand von 49 MM.

[Noth84] und [Litke96] beziehen sich hierbei auf das COCOMO-Modell. Dabei wird die Reihenfolge der zu ermittelnden Größen umgekehrt.
Zur Bestimmung der optimalen Projektdauer (*OPD*) wird eine Exponentialfunktion verwendet:

$$OPD = 2{,}5 * A^{S}, \text{ wobei } \quad S = 0{,}35 \text{ für Online-Systeme und}$$
$$S = 0{,}32 \text{ für Realtime-Systeme}$$

Um nun den durchschnittlichen Personalbedarf (*PB*) zu ermitteln, wird der Aufwand (*A*) durch die optimale Projektdauer (*OPD*) geteilt.
Auch wenn eine Exponentialfunktion die in Abbildung 2-8 gezeigte Kurve des Kommunikationsaufkommens besser beschreibt als eine lineare Relation, so ist dieses Vorgehen jedoch nur schwer nachvollziehbar. Der Faktor vor *A* und der Exponent *S* sind nicht ohne weiteres überprüfbar, so daß der praktische Nutzen dieser Gleichung nahe bei Null liegen dürfte.

2.4 Ansprüche an die Methoden zur Aufwandsabschätzung

An dieser Stelle soll nun geklärt werden, welche Ansprüche an die Methoden zur Aufwandsabschätzung gestellt werden.
Dabei haben die hierbei von mir gewählten Kriterien keinen Anspruch auf Vollständigkeit oder Ausschließlichkeit; es sind noch weitere bzw. andere denkbar, aber die vier an dieser Stelle vorgestellten Bewertungskriterien sind meines Erachtens die wichtigsten.

Wie bereits in Kapitel 2.1- Modelle zur Projektstrukturierung - gezeigt, werden für eine effektive Projektkontrolle und -steuerung genaue Aussagen über den zu erwartenden Aufwand benötigt. Daher müssen ‘gute’ Methoden zur Aufwandsabschätzung möglichst genaue Schätzungen ermöglichen, um eine sinnvolle Anwendung der Methoden zu bedingen.
Deshalb erachte ich das Kriterium ‘Schätzgenauigkeit’ als sehr wichtig, wenn nicht sogar als das Wichtigste. In [Noth84] ist es eines von mehreren der Kategorie ‘Ergebnisqualität’.

Für die Projektorganisation ist es wünschenswert, möglichst früh genaue Aussagen über den zu erwartenden Aufwand treffen zu können.
Daher sollten die Methoden zur Aufwandsabschätzung dahingehend untersucht werden, ab wann (vgl. Kapitel 2.1) diese eingesetzt werden kann, und ob vorher bestimmte Voraussetzungen erfüllt werden müssen (Allgemeingültigkeit). [Noth84] berücksichtigt dieses unter ‘Einsetzbarkeit’ (Teil von Benutzerfreundlichkeit).

Ferner empfehle ich als Kriterium die Nachvollziehbarkeit der Schätzvorganges bzw. des dadurch erzielten Ergebnisses, da in der Praxis eine Methode zur Aufwandsabschätzung, die ein nicht interpretierbares Ergebnis liefert oder bei der die Glaubwürdigkeit des Ergebnisses angezweifelt wird, keine Akzeptanz findet.
In der Praxis muß nicht nur diejenige Person, die die Schätzung durchführt, das Ergebnis anerkennen, ferner müssen alle mit der Realisierung betrauten Personen, sowie der Auftraggeber dem Schätzergebnis glauben schenken.
[Noth84] schlüsselt dieses in die Kriterien ‘Transparenz’, ‘Nachvollziehbarkeit’ und ‘Bewertbarkeit’ auf.

Nimmt man den Gedanken des Kapitels 2.2 - Software-Qualität / -Qualitätssicherung - auf, so ergibt sich das vierte Kriterium fast von selbst: die Berücksichtigung der geforderten Qualitätsmerkmale (Produktqualität).
Wie im o.g. Kapitel gezeigt, spielen Qualitätsmerkmale eine entscheidene Rolle bei der Softwareentwicklung und beeinflussen den Aufwand sehr stark, so daß eine Betrachtung dieser Merkmale sehr wichtig ist.
Allerdings ist es weder von Vorteil, möglichst viele Qualitätsmerkmale zu berücksichtigen und die Methode dadurch so gut wie nicht anwendbar zu machen, noch zu versuchen, wenige sehr wichtige Merkmale herauszufiltern und nur diese zu betrachten. An den aktuellen Diskussionen dieses Themas kann man sehen, daß wie schwierig es ist, zu entscheiden, welches Qualitätsmerkmal wichtiger als ein anderes es. Darüber hinaus ist auch die Frage nach der Art der Einschätzung bzw. Bewertung ebenfalls nicht eindeutig zu beantworten. So hängt die Auswahl von Merkmalen vom Einsatzkontext und von der Einschätzung der Wichtigkeit, durchgeführt durch die am Projekt beteiligten Personen ab.

Ich schlage deshalb vor, einem Katalog von in der Praxis häufig geforderten Qualitätsmerkmalen zu nehmen, die dann in leicht bewertbaren Kategorien, wie z.B. ‘für dieses Projekt wichtig’, ‘für dieses Projekt zu berücksichtigen’ und ‘für dieses Projekt nicht wichtig’ eingeteilt werden. Dabei sollte weder die Anzahl und Relevanz zu kurz kommen, noch die Anwendbarkeit der Merkmalsbewertung unnötig erschwert werden.
In [Noth84] fließt diese Betrachtung durch die Bewertung der Kriterien ‘Einflußabdeckung’ und ‘Parameterzahl’ ins Vergleichsergebnis ein.

Die vier Bewertungskriterien nochmals im Überblick:

- Schätzgenauigkeit
- unbedingte Voraussetzungen
- Nachvollziehbarkeit
- Berücksichtigung von Qualitätsmerkmalen

Nachdem nun über die Grundlagen der Projektsteuerung gesprochen und wichtige Begriffe eingeführt wurden, werden im nun folgenden Abschnitt eine Auswahl der von mir recherchierten Methoden vorgestellt. Drei dieser Methoden sollen gesondert vor dem Hintergrund der oben aufgeführten Anforderungen an diese Methoden untersucht werden.

3 Methoden zur Aufwandsabschätzung

Eine komplette Auflistung aller im Rahmen dieser Arbeit recherchierten Methoden zur Aufwandsabschätzung befindet sich im Anhang B.
In diesem Kapitel findet sich lediglich eine Auswahl an Methoden, die Aufgrund ihrer Art, ihrer Bedeutung oder ihrer Aktualität interessant sind.
Drei Methoden hieraus werden dann in Kapitel 3.3 näher untersucht und anhand der Kriterien aus Kapitel 2.3 bewertet.

3.1 Zusammenfassung ähnlicher Methoden zu Gruppen

Da sich viele Methoden zur Aufwandsabschätzung ähneln, gruppiert man diese um herauszufinden, ob es sich um gänzlich unterschiedliche Ansätze oder nur um Modifikationen des selben Ansatzes handelt.

3.1.1 klassische Kategorisierung

In der Literatur (vgl. [Noth84]) erfolgt eine Einteilung überwiegend anhand des der Methode zugrundeliegenden Verfahrens.

(a) Analogiemethoden:

Die Schätzung erfolgt durch Vergleich des Projektes bzw. von einzelnen Teilen mit bereits abgeschlossenen (Teil-) Projekten. Annahme hierbei ist, daß ähnliche Projekte einen ähnlich hohen Aufwand besitzen.

(b) Relationsmethoden:

Ähnlich der Analogiemethoden; der Vergleich mit bereits abgeschlossenen Projekten erfolgt durch methodische Ansätze der Mustererkennung.

(c) Multiplikatormethoden:

Zuerst wird das Projekt durch eine Bottom-Up-Zerlegung in einzelne Teilprojekte zerteilt. Danach wird für jeden Teil eine eigene Aufwandsabschätzung (meist mit der Analogiemethode) durchgeführt. Anschließend werden die Ergebnisse zu einem Gesamtergebnis zusammengefügt.

(d) Gewichtungsmethoden:

Die Aufwandsabschätzung geschieht durch eine Schätzgleichung, die aus gewichteten Einzelfaktoren zusammengesetzt ist.

(e) Methoden der parametrischen Schätzgleichungen:

Die gewichteten Einzelfaktoren (Parameter) der Schätzgleichungen werden durch Korrelationsanalysen bereits abgeschlossener Projekte quantifiziert, d.h. die Schätzgleichung stellt eine Näherung dar, die die Ergebnisse der abgeschlossenen Projekte am besten beschreibt.

(f) Prozentsatzmethode:

Das in einzelne Phasen (vgl. Kap 2.1) zerlegte Projekt wird anhand einer prozentualen Verteilung, die auf bereits abgeschlossenen Projekten basiert, abgeschätzt. Meistens wird hierzu eine Phase des Projektes durchgeführt und anhand der Verteilung erfolgt dann eine Schätzung des Gesamtaufwands.

3.1.2 datenbezogene Kategorisierung

Der Nachteil der klassischen Kategorisierung liegt darin, daß die meisten Methoden zur Aufwandsabschätzung nicht nur einer Gruppe angehören, sondern vielmehr eine Mischform aus mehreren Verfahrensweisen darstellen.
Daher ist meines Erachtens eine Einteilung sinnvoller, die Methoden, die die gleichen Eingangswerte benutzen, zusammenfaßt.

(a) Methoden basierend auf Erfahrungsanalogien:

Diese Gruppe umfaßt alle Methoden, die einen freien Vergleich mit bereits abgeschlossenen Projekten als Grundlage der Schätzung benutzen.
(z.B. (a), (b), (c) und (f) der klassischen Kategorisierung)

(b) Methoden basierend auf Lines-of-Code:

Die Methoden, die dieser Gruppe angehören, gehören zu den ersten, die zur Aufwandsabschätzung entwickelt wurden. Als Grundlage dient meisten eine Gleichung mit einem oder mehreren Parametern, wovon der wichtigste die Anzahl der Zeilen ausführbaren Codes (meist in KiloDeliveredSourceInstructions = KDSI) ist (vgl. [Noth84] und [Litke96]).
(z.B. (d) und (e) der klassischen Kategorisierung)

(c) Methoden basierend auf Anzahl von Befehlen

Als Weiterentwicklung der Gruppe (b) basieren diese Methoden, deren Aufbau ihren Vorgängern recht ähnlich ist, auf der Anzahl von ausgeführten Befehlen. Zum Teil wird auch die Anzahl der Entscheidungen (If-Cases) ebenfalls berücksichtigt.
(z.B. (d) und (e) der klassischen Kategorisierung)

(d) Methoden basierend auf Funktionen

Als Kenngrößen dieser Methoden dienen die zu realisierenden Funktionen (hier: Geschäftsvorfälle) im allgemeinen bzw. die Entitäten des zugrundegelegten Datenmodells im speziellen. Vorgehensweise ist die Bewertung der einzelnen Funktionen und Ermittlung eines (Roh-) Aufwandswertes. Dieser wird anschließend mit einem zu ermittelnden Faktor modifiziert und dann mit Hilfe einer, auf den Erfahrungen mit bereits abgeschlossenen Projekten basierenden Tabelle, in den Gesamtaufwand überführt.
(z.B. (a), (c) und d) der klassischen Kategorisierung)

(e) Methoden basierend auf teilprodukt-spezifischen Kenngrößen

Basierend auf einer Erfahrungsdatenbank, in der die Kenndaten bereits abgeschlossener Projekte erfaßt wurden, werden die Teilprodukte des aktuellen Projektes anhand spezifischer Werte des Entwicklungskontextes bewertet, woraus sich dann der Gesamtaufwand ergibt.
(z.B. (a) und (c) der klassischen Kategorisierung)

3.2 Methoden-Übersicht

Wie im Anhang B zu erkennen, existieren in jeder Gruppe mehrere Methoden zur Aufwandsabschätzung.
An dieser Stelle sollen nur einige wenige davon besonders hervorgehoben werden, da sie entweder das aktuellste Glied einer Entwicklungskette darstellen, besonders häufig eingesetzt werden oder aber besondere Aspekte beleuchten, die die anderen nicht berücksichtigen.

3.2.1 Methoden basierend auf Erfahrungsanalogien

(a) Analogie-Schätzung

Diese Art der Aufwandsabschätzung ist die am häufigsten verwendete, da sie einfach und intuitiv anwendbar ist.
Meistens wird ein Projekt hierzu in Teilprojekte bzw. Phasenabschnitte und Meilensteine zerlegt (vgl. Kap. 2.1). Daraufhin wird der Aufwand der einzelnen Teile abgeschätzt, indem diese Teile mit ähnlichen Teilen bereits abgeschlossener Projekte verglichen werden. Häufig wird hierzu eine Erfahrungsdatenbank aufgebaut, die den Aufwand bereits realisierter Projekte enthält, um zukünftige Schätzungen zu verbessern.
Den Gesamtaufwand erhält man dann durch Zusammensetzen der Teilergebnisse.

Dennoch wird die mit dieser Methode durchgeführte Schätzung durch eine Vielzahl von äußeren Faktoren beeinflußt, so daß dieses Verfahren einen hohen Unsicherheitsgrad besitzt.
In [Kellner94] wird angegeben, wie hoch die Wahrscheinlichkeit ist, daß die Schätzung mit einer Abweichung von ñ 10% stimmt (vgl. Kap. 2.1):

2 : 3	nach Phase I: Orientierung
3 : 4	nach Phase II: Untersuchung
4 : 5	nach Phase III: Funktionsbeschreibung
9 : 10	nach Phase IV: Design
9 : 10	nach Phase V: Entwicklung

Auch wenn diese Methode in der Praxis häufig eingesetzt wird, lassen sich die damit erzielten Ergebnisse nur schwer vergleichen, da man nicht das Verfahren an sich, sondern vielmehr die Fähigkeiten des Schätzers bewerten würde.

(b) Prozentsatzmethode: [UNI-H97] / [TU-B98]

Eine weitere, relativ häufig verwendete Methode, ist die Prozentsatzmethode. Der Grundgedanke hierbei ist, daß gleichartige Projekte in gleichartigen Umgebungen ähnliche Aufwandsverteilungen über die einzelnen Phasen (vgl. Kap. 2.1) besitzen.
Unter dieser Annahme wird dann eine Phase entweder durchgeführt und der Aufwand nach Beendigung dieser Phase ermittelt oder es wird versucht, eine Phase möglichst genau zu schätzen (mit anderen Methoden). Mit diesen Daten wird dann der Aufwand der anderen Phasen ‘hochgerechnet’.

Somit kann der zu erwartende Aufwand von jeder Phase aus den Daten einer beliebigen anderen Phase ermittelt werden.
Ist der Aufwand der einzelnen Phasen bekannt, kann durch Addition der Gesamtaufwand des Projektes ermittelt werden.

Als Beispiel soll hier die Verteilungen von Stahlknecht (vgl. [TU-B98]) dienen, da diese von ihrer Aufteilung dem in Kap2.1 eingeführten Phasenmodell am nächsten kommt:

Phase	relativer Aufwand
Projektbegründung	5 %
Ist-Analyse	10 %
Soll-Konzept	10 %
Systementwurf	30 %
Implementierung	35 %
Systemeinfühung	10 %

Tabelle 3-1 : Prozentuale Verteilung nach Stahlknecht

Da diese Methode geringe Anforderungen an ihre Einsetzbarkeit stellt, wird sie in die Bewertung einfließen.

3.2.2 Methoden basierend auf Lines-of-Code / Methoden basierend auf Anzahl von Befehlen

Auch wenn die Vertreter dieser beiden Gruppen aufgrund der Entwicklungsgeschichte überholt und somit eher uninteressant im Rahmen dieser Arbeit wären, haben zwei Vertreter interessante Ansätze, die an dieser Stelle kurz beleuchtet werden sollen.

(a) Aron-Verfahren: [Noth84]

Grundlage von STEPS ist, daß die Softwareentwicklung zyklisch verläuft und daß die Entwickler in diesen Zyklen ihre Produktivität durch den Lerneffekt längerer Projekte verbessern.
Um so erstaunlicher ist es, daß Aron als einer der Wenigsten bereits 1969 versuchte, diesen Einfluß des Lerneffektes zu berücksichtigen (vgl. [Noth84]).
Unter allen, im Rahmen dieser Arbeit recherchierten Methoden, kann sie sogar als einzige bezeichnet werden.

Tabelle 3-2 zeigt den Produktivitätsgewinn in Abhängigkeit von der Projektdauer und vom Schwierigkeitsgrad:

Projektdauer / Schwierigkeitsgrad	6–12 Monate	12–24 Monate	Mehr als 24 Monate
Leicht	20	500	10.000
Durchschnittlich	10	250	5.000
Schwer	5	125	1.500
	Instruktionen pro Mann-Tag	Instruktionen pro Mann-Monate	Instruktionen pro Mann-Jahr

Tabelle 3-2 : Lerneffekt nach Aron

Auch wenn diese Methode aufgrund der Berücksichtigung des Lerneffektes eine Besonderheit darstellt, ist es aufgrund der zugrundeliegenden Schätzfunktion für eine Bewertung uninteressant.

(b) Constructive-Cost-Model (COCOMO): [Noth84] / [Litke96]

COCOMO ist eine Entwicklung von Barry W. Boehm, der 1981 Direktor von Thompson-Ramo-Wooldridge, eines großen Amerikanischen Softwarehauses war.
Es gibt drei verschiedene Modelle, die sich im Zeitpunkt der Schätzung und der Granularität der Vorgehensweise unterscheiden.

Interessant an diesem Verfahren ist die Tatsache, daß zur Ermittlung des Aufwandes, neben der Eingangsgröße in KDSI (KiloDeliveredSourceInstructions, d.h. 1000 ausgeführte Befehle im Sourcecode), ebenfalls der Einsatzkontext, die Qualität und die Produktivität der Entwickler berücksichtigt werden.

Für das Basismodell sind dieses z.B. folgende Abschätzungsgleichungen:

Batchsysteme: $MM = 2.4 * KDSI^{1.05} * Q * P$

Onlinesysteme: $MM = 3.0 * KDSI^{1.12} * Q * P$

Realtimesysteme: $MM = 3.6 * KDSI^{1.2} * Q * P$

mit Q: Qualitätsfaktor und P: Produktivitätsfaktor

Da bei diesem Verfahren leider keine genauen Angaben über die Zusammensetzung der Faktoren Q und P gemacht werden und da als Eingangsgröße eine veraltete Einheit benutzt wird, kann diese Methode nicht näher bewertet werden.

3.2.3 Methoden basierend auf Funktionen

Diese Gruppe stellt die interessanteste im Rahmen dieser Arbeit dar.
Die Vorgehensweise ist bei allen Methoden dieser Gruppe ähnlich:
Es werden Gewichtungspunkte für Realisierungsmerkmale anhand von Tabellen vergeben, die dann mit Hilfe einer mathematischen Funktion in den Gesamtaufwand überführt werden.

(a) IBM-Handbuch-Verfahren: [Noth84]

Der Prototyp dieses Verfahrens wurde 1968 veröffentlicht und kann als die Ursprungsmethode dieser Gruppe angesehen werden.

Nach Abschluß des fachlichen Feinkonzeptes (vgl. Phase III, Kap 2.1) kann der Aufwand für das restliche Projekt geschätzt werden. Dabei werden bei der Schätzung der Umfang (T_{ea}), das Design (T_v), der Grad der Kenntnis des Problems (T_k) und die Erfahrung der Mitarbeiter (T_e) berücksichtigt.

Die Schätzformel für die Ermittlung des reinen Programmieraufwandes (vgl. Phase IV und V, Kap. 2.1) in MM (T_{pp}) lautet:

$$T_{pp} = (T_{ea} + T_v) * (T_k + T_e), \qquad \text{wobei}$$

- T_{ea}: Summe des Programmieraufwandes für Ein-/Ausgabeprozesse über die verschiedenen Komponenten,
- T_v: Summe des Aufwands für die Verarbeitung über die verschiedenen Komponenten,
- T_k: Problemkenntnisfaktor,
- T_e: Programmiererfahrungsfaktor.

Die einzelnen Faktoren werden anhand von Gewichtungspunkten aus verschiedenen Tabellen berechnet (siehe Anhang B.4)

Hat man nun den Aufwand für die reine Programmiertätigkeit ermittelt, so sieht das IBM-Handbuch-Verfahren einen Aufschlag von 70% bis 110% für die Systemanalyse (Phasen I und II), den Systementwurf (Phase III) und für die Einführung (Phase VI) vor. Zusätzlich kommt ein Aufschlag von 5% bis 10% für das Projektmanagement und für Schulungen und Krankheit, also durch das Umfeld verursachten Aufwand, noch einmal 20% bis 35%. Damit liegt der Gesamtaufwand des Projektes zwischen $T_{pp} * 1.95$ und $T_{pp} * 2.55$.

Aufgrund der Entwicklung in der Softwareentwicklung werden bei dem IBM-Handbuch-Verfahren Punkte berücksichtigt, die zum jetzigen Zeitpunkt nicht mehr die Bedeutung haben, welche sie 1968 noch hatten. Zwar legte diese Methode den Grundstein für weitere Entwicklungen, ist aber in dieser Form nicht mehr anwendbar und daher im Rahmen einer Bewertung ohne Bedeutung.

(b) Verfahren von End, Gotthard und Winkelmann (EGW): [Noth84]

Während viele Methoden nur den reinen technischen Entwurf und somit nur den Aufwand der Realisierung dieses Entwurfes konkret beschreiben, versucht diese Methode zur Aufwandsabschätzung bereits 1977, neben dem reinen DV-Aufwand der Realisierung ebenfalls den fachlichen Aufwand zu erfassen.

Der fachliche Aufwand (A_F in MM) wird anhand folgender Formel ermittelt:

$$A_F = FU * FT * FE, \quad \text{wobei}$$

FU: anhand einer Kurve kann aus der Anzahl der fachlichen Funktionen bzw. Aufgaben, wie z.B. Aufträge bearbeiten und Aufträge ausliefern (= 2) dieser Faktor ermittelt werden,

FT: Bewertungssumme, die den Umfang und die Schwierigkeiten der fachlichen und organisatorischen Tätigkeiten darstellt und

FE: Programmiererfahrungsfaktor.

Die Berechnung des DV-Aufwands (A_D in MM) erfolgt analog:

$$A_D = DU * DT * DE * DO * DW * DS, \quad \text{wobei}$$

DU: (Programmumfang) wird in Abhängigkeit von der zu erwartenden Anzahl von Befehlen (vgl. Anhang B.3) anhand einer Kurve ermittelt,

DT: welches den Schwierigkeitsgrad der Programmiertätigkeiten widerspiegelt, kann durch die Addition einzelner Werte aus einer Tabelle errechnet werden,

DE: ist der Faktor der Programmiererfahrung (liegt zwischen 0.8 und 2.4),

DO: steht für die Schwierigkeiten in der DV-Abwicklung (Spanne 0.8 bis 1.3),

DW: berücksichtigt die Wiederverwendbarkeit der Programme und variiert von 1.0 bis zu 1.2 und

DS: stellt den Einfluß der verwendeten Programmiersprache dar und ist:

- 0.85 - Assembler und
- 1.0 - Cobol oder Fortran.

Leider gilt für diese Methode das Gleiche wie für die in (a) genannte: auch wenn diese als erste nicht nur auf den Programmieraufwand eingeht, sondern auch das Umfeld berücksichtigt, so ist diese, aufgrund des Alters, weder verwendbar noch für eine Bewertung interessant.

(c) Function-Point-Verfahren: [Noth84] / [Litke96]

Diese, von Allan Albrecht bei der IBM Corporation entwickelte Methode, wird seit 1981 bei der IBM Deutschland GmbH eingesetzt. Es ist, neben der Weiterentwicklung Data-Point (vgl. (d)), das bekannteste Verfahren dieser Gruppe.

Die Grundidee ist, daß der Aufwand eines Projektes vom Schwierigkeitsgrad und vom Umfang abhängt. Diese werden durch die Summe der sogenannten ‘Function-Points’ (kurz: FP) dargestellt.

Die FP-Zuordnungen untereinander spiegeln den unterschiedlichen Umfang des Aufwandes für die Realisierung wieder.

Zunächst muß das Projekt in Funktionen bzw. Geschäftsvorfälle (z.B. Anlegen eines neuen Kunden) zerlegt werden.
Für diese sind folgende Größen, die nach dem Abschluß der Phase III (vgl. Kap 2.1) bekannt sind, zu ermitteln:

- Eingabedaten bzw. externe Inputs:

  jede Art von Eingabe, sei es durch direkte Eingabe durch den Anwender, durch Einlesen vom Datenträger oder ähnlichem, sowie durch Transaktion von einer anderen Anwendung aus,

- Ausgabedaten bzw. externe Outputs:

  alle, durch die Verarbeitung der Eingabe entstandenen Daten eines Geschäftsvorfalls, egal ob auf dem Bildschirm, in eine Datei oder als Transaktion an Drucker oder andere Anwendungen,

- Datenbestände bzw. interne Dateien:

  jede unterschiedliche, maschinenlesbare Datei oder bei Datenbanken, jede aus Anwendersicht logische Datengruppe, die erstellt, benutzt oder verändert wird (Entität),

- Datenbestände für andere Verfahren (Interfaces) bzw. externe Schnittstellen:

  Datenbestände für andere Anwendungen, deren Datensätze keine Transaktionen sind und

- externe Abfragen:

  Erzeugt eine Online-Eingabe eine Online-Ausgabe und dienen die Eingaben nur zu Steuerzwecken des Suchvorgangs, d.h. es werden keine Datenbestände verändert, so liegt eine Abfrage vor.

Diese werden dann anhand der drei Kriterien ‘leicht’, ‘mittel’ und ‘komplex’ klassifiziert. Dadurch wird jeder in der Software zu realisierenden Funktion ein Wert von drei bis fünfzehn zugeordnet. Summiert man alle diese Werte auf, so erhält man als Summe die (Roh-) FP.

Diese werden dann dem Entwicklungskontext zugeordnet, indem man vierzehn Faktoren entsprechend ihrer Einwirkung mit Zahlen bewertet.
Dieser sog. ‘Degree of Influence’, welcher zwischen 0 und 70 liegt, wird dann durch 100 geteilt und zu 0.65 addiert. Somit erhält man einen Multiplikator zwischen 0.65 und 1.35, der dann die (Roh-) FP anpaßt.
Diese vierzehn Faktoren und die somit entstehende Modifikationsspanne basiert auf den empirischen Untersuchungsergebnissen von Albrecht und umfassen ebenfalls ansatzweise Qualitätsmerkmale.

Anhand einer, durch die Daten bereits realisierter Projekte entstandener Kurve, werden dann die FP in den Aufwand in MM überführt. Somit wird die Produktivität des Entwicklerteams, auf dessen Daten diese Kurve basiert, ebenfalls berücksichtigt.
Die in Abbildung 3-1 gezeigte Kurve stellt die Ur-Kurve von IBM dar, die aufgrund der dort abgeschlossenen Projekte ermittelt wurde.
Diese kann auch als Grundlage dienen, die dann durch unternehmensspezifische Daten iterativ angepasst wird.

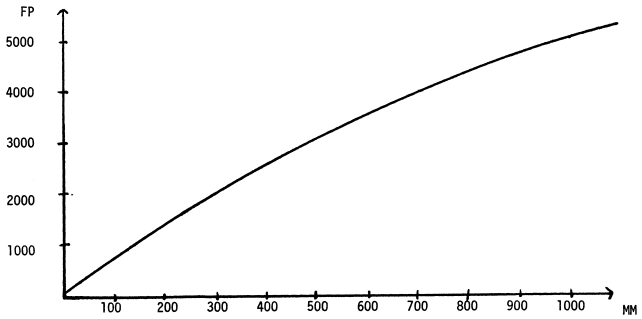

Abbildung 3-1 : Kurve zur FP-Aufwand-Transformation

In Anhang B.4 (h) sind noch zwei weitere Variationen bzw. Weiterentwicklungen dieser Methode genannt. Da dieser Methodenkomplex den Systementwurf, welcher bei einer objektorientierten Systementwicklung eine tragende Rolle spielt, als Grundlage für die Aufwandsabschätzung benutzt, wird er in die Bewertung einfließen.

(d) Data-Point-Verfahren: [Litke96]

Dieses 1991 von Sneed entwickelte Verfahren ist eine Weiterentwicklung der Function-Point-Methode (vgl. (c)).
Dadurch, daß für viele moderne Softwareentwicklungen die Verwaltung von Daten in einer Datenbank im Vordergrund steht, entstand die Annahme, daß die Komplexität eines solchen Programmes, maßgeblich von der Modellierung des Datenmodells abhängt.

Zur Berechnung der Data-Points (kurz: DP) werden also nicht die Geschäftsvorfälle betrachtet, sondern die Informationsentitäten (logische Sätze und Tabellen der Zieldatenbank, auf die das System zugreifen soll) und Nachrichten (Bildschirmmasken, Berichte, Datenübergaben und Messages an andere Programme).
Die Informationsentitäten ergeben sich aus der Daten-, die Nachrichten aus der Kommunikationsanalyse. Vorteil hieraus ist, daß diese Methode früher einsetzbar ist als die Function-Point-Methode, da ein Datenmodell meistens bereits nach der Anforderungsanalyse vorliegt.

Die Zuordnung der DP (analog zu FP) geschieht anhand von Bewertungen, die in einer Auflistung erfolgt.
Für die Informationstentitäten sieht diese wie folgt aus:

Name	Anz. Attribute	Anz. Schlüssel	Integrationsgrad	Nutzung	DP

Analog dazu werden die Nachrichten wie folgt klassifiziert:

Name	Anz. Felder	Anz. Sichten	Komplexitätsgrad	Nutzung	DP

Die Summe aller DP aus Informationsentitäten und Nachrichten wird nun mit einem Projektumgebungsfaktor multipliziert, so daß die DP in Abhängigkeit des Entwicklungskontextes um bis zu 15% erhöht oder um bis zu 25% gesenkt werden können.
Die Projektumgebungsfaktoren sind folgende:

- Projektverteilung
- Projekterfahrung
- Projektkenntnisse
- Projektautomation
- Rechenbedingungen
- Projektunterstützung
- Qualitätssicherung
- Spezifikationsformalismen
- Programmiersprache
- Testautomation

Als letzter Faktor kommt ein Qualitätsfaktor hinzu, der zwischen 0.5 und 1.5 liegt. Berücksichtigt werden hierbei:

- Funktionalität
- Zuverlässigkeit
- Benutzbarkeit
- Effizienz
- Änderbarkeit
- Übertragbarkeit

Anschließend werden die somit bewerteten DP (analog zu den FP) durch die nebenstehende Erfahrungskurve in den Aufwand in MM überführt.

Da es sich bei der Data-Point-Methode um eine neuere Form der FP-Methode und darüber hinaus auch noch um die neueste Methode aller innerhalb dieser Arbeit genannten Verfahren handelt, darf sie in der Bewertung nicht fehlen.

3.2.4 Methoden basierend auf teilprodukt-spezifischen Kenngrößen

(a) Integriertes Verfahren zur Aufwandsabschätzung (INVAS): [Noth84]

Die Idee, die 1980 Mitarbeiter der Universität Köln in Zusammenarbeit der Siemens AG zu der Entwicklung von INVAS antrieb, war, daß die aussagekräftigen Größen nicht für das gesamte Projekt gleich bleiben, sondern daß diese vielmehr von der entsprechenden Phase abhängen.

Um INVAS anwenden zu können, muß das Softwareprojekt in 'Meilenstein'-Ergebnisse bzw. Teilprodukte zerlegt werden.
Kernstück bei der Ermittlung des Aufwands ist eine Erfahrungsdatenbank, die zu jeder Maßgröße die für das Unternehmen spezifische Relation zwischen Einheit (z.B. Anzahl DIN A4-Seiten, Anzahl Diagramme etc.) und Aufwand in MM für jedes Projekt gespeichert wird. Ferner werden hier auch die zu berücksichtigen Einflußfaktoren vermerkt, die ebenfalls über das gesamte Projekt variieren können.

Wird nun für ein Teilprodukt der Aufwand gesucht, so wird das aktuelle Teilprodukt mit allen Teilprodukten in der Erfahrungsdatenbank hinsichtlich der äußeren Umstände (Einflußfaktoren) und der Ausprägung der Maßgrößen durchsucht.
Hierbei hängt es letztendlich vom Schätzenden ab, in wie weit das aktuelle Teilprojekt mit bereits realisierten Teilprojekten übereinstimmt. So kann die Ermittlung des Aufwandes auch durch Mittelwertbildung geschehen.

Auch wenn der Ansatz von INVAS interessant für einen Vergleich wäre, ist dieses im Rahmen dieser Arbeit nicht möglich, da hierzu - aufgrund des Aufbaus des Verfahrens - die notwendigen Daten fehlen.

(b) Object-Point-Verfahren: [Litke96]

Ein Verfahren, welches versucht, die Universialität von INVAS mit dem diskreten Vorgehen der Function-Point-Methode zu verbinden, ist dieses, von der Software AG entwikkelte Verfahren.
Es basiert auf definierten Projektergebnissen zu diskreten Schätzzeitpunkten, die mit den 'Meilenstein'-Ergebnissen identisch sein können, aber nicht müssen.

Zu einem Schätzzeitpunkt *X*, an dem ein definiertes Projektergebnis vorliegt, wird ein Schätzobjekt (daher: Object-Point, kurz: OP) erstellt. Definiert wird dieses durch Schätzobjekttypen, das sind Kenngrößen, die für den weiteren Projektverlauf aussagekräftig sind. In der Regel sind dies: Maske, Druckausgabe, Informationsobjekt, Datentransfer und Verarbeitungsfunktion.
Die Anforderungen werden direkt und einzeln bewertet. Ergebnis dieser Bewertung sind OP für das entsprechende Schätzobjekt.

Anschließend werden diese OP durch bewertete Einflußfaktoren modifiziert:

- unbekannte Systemumgebung
- Betreten von technischem Neuland beim technischen Entwurf und/oder bei der Realisierung
- Betreten von fachlichem Neuland
- spezielle Zugriffsmechanismen / Security
- Ausprägung der Endbenutzerhilfen
- Ausprägung RZ-Hilfen
- Fremd- oder Mehrsprachigkeit
- erschwerte Testanforderung
- umfangreiches Berichtswesen
- Beteiligung mehrerer Fachbereiche
- Anzahl der Installationen

Zuletzt werden die OP anhand einer Erfahrungskurve in den Aufwand in MM überführt (vgl. Function-Point-Methode, B.4 (h)).

Empfohlen wird die Anwendung der Object-Point-Methode erst nach Beendigung der Phase III (Funktionsbeschreibung), da zu diesem Zeitpunkt Datenmodell, Funktionsbeschreibung, Beschreibung von Schnittstellen und ggf. ein erster Oberflächenprototyp zur Verfügung stehen. Diese Vorgehensweise basiert auf den Produkten der Software AG, sollte aber auch auf andere 4GL-Entwicklungsumgebungen anwendbar sein (vgl. [Litke96]).

Da die Angaben in [Litke96] zu ungenau sind und die Software AG sich bezüglich meiner schriftlichen Anfrage nicht geäußert hatte, kann diese Methode leider nicht in die Bewertung einfließen.

3.3 Methoden-Bewertung

In diesem Kapitel wird zuerst kurz vorgestellt, wie die Bewertung erfolgt ist. Anschließend werden die in Kapitel 3.2 ausgewählten Methoden (Prozentsatzmethode, Function-Point-Methode und Data-Point-Methode) einzeln bewertet. Die hierbei erzielten Ergebnisse werden dann in Kapitel 3.3.5 dann noch einmal gegenübergestellt und abschließend bewertet.

3.3.1 Rahmenbedingungen der Bewertung

Um die drei von mir ausgewählten Methoden zur Aufwandsabschätzung bewerten zu können, habe ich eine Auswahl von mir durchgeführter Software-Projekte (vorgestellt in Anhang A) im Nachhinein bewertet, als ob ich die Daten einer Anforderungsanalyse entnommen hätte. Der tatsächlich dabei erzielte Aufwand wurde hierbei nicht berücksichtigt.
Anschließend habe ich den ex-post ermittelten Aufwand gem. der angewendeten Methoden mit dem realen Aufwand verglichen, um so die Schätzgenauigkeit zu überprüfen.
Durch das praktische Anwenden dieser Methoden konnte ich die unbedingten Voraussetzungen für die Anwendung dieser Methoden an den Eingangsgrößen für die Abschätzung festmachen.
Die Nachvollziehbarkeit des Ergebnisses bzw. des Vorgangs der Schätzung konnte ich hierdurch ebenfalls betrachten. Der geneigte Leser wird dazu an den Anhang C verwiesen, in dem alle durchgeführten Rechnungen ausführlich und nachvollziehbar dargestellt werden.
Als weiteres Kriterium sollte die Berücksichtigung von Qualitätsmerkmalen untersucht werden. Hierzu habe ich überprüft, welche der von mir in Kapitel 2.2 - Software-Qualität / -Qualitätssicherung - vorgestellten Qualitätsmerkmale direkt oder indirekt berücksichtigt wurden.

Zuerst werden die von mir gemachten Erkenntnisse vorgestellt und abschließend mit einer umgangssprachlichen, den Schulnoten angelehnte Bewertung versehen. Damit soll ein Vergleich der drei ausgewählten Methoden erleichtert werden und es ermöglichen, Vor- und Nachteile für individuelle Einsatzbereiche abzuwägen. Nur so kann eine Abschätzung des zu erwartenden Aufwands optimal erfolgen.

3.3.2 Bewertung der Prozentsatzmethode

Um diese Methode anwenden zu können, benötigt man möglichst genaue Aussagen über mind. eine Phase. Betrachtet werden hierbei meist frühe Phasen (bis Phase IV), um Aussagen über die restlichen Phasen und somit über den Gesamtaufwand treffen zu können.
Die Ausgangsdaten können geschätzt werden, was das Gesamtergebnis zusätzlich beeinträchtigt. Am besten ist es, eine abgeschlossene Phase zu betrachten und mit diesen Daten den restlichen Aufwand zu bestimmen. Bei diesem Vergleich nehme ich für die eine abgeschlossene Phase reale Daten des Projektes und ermittle die fehlenden Werte. Anschließend werden diese mit den realen Werten auf ihre Übereinstimmung überprüft.
Zum Vergleich der prozentualen Verteilungen der drei Projekte Alpha, Beta und Gamma dient die in Kapitel 3.2.1 vorgestellte Verteilung von Stahlknecht (vgl. [TU-B98]).

Die Abschätzung erfolgt zuerst unter der Annahme, die Phasen I bis III wären bereits abgeschlossen worden und der restliche Aufwand soll nun geschätzt werden. Die Werte in den Klammern dienen nur des Vergleiches, zur Berechnung wurden sie nicht herangezogen.

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
Vergleichsverteilung von Stahlknecht	5 %	10 %	10 %	30 %	35 %	10 %	100 %
Alpha, real	0 MH	0 MH	2 MH	(27 MH)	(61 MH)	(14 MH)	(104 MH)
Alpha, berechnet	-	-	-	6 MH	7 MH	2 MH	17 MH
Alpha, Abweichung	-	-	-	77,7 %	88,5 %	85,7 %	83,7 %
Alpha, Ø-Abweichung							*83,9 %*
Beta, real	0 MH	0 MH	3 MH	(5 MH)	(23 MH)	(18,5 MH)	(49,5 MH)
Beta, berechnet	-	-	-	9 MH	10,5 MH	3 MH	25,5 MH
Beta, Abweichung	-	-	-	80 %	54,3 %	83,8 %	48,5 %
Beta, Ø-Abweichung							*66,7 %*
Gamma, real	0 MH	0 MH	1 MH	(4 MH)	(35 MH)	(2 MH)	(41 MH)
Gamma, berechnet	-	-	-	3 MH	3,5 MH	1 MH	8,5 MH
Gamma, Abweichung	-	-	-	25 %	90 %	50 %	79,3 %
Gamma, Ø-Abw.							*61,1 %*

Zum Vergleich, ob diese Methode später eingesetzt werden sollte, um die Schätzergebnisse zu verbessern, wurde die Rechnung erneut durchgeführt. Als Basis dienten nun die Daten der Phase IV:

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
Vergleichsverteilung von Stahlknecht	5 %	10 %	10 %	30 %	35 %	10 %	100 %
Alpha, real	0 MH	0 MH	2 MH	27 MH	(61 MH)	(14 MH)	(104 MH)
Alpha, berechnet	-	-	-	-	31,5 MH	9 MH	86,5 MH
Alpha, Abweichung	-	-	-	-	48,4 %	33,3 %	16,8 %
Alpha, Ø-Abweichung							*32,8 %*
Beta, real	0 MH	0 MH	3 MH	5 MH	(23 MH)	(18,5 MH)	(49,5 MH)
Beta, berechnet	-	-	-	-	5,8 MH	1,7 MH	15,5 MH
Beta, Abweichung	-	-	-	-	74,8 %	90,8 %	68,7 %
Beta, Ø-Abweichung							*78,1 %*

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
Gamma, real	0 MH	0 MH	1 MH	4 MH	(35 MH)	(2 MH)	(41 MH)
Gamma, berechnet	-	-	-	-	4,7 MH	1,3 MH	11 MH
Gamma, Abweichung	-	-	-	-	86,6 %	35 %	73,2 %
Gamma, Ø-Abw.							*64,9 %*

Die zweite Berechnung konnte nur das Ergebnis von der Schätzung des Projektes Alpha verbessern; die Ergebnisse der beiden anderen verschlechterte sich. Daher kann keine Kausalität zwischen Genauigkeit und Zeitpunkt erkannt werden.

Ein weiterer Einflußfaktor der Aufwandsabschätzung mit dieser Methode ist die Verteilung, mit der geschätzt wird. Da die von Stahlknecht nicht ausreichend genaue Abschätzungen produziert, soll nun untersucht werden, ob sich die Schätzung verbessern läßt. Hierzu wird die Durchschnittsverteilung, basierend auf eigenen, realisierten Projekten, näher betrachtet. Abbildung 3-2 zeigt die Verteilungen der drei Referenz-Projekte, deren Durchschnitt und die Vergleichsverteilung von Stahlknecht. Auf der X-Achse sind die Phasen, auf der Y-Achse die Prozente abgetragen:

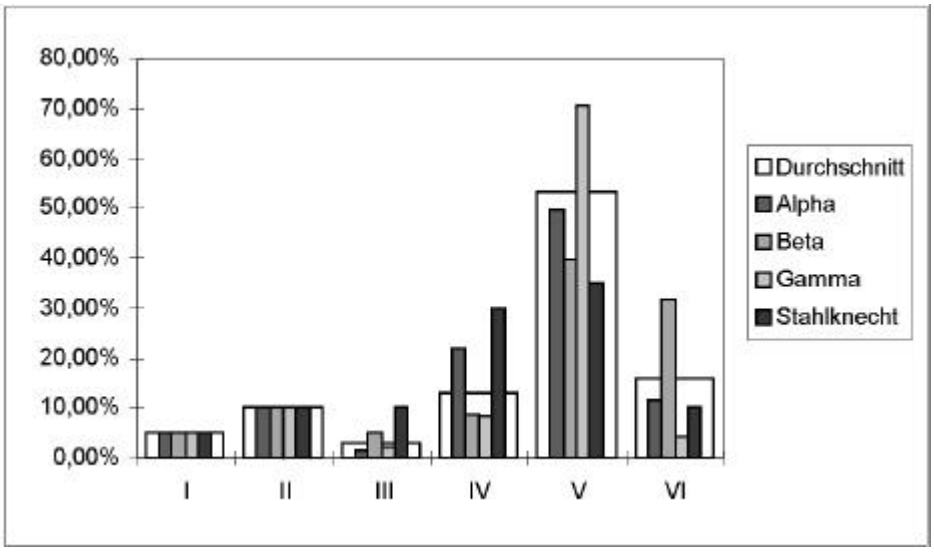

Abbildung 3-2 : Vergleich der proz. Verteilungen

Folgende Schlüsse lassen sich aus dieser Abbildung ziehen:

- Mit steigender Zahl von Projekten ergibt sich eine, dem entwicklungskontextabhängige Verteilung, die höchst wahrscheinlich nicht identisch ist mit der Vergleichsverteilung von Stahlknecht.
- Die durchschnittliche Abweichung der einzelnen Projekte von deren Durchschnitt betragen insgesamt: Alpha 4.55%, Beta 9.1% und Gamma 8.6%.
  Da die Abweichungen in den einzelnen Phasen noch höher liegt, ist die durchschnittliche Verteilung für genaue Prognosen nicht geeignet, wohl aber, um eine grobe Übersicht zu gewinnen.
- Die Verpflechtung mit den Endanwendern verschiebt durch die Variabilität das Verhältnis der Phasen III und IV zur Phase V.

- Die Einführungsphase (Phase VI) reduziert sich bei evolutionären und partizipativen Projekten meistens auf Datenübernahme und evtl. Installationen vor Ort. Daher ist diese Phase sehr variabel.

Aufgrund dieser Erkenntnisse bezeichne ich die Schätzgenauigkeit als ungenügend.

Um die Prozentsatzmethode einsetzen zu können, benötigt man eine dem Entwicklungskontext angepaßte Verteilung und möglichst genaue Aussagen über mind. eine Phase, sei es durch Abschluß dieser oder aber auch durch Schätzungen. Es liegt auf der Hand, daß die Anwendung dieser Methode basierend auf einer geschätzten Phase das Schätzresultat nicht gerade verbessert.
Daher benötigt diese Methode nur geringe ‘unbedingte Voraussetzungen’.

Wenn das Projektteam im Laufe der Zeit eine ihren Verhältnissen angeglichene Verteilung erstellt hat, ist das durch diese Methode erzielte Ergebnis für alle Beteiligten leicht nachvollziehbar. Auch die Anwendung des Verfahrens ist aufgrund der Einfachheit leicht zu überblicken.

Merkmale der Software-Qualität werden direkt keine berücksichtigt. Indirekt wird eine minimale, der durchschnittlichen Verteilung entsprechende Einwirkung der Sicherung dieser Merkmale berücksichtigt, da dieser Einfluß auf den Aufwand bereits während der Erstellung der spezifischen Verteilung mit einfließen.
Daher bewerte ich die Berücksichtigung von Qualitätsmerkmalen als ausreichend.

Fazit: Für Durchschnitts- und Grobbetrachtungen gut geeignet, genaue Abschätzungen für Kalkulationen oder Projektsteuerungen sind aber nicht möglich.

[Noth84] kommt auf Seite 64 zu einem etwas anderen Ergebnis: Ein solches Verfahren sei aufgrund fehlender Differenzierung nicht anwendbar, auch wenn eine solche prozentuale Verteilung über die Phasen hinweg aus der Praxiserfahrung heraus in etwa bestätigt werden kann. Leider kann diese Einschätzung in [Noth84] selbst nicht nachvollzogen werden.

3.3.3 Bewertung der Function-Point-Methode

Zur Aufwandsabschätzung mit der Function-Point-Methode benötigt man genaue Aussagen über die abzubildenden Funktionen, auch Geschäftsvorfälle oder Szenarien genannt. Diese erhält man in der Regel durch eine Anforderungsanalyse. Die für die drei Referenzprojekte verwendeten Aussagen werden in Anhang A aufgeführt.
Für diese Funktionen werden nun FP ermittelt, die eine Art Aufwandskennzahl darstellen. Anschließend werden diese FP durch Einflüsse des Entwicklungs- und Einsatzkontextes modifiziert ('Degree of Influence', *doi*). So erhält man die bewerteten FP, also die den Gegebenheiten angepaßte Aufwandskennzahl.
Da diese Kennzahl noch keine Aussagen über den zu erwartenden Aufwand in MM zuläßt, muß diese noch transformiert werden. Dieses geschieht durch eine Zuordnungskurve, in der die Erfahrungen mit bereits realisierten Projekten gespeichert ist. Durch Einpflechtung von neuen FP-Aufwandspaaren in die bestehende Kurve wirkt das eigene Umfeld auf die Transformation und verbessert das Ergebnis.

Während in Anhang C.2 die durchgeführten Rechnungen dargestellt sind, sollen die Ergebnisse hier kurz aufgeführt werden. Als Transformationskurve wurde die Ur-Kurve von IBM verwendet.

	Werte gem. Variante I:	Werte gem. Variante III:
Alpha, FP	150	148
Alpha, *doi*	0,92	0,73
Alpha, bewertete FP (= FP ∗ *doi*)	138	108
Alpha, Aufwand [MM]	9,5	9,3
Alpha, Abweichung von 1,01 MM	*89,4 %*	*89,1 %*
Beta, FP	102	98
Beta, *doi*	0,80	0,66
Beta, bewertete FP (= FP ∗ *doi*)	81,6	64,68
Beta, Aufwand [MM]	4,39	3,27
Beta, Abweichung von 0,49 MM	*88,8 %*	*85,0 %*
Gamma, FP	36	36
Gamma, *doi*	1,09	0,80
Gamma, bewertete FP (= FP ∗ *doi*)	39,24	28,8
Gamma, Aufwand [MM]	1,81	1,32
Gamma, Abweichung von 0,41 MM	*77,3 %*	*68,9 %*

Diese Ergebnisse zeigen, daß die Ur-Transformationskurve von IBM nicht ungeändert übernommen werden kann. Daher die Annahme, daß, wenn man eine eigene Transformationskurve mit den hier ermittelten FP-Aufwandspaaren erstellt, die Ergebnisse wesentlich verbessert werden können. Die eigene Transformationskurve muß allerdings die gleiche Charakteristik aufweisen, wie die Ur-Kurve von IBM, um anwendbar zu sein.

Dieses soll Abbildung 3-3 verdeutlichen. In dieser Abbildung wurden die eigenen, auf realen Daten basierenden Transformationskurven eingezeichnet. Dabei wurden die zwei Transformationskurven der beiden betrachteten Varianten der FP-Methode berücksichtigt, die als BFP, VI (bewertete FP, Variante I) und BFP, VIII bezeichnet werden. Auf der X-Achse wurde der Aufwand in MH (vgl. Kapitel 2.3) und auf der Y-Achse die BFP abgetragen. Die gestrichelte Fortführung der beiden Kurven stellt eine Prognose des weiteren Verlaufs, basierend auf der aus den Projektdaten entstandenen Kurven dar, um den Charakter der Kurven besser erkennen zu können.

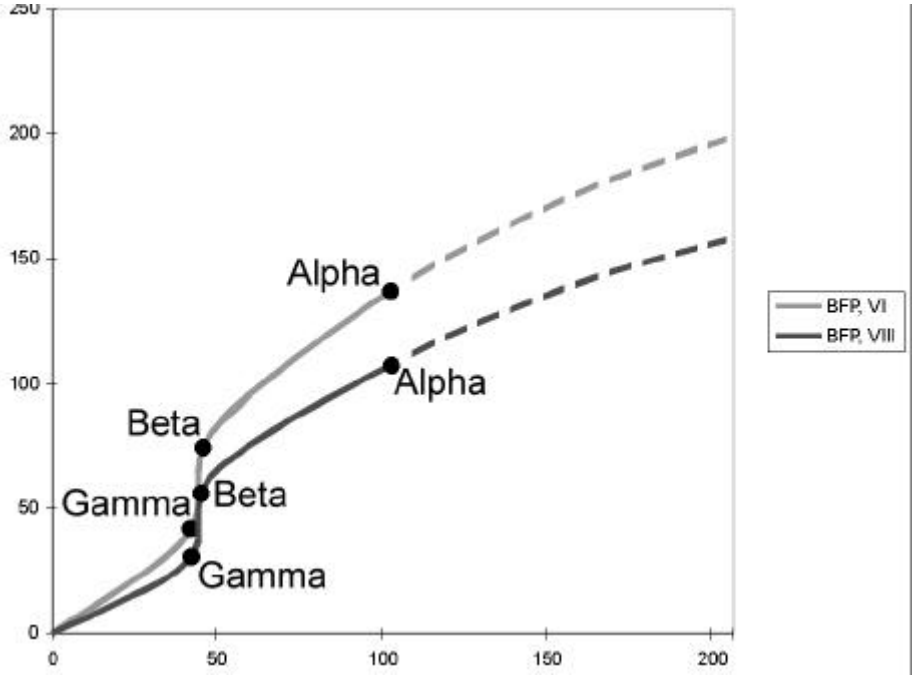

Abbildung 3-3 : FP-Transformationskurven

Folgende Schlüsse lassen sich nun aus dieser Abbildung ziehen:

- Beide Kurven ähneln sich in ihrem Verlauf und zeigen grob die gleiche Charakteristik wie die Vergleichskurve von IBM (vgl. Kapitel 3.2.3). Diese Urkurve (in obriger Abb. nicht dargestellt) kann aber, wie gezeigt, nicht unangepaßt verwendet werden, da die damit errechneten Ergebnisse erheblich von den realen Abweichen.
- Würde man den Verlauf der beiden Kurven der Vergleichskurve von IBM anpassen, so würde man ‘Gamma’ als Ausreißer bezeichnen. Dieses kann auch an dem Umstand liegen, daß das Projekt Gamma ein Teilprojekt eines größeren Projektes war. Daher nehme ich an, daß sich mit steigender Zahl von Projekten diese Kurve, die dann z.T. interpoliert werden würde, wahrscheinlich der Charakteristik der IBM-Kurve als Kurve mit stetig abnehmender Steigung annähern würde.
- Bei den großen Abweichungen meiner Erzielten BFP im Vergleich zu denen von IBM wird deutlich, daß die Kurve, nur dem Entwicklungskontext angepaßt, gute Schätzergebnisse erzielen kann. Das Verhältnis Aufwand zu BFP spiegelt die Produktivität des Entwicklerteams dar.

Zusammenfassend stufe ich die Schätzgenauigkeit der Function-Point-Methode als gut ein, eine angepaßte Transformationskurve vorausgesetzt.

Als unbedingte Voraussetzungen, um diese Methode anwenden zu können, wird eine Anforderungsanalyse mit Szenarien und einer Systemvision benötigt. Szenarien sind erforderlich, um die zu realisierenden Geschäftsprozesse zu definieren, auf denen dann die Abschätzung beruht. Hierbei ist es notwendig, möglichst konkret zu sein, da kleine Unsicherheiten das Schätzergebnis negativ beeinflussen. Die Systemvisionen werden benötigt, da auch z.T. Realisierungsdetails in die Bewertung einfließen.
Alles in allem stufe ich die Voraussetzungen für diese Methode als mittelmäßig ein, da diese weder hohe Ansprüche an das Datenmaterial stellt, noch der Zeitpunkt, zu dem diese Methode eingesetzt werden kann, sehr späten ist. Sollen anwenderspezifische Punkte, wie hier die Geschäftsvorfälle, berücksichtigt werden, so müssen diese erfaßt und konkret benannt worden sein, damit eine Abschätzung erst Sinn macht.

Das Verfahren an sich ist für denjenigen, der diese Methode anwendet, nachvollziehbar, da es sich sehr stark an den Szenarien orientiert. Das Manko, daß kein Unterschied bzgl. des Aufwands für Nur-Lese-Daten im Gegensatz zu Lese-und-Schreibdaten gemacht wird, wird durch die Variante III behoben. Einzig die Gewichtung der einzelnen Ausprägungen untereinander ist nicht trivial, aber sicherlich in endlicher Zeit nachvollziehbar.
Die Nachvollziehbarkeit des Schätzergebnissen ist zum ersten durch die Tatsache gegeben, daß Projekte mit ähnlichen BFP auch ähnliche Aufwandsergebnisse liefern und zum zweiten, daß die Kurve den Gegebenheiten innerhalb eines Entwicklungsteams ‘geformt’ wurde und somit die Arbeitsweise der Beteiligten widerspiegelt.
Daher erhält dieses Bewertungskriterium das Prädikat ‘gut’.

Die Berücksichtigung der Qualitätsmerkmale gliedert sich in zwei Teile: das Zeitverhalten wird bereits bei der Ermittlung der Roh-FP berücksichtigt, während Qualitätsmerkmale (hier: Wiederverwendung, Installationen und Anpaßbarkeit) im Katalog des ‘Degree of Influence’ berücksichtigt werden. Indirekt werden darüber hinaus keine weiteren Merkmale der Software-Qualität betrachtet.
Die Einschätzung dieser Merkmale ist durch die verfahrenseigene Kategorisierung einfach gehalten. Lediglich die Auswahl ist strittig: mit Wiederverwendung und Anpaßbarkeit sind zwar für eine fortlaufende Weiterentwicklung wichtige Merkmale berücksichtig worden, aber andere, in meinen Augen ebenfalls sehr wichtige, wie z.B. Betriebssicherheit und Korrektheit, fehlen.
Daher muß die Berücksichtigung von Qualitätsmerkmalen bei den vor mir vorgestellten Varianten der Function-Point-Methode, in Anlehnung an die Einschätzung der Prozentsatzmethode (vgl. Kapitel 3.3.2) als befriedigend bezeichnet werden.

Fazit: Das Verfahren an sich und das damit erzielte Ergebnis sprechen für die Anwendung dieser Methode, auch wenn beides sicherlich durch Verfahrensmodifikationen verbessert werden könnte. Um das Schätzergebnis zu verbessern, schlage ich vor, die Transformationskurve für jedes Team, jede Entwicklungsumgebung und Programmiersprache, sowie an die Projektstruktur (vgl. Kapitel 2.1) anzupassen. Evtl. ist eine Anpassung an die Branche der Auftraggeber hilfreich, da so die durch fortlaufende Projekte erworbenen Sachkenntnisse mit berücksichtigt werden.

[Noth84] kommt auf den Seiten 101ff. zu folgenden Ergebnissen: Die IBM-Kurve kann nicht unangepaßt verwendet werden, während die Gewichtung der einzelnen Geschäftsvorfälle durch FP den relativen Aufwandsanteil zum Gesamtprojekt realistisch wiedergibt.

Es sei, so die Autoren, leicht anwendbar und berücksichtige alle relevanten Einflüsse auf das Projekt. Daher kann es für eine Vielzahl von verschiedenen Projekten angewendet werden.
Diese Methode wird in [Noth84] als für den praktischen Einsatz empfehlenswert, aber dennoch durch Modifikationen verbesserbar eingestuft.
Dieser Teil der Bewertung wird ausführlich beschrieben, so daß die meisten Testergebnisse zur FP-Methode nachvollzogen werden können. Einziger Verbesserungsvorschlag der Autoren ist die Anpassung der Transformationskurve.

3.3.4 Bewertung der Data-Point-Methode

Die Data-Point-Methode ist von ihrer Anwendung der Function-Point-Methode ähnlich. Hierbei benötigt man genaue Aussagen über das abzubildende Datenmodell. Dieses erhält man meistens frühzeitig während einer Anforderungsanalyse. Die Datenmodelle der drei Referenzprojekte werden in Anhang A aufgeführt.
Für die Entitäten und die zugehörigen Masken werden nun DP ermittelt, die eine Art Aufwandskennzahl darstellen. Anschließend werden diese DP durch Einflüsse des Entwicklungs- und Einsatzkontextes ('Degree of Influence', *doi*) und durch die Einflüsse von Software-Qualitätsmerkmalen ('Quality Factor', *qf*) modifiziert. So erhält man die bewerteten DP, also die den Gegebenheiten angepaßte Aufwandskennzahl.
Da diese Kennzahl noch keine Aussagen über den zu erwartenden Aufwand in MM zuläßt, muß diese noch transformiert werden. Dieses geschieht durch eine Zuordnungskurve, in der die Erfahrungen mit bereits realisierten Projekten gespeichert ist. Durch Einpflechtung von neuen DP-Aufwandspaaren in die bestehende Kurve wirkt das eigene Umfeld auf die Transformation und verbessert das Ergebnis.

Während in Anhang C.3 die durchgeführten Rechnungen dargestellt sind, sollen die Ergebnisse hier kurz aufgeführt werden. Als Transformation dient die Kurve von Sneed.

	Alpha	Beta	Gamma
Data-Points (DP)	208,9	323,4	29,7
'Degree of Influence' (*doi*)	0,97	0,93	0,94
'Quality Factor' (*qf*)	0,93	0,95	1,01
BDP	188,54	291,0	18,7
Aufwand, berechnet [MM]	4,7	7,3	0,47
Aufwand, real [MM]	1,01	0,49	0,41
Abweichung	78,5 %	93,3 %	12,8 %

Diese Ergebnisse zeigen, daß zum einen das Verhältnis Komplexität Datenmodell zu Funktionalität eine große Rolle spielt und zum anderen die Transformationskurve den eigenen Gegebenheiten angepaßt werden muß. Die eigene Transformationskurve muß hierzu allerdings die gleiche Charakteristik aufweisen, wie die Ur-Kurve von Sneed.

Um die Form der angepaßten Transformationskurve einschätzen zu können, wurde in Abbildung 3-4 die durch die Daten der Referenzprojekte beschriebene Kurve dargestellt (durchgezogene Linie). Die gestrichelte Linie spiegelt einen Verlauf wieder, unter der Annahme, daß das Projekt Beta ein 'Ausreißer' ist. Dieses ließe sich durch die Eigenart des Projektes erklären, da Beta ein komplexes Datenmodell mit sehr simpler Funktionalität besitzt.

Wie im vorherigen Abschnitt ist auch hier auf der X-Achse der reale Aufwand in MH und auf der Y-Achse die BDP abgetragen:

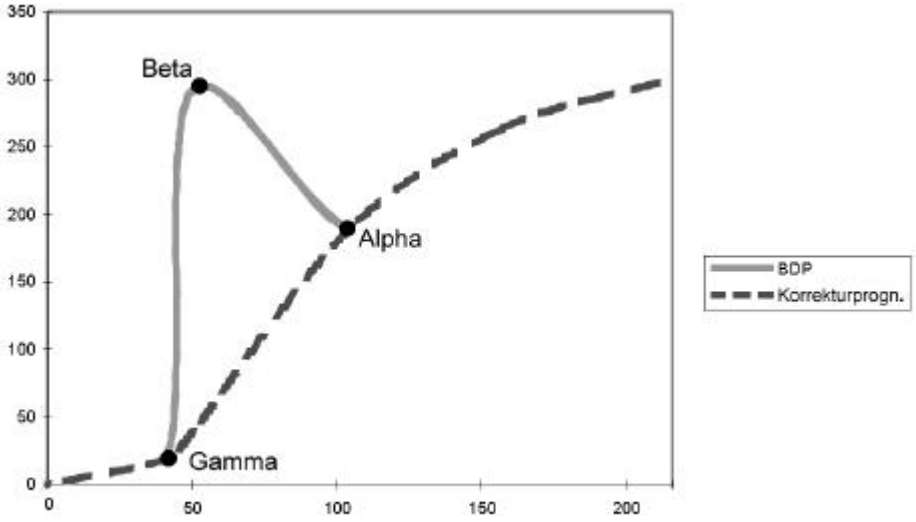

Abbildung 3-4 : BDP-Transformationskurve im Vergleich

Vergleicht man den realen Aufwand mit dem berechneten und betrachtet man die oben gezeigten Kurven, so lassen sich folgende Schlüsse daraus ziehen:

- Bei Datenbankanwendungen, die zwar ein nicht triviales Datenmodell besitzen (Alpha und Beta), aber deren Funktionalität sich weitestgehend auf die Datenerfassung beschränkt, weicht das Schätzergebnis, basierend auf der Sneed-Transformationskurve (in oberer Abb. nicht dargestellt), signifikant vom realen Aufwand ab:
  Bei Alpha (berechnet zu real) ist das 479,4 MH zu 104 MH und bei Beta ist dieses Verhältnis sogar 744,6 MH zu 49,5 MH.
  Lediglich das Teil-Projekt Gamma, bei der das Datenmodell und die Funktionalität ähnlich komplex bzw. einfach sind, hat ein gutes Verhältnis vom berechneten zum realen Aufwand, nämlich 47,7 MH zu 41 MH.
  Bei dem Projekt ‘AiRMan’, bei dem ich ebenfalls in Teilen mitgewirkt habe, war das Verhältnis Datenmodell zu Funktionalität ähnlich wie bei Gamma. Die Schätzgenauigkeit bei diesem Projekt hat sich nach Abschluß ebenfalls als sehr nahe an der Wirklichkeit herausgestellt.
- Durch meine Prognose (gestrichelte Kurve in Abbildung 3-4), unter der Annahme, daß Beta durch sein schlechtes Verhältnis vom komplexen Datenmodell zur einfachen Funktionalität kein repräsentatives Ergebnis darstellt, zeigt, daß sich eine Kurve mit ähnlicher Charakteristik wie die Sneed-Kurve bilden ließe.
  Daher kann das Schätzergebnis durch eine Anpassung der Sneed-Ausgangskurve verbessert werden.

Zum Thema Schätzgenauigkeit läßt sich also sagen, daß diese durch eine Anpassung der Transformationskurve verbessert werden kann, während die Urkurve bereits für die Neuaufnahme von Aufwandsabschätzungen in das Projektmanagement, unter bestimmten Voraussetzungen angewandt, gute Ergebnisse erzielen kann.
Daher bezeichne ich die Schätzgenauigkeit der nicht modifizierten Data-Point-Methode als ausreichend, unter Berücksichtigung der möglichen Anpassung als gut. Aufgrund der Tatsache, daß das Schätzergebnis sehr stark vom Komplexitätsverhältnis Datenmodell zu Funktionalität abhängt, tendiere ich zu der Einschätzung befriedigend.

Wie bereits im vorherigen Absatz angemerkt, ist das vom Komplexitätsverhältnis Datenmodell zu Funktionalität sehr entscheidend. Daher ist eine Voraussetzung für die Anwendung der Methode ein angemessenes Komplexitätsverhältnis.
Aufgrund der Tatsache, daß das Datenmodell als Bewertungsgrundlage dient, ist ein fertiges Datenmodell ebenfalls Voraussetzung. Dieses wird meistens bei der Anforderungsanalyse gewonnen und die Methode kann nach Erstellen der Szenarien, noch vor der ersten Systemvision, bereits angewendet werden.
Zusammenfassend schätze ich die unbedingten Voraussetzungen für das Anwenden der Data-Point-Methode als mittel bis hoch ein.

Aufgrund der Systematik dieser Methode ist deren Anwendung leicht überschaubar und somit auch leicht nachvollziehbar. Das Schätzergebnis selbst ist, ähnlich der FP-Methode (vgl. Kapitel 3.3.3), durch die Tatsache nachvollziehbar, daß Projekte mit ähnlichen BFP auch ähnliche Aufwandsergebnisse liefern und daß die Kurve den Gegebenheiten innerhalb eines Entwicklungsteams angepaßt wurde und somit die Arbeitsweise der Beteiligten widerspiegelt.
Aus diesem Grund bewerte ich die Nachvollziehbarkeit dieser Methode, angelehnt an die Einschätzung der FP-Methode, als gut.

Ein Grund dafür, daß die Data-Point-Methode so bekannt wurde, war die Tatsache, daß Sneed als Vorreiter auf dem Gebiet der Software-Qualität (vgl. Kapitel 2.2) erstmals Qualitätsmerkmale direkt und in großen Umfang in seiner Methode berücksichtigt hat. Während viele Methoden die Projektbedingungen berücksichtigen, kommt bei ihm ein Qualitätsfaktor als weiterer Modifikator hinzu.
Zur Einschätzung dieser umfangreichen Sammlung von Merkmalen dienen vier Kategorien von ‘sehr wichtig’, über ‘wichtig’ und ‘normal’ bis ‘nicht relevant’. Daher bekommt das Kriterium ‘Berücksichtigung von Qualitätsmerkmalen’ die Bewertung sehr gut.

Fazit: Für Datenbank geprägte, aber nicht datenmodellastige Projekte gut geeignet und leicht anwendbar, aber ebenfalls sicherlich durch Verfahrensmodifikationen verbesserbar.

Da diese Methode in [Noth84] nicht erwähnt wird, werden in diesem Buch natürlich auch keine Bewertungsaussagen hierüber getroffen.

3.3.5 Zusammenfassung der Bewertung

Nachdem nun die drei Methoden zuvor einzeln bewertet wurden, soll an dieser Stelle eine Zusammenfassung bzw. eine Gegenüberstellung der Ergebnisse erfolgen. Anschließend werden die einzelnen Bewertungsergebnisse als Gesamtheit betrachtet, um hieraus weitere Schlüsse ziehen zu können.

	Prozentsatzmethode	Function-Point-Methode	Data-Point Methode
Schätzgenauigkeit	*ungenügend*, da die prozentuale Verteilung einzelner Projekte von der Durchschnittsverteilung zu stark variiert	*gut*, da die Transformationskurve an die entwicklungsspezifischen Gegebenheiten ständig angepaßt und somit verbessert werden kann	*befriedigend*, da das Ergebnis sehr stark von der Beschaffenheit der Applikation abhängt (siehe Voraussetzungen)
unbedingte Voraussetzungen	*gering*, da nur eine kontextangepaßte Verteilung und Aufwandsaussagen über mind. eine Phase benötigt werden	*mittel*, da, neben einer angepaßten Transformation, die Anforderungsanalyse (Szenarien) und die Systemvision benötigt werden	*mittel bis hoch*, da, neben einer angepaßten Transformation, das Datenmodell benötigt wird. Außerdem muß ein angemessenes Verhältnis zwischen Datenmodellkomplexität und Funktionalität gegeben sein
Nachvollziehbarkeit	*leicht nachvollziehbar*, da, eine korrekte Verteilung vorausgesetzt, das Verfahren leicht anwendbar ist	*gut nachvollziehbar*, da die einzelnen aufwandsbestimmenden Faktoren der Szenarien übersichtlich sind und Projekte mit ähnlicher FP aufgrund der kontextabh. Transformation ähnliche Abschätzungsergebnisse liefern	*gut nachvollziehbar*, da die einzelnen aufwandsbestimmenden Faktoren des Datenmodells übersichtlich sind und Projekte mit ähnlicher DP aufgrund der kontextabh. Transformation ähnliche Abschätzungsergebnisse liefern
Berücksichtigung von Qualitätsmerkmalen	*ausreichend*, da nur indirekt Merkmale durch die Kontextanpassung der Verteilung berücksichtigt werden	*befriedigend*, da nur vier Q.-Merkmale indirekt berücksichtig werden und die Auswahl dieser vier verbesserungsfähig ist	*sehr gut*, da die Modifikation der DP-Methode sehr viele Merkmale berücksichtig und diese leicht bewertbar sind
Fazit	Nur für Durchschnitts- und Grobabschätzungen zu gebrauchen	Bei Anpassung der Transformationskurve separat für spezifische Projektbedingungen gute Ergebnisse, die aber sicherlich durch Verfahrensmodifikationen verbessert werden können	Für Datenbank-Applikationen gut geeignet, wenn die Transformation BDP -> Aufwand durch Anpassung an den Entwicklungskontext verbessert wird. Trotzdem aber sicherlich durch Verfahrensmodifikationen verbesserungsfähig

Tabelle 3-3 : Übersicht der Bewertungsergebnisse

In Kapitel 2.1 - Modelle zur Projektstrukturierung - wurde gezeigt, daß Aufwandsabschätzungen der Etat-Vorkalkulation, sowie der Projektkontrolle und -steuerung dienen. Da der Vergleich der Methoden zur Aufwandsabschätzung gezeigt hat, daß die Prozentsatzmethode hierfür ungeeignet ist, stellt dieses Verfahren meines Erachtens zwar einen interessanten Ansatz dar, ist aber für den praxisbezogenen Einsatz nicht zu gebrauchen.

Jeder der beiden verbleibenden Testkandidaten hat Vor- und Nachteile gegenüber dem anderen. Die Bewertung hat gezeigt, daß es die einzige, universell einsetzbare Methode mit sehr guten Schätzergebnissen nicht gibt.
Ferner dürfte das Unterscheidungsmerkmal ‘Schwerpunkt Datenbank’ ebenfalls nicht allgemeingültig sein, so daß eine Empfehlung, wann welche Methode am besten zu verwenden ist, nicht uneingeschränkt ausgesprochen werden kann.
Mögliche Entscheidungshilfe könnte sein, welche Daten zum Zeitpunkt der Abschätzung vorhanden oder leichter zu ermitteln sind.
Ich empfand das Abschätzverfahren der Data-Point-Methode als leichter durchzuführen, als das der Function-Point-Methode. Dieser Vorteil wird aber durch die höheren Voraussetzungen und dem dadurch entstehenden, schlechteren Ergebnis wieder nivelliert. Die Auswahl des bevorzugten Verfahrens allein von dem Grad der Berücksichtigung von Qualitätsmerkmalen anhängig zu machen, scheint mir ebenfalls nicht sehr sinnvoll.

Der von mir durchgeführte Vergleich hat aber gezeigt, daß Methoden den besten Ansatz für eine erfolgreiche Abschätzung des zu erwartenden Aufwandes besitzen, die einzelne Design-Einheiten, wie z.B. Entitäten oder Szenarien, als Schätzgrundlage besitzen. Daher denke ich, daß eine Verbesserung der einzelnen Methoden durch eine Anpassung der einzelnen, zu gewichtenden Aspekte, sowie der beeinflussenden Projektbedingungen an die Vorgehensweisen der modernen Software-Entwicklung erreicht werden kann.

4 Zusammenfassung

Ziel dieser Arbeit war es, Methoden zur Aufwandsabschätzung bei Softwareprojekten vorzustellen und zu vergleichen. Der Vergleich sollte hierbei auf die Bewertung der vier Kriterien Schätzgenauigkeit, unbedingte Voraussetzungen, Nachvollziehbarkeit und die Berücksichtigung von Software-Qualitätsmerkmalen beschränkt werden.

Bereits bei den Recherchen zeigte sich, daß die methodengestützte Abschätzung des zu erwartenden Aufwands, nur mäßig verbreitet, angewendet wird. Dieses spiegelt sich ebenfalls in der Anzahl der gefundenen Publikationen, sowie in deren Alter wider.
Die praxisbezogene Verzahnung zwischen Projektmanagement und Aufwandsabschätzung wird durch den Umstand verdeutlicht, daß viele Grundlagen des Projektmanagements benötigt werden, um das Thema “methodengestützte Abschätzungen“ verstehen zu können. Es gibt jedoch auch Autoren, die versuchen, das eine Thema ohne das andere zu verarbeiten.
Durch die Verwendung von Methoden zur Aufwandsabschätzung kann eine Fehleinschätzung bzw. ein Überschreiten des erwarteten Aufwandes nicht ausgeschlossen werden; allerdings werden Schätzfehler leichter erkannt und das Schätzverfahren läßt sich somit kontinuierlich verbessern.

Obwohl es für den Umfang dieser Arbeit besser war, daß ich nur drei Methoden zur Aufwandsabschätzung vergleichen konnte, hätte ich gern aus eigenem Interesse weitere Methoden ‘ausprobiert’, da einige Methoden gute Ansätze boten, aber dann im Ganzen nicht praktikabel waren.
Mein eigenes Interesse an dieser Arbeit war, abgesehen vom Hintergrund des Studiums, vielleicht die eine oder andere Methode für die praxisbezogene Anwendung innerhalb eigener Projekte finden zu können. Der Vergleich hat gezeigt, daß es keine Methode gibt, die allein allen Ansprüchen gerecht wird. Dadurch ergeben sich weitere Möglichkeiten für zukünftige Vergleiche dieser Art.
Ebenfalls habe ich dabei festgestellt, daß die wahre Qualität einer Schätzmethode erst bei der praktischen Anwendung erkennbar ist, da hierbei Unzulänglichkeiten in der Verfahrensbeschreibung aufgetaucht sind, die ich zu beheben versucht habe.
Sicherlich wäre die Untersuchung der Methoden auf die Erfüllung weiterer Kriterien interessant gewesen, aber es mußten Kriterien ausgewählt werden, um den Rahmen einer Studienarbeit nicht zu überschreiten.

Mit dem Ergebnis dieses Vergleichs wurde die zentrale Frage nach der Leistungsfähigkeit der drei betrachteten Methoden zur Aufwandsabschätzung beantwortet. Jede Methode hat ihre Stärken, aber auch ihre Schwächen, so daß keine als die beste von allen bezeichnet werden kann; für diese Arbeit waren auch die verschiedenen Ansätze und Verfahren interessanter. Es wurde gezeigt, daß der Prozeß der Aufwandsabschätzung, da von vielen Faktoren abhängig, vielschichtig ist.
Interessanterweise wurde die verborgene Frage, ob die Arbeit bzw. die Produktivität eines Menschen quantitativ und qualitativ erfaßt werden kann, ebenfalls, wenn auch indirekt, beantwortet. Bei der Function-Point- und der Data-Point-Methode werden Aufwandskennzahlen ermittelt, die den relativen Aufwand der Projektes zueinander beschreiben. Dadurch, daß die Erfahrungen über den Entwicklungskontext in eine angepaßte Transformationskurve einfließen, mit deren Hilfe dann die Aufwandskennzahlen in den zu erwartenden Aufwand überführt

werden, ist die Produktivität der im Entwicklungskontext erfaßten Personen indirekt berücksichtigt worden, ohne diese durch irgendwelche Formeln oder Zahlen greifbar machen zu müssen. Damit wurde die Problematik, wie die Arbeit anderer zu bewerten sei, elegant umgangen.

Alles in allem bietet diese Arbeit einen Überblick über Methoden zur Aufwandsabschätzung bei Softwareprojekten. Die mit dieser Thematik verbundenen Probleme werden aufgezeigt und diskutiert, Verbesserungsmöglichkeiten kurz dargestellt.
Aufgrund der Erfahrungen, die ich während des Schreibens machen konnte, ergaben sich mehrere Ansatzpunke für weitere Arbeiten.
So bietet sich die Möglichkeit, Projektstrukturierungsmodelle genauer zu untersuchen, die Problematik der Softwarequalifizierung zu erforschen oder aber Produktivitätsbetrachtungen anzustellen, um den Zusammenhang zwischen Aufwand, Menschen und Zeit besser zu erklären.
Da die Function-Point-, ebenso wie die Data-Point-Methode, Ansätze für Verbesserungen bietet, wäre auch die Modifikation einer der beiden Methoden genauso interessant, wie die Entwicklung einer Mischform aus beiden.

5 Literaturverzeichnis

[Floyd95]
C. Floyd, “Einführung in die Softwaretechnik“, Vorlesungsunterlagen, Wintersemester 1995/96

[Gryczan99]
G. Gryczan, J. Mack, C. Floyd, “Einführung in die Softwaretechnik“, Vorlesungsunterlagen, Wintersemester 1999/2000

[Kellner94]
H. Kellner, “Die Kunst, DV-Projekte zum Erfolg zu führen - Budgets, Termine, Qualität“, München, Wien: Hanser 1994

[Litke96]
H.-D. Litke, “DV-Projektmanagement - Zeit und Kosten richtig einschätzen“, München, Wien: Hanser 1996

[Noth84]
T. Noth, M. Kretschmar, “Aufwandsabschätzung von DV-Projekten - Darstellung und Praxisvergleich der wichtigsten Verfahren“, Berlin u.a.: Springer 1984

[TU-B98]
WI/AEDV, “Aufwandsabschätzungen“, Technische Universität Berlin, 1998, http://aedv.cs.tu-berlin.de/edu/se/Sollkonzept6.htm, Zugriff: 15.07.1999

[UNI-H97]
Autor nicht feststellbar, “Prozentsatzmethode der Aufwandsabschätzung“, Universität Hannover, 1997, http://www.kbs.uni-hannover.de/se/prozentsatz_methode.html, Zugriff: 15.07.1999

[Weltz92]
F. Weltz, R. G. Ortmann, “Das Software-Projekt - Projektmanagement in der Praxis“, Frankfurt/Main u.a.: Campus 1992

A N H A N G

INHALT

Tabellen- und Abbildungsverzeichnis für die Anhänge

ABBILDUNGEN IN DEN ANHÄNGEN:

TABELLEN IN DEN ANHÄNGEN:

Anhang A: Referenz-Projektdaten

Die hier aufgeführten Daten gehören zu Projekten, die ich komplett oder in großen Teilen selbst durchgeführt habe. Viele Daten konnte ich leider nicht verwenden, da sie nur einzelne Funktionen eines großen Gesamtprojektes betrafen und hierfür die Methoden zur Aufwandsabschätzung nicht geeignet sind.
Die Projektdaten dienen dazu, Modell (Abschätzung) mit Realität (abgeschlossene Projekte) zu vergleichen. Es wurde versucht, einen Kompromiß zwischen Aussagekraft und Anonymität zu erreichen.

Projekt Alpha (1998)

Es wurde eine Software entwickelt, die Analysedaten bei Verwertungsvorgängen verwalten und, in Zusammenarbeit mit einem Geoinformationssystem, bestimmte Punkte auf der Landkarte darstellen kann. Folgender Aufwand wurde hierbei, kontinuierlich über einen Zeitraum von vier Kalendermonaten, realisiert:

I. Orientierung	0 MH	÷ 0 MM
II. Untersuchung	0 MH	÷ 0 MM
III. Funktionsbeschreibung	2 MH	÷ 0,02 MM
IV. Design	27 MH	÷ 0,26 MM
V. Entwicklung	61 MH	÷ 0,60 MM
VI. Einführung	14 MH	÷ 0,13 MM
Gesamt	**104 MH**	**÷ 1,01 MM**

I. und II. wurden übersprungen, da der Auftraggeber bereits eine Funktionsbeschreibung besaß, die nur geringfügig modifiziert werden mußte.

Datenmodell:

Entität	Anzahl Attribute	Anzahl Schlüssel
Analysedaten A	6	2
Analysedaten B	6	2
Vorgangsdaten	61	1

Nachrichten:

Beschreibung	Anzahl Felder	Anzahl Sichten
Maske A	37	1
Maske B	25	1
Maske C	9	1
Maske D	6	1
Brief A	15	1
Brief B	10	1
Brief C	7	1

Realisierte Funktionen:

Funktion	Eingaben	Ausgaben	Datenbestände	Abfragen
Vorgang bearbeiten	Vorgangsdaten, Analysedaten	Vorgang	Vorgangsdaten (RW), Analysedaten (RW)	bereits existierende Vorgangsdaten
Vorgang genehmigen	benötigte Angaben	genehmigter Vorgang, drei Briefe (u.a. wg. fehlender Unterlagen)	Vorgangsdaten (RW)	bereits existierende Vorgangsdaten

Nachweis führen	Angaben erbrachter Leistungen	abgeschlossener Vorgang	Vorgangsdaten (RW)	bereits existierende Vorgangsdaten
Koppelung mit GIS	Koordinaten -	- Vorgangsdaten	Vorgangsdaten (R)	Vorgangsdaten
Datenaustausch	Vorgangsnummern Vorgänge	exportierte bzw. importierte Vorgänge	Vorgangsdaten (RW), Analysedaten (RW)	Vorgangsdaten

Projekt Beta (1999)

Aufgabe dieser Applikation ist es, alle beim industriellen Werk anfallenden Daten zu erfassen und zu verwalten. Die Masken dienen nur der Eingabe bzw. Änderung. Folgender Aufwand wurde über einen Zeitraum von vier Kalendermonaten realisiert, wobei eine zweimonatige Pause enthalten ist:

I. Orientierung	0 MH	÷ 0 MM
II. Untersuchung	0 MH	÷ 0 MM
III. Funktionsbeschreibung	3 MH	÷ 0,03 MM
IV. Design	5 MH	÷ 0,05 MM
V. Entwicklung	23 MH	÷ 0,23 MM
VI. Einführung	18,5 MH	÷ 0,18 MM
Gesamt	**49,5 MH**	**÷ 0,49 MM**

I. und II. wurden übersprungen, da der Auftraggeber bereits eine Funktionsbeschreibung besaß, die nur geringfügig modifiziert werden mußte.

Datenmodell:

Entität	Anzahl Attribute	Anzahl Schlüssel
Daten A	23	1
Daten B	25	2
Daten C	32	2
Daten D	6	1
Daten E	10	1
Daten F	6	1
Daten G	5	1

Nachrichten:

Beschreibung	Anzahl Felder	Anzahl Sichten
Maske A	0	1
Maske B	23	1
Maske C	27	1
Maske D	25	1
Maske E	32	1

Realisierte Funktionen:

Funktion	Eingaben	Ausgaben	Datenbestände	Abfragen
Werk erfassen / bearbeiten	Angaben zum Werk, Verwertungsmöglichkeiten	Werk	Daten A und D-G (alle RW)	bereits existierende Werksdaten
Verwertung erfassen	benötigte Angaben zur Verwertung	Verwertung	Daten A (R), Daten C (RW)	bereits existierende Verwertungsdaten
Inhaltsstoffe erfassen	Probendaten	Probe	Daten A (R), Daten B (RW)	bereits existierende Probendaten

Teil-Projekt Gamma (1998/99)

Im Rahmen eines größeren Projektes sollte ein Zusatztool entwickelt werden, welches zu einem geographischen Punkt eine Beschreibung verwalten kann. Neben des Anzeigens einer Koordinate zur Beschreibung, sollten diese auch noch in Kategorien einteilbar sein. Folgender Aufwand wurde bei diesem Tool über einen Zeitraum von achteinhalb Kalendermonaten realisiert, wobei für einen Zeitraum von fünfeinhalb Monaten andere Aspekte des selben Gesamtprojektes behandelt wurden:

I. Orientierung	0 MH	÷ 0 MM
II. Untersuchung	0 MH	÷ 0 MM
III. Funktionsbeschreibung	1 MH	÷ 0,01 MM
IV. Design	4 MH	÷ 0,04 MM
V. Entwicklung	35 MH	÷ 0,34 MM
VI. Einführung	2 MH	÷ 0,02 MM
Gesamt	**41 MH**	**÷ 0,41 MM**

I. und II. wurden übersprungen, da im Rahmen des Gesamtprojektes die Anforderungen an dieses Teil-Projekt nebenbei entstanden ist und diese nur geringfügig modifiziert werden mußten.

Datenmodell:

Entität	Anzahl Attribute	Anzahl Schlüssel
Einträge	3	2

Nachrichten:

Beschreibung	Anzahl Felder	Anzahl Sichten
Maske A	4	1

Realisierte Funktionen:

Funktion	Eingaben	Ausgaben	Datenbestände	Abfragen
Ort eintragen	Koordinaten, Kategorie, Text	Eintrag	Einträge (RW)	Einträge nach Kategorie
Ort anfahren	Eintrag	-	Einträge (R)	Einträge nach Kategorie

Anhang B: Übersicht über die gängigen Methoden

Die in diesem Abschnitt dargestellte Sammlung von Methoden zur Aufwandsabschätzung gibt einen Überblick der im Rahmen dieser Arbeit recherchierten Methoden und hat keinen Anspruch auf Vollständigkeit.

Da jeder Autor meistens seine eigene Ansicht über Phasenmodelle hat, entstand zwangsläufig die Notwendigkeit der Verwendung eines einheitlichen Phasenmodells. Dieses ist für die folgenden Abschnitte das Modell aus Kapitel 2.1 - Modelle zur Projektstrukturierung.
In den linken Tabellen sind immer die in der Quelle genannten Werte aufgeführt, in den rechten, doppelt eingerahmten Tabellen sind die einheitlich angepaßten Werte angegeben.
Zu beachten gilt: Die Einteilungen der Phasen sind nicht identisch. Die Anpassung erfolgte durch Inhaltsvergleich der einzelnen Phasen und unter Berücksichtigung der Verteilungen untereinander.

B.1 Methoden basierend auf Erfahrungsanalogien

(a) Analogie-Schätzung:

Diese Art der Aufwandsabschätzung ist die am häufigsten verwendete.
Meistens wird ein Projekt hierzu in Teilprojekte bzw. Phasenabschnitte und Meilensteine zerlegt (vgl. Kap. 2.1). Daraufhin wird der Aufwand der einzelnen Teile abgeschätzt, indem diese Teile mit ähnlichen Teilen bereits abgeschlossener Projekte verglichen wird. Den Gesamtaufwand erhält man dann durch Zusammensetzen der Teilergebnisse.

Da diese Art der Abschätzung sehr stark von den Erfahrungen des Schätzers und dessen Grundhaltung abhängt, gibt es drei Variationen davon, um die Ergebnisse zu verbessern:

- der Schätzer gibt eine optimistische, eine realistische und eine pessimistische Schätzung ab. In der Regel bildet dann die realistische Schätzung den Mittelwert der beiden anderen,
- eine Schätzergruppe bestehend aus zwei bis drei Personen erstellt die Aufwandsabschätzung gemeinsam. Damit soll der Schätzeinfluß der unterschiedlichen Typen von Schätzern - ängstliche und mutige - ausgeglichen werden,
- der Vergleich der Projektteile mit denen bereits abgeschlossener Projekte kann durch Verfahren der Mustererkennung optimiert werden. Dieses erleichtert dann eine Zuweisung des Aufwands.

Da die Abschätzung sehr stark vom Schätzer abhängt, beeinflußt seine Grundhaltung, sowie sein Umfeld die Schätzung ebenfalls.
Ein mutiger Schätzer wird in der Regel eher angeben, daß die gestellte Aufgabe schneller zu schaffen sei, während ein ängstlicher Schätzer erst einmal Bedenken äußern wird, daß gar nicht alle Faktoren bekannt seien und er bzw. sie daher keine Schätzung abgeben kann. Wird dann doch eine Schätzung abgegeben, liegt der geschätzte Aufwand eher zu hoch (vgl. [Kellner94]).

Auch kommt es in der Praxis nicht selten vor, daß der Aufwand zwischen Auftraggeber und Diensterbringer ausgehandelt wird. Daß dieses das Schätzergebnis verfälscht, liegt auf der Hand.
Ferner beeinflußt auch die Zeit, die ein Schätzer für seine Schätzung zur Verfügung hat, das Ergebnis (vgl. [Litke96]).

In [Kellner94] wird angegeben, wie hoch die Wahrscheinlichkeit ist, daß die Schätzung mit einer Abweichung von ñ 10% stimmt (vgl. Kap. 2.1 - Modelle zur Projektstrukturierung):

2 : 3	nach Phase I: Orientierung
3 : 4	nach Phase II: Untersuchung
4 : 5	nach Phase III: Funktionsbeschreibung
9 : 10	nach Phase IV: Design
9 : 10	nach Phase V: Entwicklung

(b) Prozentsatzmethode: [UNI-H97] / [TU-B98]

Der Grundgedanke der Prozentsatzmethode ist der, daß gleichartige Projekte in gleichartigen Umgebungen ähnliche Aufwandsverteilungen über die Phasen (vgl. Kap 2.1) besitzen. Unter dieser Annahme wird dann eine Phase entweder durchgeführt und der Aufwand nach Beendigung dieser Phase ermittelt oder es wird versucht, eine Phase möglichst genau zu schätzen. Mit diesen Daten wird dann der Aufwand der anderen Phasen ‘hochgerechnet’. Ist der Aufwand der einzelnen Phasen bekannt, kann durch Addition der Gesamtaufwand des Projektes ermittelt werden.

Als Beispiel sollen hier zwei Verteilungen dienen; erstens die der Firma Hewlett-Packard (vgl. [UNI-H97]) und zweitens die Verteilung von Stahlknecht (vgl. [TU-B98]).

Hewlett-Packard (vgl. [UNI-H97]):

Phase	relativer Aufwand
Analyse	18 % (ñ 10 %)
Entwurf	19 % (ñ 10 %)
Codierung	34 % (ñ 10 %)
Test	29 % (ñ 10 %)

angepaßte Vert. Ø [%]	Phase
8	I
10	II
8	III
11	IV
34	V
29	VI

Tabelle B - 1 : Prozentuale Verteilung nach HP

Stahlknecht (vgl. [TU-B98]):

Phase	relativer Aufwand	angepaßte Vert. Ø [%]	Phase
Projektbegründung	5 %	5	I
Ist-Analyse	10 %	10	II
Soll-Konzept	10 %	10	III
Systementwurf	30 %	30	IV
Implementierung	35 %	35	V
Systemeinfühung	10 %	10	VI

Tabelle B - 2 : Prozentuale Verteilung nach Stahlknecht

(c) Software-Lifecycle-Management (SLIM): [Noth84]

Dieses Verfahren wurde 1974 von Lawrence H. Putnam, damals Budgetdirektor des Softwarebereiches des US-Verteidigungsministeriums entwickelt.

In einer Exponentialfunktion wird der Restaufwand des Produktes anhand der Kenngrößen Lebensdauer, Entwicklungsdauer und Schätzzeitpunkt ermittelt.

Da es sich hierbei aber ebenfalls um eine (sehr kontextabhängige) Prozentsatzmethode handelt, die darüber hinaus aufgrund ihres Alters nicht mehr heutigen Projekten gerecht wird, wird auf eine detaillierte Vorstellung an dieser Stelle verzichtet. Der geneigte Leser kann dieses bei Bedarf in [Noth84] nachlesen.

(d) IfA-Pass-Verfahren: [Noth84]

Bei dem 1977 am Züricher Institut für Automation entwickelten Verfahren handelt es sich um eine Variante der Prozentsatzmethode. Daher werden hier nur die speziellen Prozentsätze genannt:

Phase	relativer Aufwand in Prozent			angepaßte Vert.	
	Min	Durchschnitt	max	Ø [%]	Phase
1 Voranalyse	4	5	6	6	I
2 Methoden und Verfahren	14	18	20	20	II
3 Systemspezifikation	20	27	35	20	III
4 Programmierung	25	30	35	12	IV
5 Benutzerorganisation	10	12	15	33	V
6 Einführung	5	8	10	9	VI

Tabelle B - 3 : Prozentuale Verteilung IfA-Pass

Der linke Tabelle zeigt die prozentuale Verteilung, die am Züricher Institut für Automation entwickelt wurde, während die rechte, doppelt eingerahmte Tabelle, wieder die Vergleichswerte (vgl. B.1 (b)) angibt.

B.2 Methoden basierend auf Lines-of-Code

Da bei der Software-Entwicklung mit heutigen Entwicklungsumgebungen und GUI-Buildern die ‘Programmierwerkzeuge’ ebenfalls Lines of Code (kurz: LoC) erzeugen, ist dieses eine Größe, die heute nur noch schwer greifbar ist. Im Normalfall wird kein Entwickler heute noch damit konfrontiert werden und es ist sogar am fertigen Produkt kaum nachvollziehbar, wieviele LoC es hat.
Diese Verfahren sind in einer Zeit entstanden, in der in Maschinensprache oder sehr stark verwandten Makro-Sprachen entwickelt wurde. Zu dieser Zeit war LoC, unter der Annahme, daß über ein größeres Projekt der Aufwand jeder Code-Zeile sich einem Durchschnittswert nähert, eine akzeptable Kenngröße für Abschätzungen.
Die Ungenauigkeit dieser Methoden wird noch weiter durch die Tatsache gesteigert, daß die LoC ebenfalls - meist nach der Analogieschätzung - geschätzt werden müssen, so daß eigentlich eine Schätzung einer Schätzung durchgeführt wird.

Obwohl diese Verfahren aufgrund ihrer bedingten Verwendbarkeit heute nicht mehr zu Schätzung des Aufwandes verwendet werden können, seien an dieser Stelle trotzdem der Vollständigkeit halber einige aufgeführt.

(a) Verfahren von Walston und Felix: [Litke96]

Dieses Verfahren wurde von Walston und Felix bei IBM entwickelt. Grundlage für die Schätzgleichung bilden die Daten aus 60 Projekten, die mit einem von ihnen entwickelten, vereinheitlichen Verfahren gesammelt wurden.

$$E = 5.2 * L^{0.91} \quad \text{mit } E\text{: Aufwand in Mannmonaten und } L\text{: je 1000 LoC}$$

Um diese Gleichung für verschiedene Projekte anwenden zu können, haben Walston und Felix Annahmen über das Umfeld getroffen.

Die schwankende Produktivität der Mitarbeiter gleicht sich durch eine hohe Anzahl der Projektmitarbeiter aus. Als Grenzwert hierzu werden in [Litke96] sieben Mitarbeiter genannt.
Ferner wird angenommen, daß bei mehreren Schätzungen, immer im selben Kontext, alle anderen Faktoren (außer LoC) vernachlässigbar sind. Dieses träfe aber nur innerhalb des selben Projektteams mit den selben Entwicklungswerkzeugen zu.

(b) Verfahren der Universität von Maryland: [Litke96]
Hierbei handelt es sich um ein Verfahren, welches von Bailey und Basili an der Universität von Maryland entwickelt wurde.

Da der Kontext dieses Verfahrens ähnlich des in (a) genannten ist, sei an dieser Stelle nur die Schätzgleichung angegeben:

$$E = 3.4 + 0.72 * DL^{1.17} \text{ ñ } 25\ \%$$

mit E: Aufwand in Mannmonaten und DL: zusammengesetzter Wert aus Arbeit und LoC, an dieser Stelle nicht näher quantifiziert

B.3 Methoden basierend auf Anzahl von Befehlen

Für diese Gruppe von Verfahren gilt vom Prinzip her das gleiche, wie für die in B.2 genannten. Durch die Weiterentwicklung der Programmiersprachen Mitte der 70er Jahre wurde LoC (vgl. vorheriges Kap.) mehr und mehr durch die Anzahl von ausgeführten Instruktionen / Befehlen ersetzt.
Auch hierbei ging man von der Annahme aus, daß sich der Schwierigkeits- bzw. Aufwandsunterschied der einzelnen Befehle, durch die steigende Anzahl einem Durchschnitt nähert. Genauso häufig galt die Anzahl von bedingten Verzweigungen (If-Anweisungen) als geeignete Kenngröße proportional zum Aufwand.
Daß beide Größen heutzutage im nachhinein schwer ermittelbar und im voraus extrem schwer schätzbar sind, liegt auf der Hand. Daher werden diese Methoden an dieser Stelle nur kurz aufgeführt. Der geneigte Leser möge bei Interesse auf die vorhandene Literatur zurückgreifen (siehe Anhang D).

(a) Verfahren der System Development Corporation (SDC):

Aufbauen auf dem Ergebnis einer Untersuchung der System Development Corporation aus dem Jahre 1967 entstand folgende Gleichung zur Abschätzung des Aufwandes:

$$Y_1 = -33.63 + 9.15 * X_2 + 10.73 * X_8 + 0.51 * X_{26} + 0.46 * X_{30} + 0.4 * X_{41} + 7.28 * X_{42} - 21.45 * X_{48.1} + 13.53 * X_{48.5} + 12.35 * X_{51} + 58.82 * X_{53} + 30.61 * X_{56} + 29.55 * X_{72} + 0.54 * X_{75} + 25.2 * X_{76}$$

mit

- Y_1: Aufwand in MM (= Mannmonaten)
- X_2: Mängel an Kenntnis der Betriebsanforderungen: 0 - 2
- X_8: Stabilität des Entwurfs: 0 - 3
- X_{26}: Prozentanteil mathematischer Funktionen
- X_{30}: Prozentanteil Informationsspeicherungs- und Wiedergew.-Instruktionen
- X_{41}: Anzahl der Unterprogramme
- X_{42}: Programmiersprache: MOS $\hat{=}$ 1, POS $\hat{=}$ 0
- $X_{48.1}$: Programmtyp ist kommerziell: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- $X_{48.5}$: Standalone-Entwicklung: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- X_{51}: erstes Programm auf Computer: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- X_{53}: gleichzeitige Entwicklung von Hardware: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- X_{56}: Benutzung im Direktzugriff: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- X_{72}: Verschiedene Computer für Programmierung: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0
- X_{75}: Anzahl der Reisen, die zur Projektdurchführung nötig sind
- X_{76}: Programmentwicklung im militärischen Bereich: Ja $\hat{=}$ 1, Nein $\hat{=}$ 0

Hieran ersichtlich ist, daß zur Zeit der Entwicklung dieser Methode andere Faktoren als wichtig erachtet wurden, als bei heutigen Betrachtungen. Ferner spiegelt sich auch die Unterschiedlichkeit der Softwareentwicklung in den USA zu der in Europa wieder.

(b) Verfahren zur Terminierung von Organisation und Programmierung (T.O.P.): [Noth84]

Diese Methode wurde 1971 von der Kölner Unternehmensberatung Peter Sattler und Partner entwickelt. Es gibt hiervon drei Varianten, die je nach Abschluß einer Phase den verbleibenden Rest-Aufwand abschätzen.

- T.O.P. I - nach der 'Durchführbarkeitsstudie' (vgl. Kap2.1, nach Phase I und II für Phase III),
- T.O.P. II - nach dem 'Grobentwurf' (vgl. Kap2.1, nach Phase III für Phase IV),
- T.O.P. III - nach dem 'Feinentwurf' (vgl. Kap2.1, nach Phase IV für Phase V).

T.O.P. I soll hier kurz vorgestellt werden.

Das zu entwickelnde System wird hierzu in Module zerlegt, deren Anzahl von Befehlen abgeschätzt werden. Um eine durchschnittliche Größe zu ermitteln, gibt der Schätzer einen kleinsten, einen wahrscheinlichsten und einen größten Wert an, der dann mit folgender Formel gemittelt wird:

$$GA = \frac{\text{min.} + 4 * \textit{warsch.} + \text{max.}}{6}$$ mit *GA*: geschätzter Aufwand

Den Gesamtaufwand ermittelt man nun durch die Division von *GA* durch die geschätzte Produktivität (100 - 700 Befehle / MM).

(c) Aron-Verfahren: [Noth84]

Der Aufbau dieses Verfahrens, das Aron 1969 entwickelte, ähnelt sehr stark dem T.O.P.-Verfahren (siehe B.3 (b)).

Auch hier wird das Produkt in Module zerlegt, deren Umfang (zwischen 400 und 1000 Befehlen) geschätzt wird. Anschließend wird der mittlere Umfang mit der Anzahl der Module multipliziert. Dieser Durchschnitt wird nun durch einen Produktivitätsfaktor dividiert, der von der verwendeten Programmiersprache, dem Schwierigkeitsgrad der Realisierung und der Dauer des Projektes abhängt.
Der Gesamtaufwand des Projektes wird nun anhand einer prozentualen Verteilung ermittelt, für die Aron eine Funktionskurve angibt (vgl. [Noth84]).

(d) Shell-Verfahren: [Noth84]

Richard L. Shell entwickelte 1972 ein Verfahren, das es ermöglichen sollte, den Aufwand für Programmierung und Modultest (vgl. Phase V, Kap2.1) abzuschätzen.

Hierzu wurde auf der Datenbasis bereits abgeschlossener Projekte ermittelt, wie die Aufwandsverteilung von reiner Programmierarbeit zu anderer, produktiver Arbeit und zu persönlicher, unproduktiver Arbeit ist.
Genauso wurde das Verhältnis zwischen Programmgröße, gemessen an der Anzahl von Befehlen, und der reinen Programmierarbeit ermittelt. Um diese Abschätzung genauer zu machen, ermittelte Shell dieses Verhältnis für drei Schwierigkeitsgrade.
Der Gesamtaufwand konnte nun anhand dieser Daten ermittelt werden.

(e) TRW-Verfahren von Wolverton: [Noth84]

Auch wenn dieses 1974 von Wolverton entwickelte Verfahren von einer direkten Beziehung zwischen der Anzahl der Befehle und dem Aufwand ausgeht, so wird dennoch versucht, den Einzelaufwand der einzelnen Module unter Berücksichtigung ihrer Komplexität zu ermitteln.

Dazu sind die Module in funktionale Gruppen zu gliedern. Diese sind:

- Steuerprogramme,
- E/A-Programme,
- Programme, die Daten für die Verarbeitung vorbereiten,
- Algorithmen,
- Datenverwaltungsroutinen,
- zeitkritische Programme.

Unter Berücksichtigung ihres Schwierigkeitsgrades werden dann die Module, aufgeschlüsselt nach der geschätzten Anzahl von Befehlen, in sechs Klassen, von ‘wiederverwendet und leicht zu realisieren’ bis hin zu ‘neu und schwer zu realisieren’ eingestuft. So erhält man eine 6x6-Matrix, die in jeder Zelle die kumulierte Anzahl von Befehlen enthält.

Wolverton hat nun, basierend auf den Untersuchungen bereits abgeschlossener Projekte, für jede Zelle dieser Matrix den Aufwand ermittelt. Somit erhält man den Gesamtaufwand durch Multiplikation jeder Zelle mit dem zugehörigen Aufwand und durch Addition der einzelnen Zellen-Produkte.

(f) Boing-Verfahren: [Noth84]

Dieses 1977 bei der Firma Boing entwickelte Verfahren setzt eine Schätzung der Anzahl der Befehle jeder einzelnen Routine zum Zeitpunkt des Designs voraus.
Daraus wird dann anhand folgender Tabelle der Aufwand erst für jede Routine und damit dann der Gesamtaufwand der Entwicklungsphase geschätzt. Der Aufwand des gesamten Projektes wird dann anhand einer Prozentsatzmethode (vgl. B.3 (b)) hochgerechnet.

6 MM je 1000 Befehle	für mathematische Routinen
8 MM je 1000 Befehle	für Ausgabe-Routinen
12 MM je 1000 Befehle	für Steuerungs-Routinen
20 MM je 1000 Befehle	für Prozeßdaten-Routinen
40 MM je 1000 Befehle	für Realtime-Routinen

Quelle: [Noth84]

Die prozentuale Verteilung wird wie folgt angegeben (vgl. [Noth84]):

Phase	relativer Aufwand	angepaßte Vert. Ø [%]	Phase
Anforderungsdefinition	5 %	2	I
Design und Spezifikation	25 %	3	II
Codierung	10 %	8	III
Modultest	25 %	17	IV
Integration und Test	25 %	35	V
Systemtest	10 %	35	VI

Tabelle B - 4 : Prozentuale Verteilung nach Boing

Zuletzt werden die so ermittelten Aufwandszahlen anhand einer Tabelle an projektspezifische Faktoren angepaßt, die aber an dieser Stelle nicht aufgeführt wird, da sie für diese Arbeit nicht weiter relevant ist.

(g) Doty-Verfahren: [Noth84]

Aufbauend auf der SDC-Studie (vgl. B.3 (a)) entwickelte Doty 1977 folgende Formel für standardmäßge Softwareprojekte:

$$\text{bis 10.000 Befehle:} \quad MM = 5.288 * \left(\frac{Befehle}{1000}\right)^{1.047} \quad \text{und}$$

$$\text{über 10.000 Befehle:} \quad MM = 2.06 * \left(\frac{Befehle}{1000}\right)^{1.047} * f_i$$

wobei f_i ein Korrekturfaktor zwischen 0.83 und 1.92 ist, der einer Tabelle entnommen werden kann und die individuellen Einflüsse des Entwicklungskontextes wiedergeben soll.

(h) Verfahren von Griffin: [Noth84]

Griffin hat 1977 dieses Verfahren für den Einsatz im militärischen Kontext entwickelt. Ähnlich der bereits vorgestellten Methoden (z.B. SDC, B.3 (a)) wird hier der Aufwand anhand der Anzahl von Befehlen, dividiert durch die Produktivität und modifiziert durch Korrekturfaktoren, ermittelt.

(i) Schneider-Verfahren: [Noth84]

Die Arbeiten von Schneider basieren auf den Arbeiten von Halstead (vgl. [Noth84]). Das 1978 von Schneider entwickelte Verfahren basiert auf Durchschnittswerten bereits realisierter Projekte. Sein Kernstück bildet folgende Funktion:

$$\text{Gesamtaufwand in MM} = 0.3 * \left(\frac{Befehle}{1000}\right)^{1.83}$$

(j) BANG (Tom DeMarco): [Litke96]

Mit der Kennzahl BANG versuchte Tom DeMarco eine implementationsunabhängige Größe für den Umfang eines Software-Systems zu schaffen.

Der Grundgedanke hierbei ist, daß der Informationsgehalt - und somit der mit der Realisierung verbundene Aufwand - eine direkte Funktion der Eingabe / Spezifikation ist. Zunächst muß das System in elementare Einheiten zerlegt werden, denen dann anhand eines Kriterienkataloges sogenannte p-counts zugeordnet werden. Diese werden dann mit Korrekturfaktoren dem Kontext und der Produktivität angepaßt.

Um diese nun in BANG gewonnene Kennzahl für den Umfang des Systems in den für die Entwicklung benötigten Aufwand zu überführen, kommen nun zwei weitere Kennzahlen hinzu: Design-Weight (DW) und Implementation-Weight (IW).
DW soll den Einfluß des Code-Entwurfes wiedergeben und ist sehr stark an der Anzahl von Entscheidungen innerhalb der betrachteten, elementaren Einheit orientiert.
IW hingegen, als Kennzahl für den eigentlichen Programmieraufwand, basiert auf der Anzahl der Befehle.

Aus BANG, DW und IW läßt sich nun der Gesamtaufwand für die Entwicklung (vgl. Phase V, Kap. 2.1) abschätzen.
Die genaue Vorstellung dieses Verfahren würde den Rahmen dieser Arbeit sprengen und wäre auch, angesichts der heutigen Anwendbarkeit dieser Methode, sicherlich nicht angebracht. Der geneigte Leser wird daher entweder auf [Litke96] oder aber auch die Ursprungsliteratur von Tom deMarco verwiesen, der auf diesem Gebiet sicherlich einer der Vorreiter für heutige und zukünftige Methodenentwicklungen ist.

(k) Constructive-Cost-Model (COCOMO): [Noth84] / [Litke96]

COCOMO ist eine Entwicklung von Barry W. Boehm, der 1981 Direktor von Thompson-Ramo-Wooldridge, eines großen amerikanischen Softwarehauses war.

Es gibt drei verschiedene Modelle, die sich im Zeitpunkt der Schätzung und der Granularität der Vorgehensweise unterscheiden:

- Basismodell: für schnelle, frühzeitige Schätzungen des Aufwandes in groben Zügen. Besonders geeignet für kleinere bis mittlere Projekte,
- Zwischenmodell: es besitzt eine detailliertere Vorgehensweise als das Basismodell. Daher ist es erst später, wenn ein gewisser Detailgrad bei der Produkt-Definition erreicht wurde, einsetzbar, liefert dann aber auch genauere Aussagen,
- Detailmodell: dieses Modell ist erst zu Beginn der Entwicklungsphase einsetzbar, wenn der Entwurf des Systems im Detail vorliegt. Aufgrund dieser Voraussetzung liefert dieses Modell die genauesten Ergebnisse im Vergleich mit den anderen beiden.

Als Schätzgrundlage dient allen Modellen eine Größe Namens KDSI (KiloDeliveredSourceInstructions, d.h. 1000 ausgeführte Befehle im Sourcecode).

Das Basismodell soll für eine kurze Vorstellung genügen. Hierbei werden drei verschiedene Einsatzkontexte berücksichtigt, die jeweils eine eigene Abschätzungsgleichung besitzen:

Batchsysteme: $MM = 2.4 * KDSI^{1.05} * Q * P$

Onlinesysteme: $MM = 3.0 * KDSI^{1.12} * Q * P$

Realtimesysteme: $MM = 3.6 * KDSI^{1.2} * Q * P$

mit Q: Qualitätsfaktor und P: Produktivitätsfaktor

B.4 Methoden basierend auf Funktionen

Die grundlegende Vorgehensweise bei allen Methoden dieser Gruppe besteht darin, daß das zu realisierende Produkt in Funktionen (meistens nicht aus softwaretechnischer Sicht) bzw. Geschäftsvorfälle aufgeteilt werden muß. Funktion und Geschäftsvorfall werden im weiteren Verlauf synonym (mit der Bedeutung Geschäftsvorfall) verwendet.
Anschließend werden diese anhand von Kriterien bewertet und gewichtet. Die so entstandene Kenngröße wird dann dem Entwicklungskontext mittels Korrekturfaktoren angeglichen und dann mit Hilfe einer Funktion in den Aufwand überführt.
Somit spielt der Entwurf und das Entwurfsumfeld, nicht aber implementierungsspezifische Details eine Rolle.

(a) IBM-Handbuch-Verfahren: [Noth84]

Der Prototyp dieses Verfahrens wurde 1968 veröffentlicht. Verschiedentlich wurde versucht, dieses durch Modifikationen zu verbessern, was aber nicht von Erfolgt gekrönt war. Daher beziehen sich die verschiedenen Autoren meistens auf diesen Prototypen.

Voraussetzung für dieses Verfahren ist das abgeschlossene fachliche Feinkonzept (vgl. Phase III, Kap. 2.1), aufgrund dessen der Aufwand für das restliche Projekt geschätzt ist. Dieses ist typisch für die Methoden dieser Gruppe.

Somit lautet die Formel für die Ermittlung des reinen Programmieraufwandes in MM (T_{pp}):

$$T_{pp} = (T_{ea} + T_v) * (T_k + T_e), \qquad \text{wobei}$$

T_{ea}: Summe des Programmieraufwandes für Ein-/Ausgabeprozesse über die verschiedenen Komponenten.
Ermittelt wird diese anhand nebenstehender .Tabelle B - 5

T_v: Summe des Aufwands für die Verarbeitung über die verschiedenen Komponenten
(Tabelle B - 6)

T_k: Problemkenntnisfaktor
(Tabelle B - 8)

		Gewicht
EINGABE	Karte - ein Format	1
	- mehrere Formate	2
	Jedes Magnetband je Datei	1
	Jede Magnetplatte je Datei	1
AUSGABE	Drucker je Druckformat	1
	Jedes Magnetband je Datei	1
	Karte - ein Format	1
	- mehrere Formate	2
	Jede Magnetplatte je Datei	1

***Tabelle B - 5** :Tabelle zur Bestimmung des PA bei E/A-Operationen*

T_e: Programmier-erfahrungsfaktor (Tabelle B - 7)

Funktion	GEWICHT		
	Einfach	Mittelschwer	Schwer
Datenbewegung	1	3	4
Prüfungen	1	4	7
Dat. suchen, Tab. lesen	2	5	8
Arithm. Operationen	1	3	5
Programmverbindungen	1	2	3

Tabelle B - 6 : Tabelle zur Bestimmung des Programmieraufwands (COBOL)

Verfügbare Kenntnisse	Erforderliche Kenntnisse		
	Viel	Einiges	Nichts
Detaillierte Kenntnisse der speziellen Aufgabe	0,75	0,25	0,00
Gute generelle Kenntnisse mit fragment Detail Kenntnissen der speziellen Aufg.	1,25	0,50	0,00
Generelle Kenntnisse, keine Detail-Kenntnisse der speziellen Aufgaben	1,50	0,75	0,00
Keine Kenntnisse der speziellen Aufgabe, aber generelle Kenntnisse der angrenzenden Gebiete.	1,75	1,00	0,25
Keine Kenntnisse der speziellen Aufgabe, auch ohne Kenntnisse in den angrenzenden Gebieten.	2,00	1,25	0,25

Tabelle B - 8 : Tabelle zur Ermittlung des Problemkenntnis-faktors

	GEWICHT
CHEFPROGRAMMIERER	0,50 - 0,75
PROGRAMMIERER	1,00 - 1,50
PROGRMMIER-ANFÄNGER	2,00 - 3,00
IN PROGRAMMIER AUSBILDUNG	3,50 - 4,00

Tabelle B - 7 : Tabelle zur Ermittlung des Programmiererfahrungsfaktors

Der endgültige Faktor ergibt sich aus dem Durchschnitt der Einzelfaktoren.

Hat man nun den Aufwand für die reine Programmiertätigkeit ermittelt, so sieht das IBM-Handbuch-Verfahren einen Aufschlag von 70% bis 110% für die Systemanalyse, den Systementwurf und für die Einführung vor. Zusätzlich kommt ein Aufschlag von 5% bis 10% für das Projektmanagement und für Schulungen und Krankheit, also durch den durch das Umfeld verursachten Aufwand, noch einmal 20% bis 35%.
Damit liegt der Gesamtaufwand des Projektes zwischen $T_{pp} * 1.95$ und $T_{pp} * 2.55$.

(b) Futh-Verfahren: [Noth84]

Dieses 1975 von Futh entwickelte Verfahren dient der Ermittlung der reinen Programmierzeit.
Durch prozentuale Zuschläge für Analyse, Entwurf, Test etc. und mit Hilfe von Zuschlägen für die Personalqualität kann hieraus aber die gesamte Projektdauer ermittelt werden.

Die Angaben über die codierten Programme gliedern sich in die drei Bereiche Dateidefinitionen, Programmstruktur und Instruktionen.

Die Aufwandsabschätzung wird aber erleichtert durch Formulare, die Futh aufgrund seiner Erfahrungen mit dieser Methode entwickelte (Beispiel in Tabelle B - 10 und Tabelle B - 9).

FUTH	ZEITSCHÄTZUNG PROGRAMMIERUNG I	erstellt am: 22.12.75 erstellt durch: Gaumann 01
System: Rechnungswesen	Teilsystem: Buchhaltung	Untersystem: Verkaufsabrechnung
Auftrags-Nr.: 074.1.08	Auftragsbezeichnung: Programmieren Eingabeprogramm	Projektstufe: Programmierung

	Programmelemente	Wert	Codierzeit in Stunden = n x k	Summe in Stunden
1. Dateidefinitionen	1.1 Anzahl der Dateien (= Dateispezifikation)	n = 5, k = (0,15)	0,75	
	1.2 Anzahl der Datenelemente aller Dateien	n = 108, k = (0,02)	2,16	
	1.3 Tabellen, Summen aller Stufen	n =, k = (0,1)		
	1.4 Anzahl der zu definierenden Tabellenwerte	n =, k = (0,02)		S1 = 2,91
2. Programmstruktur	2.1 Anzahl der Eingabedateien	n = 3, k = (0,25)	0,75	
	2.2 Anzahl der Gruppenstufen	n = 2, k = (0,25)	0,50	
	2.3 Anzahl der unterschiedlich zu verarbeitenden Satzarten	n = 6, k = (0,1)	0,60	1,85 + 0,50 S2 = 2,35
3. Instruktionen	3.1 Anzahl der sequentiellen E/A-Operationen	n = 9, k = (0,15)	1,35	
	3.2 Anzahl der E/A-Operationen im Direktzugriff	n = 7, k = (0,25)	1,75	
	3.3 Anzahl der Tabellenzugriffe	n =, k = (0,30)		
	3.4 Anzahl der Tabellenzyklen	n =, k = (0,50)		
	3.5 Anzahl der Funktionen, unterteilt nach 3 Schwierigkeitsgraden	n = 12, k = (0,1); n = 42, k = (0,25); n =, k = (0,50)	1,20; 10,50	S3 = 14,80

Tabelle B - 10 : Formular von Futh, Vorderseite

FUTH	ZEITSCHÄTZUNG PROGRAMMIERUNG II	erstellt am: 22.12.75 erstellt durch: Gaumann 02
System: Rechnungswesen	Teilsystem: Buchhaltung	Untersystem: Verkaufsabrechnung
Auftrags-Nr.: 074.1.08	Auftragsbezeichnung: Programmieren Eingabeprogramm	Projektstufe: Programmierung

S1	+ S2	+ S3	= Summe Codierzeit in Stunden
2,91	2,35	14,80	~ 20,00

Programmiertätigkeit	%Satz von Summe Codierzeit in Stunden: Standard %Satz	eigener %Satz	Programmierzeit in Stunden (= %Satz von Su. Codierzeit)
Analyse der Programmvorgabe	10%		2,00
Entwurf des Programmablaufs	80%		16,00
Codieren problemorientiert = S1 + S2 + S3	100%		20,00
Codieren maschinenorientiert	140%		/
Erstellen Testdaten	-	15%	3,00
Programmtests problemorientiert	50%		10,00
Programmtests maschinenorientiert	90%		/
Ausarbeiten der Dokumentation	20%		4,00
Programmeinführung (Starthilfe)	10%		2,00
Summe S4 =			57,00

Zuschlagsart	%Satz lt. Tabelle von Summe S4	Zuschlagszeit in Stunden
Programmiererkenntnis (= Wert X)	20 %	11,40
Systemerfahrung (= Wert Y)	10 %	5,70
Verlustzeit (= Wert Z)	10 %	5,70
Summe S5 =		22,80

S4	+ S5	= Gesamtzeit Programmierung in Stunden
57,00	22,80	~ 80,00

Tabelle B - 9 : Formular von Futh, Rückseite

(c) Software-Partner-Verfahren: [Noth84]

Dieses 1976 von der Darmstadter Unternehmensberatung Software-Partner GmbH entwickelte Verfahren wird in der Literatur etwas stiefmütterlich behandelt.
Wahrscheinlich liegt es daran, daß es sich hierbei um eine Verfeinerung der Analogiemethode handelt; zwar wird das Projekt hierbei in Aufgaben untergliedert, denen dann unter Berücksichtigung von Erschwernissen, der Aufwand zugeteilt ist.
Die hierfür benötigten Formulare sind nicht mehr verfügbar; daher kann dieses Verfahren an dieser Stelle nicht genauer vorgestellt werden.

(d) Verfahren von End, Gotthard und Winkelmann (EGW): [Noth84]

Die Autoren haben 1977 dieses Verfahren bei der Siemens AG entwickelt. Es basiert dabei auf empirischen Untersuchungen, die Projekte bis zu einer Größe von bis zu 40.000 Befehlen umfaßten.

Besonderheit dieser Methode zur Aufwandsabschätzung ist, daß hierbei der fachliche vom DV-Aufwand unterschieden wird.

Der fachliche Aufwand (A_F in MM) wird anhand folgender Formel ermittelt:

$$A_F = FU * FT * FE, \quad \text{wobei}$$

FU: Wird mit Hilfe der Anzahl der fachlichen Funktionen bzw. Aufgaben, wie z.B. Aufträge bearbeiten und Aufträge ausliefern (= 2), ermittelt. Anhand dieser Anzahl kann der in Abbildung B - 1 angegebenen Kurve der Faktor FU ermittelt werden.

FT: Wird durch Addition der einzelnen aus Tabelle B - 11 ermittelten Werte gewonnen und stellt den Umfang und die Schwierigkeiten der fachlichen und organisatorischen Tätigkeiten dar.

Fachliche und organisatorische Tätigkeiten	Wenig	Mittel	Viel
Definition der Aufgabenstellung	0	0,1	0,2
Umfang der Istaufnahme	0,9	1,0	1,1
Organisatorische Änderungen	0	0,15	0,3
Fachliche Grundlagenarbeit	0	0,15	0,3
Organisatorische Abwicklungsschwierigkeiten	0	0,1	0,3

Tabelle B - 11 : Tabelle zur Ermittlung von FT

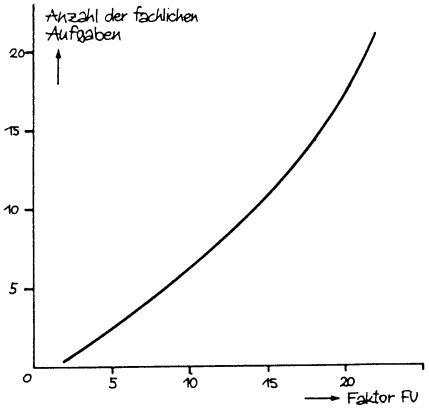

Abbildung B - 1 : Kurve zur Bestimmung von FU
Quelle: [Noth84]

FE: Programmiererfahrung; kann aus folgender Tabelle abgelesen werden:

Verfügbare Erfahrung	Erforderliche Erfahrung		
	Wenig	Einige	Viel
Gute Kenntnisse des Aufgabengebiets und Projekterfahrung	1.0	1.0	1.05
Kenntnisse des Aufgabengebiets und Projekterfahrung	1.0	1.1	1.15
Keine Kenntnisse des Aufgabengebiets, aber Projekterfahrung	1.05	1.15	1.25
Keine Kenntnisse des Aufgabengebiets und keine Projekterfahrung	1.15	1.3	1.5

Tabelle B - 12 : Tabelle zur Bestimmung von FE

Die Berechnung des DV-Aufwands in MM erfolgt durch die Multiplikation der Faktoren *DU*, *DT*, *DE*, *DO*, *DW* und *DS*.

DU (Programmumfang) wird in Abhängigkeit von der zu erwartenden Anzahl von Befehlen (vgl. Anhang B.3) anhand der Kurve in Abbildung B - 2 ermittelt.

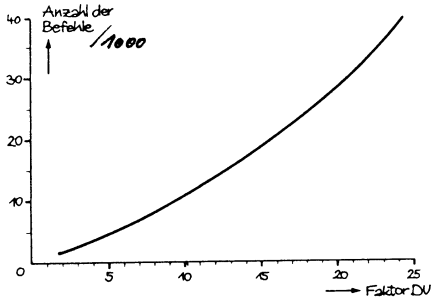

Abbildung B - 2 : Kurve zur Ermittlung von DU
Quelle: [Noth84]

DT, welches den Schwierigkeitsgrad der Programmiertätigkeiten widerspiegelt, kann durch die Addition der einzelnen Werte aus Tabelle B - 13 errechnet werden.

Programmiertätigkeiten	Faktor		
	Einfach/ Wenig	Mittel	Schwer/ Viel
Datenbewegung	0.2	0.3	0.4
Kontroll- und Sicherheitsprüfungen	0.2	0.3	0.4
Dateiorganisation	0.4	0.6	0.8
Arithmetische Operationen	0.1	0.3	0.5
Komponentenstruktur	0.1	0.2	0.3
Erwartete Änderungen in der Leistungsbeschreibung	0.2	0.4	0.6

Tabelle B - 13 : Tabelle zur Bestimmung von DT

DE ist der Faktor der Programmiererfahrung und kann folgende Werte annehmen:

- 0.8 - Chefprogrammierer,
- 1.0 - Programmierer,
- 1.6 - Programmieranfänger,
- 2.4 - in Programmierausbildung.

DO steht für die Schwierigkeiten in der DV-Abwicklung. Hierfür gibt es drei Klassen:

- 0.8 - keine oder wenige,
- 1.0 - normal und
- 1.3 - viele.

DW berücksichtigt die Wiederverwendbarkeit der Programme und variiert von 1.0 bis zu 1.2.

DS stellt den Einfluß der verwendeten Programmiersprache dar und ist:

- 0.85 - Assembler und
- 1.0 - Cobol oder Fortran.

(e) Verfahren von Surböck: [Noth84] / [Litke96]

1978 versuchte Surböck durch Modifikation des IBM-Handbuch-Verfahrens (vgl. Anhang B.4 (a)) eine Aufwandsabschätzung für die dort nicht erfaßten Phasen bis zur Entwicklung eines fachlichen Feinkonzeptes (vgl. Phase III, Kap2.1) zu erreichen.

Mit diesen Informationen, so Surböck, lägen dann zum Zeitpunkt der Projektfreigabe genügend Informationen vor, auch den Aufwand für die DV-technische Realisierung (vgl. Phase IV und V, Kap. 2.1) zu schätzen.

Mit seiner Methodik kann dann der Aufwand der Orientierung (T_O) (vgl. Phase I, Kap. 2.1), den der Systemplanung (T_S) (vgl. Phase II, Kap. 2.1) und den der fachlichen Realisierung (T_t) (vgl. Phase III, Kap2.1) in Manntagen (MT) geschätzt werden.
Für die Schätzung der DV-technischen Realisierung (vgl. Phase IV und V, Kap2.1) hat Surböck das IBM-Handbuch-Verfahren bis auf kleine Änderungen übernommen. Diese Modifikationen beziehen sich hauptsächlich auf den dort genannten Dateifaktor, der Tabelle B - 18 zu entnehmen ist. Die Tabelle B - 14 bis Tabelle B - 18 stammen aus [Noth84].

Orientierung: $T_O = t_O * (1 + g_a / 100)$, wobei t_O der Tabelle B - 14 und g_a der Tabelle B - 15 zu entnehmen sind

1. Wieviele Abteilungen sind beteiligt?	..... × 2 = ...
2. Wieviele Geschäftsstellen, Filialen usw. sind beteiligt?	..... × 6 = ...
3. Wieviel 100 Mitarbeiter sind durch das neue System betroffen?	.....
	t_o = ...

Tabelle B - 14 : Schätzung des Aufwands für Orientierung

1. Wie vertraut ist das Projektteam mit dem Arbeitsgebiet? völlig neu = 30, einiges Wissen = 15, schon einmal gemacht = 0, mehr als einmal = - 20	.....
2. Haben die zukünftigen Benutzer an der Vorstudie mitgearbeitet? nein = 20, teilweise = 10, intensiv = 0	.....
3. Wo wird gearbeitet? Am gewohnten Arbeitsplatz = 0, in der selben Stadt = 10, in einer anderen Stadt = 20	.....
4. Wird die Zusammenarbeit mit dem zukünftigen Benutzer leicht sein? ja = 0, nein = 10	.....
	g_a = ...

Tabelle B - 15 : Schätzung der Vertrautheit mit dem Arbeitsgebiet

Systemplanung: $T_S = t_S * (1 + g_a / 100 + g_k / 100)$, wobei t_S der Tabelle B - 15, g_a der Tabelle B - 16 und g_k der Tabelle B - 17 zu entnehmen sind

1. Art des zu entwickelnden Systems batch = 0, online = 5, realtime = 10				.....
2. Komplexität der Verarbeitung	einfach	mittel	komplex	
Daten kombinieren, verdichten, umordnen löschen	0	2	3	.....
Daten aufsuchen (Indextabellen, Adressketten usw.)	0	3	5	.....
Prüfung auf Formate, Plausibilität, Fehlerroutinen	0	3	5	.....
Logik (Rechnen, Vergleiche)	0	2	4	.....
Ansprüche an Optimierung und Sicherung (Overlays, Checkpoints)	0	2	3	.....
3. Behandlung von Ausnahmen manuell = 0, automatisch = 10				.....
4. Systembelastung unkritisch = 0, kritisch = 5				.....
5. Zahl von Subsystemen	..... × 5 =			.....
6. Zahl der Eingaben	..... × 1.5 =			.....
7. Zahl der Ausgaben	..... × 1.5 =			.....
8. Zahl der Transaktionen	..... × 2 =			.....
9. Zahl der verwendeten Datenbestände	..... × 3.5 =			.....
		t_s =		.....

Tabelle B - 16 : Schätzung des Aufwands für die Systemplanung

1. Systemverfügbarkeit: normal = 0, wichtig = 4, kritisch = 8	.....
2. Backupverfahren: manuell = 0, automatisch = 6	.....
3. Werden bestehende Arbeitsverfahren automatisiert? ja = 0, nein = 10	.....
4. Existiert bereits ein EDV-System? ja = 0, nein = 6	.....
5. Ist das zu entwickelnde System das erste seiner Art? ja = 10, nein = 0	.....
6. Werden neue DV-Verfahren eingesetzt? ja = 10, nein = 0	.....
7. Wird ein Verbundsystem oder Netzwerk entwickelt? ja = 10, nein = 0	.....
8. Ist spätere Erweiterung auf Online- oder Dialogbetrieb geplant? ja = 10, nein = 0	.....
9. Geplante Hardwaregröße: bis z. B. /370-135 = 0, z. B. /370-145 = 4, größer als z. B. /370-158 = 8	.....
	g_k =.....

Tabelle B - 17 : Schätzung der Komplexität

Für die Erstellung des Systemanforderungskataloges werden zusätzlich 20% von T_S benötigt, während für die Systementwicklung weitere 75% hinzukommen. Zuletzt wird für die Erarbeitung der funktionellen Spezifikation nochmals ein Zuschlag von 40% von T_S erhoben, so daß sich der Gesamtaufwand für die fachliche Realisierung T_t wie folgt errechnet:

$$T_t = T_O + 2.35 * T_S$$

Für Managementtätigkeiten sind weitere 16% des Gesamtaufwandes vorgesehen.

Neben einer Anpassung des Dateifaktors (Tabelle B - 18) des IBM-Handbuch-Verfahrens ersetzt Surböck den Zuschlag von 70% bis 110% zur Ermittlung der Rohprojektzeit durch eine Spanne von 50% bis 80%.
Damit läge der Gesamtaufwand des Projektes (analog zu B.3 (a)) zwischen

$$T_{pp} * 1.75 \text{ und } T_{pp} * 2.25.$$

	„Schwierigkeits-punkte"
Sequentieller Datenbestand	
ein Datensatzformat	1
mehrere Datensatzformate	2
variable Satzlänge	3
Nichtsequentieller Datenbestand (indexiert, direkt, usw.)	
ein Datensatzformat	3
mehrere Datensatzformate	4
Datenbestand mit Nicht-Standard-Zugriffsmethode (z. B. im PIOCS* programmiert)	8
jedes Druckformat	1

* Physical Input Output Control System

Tabelle B - 18 : Dateifaktor nach Surböck

(f) Zeit-Kosten-Planung: [Noth84]

Dieses Verfahren, das 1980 ebenfalls in einer Abteilung der Siemens AG entwickelt wurde, greift den Grundgedanken des IBM-Handbuch-Verfahren auf.
Nach Abschluß des fachlichen Feinkonzeptes (vgl. Phase III, Kap. 2.1) kann hiermit der Aufwand für

- das DV-Grobkonzept (vgl. Kap2.1, Phase I und II),
- das DV-Feinkonzept (vgl. Kap2.1, Phase III und IV),
- die Programmierung und Codierung (vgl. Kap2.1, Phase V), sowie
- den Test (vgl. Kap. 2.1, Phase VI),

abgeschätzt werden.

Für die Schätzung ist eine Zergliederung des Projektes in elementare Einheiten (mit ca. 1000 bis 1500 LoC bzw. mit einer mittleren Bearbeitungszeit von drei Monaten, gem. Siemens AG) nötig.
Danach wird der Programmieraufwand (P_A) im MM nach folgender Formel berechnet:

$$P_A = (D + V) * (P_k + P_e), \qquad \text{mit}$$

D: Dateifaktor; Summe der Gewichte für jede Ein- und Ausgabedatei (Tabelle B - 19)

	Dateien	Schwierigkeitsgrad/Gewicht	
		ein Format (einfacher Aufbau) wenig Datenfelder	mehrere Formate/komplizierter oder variablar Aufbau/viele Datenfelder
Anzahl Eingabe-Dateien mal Gewicht = Produkt Eingabe	Lochkarte	0,2 - 0,4	0,5 - 1,0
	Magnetband	0,5 - 0,9	1,0 - 1,5
Anzahl Ausgabe-Dateien mal Gewicht = Produkt Ausgabe	Magnetplatte sequentiell	0,5 - 0,9	1,0 - 1,5
	Magnetplatte indexsequentiell	1,0 - 1,4	1,5 - 1,9
Produkt Eingabe + Produkt Ausgabe = Produkt Summe	Kriterienbank	1,0 - 1,4	1,5 - 1,9
	Datenbank	1,2 - 1,9	2,0 - 2,5
Addition aller Produktsummen = Ergebnis Schritt 1	Terminal (Masken mit IFG, DPG oder FORMPLAG)		
	- erste Maske	1,0 - 1,5	-
	- Folgemasken	0,1 - 0,3	-
	Listen	0,2 - 0,9	1,0 - 1,5
	Sonstige [1]		

Tabelle B - 19 : Gewichte für Ein- und Ausgabedateien

V: Verarbeitungsfaktor;
Die Addition der Aufwandsstufen ergibt die ‘Summe je Aufwands-art’, deren Gewichtung der rechten Hälfte der Tabelle entnommen werden kann (Tabelle B-20)

Verarbeitung

Schritt 2.1
Bewertung jeder Zeile gemäß Aufwandsstufen.

Summe je Aufwandsart.

Aufwandsstufen	
Kein Aufwand	0
Aufwand gering	1
Aufwand mittel	2
Aufwand groß	3
Aufwand ungewöhnlich groß	4

Aufwandsarten/Zeilen	Stufe	Summe je Aufwandsart
o Datenbewegung Aufbau von Zwischenbereichen		
Löschroutinen von Datenfeldern, Tabellen		
Verdichten von Daten		
Selektieren / Umformatieren		
Sortierbereiche aufbauen		
Aufbau von Ausgabebereichen		
o Prüfungen Plausibilitätsprüfungen		
Grenzwertprüfungen		
Formatprüfungen		
o Tabellenverarbeitung (ein-, zwei-, dreistufige Tabellen) Tabellen füllen		
Tabellen lesen		
Technik des direkten Zugriffs und damit verbundenes Subskribieren		
o Arithmetische Operationen Addition/Subtraktion		
Multiplikation/Division		
weitere Rechenarten		
o Programmsteuerung/Programmverbindung Aufteilung der Funktionen in Steuer-, Teil- und Unterprogramme		
Gruppenwechsel		
Wiederanlaufroutinen		
Programmverbindungen		

Schritt 2.2
Summe je Aufwandsart in nebenstehender Tabelle aufsuchen, zugehöriges Gewicht ablesen, Plausibilität durch Vergleich mit den Grob-Kategorien „gering-mittel-schwer" überprüfen, in Gewicht-Liste eintragen. Addition der Gewichte je Aufwandsart = Faktor Verarbeitung.

Je Baustein, Schritt 2

GEWICHTE					
Oberer Teil	Gewichte-Kategorie „gering"				
Mittelteil	„mittel"				
Unterer Teil	„schwer"				
Summe	Daten-beweg.	Prü-fung	Tabel-len	Arithm. Op.	Progr. Steuer.
0	0,10	0,10	0,10	0,10	0,10
1	0,20	0,53	0,70	0,37	0,37
2	0,30	0,96	1,30	0,64	0,64
3	0,40	1,40	1,90	0,90	0,90
4	0,50	1,50	2,00	1,00	1,00
5	0,64	2,10	2,80	1,47	1,32
6	0,78	2,70	3,60	1,94	1,64
7	0,92	3,30	4,40	2,40	1,96
8	1,06	3,90	4,50	2,50	2,28
9	1,20	4,00	5,25	3,00	2,60
10	1,34	4,83	6,00	3,50	2,90
11	1,48	5,66	6,75	4,00	3,00
12	1,62	6,50	7,50	4,50	3,38
13	1,78	-	-	-	3,76
14	1,90	-	-	-	4,14
15	2,00	-	-	-	4,52
16	2,15	-	-	-	4,90
17	2,30				
18	2,45				
19	2,60				
20	2,75				
21	2,90				
22	3,05				
23	3,20				
24	3,50				

Gewichte-Liste

Datenbewegung	
Prüfungen	
Tabellen-V.	
Arithm. Operat.	
Programmsteuer.	
Faktor Verarbeit.	

Zeile 2 Ergebnisblatt →

Tabelle B - 20 : Detaillierte Ermittlung des Verarbeitungsfaktors

P_k: Problemkenntnisfaktor
(Tabelle B - 22)

P_e: Programmiererfahrungs-faktor
(Tabelle B - 21)

Erforderliche Problem-Kenntnisse	Vorhandene Problem-Kenntnisse keine/geringe	einige	viel
gering	0,75	0,25	0,10
einige	1,25	0,75	0,25
viel	2,00	1,25	0,75

Tabelle B - 21 : Gewichtung zur Bestimmung des Problem-kenntnisfaktors

Programmier-Praxis, auch in Bausteintechnik, Strukturierter Programmierung, Benutzung von Programmierkonventionen	Gewicht
Programmierer (über 2 Jahre Praxis)	0,75 — 1,50
Programmierer (bis zu 2 Jahren Praxis)	2,00 — 3,00
Programmier-Anfänger (bis zu 1 Jahr Praxis)	3,50 — 4,00

Tabelle B - 22 : Ermittlung des Programmiererfahrungsfak-tors

Als letztes werden dann Zuschlagssätze für die interne Organisation (Tabelle B - 23) und für interne Verluste (Tabelle B - 23) berücksichtigt.

Zuschlagssätze:

Zeitaufwand für Organisation

Je Baustein, Schritt 5

Aufwandsarten	Stufe
Phasenorganisation	
Standards, Konventionen festlegen	
Einlesen in fachliches Grob-/Feinkonzept	
DV-technische Systemanalyse	
Einarbeitung in bestehende Software	
Unvorhergesehene fachliche Änderungen	
Erstellen Testdaten	
Unterstützung der Datenübernahme	
Summe	

Aufwands-Stufen	
Kein Aufwand	0
Aufwand gering	1
Aufwand mittel	2
Aufwand groß	3
Aufwand ungewohnlich groß	4

Summe	0	1-3	4	5-7	8	9-11	12	13-15	
Prozent	10	11-19	20	21-29	30	31-39	40	41-49	
Summe	16	17-19	20	21-23	24	25-27	28	29-32	
Prozent	50	51-74	75	76-99	100	101-124	125	126-150	

Ermittelten Prozentsatz in Zeile 8 des Ergebnisblattes eintragen.

Tabelle B - 23 : Bestimmung des Zuschlagssatzes für Organisation

Zeitaufwand für Verlustzeiten innerhalb des Auftrages

Je Baustein, Schritt 6

Aufwandsarten	Stufe
Fehlende Maschinenzeit/ Systemausfall	
Keine freien Terminals	
Wartezeit auf Gesprächspartner	
Verwaltung fur das Projekt (Beschaffen, Abrechnen)	
Sonstiges für das Projekt (Vorträge, Besprechungen)	
Summe	

Aufwands-Stufen	
Kein Aufwand	0
Aufwand gering	1
Aufwand mittel	2
Aufwand groß	3
Aufwand ungewohnlich groß	4

Summe	0	1-2	3-4	5-6	7-8	9-10	11	12	13	14	15	16
Prozent	5	6	7	8	9	10	12	14	16	18	20	22
Summe	17	18	19	20								
Prozent	24	26	28	30								

Ermittelten Prozentsatz in Zeile 10 des Ergebnisblattes eintragen.

Tabelle B - 24 : Ermittlung des Zuschlagssatzes für interne Verluste

(g) Business-Objectives-Verfahren: [Litke96]

Dieses Verfahren, Anfang der 80er Jahre von Howard A. Rubin entwickelt, orientiert sich an den ‘Business Objectives’ auf Basis eines ‘Business Information Flow’-Diagrammes.

Vorgehensweise hierbei ist die Beantwortung eines Fragenkataloges, bestehend aus 25 Fragen, die drei Gruppen zugeordnet sind.

Gruppe 1:

1. Anzahl der betroffenen funktionalen Einheiten der Firma
   Wieviele organisatorische Einheiten (außer Entwicklung) sind in das Projekt involviert (Projektdefinition, Entwurf, Abnahme etc.) ?
2. Anzahl von Firmenstandorten
   Zum Zwecke der Informationsgewinnung. Mehrere Standorte in der gleichen Stadt, die nicht weiter als eine halbe Stunde voneinander entfernt sind, zählen als ein Standort.
3. Wieviele Menschen (in 100) sind vom Projekt betroffen ?
4. Den Entwicklern ist diese Anwendung (a) völlig neu, (b) einigermaßen vertraut oder (c) sehr vertraut
5. War die Analyse der Projektziele (a) eingehend deutlich, (b) gut oder (c) schwach ?

6. Wird die Reisetätigkeit (a) normal, d.h. 0-2 Besuche in anderen Standorten, (b) von Bedeutung oder (c) werden Übersiedlungen nötig sein ?
7. Personal wird (a) vom Entwickler oder (b) vom Entwickler und Anwender gestellt
8. Wieviele Personen müssen zur Durchführung der Arbeit umgesiedelt werden ?

Gruppe II:

9. Anzahl der größeren Subsysteme
10. Anzahl der externen Eingaben
11. Anzahl der externen Ausgaben
12. Anzahl der logischen Dateien
13. Anzahl der Online-Abfragetypen

Gruppe III:

14. Wie beeinflußt das Vorwissen der Anwender das Projekt ?
    (a) helfend oder (b) störend
15. Führt der Anwender bereits die beabsichtigten Geschäftsfunktionen durch ?
    (a) ja oder (b) nein
16. Wenn ja, sind sie automatisiert ?
    (a) ja oder (b) nein
17. Ist das System das erste seiner Art ?
    (a) ja oder (b) nein
18. Ist die Hardware/Software-Umgebung (a) komplex, (b) durchschnittlich oder (c) einfach ?
19. Ist die Systemverfügbarkeit (a) kritisch, (b) wichtig oder (c) normal ?
20. Gibt es spezielle Backup-Anforderungen ?
    (a) ja oder (b) nein
21. Gibt es spezielle Recovery-Anforderungen ?
    (a) ja oder (b) nein
22. Sind Data Traffic Load oder Performance kritisch ?
    (a) ja, (b) nein oder (c) sowohl als auch
23. Ist die Anwendung (a) Batch, (b) Online oder (c) Realtime ?
24. Die Verarbeitungslogik ist (a) einfach, (b) durchschnittlich oder (c) komplex
25. Die Behandlung eines Ausnahmefalls erfolgt (a) von Hand oder (b) automatisch

Leider ist in der einzigen Quelle für dieses Verfahren (vgl. [Litke96]) keine Werte bzw. keine Umrechnung des Fragenkataloges in den damit verbundenen Aufwand angegeben. Es wird lediglich auf das Software-Tool ‘Estimacs’ verwiesen, daß eine solche Auswertung, basierend auf einer Datenbasis aus 15.000 Projekten, vornehmen kann. Die Schätzungenauigkeit dieses Tools wird mit ñ 15% angegeben.

(h) Function-Point-Verfahren: [Noth84] / [Litke96]

Diese, von Allan Albrecht bei der IBM Corporation entwickelte Methode, wird seit 1981 bei der IBM Deutschland GmbH eingesetzt.
Es ist, neben der Weiterentwicklung Data-Point (vgl. B.4 (i)), das bekannteste Verfahren dieser Gruppe.

Die Grundidee ist, daß der Aufwand eines Projektes vom Schwierigkeitsgrad und vom Umfang abhängt. Diese werden durch die Summe der sogenannten 'Function-Points' (kurz: FP) dargestellt.
Zunächst muß das Projekt in Funktionen bzw. Geschäftvorfälle (z.B. Anlegen eines neuen Kunden) zerlegt werden. Diese werden dann anhand der drei Kriterien 'leicht', 'mittel' und 'komplex' aus Sicht des zukünftigen Anwenders klassifiziert. Dadurch wird jeder in der Software zu realisierenden Funktion ein Wert von drei bis fünfzehn zugeordnet. Summiert man alle diese Werte auf, so erhält man als Summe die (Roh-) FP.
Diese werden dann dem Entwicklungskontext zugeordet, indem man vierzehn Faktoren entsprechend ihrer Einwirkung mit Zahlen bewertet. Dieser sog. 'Degree of Influence', welcher zwischen 0 und 70 liegt, wird dann durch 100 geteilt und zu 0.65 addiert. Somit erhält man einen Multiplikator zwischen 0.65 und 1.35, der dann die (Roh-) FP anpaßt. Diese vierzehn Faktoren und die somit entstehende Modifikationsspanne basiert auf den empirischen Untersuchungsergebnissen von Albrecht.
Anhand einer, durch die Daten bereits realisierter Projekte entstandener Kurve, werden dann die FP in den Aufwand in MM überführt.

Nun soll das Verfahren im Detail vorgestellt werden:

Für jede Funktion bzw. Geschäftsvorfall sind folgende Größen, die nach dem Abschluß der Phase III - Funktionsbeschreibung (vgl. Kap2.1) bekannt sind, zu ermitteln:

- Eingabedaten bzw. externe Inputs,
- Ausgabedaten bzw. externe Outputs,
- Datenbestände bzw. interne Dateien,
- Datenbestände für andere Verfahren (Interfaces) bzw. externe Schnittstellen und
- externe Abfragen.

Für die Summe der FP werden nur sinnvolle Klassifizierungen addiert, da nicht von jeder Funktion Eingabedaten, Ausgabedaten, Datenbestände, Interfaces und Abfragen berührt sind (vgl. Anhang C.2, um dieses zu verdeutlichen).

Eingabedaten: Hierzu gehören jede Art von Eingabe, sei es durch direkte Eingabe durch den Anwender, durch Einlesen vom Datenträger oder ähnlichem, sowie durch Transaktion von einer anderen Anwendung aus.

Anforderungen:	einfach	mittel	komplex
Datenelementtypen	wenige (1-5)	mehrere (6-10)	viele (> 10)
logische Datengruppen (bei Bildschirminhalten)	wenige	mehrere	viele
Eingabeprüfung	formal	formal / logisch	form./log./ DB-Zugriff
Ansprüche an die Bedienerführung	gering	mittel	hoch
Cursorhandhabung	einfach	mittel	schwierig
Gewicht	3	4	6

Tabelle B - 25 : Klassifizierung der Eingabedaten

Ein Geschäftsvorfall wird nun anhand jeder einzelnen Zeile der o.g. Tabelle klassifiziert. Die Spalte, die anschließend als einzige die meisten 'Treffer' besitzt, gibt dann die Klassifikation des ganzen Geschäftsvorfalls wider.
Beispiel: Datenelementtypen - wenige, logische Dateigruppen - wenige, Ansprüche an die Bedienerführung - hoch, Cursorhandhabung - mittel
$\Rightarrow$ Klassifizierung - einfach, Gewicht - 3.

Anhand der Bezeichnungen ist deutlich, daß jemand, der dieses Verfahren anwenden möchte, dieses im Geiste an heutige Verhältnisse anpassen muß. So würde man z.B. nicht von Cursorhandhabung, sondern eher von Bedienbarkeit oder Handhabung reden.

Zu den Eingabedaten läßt sich noch bemerken, daß z.B. Transaktionen, die dem Hinzufügen, Löschen oder Ändern von Daten dienen, jeweils als verschiedene Eingaben zu betrachten sind, auch wenn dieses in der fertigen Software über ein und den selben Dialog realisiert werden würde. Ist aber die Verarbeitungslogik beim Neuanlegen die selbe, wie beim Verändern, so werden diese Funktionen als eine betrachtet. Daher wird jede Eingabe nur einmal betrachtet, auch wenn diese auf verschiedenen Wegen erfolgen kann, wie z.B. per Funktionstaste und aus dem Menü heraus.

Ausgabedaten: Dieses sind alle, durch die Verarbeitung der Eingabe entstandenen Daten eines Geschäftsvorfalls, egal ob auf dem Bildschirm, in eine Datei oder als Transaktion an Drucker oder andere Anwendungen.

Anforderungen:	einfach	mittel	komplex
Listenspalten	wenige (1-6)	mehrere (7-15)	viele (> 15)
Umsetzen von Datenelementen	einfach	mehrfach	komplex
Dateizugriffe und Verknüpfungen	wenige	mehrere	viele
Anforderungen an die Performance	keine	wenige	mehrere
Gewicht	4	5	7

Tabelle B - 26 : Klassifizierung der Ausgabedaten

Grundsätzlich geschieht die Klassifikation der Ausgabedaten analog zu der der Eingabedaten.
Es muß aber jede Ausgabe gezählt werden, wenn sie unterschiedliches Format besitzt oder aus unterschiedlichen Verarbeitungsteilen entstanden ist.
Nachrichten, die ohne eigene Verarbeitungslogik entstehen, wie z.B. Bestätigungsanzeigen, sind nicht zu berücksichtigen, während Fehlermeldungen, die sich auf logische Fehler beziehen, als zusätzliche Ausgabe gelten.

Datenbestände: Als Datenbestände gelten jede unterschiedliche, maschinenlesbare Datei oder bei Datenbanken, jede aus Anwendersicht logische Datengruppe, die erstellt, benutzt oder verändert wird (Entität).

Anforderungen:	einfach	mittel	komplex
Anzahl versch. Datenelemente	wenige (1-20)	mehrere (21-40)	viele (> 40)
Performanceeinflüsse	keine	wenige	viele
Wiederanlaufeinflüsse	keine	wenige	viele
DDP-Konzept der Daten	nein	-	ja
Gewicht	7	10	15

Tabelle B - 27 : Klassifizierung der Datenbestände

Werden Datengruppen während des Programmlaufes nicht neu aufgebaut, sondern lediglich verwendet oder gepflegt, so ist generell die Klasse ‘einfach’ anzusetzen.

Interfaces: Datenbestände für andere Anwendungen, deren Datensätze keine Transaktionen sind, werden zu den Interfaces gezählt.
Ihre Bewertung verläuft analog zu Bewertung der Datenbestände der eigenen Applikation.

Anforderungen:	einfach	mittel	komplex
Datenelementtypen	wenige (1-20)	mehrere (21-40)	viele (> 40)
Anzahl Schlüsselbegriffe / Satzarten	wenige (1)	mehrere (2)	viele (> 2)
Tabellendimension	einfach (1)	mittel (2)	hoch (3)
Gewicht	5	7	10

Tabelle B - 28 : Klassifizierung der Interfaces

Abfrage: Erzeugt eine Online-Eingabe eine Online-Ausgabe und dienen die Eingaben nur zu Steuerzwecken des Suchvorgangs, d.h. es werden keine Datenbestände verändert, so liegt eine Abfrage vor.
Gezählt wird jede unterschiedlich formatierte Abfrage, wobei Abfragen, die viele Schlüsselangaben und Verarbeitungsschritte benötigen, als Ein- und Ausgabe zählen.

Anforderungen:	einfach	mittel	komplex
Datenelementtypen	wenige (1-5)	mehrere (6-10)	viele (> 10)
logische Datengruppen (bei Bildschirminhalten)	wenige	mehrere	viele
Eingabeprüfung	formal	formal / logisch	form./log./ DB-Zugriff
Ansprüche an die Bedienerführung	gering	mittel	hoch
Cursorhandhabung	einfach	mittel	schwierig
Gewicht	3	4	6

Tabelle B - 29 : Klassifizierung der Abfragen

Die so gewonnene Summe von (Roh-) FP wird dann anhand eines Punktekataloges dem Entwicklungkontext angepaßt.
Wichtig hierbei ist, daß nicht das bloße Vorhandensein eines Zustandes, sondern vielmehr sein Einfluß auf die Entwicklung bewertet wird.
Folgende Skala wird hierbei verwendet (Ausnahmen: Punkte 9 und 10 des Kataloges):

0 = kein Einfluß,

1 = gelegentlicher Einfluß,

2 = mäßiger Einfluß,

3 = mittlerer Einfluß,

4 = bedeutsamer Einfluß und

5 = starker Einfluß.

Somit läßt sich der ‘Degree of Influence’ wie folgt bestimmen:

$$doi = \sum_{Punkte1-14} \div 100 + 0.65$$

1. Die in der Anwendung verwendeten Daten werden über Kommunikationseinrichtungen empfangen oder gesendet
2. Die Verwaltung oder die Verarbeitung der Daten wird nach dem DDP-Konzept durchgeführt (Verteilte Daten- oder Verarbeitungsfunktionen; DistributedData-Processing)
3. Bezüglich Design, Implementierung und Wartung nimmt ein günstiges Antwortzeitverhalten einen hohen Stellenwert ein
4. Das in Punkt 3 genannte Antwortzeitverhalten soll auf einem stark ausgelasteten System erreicht werden
5. Design, Implementierung und Wartung werden durch eine hohe Transaktionsrate beeinflußt
6. Die Dateneingabe erfolgt online
7. Die Online-Dateneingabe ist dialogorientiert, und eine Transaktion wird aus mehreren Verarbeitungsschritten aufgebaut
8. Die Dateien und Datenbanken werden online gewartet
9. Es liegt eine komplexe Verarbeitung vor, die viele Verknüpfungen, Entscheidungen, logische und mathematische Gleichungen und Ausnahmeregelungen beinhaltet. Es ist für die Bewertung eine Spanne von 0 bis 50 vorgesehen, wobei Werte zwischen 40 und 50 Systeme beschreiben, die in hohem Maße komplexe Funktionen, wie z.B. Stücklistenauflösung, Lageroptimierung und ähnliches enthalten
10. Bei Entwicklung, Implementierung und Wartung ist eine Wiederverwendung der Programme in anderen Anwendungen zu berücksichtigen. Die Bewertung erfolgt nach dem Prozentanteil des wiederverwendbaren Codes: bis 10% = 0, 11% bis 20% = 1, 21% bis 30% = 2, 31% bis 40% = 3, 41% bis 50% = 4 und über 50% = 5
11. Bei Design, Implementierung und Wartung sind Konvertierungen zu berücksichtigen
12. Für die Erleicherung der späteren Bedienung sind bestimmte Prozeduren vorzusehen, wie z.B. automatischer Systemstart, Backup, Wiederanlauf, Minimierung von Bandbedienung, Papierbereitstellung
13. Bei Design, Implementierung und Wartung muß berücksichtigt werden, daß die Anwendung in mehreren Funktionen oder lokal getrennten Stationen installiert wird
14. Bei der Entwicklung der Anwendung wird vorgesehen, dem Benutzer die Bedienung und einen eventuellen Änderungsdienst zu erleichtern, wie z.B. variable Abfragemöglichkeiten, Verwendung von Benutzertabellen für Parameter

Hat man so die endgültige Zahl von Function-Points ermittelt, so kann diese anhand einer entwicklungskontextsensitiven Kurve in den Aufwand überführt werden.

Diese Kurve entsteht, indem man die FP bereits abgeschlossener Projekte in Relation zum tatsächlich aufgebrachten Aufwand stellt. Diese Daten - je mehr, um so genauer die Transformation - dienen dann als Stützpunkte der Kurve. FP-Aufwand-Zuordnungen, die nicht in der Tabelle vorhanden sind, können somit entweder mathematisch oder graphisch interpoliert werden.

Die in Abbildung B - 3 gezeigte Kurve stellt die Ur-Kurve von IBM dar, die aufgrund der dort abgeschlossenen Projekte ermittelt wurde.
Diese kann auch als Grundlage dienen, die dann durch unternehmensspezifische Daten iterativ angepasst wird.

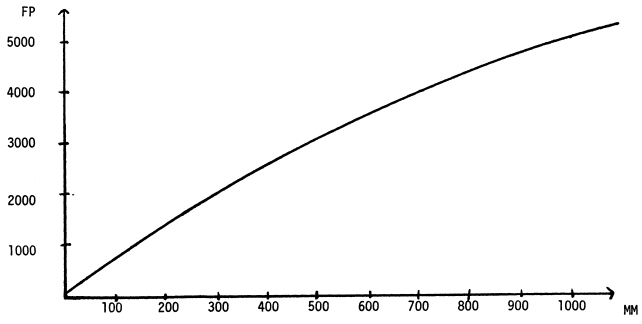

Abbildung B - 3 : Kurve zur FP-Aufwand-Transformation
Quelle: [Noth84]

FP	MM	FP	MM	FP	MM
50	2,3	1.200	145,2	3.300	547,0
100	5,6	1.300	161,3	3.400	568,8
150	9,5	1.400	177,7	3.500	590,8
200	13,9	1.500	194,6	3.600	613,1
250	18,6	1.600	211,7	3.700	635,5
300	23,6	1.700	229,3	3.800	658,1
350	28,9	1.800	247,1	3.900	680,9
400	34,4	1.900	265,3	4.000	703,9
450	40,1	2.000	283,7	4.100	727,0
500	46,1	2.100	302,4	4.200	750,4
550	52,2	2.200	321,5	4.300	773,9
600	58,5	2.300	340,7	4.400	797,5
650	65,0	2.400	360,3	4.500	821,4
700	71,6	2.500	380,1	4.600	845,4
750	78,4	2.600	400,1	4.700	869,6
800	85,3	2.700	420,4	4.800	893,9
850	92,4	2.800	441,0	4.900	918,4
900	99,6	2.900	461,7	5.000	943,1
950	106,9	3.000	482,7	5.100	967,9
1.000	114,4	3.100	503,9	5.200	992,8
1.100	129,6	3.200	525,4		

Tabelle B - 30 : Zuordnung FP zum Aufwand

Es gibt inzwischen verschiedene Ansätze, dieses Verfahren zu optimieren bzw. der Entwicklung der Softwareentwicklung anzupassen.
Zwei seien an dieser Stelle genannt. Um im weiteren Verlauf dieser Arbeit dazwischen unterscheiden zu können, bekommt die Grundvariante, wie sie hier vorgestellt wurde, die Kennziffer I.

Während Tabelle B - 30 die in [Noth84] angegebene Zuordnungskurve von FP nach Aufwand in MM beschreibt, wird in [Litke96] folgende Kurve angegeben (Kennziffer II):

FP	MM	FP	MM	FP	MM
150	5	900	65	2.300	205
200	9	950	70	2.400	217
250	13	1.000	75	2.500	229
300	17	1.100	84	2.600	242
350	21	1.200	93	2.700	255
400	25	1.300	102	2.800	269
450	29	1.400	111	2.900	284
500	33	1.500	121	3.000	299
550	37	1.600	131	3.100	315
600	41	1.700	141	3.200	331
650	45	1.800	151	3.300	350
700	49	1.900	161	3.400	369
750	53	2.000	171	3.500	390
800	57	2.100	182	3.600	415
850	61	2.200	193	3.700	441

Tabelle B - 31 : Zuordnung FP zum Aufwand

Als dritte Variante (Kennziffer III) kommt eine Modifikation der Klassifizierungstabellen, die Mitte der 80er-Jahre bei IBM Deutschland entwickelt wurde:

- die gesonderte Bewertung der ‘Datenbestände für andere Anwendungen’ (Interfaces) wird in die Gruppe ‘Datenbestände’ integriert,
- neu hinzu kommt die Klasse der ‘Referenzdaten’, also der Daten, die nicht komplett verarbeitet werden, sondern nur der Bereitstellung von Informationen dienen.

  Sie werden nach folgender Tabelle klassifiziert:

Anforderungen:	einfach	mittel	komplex
Anzahl der unterschiedlichen Datenelemente	einige	mehrere	viele
Anzahl Schlüsselbegriffe / Satzarten bzw. Dimension	1	2	> 2
Gewicht	5	7	10

Tabelle B - 32 : Klassifizierung der Referenzdaten

- die Einflußfaktoren werden von 14 auf 7 reduziert, während die Formel für die Berechnung des ‘Degree of Influence’ unverändert bleibt:
  1. Verpflechtung mit anderen Projekten
  2. Die Verwaltung der Daten oder die Verarbeitung wird dezentral (DDP-Konzept) durchgeführt
  3. Design, Implementierung und Wartung werden durch eine hohe Transaktionsrate beeinflußt
  4. Es liegt eine komplexe Verarbeitung vor, die viele Verknüpfungen, Entscheidungen, logische und mathematische Gleichungen und Ausnahmeregelungen beinhaltet. Spanne hierbei: 0 - 30
  5. Bei Entwicklung, Implementierung und Wartung ist eine Wiederverwendung der Programme in anderen Anwendungen zu berücksichtigen. Die Bewertung erfolgt nach dem Prozentanteil des wiederverwendbaren Codes: bis 10% = 0, 11%, bis 20% = 1, 21% bis 30% = 2, 31% bis 40% = 3, 41% bis 50% = 4 und über 50% = 5
  6. Bei Design und Implementierung sind Konvertierungen zu berücksichtigen
  7. Bei der Entwicklung der Anwendung wird vorgesehen, dem Benutzer die Bedienung und einen eventuellen Änderungsdienst zu erleichtern, wie z.B. variable Abfragemöglichkeiten, Verwendung von Benutzertabellen für Parameter

IBM führte diese Modifikationen hauptsächlich durch, um die Handhabbarkeit des Verfahrens zu erhöhen. Leider wird in [Noth84], einzige Quelle für diese Modifikationen, keine Zuodnungskurve für die ‘neuen’ FP angegeben.

(i) Data-Point-Verfahren: [Litke96]

Dieses 1991 von Sneed entwickelte Verfahren ist eine Weiterentwicklung der Function-Point-Methode (vgl. B.4 (h)).
Grundgedanke hinter dieser Methode ist die Annahme, daß die Komplexität eines Programmes, welches daten- oder objektbezogen entwickelt wird, maßgeblich von der Modellierung des Datenmodells abhängt.

Zur Berechnung der Data-Points (kurz: DP) werden nicht die Geschäftsvorfälle betrachtet, sondern die Informationsentitäten (logische Sätze und Tabellen der Zieldatenbank, auf die das System zugreifen soll) und Nachrichten (Bildschirmmasken, Berichte, Datenübergaben und Messages an andere Programme).
Die Informationsentitäten ergeben sich aus der Daten-, die Nachrichten aus der Kommunikationsanalyse. Vorteil hieraus ist, daß diese Methode früher einsetzbar ist als die Function-Point-Methode. Ein weiterer Vorteil der Data-Point-Methode gegenüber der Function-Point-Methode ist, daß diese Qualitätsmerkmale stärker berücksichtigt[1].

Die Klassifizierung der Informationstentitäten geschieht hierbei in einer Liste, die folgende Spalten besitzt:

Name	Anz. Attribute	Anz. Schlüssel	Integrationsgrad	Nutzung	DP

Bezeichnung	Bedeutung	Bewertung [DP]
Anz. Attribute	Anzahl der Felder der Entität	1 DP / Attribut
Anz. Schlüssel	Anzahl der Primär- und Sekundärschlüssel	4 DP / Schlüssel
Integrationsgrad	Integrationsgrad der Entität: niedrig	2 DP
	mittel	4 DP
	hoch	8 DP
Nutzung	Entität wird zur Eingabe, zur Ausgabe oder sowohl zur Ein- als auch zur Ausgabe benutzt	Ausgabe, Ein- und Ausgabe (Objekt wird geschrieben): $f = 1{,}1$ nur Eingabe: $f = 1$
Die DP der Informationsentität errechnen sich nun aus der Summe der 2. bis 4. Spalte, multipliziert mit f.		

Tabelle B - 33 : Klassifizierung der Informationsentitäten

[1] Die im weiteren Verlauf angegebenen Qualitätsmerkmale stammen aus dem Pflichtenheft der Firma TEC für das Projekt ‘AiRMan’ vom 11.02.1999 und stellen einen Merkmalskatalog dar, der angelehnt an die ISO 9000ff für Software-Qualität ist.

Analog dazu werden die Nachrichten wie folgt klassifiziert:

Name	Anz. Felder	Anz. Sichten	Komplexitätsgrad	Nutzung	DP

Bezeichnung	Bedeutung	Bewertung [DP]
Anz. Felder	Bei Masken alle Felder, die vom Programm oder vom Benutzer gefüllt werden müssen, bei Listen alle Kopf- und Fußfelder, alle Datenreihen und alle Gruppenwechselfenster	1 DP / Feld
Anz. Sichten	Anzahl der Informationsobjekte, die in der Nachricht vertreten sind	4 DP / Sichten
Komplexitätsgrad	niedrig	2 DP
	mittel	4 DP
	hoch	8 DP
Nutzung	Nachricht dient der Eingabe, der Ausgabe oder sowohl der Ein- als auch Ausgabe	Eingabe, Ein- und Ausgabe: $f = 1{,}1$ Ausgabe: $f = 1$
Die DP der Informationsentität errechnen sich nun aus der Summe der 2. bis 4. Spalte, multipliziert mit *f*.		

Tabelle B - 34 : Klassifizierung der Nachrichten

Die Summe aller DP aus Informationsentitäten und Nachrichten wird nun mit einem Projektumgebungsfaktor multipliziert, so daß die DP in Abhängigkeit des Entwicklungskontextes um bis zu 15% erhöht oder um bis zu 25% gesenkt werden können.
Diese Faktoren sind folgende:

Anforderung					
Projektverteilung	mehrere Länder	mehrere Orte	ein Ort	mehrere Räume	ein Raum
Projekterfahrung	keine	mäßig	mittel	gut	sehr gut
Projektkenntnisse	keine	mäßig	mittel	gut	sehr gut
Projektautomation	keine	teilweise	halb	überwiegend	voll
Rechenbedingungen	schlecht	mäßig	mittel	gut	sehr gut
Projektunterstützung	keine	wenig	mittel	gut	sehr gut
Qualitätssicherung	keine	teilw. autom.	halb autom.	überwiegend autom.	voll autom.
Spezifikationsformalismen	informal	strukturiert	semi-formal	formal	formal-grafisch
Programmiersprache	1. Generation	2. Generation	3. Generation	4. Generation	5. Generation
Testautomation	keine	bis 26%	bis 50%	bis 75%	bis 100%
Gewicht	1	2	3	4	5

Tabelle B - 35 : Gewichtung der Qualitätsmerkmale

Somit läßt sich der ‘degree of Influence’ bei der Data-Point-Methode wie folgt bestimmen:

$$doi = \left(125 - \sum_{Qualitätsmerkmale} \right) \div 100$$

Desweiteren wird darüber hinaus noch ein Qualitätsfaktor (*qf*) berücksichtigt[2], der ebenfalls die Roh-DP modifiziert.
Hierzu werden für das Projekt nachfolgende Gruppen von Merkmalen bewertet, danach wird der Durchschnitt der einzelnen Gruppen errechnet und der Durchschnitt von diesen Werten wiederum ergibt den *qf*.
Die Bewertung erfolgt in den Stufen sehr gut (1.5), gut (1.25), normal (1.0) und nicht relevant (0.5):

- Funktionalität:
  - Angemessenheit
  - Richtigkeit
  - Interoperabilität
  - Ordnungsmäßigkeit
  - Sicherheit
- Zuverlässigkeit:
  - Reife
  - Fehlertoleranz
  - Wiederherstellbarkeit
- Benutzbarkeit:
  - Verständlichkeit
  - Erlernbarkeit
  - Bedienbarkeit
- Effizienz:
  - Zeitverhalten
  - Verbrauchsverhalten
- Änderbarkeit:
  - Analysierbarkeit
  - Modifizierbarkeit
  - Stabilität
  - Prüfbarkeit
- Übertragbarkeit:
  - Angepaßtheit
  - Installierbarkeit
  - Konformität
  - Austauschbarkeit

Die gewichtete DP-Zahl erhält man nun durch die Multiplikation der DP-Summe mit dem *doi* und dem *qf*.

Anschließend werden die DP (analog zu den FP) durch eine Erfahrungskurve (nebenstehende Tabelle B - 36 gibt die Werte hierzu an) in den Aufwand in MM überführt.

DP	MM	DP	MM	DP	MM
80	2	1.440	44	2.800	115
160	4	1.520	47	2.880	120
240	6	1.600	50	2.960	125
320	8	1.680	54	3.040	130
400	10	1.760	58	3.120	135
480	12	1.840	62	3.200	140
560	14	1.920	66	3.280	146
640	16	2.000	70	3.360	152
720	18	2.080	74	3.440	158
800	20	2.160	78	3.520	164
880	23	2.240	82	3.600	170
960	26	2.320	86	3.680	176
1.040	29	2.400	90	3.760	182
1.120	32	2.480	95	3.840	188
1.200	35	2.560	100	3.920	194
1.280	38	2.640	105	4.000	200
1.360	41	2.720	110		

Tabelle B - 36 : Data-Point-Produktivitätstabelle

[2] Die angegebenen Qualitätsmerkmale stammen aus dem Pflichtenheft der Firma TEC für das Projekt ‘AiRMan’ vom 11.02.1999 und stellen einen Merkmalskatalog dar, der angelehnt an die ISO 9000ff für Software-Qualität ist.

B.5 Methoden basierend auf teilprodukt-spezifischen Kenngrößen

(a) Integriertes Verfahren zur Aufwandsabschätzung (INVAS): [Noth84]

Diese Methode wurde 1980 im Rahmen eines Forschungsprojektes an der Universität Köln in Zusammenarbeit der Siemens AG entwickelt.
Hierbei handelt es sich nicht um ein festgeschriebenes Verfahren, sondern um eine Anleitung, wie die in verschiedenen Entwicklungsumgebungen signifikanten Kenngrößen ermittelt und zur Auwandsabschätzung verwendet werden können.

Um INVAS anwenden zu können, muß das Softwareprojekt in 'Meilenstein'-Ergebnisse bzw. Teilprodukte zerlegt werden. Diese eindeutig beschriebenen Leistungseinheiten des Entwicklungsprozesses eigenen sich zur Ermittlung des Aufwands, da

- eindeutig Anfang und Ende ermittelbar sind,
- eine weitgehend unmißverständliche Beschreibung möglich ist,
- eine überschaubare Auswahl aus den verschiedensten Einflußfaktoren und deren Ausprägung getroffen werden kann und
- sinnvolle und aussagefähige Kenngrößen zur Bestimmung des Aufwandes festgelegt werden können.

Als Beispiel hierzu diene die bei der Siemens AG vorgenommene Softwarestrukturierung:

MEILENSTEIN - ERGEBNISSE	TÄTIGKEITS - ARTEN	MASSGRÖSSEN
Anforderungs-katalog	- Requirements erstellen - Review	. Anforderungen . DIN A4-Seiten
Pflichten-heft	- Studie konzipieren - Review	. Benutzer-funktionen . DIN A4-Seiten
Leistungs-beschreibung/ Systementwurf	- Fachkonzept erstellen - Review	. Systeme . System-schnittstellen
Design/ Funk-tionsentwurf	- Design erstellen - Review	. Komponenten . Subsystem-schnittstellen
Komponenten-spezifikation	- Spezifizieren - Review	. Komponenten-schnittstellen . Programmver-zweigungen
Komponenten codiert und getestet	- Codieren - Code-Review - Testen - Diagnostizieren	. LOC . Programmver-zweigungen . Testfälle

Abbildung B - 4 : Teilproduktstrukturierung, Bsp. Siemens AG (a), Quelle: [Noth84]

MEILENSTEIN - ERGEBNISSE	TÄTIGKEITS - ARTEN	MASSGRÖSSEN
Subsysteme/Funk-tionen integriert und getestet	- Integrieren - Technische Dokumentation - Testen - Diagnostizieren	. Komponenten . Subsystem-schnittstellen . Testfälle
Systeme integriert und getestet	- Integrieren - Technische Dokumentation - Testen - Diagnostizieren	. Subsysteme . System-schnittstellen . Testfälle
Gesamtsystem getestet und gemessen	- Testen - Diagnostizieren - Messen (Performance) - Technische Dokumentation	. Testfälle . Systeme
Produktabnahme/ Benutzerdoku-mentation	- Produkt-freigabe - Anwender-Dokumentation	. Benutzer-funktionen . DIN A4-Seiten

Abbildung B - 5 : Teilproduktstrukturierung, Bsp. Siemens AG (b), Quelle: [Noth84]

Kernstück bei der Ermittlung des Aufwands ist eine Erfahrungdatenbank, in der die, zu jeder Maßgröße für das Unternehmen spezifische Relation zwischen Einheit (z.B. Anzahl DIN A4-Seiten, Anzahl Diagramme etc.) und Aufwand in MM, für jedes Projekt gespeichert wird. Ferner werden hier auch die zu berücksichtigen Einflußfaktoren vermerkt.

Wird nun für ein Teilprodukt der Aufwand gesucht, so wird das aktuelle Teilprodukt mit allen Teilprodukten in der Erfahrungsdatenbank hinsichtlich der äußeren Umstände (Einflußfaktoren) und der Ausprägung der Maßgrößen durchsucht.
Hierbei hängt es letztendlich vom Schätzenden ab, in wie weit das aktuelle Teilprojekt mit bereits realisierten Teilprojekten übereinstimmt. So kann die Ermittlung des Aufwandes auch durch Mittelwertbildung geschehen.

(b) Object-Point-Verfahren: [Litke96]

Dieses, von der Software AG entwickelte Verfahren basiert auf definierten Projektergenissen zu diskreten Schätzzeitpunkten, die mit den ‘Meilenstein’-Ergebnissen identisch sein können, aber nicht müssen.

Zu einem Schätzzeitpunkt *X*, an dem ein definitertes Projektergebnis vorliegt, wird ein Schätzobjekt (daher: Object-Point, kurz: OP) erstellt. Definiert wird dieses durch Schätzobjekttypen, das sind Kenngrößen, die für den weiteren Projektverlauf aussagekräftig sind. In der Regel sind dies: Maske, Druckausgabe, Informationsobjekt, Datentransfer und Verarbeitungsfunktion.
Die Anforderungen werden direkt und einzeln bewertet. Ergebnis dieser Bewertung sind OP für das entsprechende Schätzobjekt.
Ebenso werden den Entwicklungsaufwand erhöhende Einflußfaktoren direkt und einzeln bewertet. Andere Einflußfaktoren zu betrachten macht keinen Sinn, da die Roh-OP (vor der Berücksichtigung des Einflusses) den Minimal-Aufwand für dieses Schätzobjekt beschreiben.
Anschließend werden die OP anhand einer Erfahrungskurve in den Aufwand in MM überführt (vgl. Function-Point-Methode, B.4 (h)).

Empfohlen wird die Anwendung der Object-Point-Methode erst nach Beendigung der Phase III (Funktionsbeschreibung), da zu diesem Zeitpunkt Datenmodell, Funktionsbeschreibung, Beschreibung von Schnittstellen und ggf. ein erster Oberflächenprototyp zur Verfügung stehen. Diese Vorgehensweise basiert auf den Produkten der Software AG, sollte aber auch auf andere 4GL-Entwicklungsumgebungen anwendbar sein (vgl. [Litke96]). Es werden verschiedene Typen von Schätzobjekten unterschieden.

Zur Bestimmung des Schätzobjekttyps ‘Maske’ wird davon ausgegangen, daß eine Maske die Grundlage zur Ausführung einer oder mehrerer Funktionen darstellt. Der Typ bezieht sich auf das Masken-Layout und die damit verbundene Funktionalität.

Anforderung		Formel zur Bestimmung der OP
Anzahl einfacher Maskenfelder		Anzahl * 0,2
Anzahl Felder je Feldgruppe mit Mehrfachausprägung		Anzahl * 0,4
Datenaufbereitung, keine		= 0
	bis 30%	
	bis 50%	
	bis 70%	
	bis 100%	
Selektionen,	keine	= 0
	Werteingabe	
	PF-Tasten	
	einfaches Ankreuzen	
	Cursor-Positionierung	
	Zeilenkommandos	
Blättern,	nein	= 0
	vorwärts	
	vorwärts, Startwert	
	vorwärts und rückwärts	
	vorwärts, rückwärts, Startwert	
	Seitensteuerung, einfach	
	Seitensteuerung, schwer	
Funktionen		Anzahl gelesener / geschriebener Informationsobjekte
	Anzeigen	Anzahl * 2 / Anzahl * 4
	Eingeben	Anzahl * 2 / Anzahl * 4
	Ändern	Anzahl * 2 / Anzahl * 4
	Löschen	Anzahl * 2 / Anzahl * 4

Tabelle B - 37 : Ermittlung der OP für den Typ Maske

Unter dem Schätzobjekttyp ‘Druckerausgabe’ wird jede schriftliche Ausgabe auf einem externen Gerät zusammengefaßt. Nicht berücksichtigt werden Trivialmeldungen über erfolgreich / erfolglos ausgeführte Operationen.

Anforderung		Formel zur Bestimmung der OP
Anzahl einfacher Maskenfelder		Anzahl * 0,2
Anzahl Felder je Feldgruppe mit Mehrfachausprägung		Anzahl * 0,4
Datenaufbereitung, keine		= 0
	bis 30%	
	bis 50%	
	bis 70%	
	bis 100%	
Seitensteuerung,	keine	= 0
	einfach	= 1
	schwer	
Sortierung,	keine	= 0
	einfach	= 1
	schwer	
Anzahl gelesener Informationsobjekte		Anzahl * 3
Anzahl geschriebener Informationsobjekte		
Handelt es sich um ein Formular ? Ja		
	Nein	= 2

Tabelle B - 38 : Ermittlung der OP für den Typ Druckausgabe

Der Umfang und die Komplexität der im Anwendungssystem modulierten Daten wird auf der Basis von Informationsobjekten geschätzt. Bedingung hierfür ist das Vorliegen der dritten Normalform (Entity-Relationship-Model) und Spezialisierung als einzige semantische Beziehung untereinander.

Anforderung	Formel zur Bestimmung der OP
Anzahl Datenelemente	Anzahl * 0,7
Integration (Anzahl von Beziehungen), 1:1 und 1:n	Anzahl * 2,0
n:m	
Datenvolumen, bis 100	
101 - 1.000	
1.001 - 10.000	= 3
10.001 - 100.000	
100.001 - 1 Mio.	
1 Mio. - 16 Mio.	
> 16 Mio.	

Tabelle B - 39 : Ermittlung der OP für den Typ Informationsobjekt

Jeder nicht weiter zerlegbare Vorgang der Übernahme oder Bereitstellung externer Datenbestände wird im Schätzobjekttyp ‘Datentransfer’ zusammengefaßt.
Datenbestandsmanipulationen oder -auswertungen, die keinem anderen Typ zugeordnet werden können, werden als Typ ‘Verarbeitungsfunktion’ behandelt.
Die Bewertung dieser beiden Typen erfolgt analog:

Anforderung	Formel zur Bestimmung der OP
Anzahl der Datenelemente	Anzahl * 0,5
Prüfung der Datenelemente, keine	= 0
bis 30%	
bis 50%	
bis 70%	
bis 100%	
Dateityp, ADABAS	
sequentielle Datei	=5
sonstige	
Übergabedateien, Anzahl der Datenelemente	
Übergabedateien, Dateityp, ADABAS	
sequentielle Datei	
sonstige	
Anzahl gelesener Informationsobjekte	Anzahl * 3
Anzahl geschriebener Informationsobjekte	Anzahl * 6
Testbarkeit bei Datenübernahme:	
Zielumgebung mit Realdaten	= 0
Realdatenbereitstellung	
Testdatenbereitstellung	
keine Testmöglichkeit	
Restartfähigkeit, keine	
einfach	= 3
komplex	
Komplexe Transaktionslogik ? Ja	
Nein	= 0
Hohe Performance-Anforderung ? Ja	
Nein	= 0

Tabelle B - 40 : Ermittlung der OP für die Typen Datentransfer und Verarbeitungsfunktion

Nach Addition aller OP für die o.g. Typen werden diese noch anhand von Einflußfaktoren modifiziert. Dabei sind nur solche zu betrachten, die den Gesamtaufwand erhöhen. Eine Verringerung des Aufwands ist nicht möglich, da die OP-Summe bereits ein Mindestmaß für Qualität und Rahmenbedingungen beinhaltet.

Einflußfaktor	Erhöhung der OP-Summe in %
Unbekannte Systemumgebung, Batchanteil der Anwendung	bis 20 % 21 - 40 % 41 - 50 % > 50 %
Betreten von technischem Neuland, techn. Entwurf	
Realisierung	
Betreten von fachlichem Neuland	
Spezielle Zugriffsmechanismen	
Funktionalität vorhanden	
Funktionalität vorhanden, aber nur selten oder gar nicht eingesetzt	
keine Funktionalität vorhanden (funktionale Security, datenabhängige Security, 4-Augen-Prinzip)	
Endbenutzerhilfen, maskenbezogene Hilfe	
feldbezogene Hilfe	
seperates Benutzerhandbuch	÷ 5 %
RZ-Hilfen, Operator-Handbuch	÷ 2 %
Restart-Verfahren	÷ 3 %
Fremd- oder Mehrsprachigkeit	
Benutzeroberfläche, nicht in Muttersprache entwickelt	
Projektdokumentation, nicht in Muttersprache entwikkelt	
je Zusatzsprache: Benutzeroberfläche	
Projektdokumentation	
Erschwerte Testanforderungen	
Code Inspection	
Testdeckungsgrad	
Testdatenbereitstellung, ausreichend	
nicht ausreichend, aber keine besonderen Anforderungen	
nicht ausreichend, trotz besonderer Anforderungen	
Umfangreiches Berichtswesen	
Beteiligung mehrerer Fachbereiche, 2-4 Fachbereiche	je FB:
ab 5. Fachbereich	je FB:
Installation (Anzahl der Installationen je Anwendung)	je Installation:

Tabelle B - 41 : Einflußfaktoren zur Anpassung der OP

Die durch die Modifikation der OP-Summe mit Hilfe des Einflußfaktors gewonnene OP-Zahl wird zuletzt anhand einer Erfahrungskurve in den Aufwand in MM überführt.

Anhang C: Rechenbeispiele

In diesem Anhang sind die ausführlichen Rechnungen zu finden, deren Ergebnisse bereits im Kapitel *3.3 Methoden-Bewertung* als Grundlage der Bewertung dienten.

C.1 Prozentsatzmethode

Verglichen werden alle Projekte anhand deren relativer Aufwandsverteilung. Bei allen drei Referenzprojekten sind die ersten beiden Phasen ausgelassen worden. Zum Vergleich wurden in der letzten Zeile für die fehlenden Werte die der Stahlknecht-Verteilung (siehe Tabelle 3-1, Seite 25) eingesetzt, so daß die Werte der restlichen Phasen (Phasen III bis VI) die gleiche Charakteristik wie die Vergleichsverteilung aufzeigen müßten. Alle Prozentangaben sind auf eine Nachkommastelle gerundet.

Projekt Alpha

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
absolut	0 MH	0 MH	2 MH	27 MH	61 MH	14 MH	104 MH
relativ, alle Phasen	0 %	0 %	1,9 %	26,0 %	58,7 %	13,4 %	100 %
relativ, nur Phase III-VI	*5 %*	*10 %*	1,6 %	22,1 %	49,9 %	11,4 %	100 %

Projekt Beta

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
absolut	0 MH	0 MH	3 MH	5 MH	23 MH	18,5 MH	49,5 MH
relativ, alle Phasen	0 %	0 %	6 %	10,1 %	46,5 %	37,4 %	100 %
relativ, nur Phase III-VI	*5 %*	*10 %*	5,1 %	8,6 %	39,5 %	31,8 %	100 %

Teil-Projekt Gamma

	Phase I	Phase II	Phase III	Phase IV	Phase V	Phase VI	Gesamt
absolut	0 MH	0 MH	1 MH	4 MH	35 MH	2 MH	41 MH
relativ, alle Phasen	0 %	0 %	2,4 %	9,8 %	82,9 %	4,9 %	100 %
relativ, nur Phase III-VI	*5 %*	*10 %*	2 %	8,3 %	70,5 %	4,2 %	100 %

C.2 Function-Point-Methode

Zuerst werden die FP für die Funktionen, sortiert nach Daten, ermittelt. Dieses geschieht unter Berücksichtigung der Varianten I (IBM-Ursprungsfunktion) und III. (IBM-Modifikation). Daraus ergeben sich zwei verschiedene Aufwandsmaße für jedes Projekt.

Projekt Alpha

Eingaben:

Daten	Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1. Vorgangsdaten	viele	mehrere	formal	mittel	einfach	4
Analysedaten A und B	mehrere	wenige	formal	mittel	einfach	$2 * 3 = 6$
2. benötigte An-gaben	mehrere	wenige	formal	gering	einfach	3
3. Leistungs-angaben	wenige	wenige	formal	gering	einfach	3
4. Koordinaten	wenige	wenige	formal	gering	einfach	3
5. Vorgangs-nummern	wenige	wenige	formal	gering	einfach	3
Vorgänge	viele	viele	formal	gering	einfach	3

Ausgaben:

Daten	Listenspalten	Datenkon-vertierungen	Datenzugriffe und Verknüp-fungen	Anforderung a.d. Performance	FP
1. Vorgang	viele	einfach	wenige	keine	4
2. genehmigter Vorgang	viele	einfach	wenige	keine	4
Brief A bis C	mehrere	mehrfach	wenige	keine	$3 * 5 = 15$
3. abgeschl. Vor-gang	viele	einfach	wenige	keine	4
4. Vorgangsdaten	viele	einfach	wenige	keine	4
5. exportierte Vorgänge	viele	komplex	viele	keine	7
importierte Vorgänge	viele	komplex	viele	keine	7

Datenbestände:

Daten	Anzahl versch. Datenelemente	Performance-einflüsse	Wiederanlauf-einflüsse	DDP-Konzept der Daten	FP
1. Vorgangsdaten	viele	keine	keine	nein	7
Analysedaten A und B	wenige	keine	keine	nein	$2 * 7 = 14$

2.	Vorgangsdaten	viele	keine	keine	nein	7
3.	Vorgangsdaten	viele	keine	keine	nein	7
4.	Vorgangsdaten	viele	keine	keine	nein	7
5.	Vorgangsdaten	viele	keine	keine	nein	7
	Analysedaten A und B	wenige	keine	keine	nein	$2 * 7 = 14$

Abfragen:

Daten		Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1.	existierende Vorgangsdaten	viele	mehrere	formal	mittel	einfach	4
2.	existierende Vorgangsdaten	mehrere	wenige	formal	gering	einfach	3
3.	existierende Vorgangsdaten	wenige	wenige	formal	gering	einfach	3
4.	Vorgangsdaten	wenige	wenige	formal	gering	einfach	3
5.	Vorgangsdaten	viele	mehrere	formal	mittel	einfach	4

Referenzdaten (Datenbestände, die nur lesend geöffnet werden, nur für Variante III):

Daten		Anzahl versch. Datenelemente	Anzahl Schlüsselbegriffe bzw. Dimension	FP
4.	Vorgangsdaten	einige	1	5

Somit besitzen die Funktionen folgende FP:

Funktion	FP gem. Variante I	FP gem. Variante III
1.	39	39
2.	32	32
3.	17	17
4.	17	15
5.	45	45
Gesamt	150	148

Bestimmung des ‘Degree of Influence’ für Variante I:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Daten per Komm.	x						0
2. DDP-Konzept	x						0
3. Antwortzeitverhalten	x						0
4. Pkt. 3 bei starker Auslastung	x						0
5. hohe Transaktionsrate	x						0
6. Online-Dateneingabe						x	5
7. Online-Dateneing. dialogorientiert	x						0
8. Online-Wartung						x	5

9. komplexe Verarbeitungsfunktion		10
10. Wiederverwendung	bis 10%	0
11. Konvertierungen	x	4
12. Vorbereitung best. Prozeduren	x	2
13. mehrere Installationen	x	1
14. Änderungsdienst	x	0
Gesamt		27

$$doi_{VI} = (27 / 100) + 0{,}65 = 0{,}92$$

Bestimmung des ‘Degree of Influence’ für Variante III:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Verpflechtung	x						0
2. DDP-Konzept	x						0
3. hohe Transaktionsrate	x						0
4. komplexe Verarbeitung							5
5. Wiederverwendung			bis 10%				0
6. Konvertierungen				x			3
7. Änderungsdienst	x						0
Gesamt							8

$$doi_{VIII} = (8 / 100) + 0{,}65 = 0{,}73$$

Bestimmung des Aufwands:

	gem. Variante I:	gem. Variante III:
FP	150	148
doi	0,92	0,73
bewertete FP (= FP ∗ *doi*)	138	108
Aufwand [MM]	9,5	9,3

Projekt Beta

Eingaben:

Daten	Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1. Angaben zum Werk	viele	mehrere	formal	gering	einfach	3
Verwertungs-möglichkeiten	mehrere	wenige	formal	gering	einfach	3 * 3= 9
2. Angaben zur Verwertung	viele	mehrere	formal	gering	einfach	3
3. Probendaten	viele	mehrere	formal	gering	einfach	3

Ausgaben:

Daten	Listenspalten	Datenkon-vertierungen	Datenzugriffe und Verknüp-fungen	Anforderung a.d. Performance	FP
1. Werk	viele	einfach	mehrere	keine	4
2. Verwertung	viele	einfach	mehrere	keine	4
3. Probe	viele	einfach	mehrere	keine	4

Datenbestände:

Daten	Anzahl versch. Datenelemente	Performance-einflüsse	Wiederanlauf-einflüsse	DDP-Konzept der Daten	FP
1. Daten A	mehrere	keine	keine	nein	7
Daten D bis G	wenige	keine	keine	nein	4 * 7=28
2. Daten A	mehrere	keine	keine	nein	7
Daten C	mehrere	keine	keine	nein	7
3. Daten A	mehrere	keine	keine	nein	7
Daten B	mehrere	keine	keine	nein	7

Abfragen:

Daten	Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1. existierende Werksdaten	viele	mehrere	formal	gering	einfach	3
2. existierende Verwertungsd.	viele	mehrere	formal	gering	einfach	3
3. existierende Probendaten	viele	mehrere	formal	gering	einfach	3

Referenzdaten (Datenbestände, die nur lesend geöffnet werden, nur für Variante III):

Daten	Anzahl versch. Datenelemente	Anzahl Schlüsselbegriffe bzw. Dimension	FP
2. Daten A	einige	1	5
3. Daten A	einige	1	5

Somit besitzen die Funktionen folgende FP:

Funktion	FP gem. Variante I	FP gem. Variante III
1.	54	54
2.	24	22
3.	24	22
Gesamt	102	98

Bestimmung des ‘Degree of Influence’ für Variante I:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Daten per Komm.	x						0
2. DDP-Konzept	x						0
3. Antwortzeitverhalten	x						0
4. Pkt. 3 bei starker Auslastung	x						0
5. hohe Transaktionsrate	x						0
6. Online-Dateneingabe						x	5
7. Online-Dateneing. dialogorientiert			x				3
8. Online-Wartung						x	5
9. komplexe Verarbeitungsfunktion							0
10. Wiederverwendung			bis 10%				0
11. Konvertierungen			x				2
12. Vorbereitung best. Prozeduren	x						0
13. mehrere Installationen	x						0
14. Änderungsdienst	x						0
Gesamt							15

$$doi_{VI} = (15 / 100) + 0{,}65 = 0{,}80$$

Bestimmung des ‘Degree of Influence’ für Variante III:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Verpflechtung	x						0
2. DDP-Konzept	x						0
3. hohe Transaktionsrate	x						0
4. komplexe Verarbeitung							0
5. Wiederverwendung			bis 10%				0
6. Konvertierungen		x					1
7. Änderungsdienst	x						0
Gesamt							1

$$doi_{VIII} = (1 / 100) + 0{,}65 = 0{,}66$$

Bestimmung des Aufwands:

	gem. Variante I:	gem. Variante III:
FP	102	98

doi	0,80	0,66
bewertete FP (= FP ∗ *doi*)	81,6	64,68
Aufwand [MM]	4,39	3,27

Teil-Projekt Gamma

Eingaben:

	Daten	Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1.	Koordinaten	viele	wenige	formal	hoch	einfach	3
	Kategorie	viele	wenige	DB-Zugriff	mittel	einfach	3
	Beschreibung	wenige	wenige	formal	gering	einfach	3
2.	Eintrag	wenige	wenige	formal	gering	einfach	3

Ausgaben:

	Daten	Listenspalten	Datenkon-vertierungen	Datenzugriffe und Verknüp-fungen	Anforderung a.d. Performance	FP
1.	Eintrag	wenige	komplex	wenige	keine	4

Datenbestände:

	Daten	Anzahl versch. Datenelemente	Performance-einflüsse	Wiederanlauf-einflüsse	DDP-Konzept der Daten	FP
1.	Einträge	wnige	keine	keine	nein	7
2.	Einträge	wenige	keine	keine	nein	7

Abfragen:

	Daten	Datenelement-typen	logische Daten-gruppen	Eingabeprüfung	Ansprüche a.d. Bedienerführung	Bedienung	FP
1.	Einträge nach Kategorie	wenige	wenige	formal	gering	einfach	3
2.	Einträge nach Kategorie	wenige	wenige	formal	gering	einfach	3

Referenzdaten (Datenbestände, die nur lesend geöffnet werden, nur für Variante III):

	Daten	Anzahl versch. Datenelemente	Anzahl Schlüsselbegriffe bzw. Dimension	FP
2.	Einträge	mehrere	2	7

Somit besitzen die Funktionen folgende FP:

Funktion	FP gem. Variante I	FP gem. Variante III
1.	23	23
2.	13	13
Gesamt	36	36

Bestimmung des ‘Degree of Influence’ für Variante I:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Daten per Komm.						x	5
2. DDP-Konzept	x						0
3. Antwortzeitverhalten	x						0
4. Pkt. 3 bei starker Auslastung	x						0
5. hohe Transaktionsrate	x						0
6. Online-Dateneingabe						x	5
7. Online-Dateneing. dialogorientiert			x				3
8. Online-Wartung						x	5
9. komplexe Verarbeitungsfunktion							20
10. Wiederverwendung			> 50 %				5
11. Konvertierungen		x					1
12. Vorbereitung best. Prozeduren	x						0
13. mehrere Installationen	x						0
14. Änderungsdienst	x						0
Gesamt							44

$doi_{VI} = (44 / 100) + 0{,}65 = 1{,}09$

Bestimmung des ‘Degree of Influence’ für Variante III:

Merkmal	kein Einfluß	gelegentl. Einfluß	mäßger Einfluß	mittlerer Einfluß	bedeutsamer Einfluß	starker Einfluß	Wert
1. Verpflechtung	x						0
2. DDP-Konzept	x						0
3. hohe Transaktionsrate	x						0
4. komplexe Verarbeitung							10
5. Wiederverwendung			> 50 %				5
6. Konvertierungen	x						0
7. Änderungsdienst	x						0
Gesamt							15

$doi_{VIII} = (15 / 100) + 0{,}65 = 0{,}80$

Bestimmung des Aufwands:

	gem. Variante I:	gem. Variante III:
FP	36	36
doi	1,09	0,80
bewertete FP (= FP ∗ *doi*)	39,24	28,8
Aufwand [MM]	1,81	1,32

C.3 Data-Point-Methode

Die Berechnung des Aufwands basiert auf dieser Methode unter Einbeziehung der Qualitätsfaktor-Erweiterung.

Projekt Alpha

Bestimmung der DP für das Datenmodell:

Name	Anz. Attribute	Anz. Schlüssel	Integrationsgrad	Nutzung	DP
Analysen A	6	2	niedrig	1.1	17,6
Analysen B	6	2	niedrig	1.1	17,6
Vorgänge	61	1	mittel	1.1	75,9

Bestimmung der DP für die Nachrichten:

Name	Anz. Felder	Anz. Sichten	Komplexitätsgrad	Nutzung	DP
Maske A	37	1	niedrig	1.1	47,3
Maske B	25	2	niedrig	1.1	34,1
Maske C	9	1	niedrig	1.1	16,5

Bestimmung des 'Degree of Influence':

		Wert
Verteilung	ein Raum	5
Erfahrung	mittel	3
Kenntnisse	maßig	2
Automation	keine	1
Rechenbedingungen	gut	4
Unterstützung	mittel	3
Q.-Sicherung	teilw. autom.	2
Spezifikationsmech.	semi-formal	3
Prog.-Sprache	4. Generation	4
Testautomation	keine	1
Summe		28

doi = (125 - 28) / 100 = 0,97

Berechnung des Aufwands:

DP, Entitäten	111,1
DP, Nachrichten	97,9
doi	0,97
qf (Durchschnitt aller Faktoren)	0,93
BDP (= (DPE + DPN) * *doi* * *qf*)	188,54
Aufwand [MM]	4,7

Bestimmung des Qualitätsfaktors (*qf*):

	sehr gut (1,15)	gut (1,25)	normal (1,10)	nicht relevant (0,15)
Funktionalität				
Angemessenheit		x		
Richtigkeit		x		
Interoparabilität				x
Ordnungsmäßigkeit			x	
Sicherheit				x
Zuverlässigkeit				
Reife		x		
Fehlertoleranz			x	
Wiederherstellbark.				x
Benutzbarkeit				
Verständlichkeit		x		
Erlernbarkeit		x		
Bedienbarkeit		x		
Effizienz				
Zeitverhalten				x
Verbrauchsverhalten				x
Änderbarkeit				
Analysierbarkeit			x	
Modifizierbarkeit			x	
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit				
Anpaßbarkeit		x		
Installierbarkeit		x		
Konformität				x
Austauschbarkeit				x

Projekt Beta

Bestimmung der DP für das Datenmodell:

Name	Anz. Attribute	Anz. Schlüssel	Integrationsgrad	Nutzung	DP
Daten A	23	1	niedrig	1.1	31,9
Daten B	25	2	mittel	1.1	40,7
Daten C	32	2	mittel	1.1	48,4
Daten D	6	1	niedrig	1.1	13,2
Daten E	10	1	niedrig	1.1	17,6
Daten F	6	1	niedrig	1.1	13,2
Daten G	5	1	niedrig	1.1	12,1

Bestimmung der DP für die Nachrichten:

Name	Anz. Felder	Anz. Sichten	Komplexitätsgrad	Nutzung	DP
Maske A	0	1	niedrig	1.1	6
Maske B	23	1	niedrig	1.1	31,9
Maske C	27	1	niedrig	1.1	38,5
Maske D	25	1	niedrig	1.1	34,1
Maske E	32	1	niedrig	1.1	41,8

Bestimmung des ‘Degree of Influence’:

		Wert
Verteilung	ein Raum	5
Erfahrung	gut	4
Kenntnisse	mittel	3
Automation	teilweise	2
Rechenbedingungen	gut	4
Unterstützung	wenig	2
Q.-Sicherung	teilw. autom.	2
Spezifikationsmech.	form.-grafisch	5
Prog.-Sprache	4. Generation	4
Testautomation	keine	1
Summe		32

doi = (125 - 32) / 100 = 0,93

Berechnung des Aufwands:

DP, Entitäten	171,1
DP, Nachrichten	152,3
doi	0,93
qf (Durchschnitt aller Faktoren)	0,95
BDP (= (DPE + DPN) ∗ *doi* ∗ *qf*)	291,0
Aufwand [MM]	7,3

Bestimmung des Qualitätsfaktors (*qf*):

	sehr gut (115)	gut (1125)	normal (110)	nicht relevant (015)
Funktionalität				
Angemessenheit	x			
Richtigkeit				x
Interoparabilität				x
Ordnungsmäßigkeit			x	
Sicherheit				x
Zuverlässigkeit				
Reife			x	
Fehlertoleranz			x	
Wiederherstellbark.				x
Benutzbarkeit				
Verständlichkeit	x			
Erlernbarkeit	x			
Bedienbarkeit			x	
Effizienz				
Zeitverhalten				x
Verbrauchsverhalten				x
Änderbarkeit				
Analysierbarkeit			x	
Modifizierbarkeit			x	
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit				
Anpaßbarkeit	x			
Installierbarkeit			x	
Konformität			x	
Austauschbarkeit			x	

Teil-Projekt Gamma

Bestimmung der DP für das Datenmodell:

Name	Anz. Attribute	Anz. Schlüssel	Integrationsgrad	Nutzung	DP
Einträge	3	2	niedrig	1.1	14,3

Bestimmung der DP für die Nachrichten:

Name	Anz. Felder	Anz. Sichten	Komplexitätsgrad	Nutzung	DP
Maske A	4	1	hoch	1.1	15,4

Bestimmung des 'Degree of Influence':

		Wert
Verteilung	ein Raum	5
Erfahrung	gut	4
Kenntnisse	sehr gut	5
Automation	keine	1
Rechenbedingungen	gut	4
Unterstützung	mittel	3
Q.-Sicherung	teilw. autom.	2
Spezifikationsmech.	informal	1
Prog.-Sprache	5. Generation	5
Testautomation	keine	1
Summe		31

doi = (125 - 31) / 100 = 0,94

Berechnung des Aufwands:

DP, Entitäten	14,3
DP, Nachrichten	15,4
doi	0,94
qf (Durchschnitt aller Faktoren)	1,01
BDP (= (DPE + DPN) ∗ *doi* ∗ *qf*)	18,7
Aufwand [MM]	0,47

Bestimmung des Qualitätsfaktors (*qf*):

	sehr gut (115)	gut (125)	normal (110)	nicht relevant (015)
Funktionalität				
Angemessenheit		x		
Richtigkeit		x		
Interoparabilität				x
Ordnungsmäßigkeit			x	
Sicherheit				x
Zuverlässigkeit				
Reife		x		
Fehlertoleranz			x	
Wiederherstellbark.				x
Benutzbarkeit				
Verständlichkeit	x			
Erlernbarkeit	x			
Bedienbarkeit	x			
Effizienz				
Zeitverhalten				x
Verbrauchsverhalten				x
Änderbarkeit				
Analysierbarkeit		x		
Modifizierbarkeit		x		
Stabilität		x		
Prüfbarkeit		x		
Übertragbarkeit				
Anpaßbarkeit		x		
Installierbarkeit		x		
Konformität				x
Austauschbarkeit				x