Aufwandsschätzung

Udo Kelter

18.01.2001

Zusammenfassung dieses Lehrmoduls

Dieses Lehrmodul führt Techniken zur Aufwandsschätzung ein. Wir untersuchen zunächst allgemeine Faktoren, die einen Einfluß auf den Aufwand ausüben, und Bedingungen, unter denen die Schätzverfahren anwendbar sind. Es sind sehr viele Methoden zur Aufwandsschätzung vorgeschlagen worden; wir konzentrieren uns auf die wichtigsten elementaren Konzepte in diesen Methoden und stellen nur eine komplexere Methode, die Function-Point-Methode, detaillierter vor.

Vorausgesetzte Lehrmodule:

obligatorisch:
1. Vorgehensmodelle
2. Projektplanung

Stoffumfang in Vorlesungsdoppelstunden: 1.2

Inhaltsverzeichnis

1 Einführung

Bei der Entwicklung eines Systems treten recht unterschiedliche Arten von Kosten auf; diese wiederum bilden nur einen Teil einer umfassenderen Kosten-Nutzen-Analyse (s. Abschnitt 1 in [PPL]). Wir betrachten in diesem Lehrmodul nur die Personalaufwände, die i.d.R. die mit Abstand wichtigsten Kosten sind.

Es sind sehr viele Methoden zur Aufwandsschätzung vorgeschlagen worden ([NoK86] bspw. gibt eine Übersicht über 20 Methoden). In diesem Lehrmodul konzentrieren wir uns auf die wichtigsten elementaren Konzepte in diesen Methoden und stellen nur eine komplexere Methode, die Function-Point-Methode, detaillierter vor.

Aufwandsschätzungen müssen sehr früh stattfinden, typischerweise nach Vorliegen eines ersten Entwurfs des Lastenhefts. Aufwandsschätzungen haben daher generell mit den folgenden Problemen zu kämpfen:

1. In den frühen Phasen der Entwicklung liegen nur *unsichere und ungenaue Daten* über das entstehende System vor. Dementsprechend unsicher sind die daraus abgeleiteten Schätzwerte.
2. Die Schätzungen stehen meist unter *Zeitdruck*. Es kann z.B. sein, daß bei einer Ausschreibung bis zu einem bestimmten Termin ein Angebot eingereicht werden muß oder daß bei einer internen Entwicklung der Entscheidungsprozeß vorangebracht werden muß.
3. Die Durchführung des Schätzverfahrens, wozu insb. gehört, die nötigen Eingangsdaten zu erheben, kostet Zeit und Aufwand. Die *Kosten der Schätzung* müssen sich natürlich in einem angemessenen Rahmen halten.

Hieraus ergibt sich, daß nicht einseitig eine hohe Prognosegenauigkeit angestrebt werden darf, sondern daß rasche Durchführbarkeit und geringer Bearbeitungsaufwand mindestens ebensowichtige Merkmale eines Schätzverfahrens sind.

Ein sehr simples Verfahren könnte in einem Beispiel folgendermaßen arbeiten:

1. Man schätzt den Umfang in Codezeilen (LOC), z.B. 28.000

2. Man schätzt die Produktivität der Entwickler, z.B. 400 LOC/PM (PM = Personenmonat). Hieraus ergibt sich ein Aufwand von 70 PM.
3. Wenn wir zusätzlich die Projektdauer schätzen sollen, teilen wir den Aufwand durch die Zahl der verfügbaren Entwickler. Ist diese im Beispiel 3, dauert das Projekt ca. 23 Monate, also ca. 2.3 Jahre (1 Jahr = ca. 10 Monate).

Eine so einfache Schätzmethode hat gravierende Nachteile:

- Der Umfang des entstehenden Systems kann am Anfang nicht genau geschätzt werden; jeder Fehler geht linear in den Aufwand ein.
- Die Produktivität von Mitarbeitern schwankt erheblich. Unterschiede um den Faktor 10 sind ohne weiteres möglich. Ferner wird die Kompliziertheit des Systems nicht berücksichtigt.
- Für die einzelnen Entwicklungsphasen sind unterschiedliche Qualifikationen erforderlich, ferner variiert der Umfang der Arbeit erheblich.

Bessere Schätzverfahren versuchen, diesen Problemen entgegenzuwirken. Eine erste Maßnahme besteht darin, solche Einflußfaktoren zu erfassen, die gegenüber dem Durchschnittsfall einen höheren oder niedrigeren Aufwand erwarten lassen. Derartige Einflußfaktoren werden wir vorab in Abschnitt 2 analysieren. Wunder kann die Berücksichtigung solcher Einflußfaktoren nicht bewirken, denn sie müssen für den Einsatz in den Aufwandsformeln quantifiziert werden, d.h. ihre konkreten Werte müssen geschätzt werden, was die Unsicherheit nur verlagert.

2 Einflußfaktoren

Systemgröße. Klar ist, daß ein größeres System mehr Aufwand verursacht als ein kleineres. Es liegt nun nahe anzunehmen, daß der Aufwand linear mit der Systemgröße wächst, also ein viermal so großes System braucht den vierfachen Aufwand[1]. Dies ist aber i.a. falsch, der

[1] Wir gehen hier noch nicht auf die Frage ein, in welcher Einheit die Größe des Systems gemessen werden kann.

Aufwand steigt normalerweise etwas mehr als linear an. Unser viermal so großes System könnte z.B. den 4.3-fachen Aufwand verursachen. Dieser vielleicht überraschende Befund ist leicht zu erklären:

- Bei einer Vervierfachung der Systemgröße kann man nicht etwa die vierfache Projektdauer ansetzen, sondern z.B. nur die 1.4-fache Projektdauer. Hieraus folgt, daß die Zahl der parallel arbeitenden Entwickler um knapp den Faktor 3 steigen muß. Ein größeres Entwicklungsteam hat indessen einen höheren internen Kommunikations- und Koordinierungsaufwand. Bei einer Verdreifachung der Größe des Teams kann es z.B. erforderlich sein, eine zusätzliche Leitungsebene einzuziehen.
- Große Systeme haben tendenziell eine höhere innere Komplexität als kleinere. Dies erhöht die Chancen, Fehler bei der Entwicklung zu machen, und die Kosten für die Aufdeckung und Beseitigung eines Fehlers.

Qualität. Höhere Qualität verursacht bei gegebener Systemgröße mehr Aufwand und Kosten.

Diese Aussage ist zwar als qualitative Aussage leicht zu bestätigen, leider ist sie aber nur schwer als quantitative Aussage zu konkretisieren. Zum einen hat der Begriff Qualität viele Dimensionen (Bedienbarkeit, Performanz, Zuverlässigkeit, ...), d.h. man kann die Qualität eines Systems nicht auf eine einzige Note reduzieren. Sodann ist es selbst innerhalb der einzelnen Dimensionen meist schwierig, Qualitätsmerkmale in Kennzahlen umzusetzen, die als Eingangsgrößen für Schätzformeln brauchbar sind.

Zeitrahmen. Testfrage: kann man ein System, für das 2 Entwickler 7 Monate brauchen, auch mit 7 Entwicklern in 2 Monaten schaffen? Antwort: sehr wahrscheinlich nicht.

Entwicklungsaufgaben lassen sich vielfach gar nicht oder nur in geringem Ausmaß parallelisieren. Dies gilt vor allem für die Analyse und Produktdefinition – der Kunde kann z.B. nicht von drei Systemanaly-

tikern zugleich nach seinen Wünschen befragt werden –, aber auch für die Entwurfs- und Integrationsphase.

Selbst da, wo eine Parallelisierung prinzipiell möglich ist, führt eine Zerlegung in sehr viele Teilaufgaben dazu, daß sehr viele Querbezüge zwischen den Teilaufgaben entstehen und dadurch der Kommunikationsaufwand stark erhöht wird.

Aus diesen Beobachtungen folgt, daß es für jedes Projekt einen *Mindestzeitrahmen* gibt, der entweder prinzipiell nicht oder der nur unter starker Erhöhung der Gesamtaufwände unterschritten werden kann. Da man generell an kurzen Projektlaufzeiten interessiert ist (u.a. wegen der Kapitalkosten), nennt man diesen Mindestzeitrahmen auch "optimal". Das Schätzverfahren COCOMO gibt z.B. für eine gängige Klasse von Systemen folgende Formel für den optimalen Zeitrahmen Z_{opt} an:

$$Z_{opt} = 2.5 * A^{0.38}$$

Darin ist A der Aufwand in Personenmonaten. Die folgende Tabelle zeigt einige Beispiele:

A (PM)	Z_{opt} (Monate)
10	6.0
100	14.4
1000	34.5

Entwicklerproduktivität. Die Produktivität eines Entwicklers ist beeinflußt von

- der Ausbildung
- der Lernfähigkeit
- der Motivation für das Projekt
- dem allgemeinen Betriebsklima
- der Gestaltung des Arbeitsplatzes

und diversen weiteren Faktoren. Letztlich relevant ist, wieviel Zeit ein Entwickler für eine bestimmte Leistungseinheit braucht, z.B.

- für das Lokalisieren eines Fehlers im System

- für das initiale Ausprogrammieren von je 100 Codezeilen
- für das Schreiben einer Seite eines Handbuchs

Da derartige Produktivitätskennziffern linear in den Aufwand einfließen, ist es bei geringer Produktivität der Entwickler sinnvoll, zunächst in die Produktivität der Entwickler zu investieren, z.B. in zusätzliche Ausbildung oder Werkzeuge.

3 Anwendbarkeit der Schätzverfahren

Generell wird man durch die Schätzverfahren nicht zu einem Hellseher, der den Aufwand für beliebige Projekte prophezeien kann. Die Schätzformeln der Verfahren beruhen ganz im Gegenteil auf Erfahrung, also früheren Projekten,

- die inhaltlich vergleichbar sind,
- die in der gleichen oder einer vergleichbaren Arbeitsgruppe durchgeführt wurden und
- für die die Aufwandsdaten und Ausprägungen der Einflußfaktoren gemessen und aufgezeichnet wurden.

Hieraus ergeben sich teilweise gravierende *Einschränkungen für die praktische Anwendbarkeit der Verfahren*, denen man sich stets bewußt sein sollte. Wir diskutieren sie im folgenden.

Etablierte Anwendungsgebiete. Einigermaßen zuverlässig schätzen kann man nur in Anwendungsgebieten, die in dem Sinn "etabliert" sind, daß man die prinzipiellen Wege zur Lösung gängiger Aufgabenstellungen gut kennt und daß keine gravierenden Überraschungen zu erwarten sind. Daß in den etablierten Gebieten die Systementwicklung relativ gut planbar ist, hängt typischerweise auch damit zusammen, daß dort standardisierte Vorgehensmodelle und Standardarchitekturen vorhanden sind.

Eine zumindest grobe Zerlegung des zu entwickelnden Systems in Komponenten, also eine "Architektur", ist normalerweise eine unabdingbare Voraussetzung, wenn man die Systementwicklung hinrei-

chend präzise planen bzw. den Aufwand schätzen will. Standardarchitekturen bieten zunächst einmal den großen Vorteil, daß die (Grob-) Struktur des Systems nicht jedesmal neu entwickelt werden muß und daß standardisierte Komponenten eingesetzt können[2]. Hinsichtlich der Projektplanung bieten sie den großen Vorteil, daß die zu realisierenden Systemkomponenten sehr früh bekannt sind und relativ gut eingeschätzt werden können; generell führen Standardarchitekturen also zu einer höheren Planungssicherheit.

Sammlung einer Erfahrungsdatenbasis. Ein ganz entscheidendes Problem ist die Sammlung von Daten früherer Projekte. Da jedes Projekt Zufälligkeiten aufweist, muß man eine ausreichend hohe Zahl von Projekten erfaßt haben, um sinnvolle Mittelwerte – die letztlich immer auf Produktivitätskennziffern herauslaufen – bilden zu können.

Die bittere Erkenntnis ist also, daß dann, wenn man erst sehr wenige Projekte durchgeführt hat, keine ausreichende Sammlung vorliegt und man die Verfahren nicht anwenden kann. Man kann natürlich statt eigener Kennziffern solche verwenden, die bei anderen Gruppen erfaßt worden sind; allerdings weichen die Kennziffern verschiedener Gruppen deutlich voneinander ab. Ursachen dieser Differenzen können sein:

- Der Umfang der zu erbringenden Leistungen ist verschieden definiert.
- Die Entwicklungsphasen werden anders gegliedert, und Aufwände daher anders zugeordnet.
- Es werden unterschiedliche Maßstäbe für die Größe der Projekte bzw. die Einflußfaktoren verwendet.

Einschränkung bei Vorgehensmodellen. Die klassischen Schätzverfahren unterstellen ein relativ fixiertes Vorgehen, praktisch immer eine Variante des klassischen Phasenmodells.

[2] Da die Standardkomponenten nicht immer komplett neu entwickelt, sondern ggf. nur noch angepaßt werden müssen, wirken Standardarchitekturen unmittelbar kostensenkend. Dies erhöht die Planungssicherheit, um die es uns hier geht, nur insofern, als die kleineren Arbeitsaufwände etwas sicherer planbar sein sollten.

Wartungsaufgaben. Die Schätzverfahren behandeln primär den Fall einer Neuentwicklung eines Systems. Sie sind weniger geeignet für den – in der Praxis viel häufigeren – Fall einer (inkrementellen) Weiterentwicklung eines vorhandenen Systems. Auswirkungen einer Änderung oder Erweiterung auf ein vorhandenes System sind schwer einschätzbar.

4 Basismethoden

In diesem Abschnitt stellen wir einige grundlegende Methoden vor, anhand derer Aufwände geschätzt werden können. Für sich alleine genommen sind diese Methoden zu beschränkt bzgl. ihres Einsatzbereiches oder zu unpräzise. Sie treten aber als Komponenten komplexerer Methoden auf, daher ist es durchaus sinnvoll, sie zunächst isoliert zu betrachten.

4.1 Die Analogiemethode

Der Ansatz besteht darin, das zu schätzende Projekt mit einem ähnlichen früheren Projekt zu vergleichen.

Es muß in möglichst allen relevanten Merkmalen (Anwendungsgebiet, Programmiersprache, Methodeneinsatz, Umfang usw.) ähnlich wie das frühere Projekt sein. Ob und in welchem Ausmaß diese Ähnlichkeit vorliegt, entscheidet ein "Experte" aufgrund seiner Erfahrung. Für Abweichungen gibt es folgende Zu-/Abschläge:

- 50 % Zuschlag für erhöhte Komplexität
- 75 % Abschlag, falls das neue System i.w. aus vorhandener Software konstruiert wird, die nur leicht angepaßt werden muß.

Statt des ganzen Systems können auch die Komponenten einzeln nach diesem Verfahren geschätzt werden.

4.2 Die Einflußfaktorenmethode

Auch hier besteht der Ansatz darin, das zu schätzende Projekt mit einem ähnlichen früheren Projekt zu vergleichen. Die beiden Projek-

te können sich in mehreren Einflußfaktoren unterscheiden. Für jeden Einflußfaktor wird eine Punkteskala gebildet. Beispiele für berücksichtigte Einflußfaktoren und Punkteskalen sind:

die Programmiersprache: Ada entspricht 80 Punkten, Pascal 100 Punkten, C/C++ 110 Punkten, COBOL 120 Punkten, Assembler 150 Punkten

die Programmiererfahrung des Teams: 5 Jahre entspricht 80 Punkten, 3 Jahre 100 Punkten, ein Jahr 140 Punkten

Für ein Projekt werden nun die Punktzahlen aller Einflußfaktoren bestimmt und dann *addiert*.

Nehmen wir nun z.B. an, das frühere Projekt sei in Ada von einem Team mit 3 Jahren Erfahrung realisiert worden; wir erhalten somit 180 Punkte. Das neue Projekt sei in Ada von einem Team mit einem Jahr Erfahrung zu realisieren; wir erhalten somit 220 Punkte.

Der zweite Schritt besteht nun darin, die Differenz von 40 Punkten zwischen beiden Projekten in einem Korrekturfaktor umzusetzen. Hierfür müssen wiederum firmenspezifische Umsetzungsregeln verwendet werden.

4.3 Die Komponentenartenmethode

Diese Methode unterstellt, daß eine Größenschätzung des Systems in LOC vorliegt, ferner eine Tabelle, wie LOC-Umfänge in Aufwände umzurechnen sind.

Die Idee ist hier, die Schätzung zu verbessern, indem "schwieriger" Code stärker gewichtet wird, man also letztlich von tatsächlichen LOC zu gewichteten LOC übergeht.

Hierzu unterscheidet man mehrere Komponentenarten und ordnet jeder einen Aufwandsfaktor zu. Beispiel:

Komponentenart	Faktor
Datenverwaltungsroutinen	1.0
Auswertungsalgorithmen	2.0
Steuerprogramme	1.5
graphische Schnittstellen	2.4

Das zu entwickelnde System muß hier bereits in Module zerlegt sein (z.B. anhand einer Standardarchitektur), und die Größe jeder Komponente muß einzeln geschätzt werden (z.B. aufgrund einer Analyse existierender Produkte). Ferner muß jede Komponente einer Komponentenart zugeordnet werden. Die Größen der Komponenten werden mit dem Aufwandsfaktor der jeweiligen Komponentenart gewichtet aufsummiert, und man erhält die gewichtete Größe des Gesamtsystems.

Diese gewichtete Gesamtgröße muß anhand firmenspezifischer Umsetzungsregeln und Erfahrungswerte in Aufwände umgerechnet werden. Ebenso müssen die Aufwandsfaktoren firmenspezifisch angepaßt werden.

4.4 Die Prozentsatzmethode

Diese Methode basiert darauf, daß irgendeine Variante des Phasenmodells, diese dann allerdings durchgängig, verwendet wird, und daß für (möglichst viele) frühere Projekte die Verteilung der Aufwände auf die Phasen bekannt ist. Derartige Aufwandsverteilungen sind z.B. von den Firmen Bertelsmann und Hewlett-Packard publiziert worden:

Phase	Bertelsmann	Hewlett-Packard
Definition	18%	30%
Entwurf	19%	30%
Codierung	34%	15-20%
Test	29%	20-25%

Die entscheidende Annahme ist nun, daß die Verteilung der Aufwände auf die Phasen stets gleich ist. Dann kann man den Aufwand schätzen, indem man

1. die erste Phase abschließt und den aufgetretenen Aufwand mißt oder, falls die Schätzung früher erforderlich ist, den Aufwand für die erste Phase möglichst genau schätzt und
2. gemäß der Verteilung den Aufwand der folgenden Phasen und damit des Gesamtprojekts schätzt.

Die Prozentsatzmethode ist relativ früh einsetzbar und sehr leicht anwendbar; vorteilhaft ist ferner, daß die Schätzungen inkrementell verbessert werden können. Die Methode ist nur anwendbar, wenn der Entwicklungsprozeß anhand des Phasenmodells strukturiert wird.

4.5 Die Expertenschätzung

Die Vorgehensweise besteht hier darin, einen Experten mit Erfahrung auf dem jeweiligen Gebiet den Aufwand schätzen zu lassen.

Diese "Methode" ist in der Praxis sehr verbreitet.

4.6 Zusammenfassung

Zusammenfassend läßt sich sagen, daß alle Basismethoden (meist gravierende) Nachteile haben und daher allenfalls für grobe Vorabschätzungen geeignet sind.

Die besseren Methoden sind im Detail weiter ausgearbeitet; sie kombinieren i.a. mehrere Basismethoden.

5 Die Function-Point-Methode

5.1 Einleitung

Die Function-Point-Methodegeht auf die Autoren Albrecht und Gaffney zurück [Al79, AlG83].

Ausgehend von den vorgenannten Veröffentlichungen wurden später über 10 Varianten und/oder Erweiterungen publiziert. Wir stellen hier eine der neueren Versionen der Methode vor.

Entwickelt worden ist die Function-Point-Methode in einem Anwendungsbereich, in dem sehr viele Projekte durchgeführt werden und dementsprechend ausreichend statistisches Material entsteht: üblichen betrieblichen Informationssystemen. Nur in diesem Bereich ist die Methode sinnvoll anwendbar.

Ein sehr großer Vorteil der Function-Point-Methode ist, daß *keine* Schätzung der Größe des geplanten Programms – die immer ein zusätzlicher Unsicherheitsfaktor ist – notwendig ist, denn sie geht im Gegen-

satz zu vielen anderen Methoden nicht von der Größe des Systems aus. Stattdessen dienen Produktanforderungen und deren Merkmale als Ausgangsdaten. Eine erste Schätzung ist daher schon auf Basis des Lastenhefts möglich; eine verbesserte Schätzung später auf Basis des Pflichtenhefts.

5.2 Vorgehen

Das grundlegende Vorgehen bei einer Aufwandsschätzung mit der Function-Point-Methode kann in 6 Schritte gegliedert werden:

1. Identifizierung von Diensten bzw. "Funktionen": Folgende Klassen von Funktionen werden unterschieden:
   - Eingabe eines Datenelements
   - Ausgabe eines Datenelements
   - Abfragen, d.h. Suche in einem Datenbestand
   - Verwaltung eines Datenbestands
   - Benutzung von Referenzdaten

   Auf die genaueren Merkmale dieser Funktionen gehen wir anschließend ein.
2. Gewichtung der Funktionen: jede einzelne Funktion wird anhand ihrer Größe bzw. ihres Umfangs als einfach, mittel oder komplex eingestuft; jeder Gewichtungsstufe ist eine gewisse Zahl von **Funktionspunkten** gemäß folgender Tabelle zugeordnet:

Funktion	einfach	mittel	komplex
Eingabe	3	4	6
Ausgabe	4	5	7
Abfrage	3	4	6
Datenbestand	7	10	15
Referenzdatei	5	7	10

3. Addition der Funktionspunkte der einzelnen Funktionen; Ergebnis ist die Zahl der **Roh-Funktionspunkte** (**RFP**) des Gesamtsystems.

4. Bewertung des Schwierigkeitsgrads anhand globaler Einflußfaktoren; anhand verschiedener Kriterien werden Punkte für den Schwierigkeitsgrad vergeben und aufsummiert. Die Summe ist eine Zahl E, für die gilt: $0 \leq E \leq 60$.
5. Berechnung der **bewerteten Funktionspunkte** als

   $$BFP := RFP * \frac{70+E}{100}$$

   M.a.W. werden die Roh-Funktionspunkte durch den Schwierigkeitsgrad, der durch die Zahl E ausgedrückt wird, um maximal 30 % nach oben oder nach unten adjustiert.
6. Umsetzung der bewerteten FPs anhand einer Tabelle oder Kurve in den Aufwand (gemessen in Personenmonaten). Die Umsetztabelle ist firmenspezifisch; ein Funktionspunkt kostet zwischen 1 und 4 Personentagen Aufwand.

Die Methode wirkt schon intuitiv plausibel. Bei betrieblichen Informationssystemen werden Daten eingegeben, in einer Datenbank gespeichert und später entweder einzeln oder als Liste wieder ausgegeben. Hierfür sind jeweils Formulare und Tabellen zu gestalten und zu realisieren. Daß der Aufwand also linear (modulo qualitativer Korrekturen) von der Zahl der Formulare und Tabellen abhängt, ist daher naheliegend.

In Schritt 1 sind die einzelnen Funktionen vergleichsweise einfach zu identifizieren. Wenn z.B. im Lastenheft steht, daß man die Adressen und andere Stammdaten von Kunden verwalten muß, dann braucht man offensichtlich ein Formular, in dem die Daten eines Neukunden erfaßt und die eines Altkunden modifiziert werden können. Ferner muß ein Datenbankschema für die Kunden angelegt werden.

Jedes Formular muß graphisch entworfen werden, die Datenbankanbindung muß hergestellt werden, eventuelle Fehlerfälle müssen ausgewertet und dem Benutzer angezeigt werden usw., alles muß ausprogrammiert und getestet werden. Das Formular muß im Benutzerhandbuch erklärt werden, d.h. man muß sich ein Fallbeispiel ausdenken, einen Text schreiben und ggf. noch Bildschirmabzüge herstellen und in den Text einbinden. Hinzu kommt die Programmdokumentation.

Da eine Eingabe ca. 4 Funktionspunkte kostet und jeder Funktionspunkt - wie wir später sehen werden - ca. 1 - 2 Arbeitstage, kommt man pro Eingabe auf ca. 4 - 8 Tage anteiligem Gesamtaufwand, was durchaus realistisch ist.

5.3 Zählung und Gewichtungen

In diesem Abschnitt behandeln wir die Frage, welche Funktionen bei der Bildung der Roh-Funktionspunkte mitgezählt werden und wie man sie gewichtet.

Eingaben: Mitgezählt werden Eingaben über alle Medien; Beispiele für Medien sind die Tastatur, Disketten oder andere Datenträger, Schnittstellen, sequentielle Dateien, Belegleser, usw.

Oft werden in einem Formular bzw. einer Eingabemaske unterschiedliche Funktionen zusammengefaßt. Beispielsweise kann man oft über eine einzige Eingabemaske Daten neu eingeben oder vorhandene Daten korrigieren oder löschen, wobei die Funktion z.B. durch einen Menüeintrag gewählt wird. Abhängig von der gewählten Funktion sind ggf. einzelne Felder unbenutzt oder bekommen eine spezielle Bedeutung. Entscheidend ist hier, daß für die einzelnen Funktionen eine andere Verarbeitungslogik notwendig ist. In solchen Fällen zählt jede Funktion einzeln mit.

Ebenfalls als jeweils eigene Eingabe zählen Funktionen, bei denen für die Eingabe andere Formate verwendet werden.

Die bei der Gewichtung anzuwendenden Kriterien sind in der folgenden Tabelle angegeben.

Kriterium	einfach	mittel	komplex
Anzahl Datenelemente	1-5	6-10	> 10
Korrektheitsprüfung der Eingabe	nur Syntax	Syntax und Logik	mit Datenbankzugriff
Bedienerführung	einfach	normal	luxuriös

Die vorstehenden Kriterien korrespondieren nicht notwendigerweise miteinander, d.h. es kann eine Eingabe mit sehr vielen Fällen, syn-

taktischer und logischer Korrektheitsprüfung und sehr einfacher Bedienerführung geben. Die Kriterien sind daher eher als Indikatoren zu behandeln, bei unterschiedlichen Indikatoren kommt es auf den Gesamteindruck an, also das im konkreten Fall wichtigste und einflußreichste Kriterium.

Abfragen. Eine Abfrage führt zu einer Suche in einem Datenbestand und zu einer anschließenden Darstellung des Ergebnisses für den Benutzer; sie beinhaltet keine Änderung der Datenbestände.

Sofern eine Abfrage in mehreren Schritten abgewickelt wird, zählt jeder Schritt als eine Eingabe und eine Ausgabe.

Die bei der Gewichtung anzuwendenden Kriterien sind:

Kriterium	einfach	mittel	komplex
Anzahl unterschiedlicher Schlüssel	1	2	≥ 3
Bedienerführung	einfach	normal	luxuriös

Ausgaben. Mitgezählt werden Ausgaben auf beliebigen Medien: Bildschirmausgaben, Ausgaben über Schnittstellen an andere Anwendungen, Berichte (Listen, Formulare), Druckausgaben usw.

In Dialogen zählt ein Dialogschritt (eine Ein-/Ausgabe) nur als eine Ausgabe.

Die bei der Gewichtung anzuwendenden Kriterien sind:

Kriterium	einfach	mittel	komplex
Zahl Spalten	1-6	7-15	≥ 16
unterschiedliche Datenelemente	1-5	6-10	≥ 11
Gruppenwechsel	1	2-3	≥ 4
Datenelemente für die Druckaufbereitung	keine	einige	viele

Datenbestände. Mitzuzählen ist hier jede logische Datengruppe aus Anwendungssicht. Nicht mitzuzählen sind Hilfsdaten oder systemtechnisch erforderliche Datenbestände. Die bei der Gewichtung anzuwendenden Kriterien sind:

Kriterium	einfach	mittel	komplex
unterschiedliche Datenelemente	1-20	21-40	≥ 41
Anzahl der Schlüssel / Satzarten	1	2	≥ 3
Datenbestand vorhanden?	ja	-	nein
vorhandener Datenbestand wird verändert?	nein	ja	

Daß es aufwandserhöhend ist, wenn kein Datenbestand vorhanden ist, erklärt sich daraus, daß dann eigene Testdaten erstellt oder initial echte Daten erfaßt werden müssen. Wenn zwar Daten vorhanden sind, aber nicht exakt im benötigten Format (z.B. weil andere Codiervorschriften gelten), werden Konversionen notwendig, die meist viel Zeit kosten.

Referenzdateien. Referenzdateien unterscheiden sich von normalen Datenbeständen dahingehend, daß es sich hier eher um Zusatz- oder Steuerinformationen handelt. Sie werden meist nicht komplett verarbeitet und nur gelesen. Die Gewichte bestimmen sich wie folgt:

Kriterium	einfach	mittel	komplex
unterschiedliche Datenelemente	1-5	6-10	≥ 11
Anzahl der Schlüssel	1	2	≥ 3

Bei tabellarischen Strukturen zählt jede Dimension wie ein Schlüssel.

5.4 Einflußfaktoren

Wie schon oben erläutert, schätzt man mit den Einflußfaktoren Merkmale ein, die den Aufwand gegenüber dem Durchschnitt senken oder erhöhen. Die Zusammensetzung dieser Faktoren ist bei den Varianten

der Function-Point-Methode nicht ganz einheitlich. Einen ganz wesentlichen Bereich bilden dabei, was nicht überrascht, Merkmale, die die Komplexität des Systems einschätzen.

Der Einfluß wird stets global eingeschätzt und in Form von Punkten quantifiziert. 0 Punkte bedeutet kein Einfluß, maximale Punktezahl bedeutet starker Einfluß; die übrigen Werte sind entsprechend zu interpolieren. Hier zunächst eine Übersicht über die Einflußfaktoren und die Punkteskala:

1. Verflechtung mit anderen Anwendungen 0-5
2. dezentrale Verwaltung/Verarbeitung von Daten 0-5
3. hohe Transaktionsraten (erfordert Optimierungen) 0-5
4. Komplexität
   1. komplexe Rechnungen, Simulationen, Hochrechungen 0-10
   2. umfangreiche Kontrollverfahren, sensitive Anwendungen 0-5
   3. viele Sonderfälle / Ausnahmeregelungen 0-10
   4. komplexe Logik (z.B. Verknüpfung von Datengruppen) 0-5
5. geplante Wiederverwendung 0-5
6. Konvertierung von Datenbeständen 0-5
7. Anwendung durch Benutzer adaptierbar 0-5

Die Merkmale treffen ggf. nur auf Teile des Systems zu und sind dann anteilig zu gewichten. Bei der geplanten Wiederverwendung ist pro 10 % betroffenem Code 1 Punkt zu vergeben, maximal aber 5 Punkte.

5.5 Umsetzung

Die bewerteten Funktionspunkte können leider nicht einfach mit einem Faktor in eine Aufwandszahl umgerechnet werden. Wie schon früher erwähnt, ist systembedingt bei größeren Projekten mit einer geringeren Produktivität zu rechnen.

Die folgende Tabelle gibt einen Auszug aus einer Umsetztabelle der Firma IBM an [NoK86], der eine große Zahl von Projekten zugrundelag, in denen betriebliche Informationssysteme realisiert worden waren. Für verschiedene Systemgrößen (gemessen in bewerteten Funktionspunkten) wird der Gesamtaufwand (gemessen in Personenmonaten)

angegeben. Die dritte bzw. sechste Spalte enthält den Aufwand pro Funktionspunkt (gemessen in Personentagen pro Funktionspunkt, unter der Annahme 1 Personenmonat = 20 Personentage). Man erkennt, daß sich die Produktivität der Entwickler bei einer Verzehnfachung der Systemgröße in etwa halbiert.

BFP	PM	PT/BFP	BFP	PM	PT/BFP
50	2.3	0.92			
100	5.6	1.12	1000	114.4	2.29
150	9.5	1.26	1500	194.6	2.59
200	13.9	1.39	2000	283.7	2.84
250	18.6	1.49	2500	380.1	3.04
300	23.6	1.57	3000	482.7	3.22
400	34.4	1.72	4000	703.9	3.52
500	46.1	1.84	5000	943.1	3.77

Tendenziell ähnliche, absolut etwas höhere Zahlen wurden aus der Firma VW berichtet (s. [Ba96]). Wie schon erwähnt fließen in derartige statistische Kennzahlen Charakteristika ein wie die benutzte Programmiersprache, das lokale Vorgehensmodell, der darin festgelegte Leistungsumfang (u.a. Dokumentierungsstandards), Ausbildungsstand der Mitarbeiter usw.; die Zahlen sind daher außerhalb ihres Entstehungskontextes nur bedingt anwendbar, d.h. auf ihnen basierende Schätzungen sind unsicherer.

5.6 Zusammenfassung

Die Function-Point-Methode ist eine gute Schätzmethode und daher sehr verbreitet in der Industrie:

- Die Prognosegenauigkeit ist gut.
- Der Aufwand zur Durchführung einer Schätzung ist relativ klein.
- Die Reproduzibilität der Ergebnisse ist gut, d.h. wenn verschiedene Personen den Aufwand schätzen, kommen sie zu recht ähnlichen Ergebnissen.
- Die Methode kann sehr früh (nach Vorliegen des Lastenhefts) eingesetzt werden; die initiale Schätzung kann später verbessert werden.

Das gravierendste Problem ist die Erstellung einer firmenspezifischen Umsetztabelle.

Der Aufwand zur Durchführung von Schätzungen kann teilweise durch Werkzeuge reduziert werden. Die Datenbestände werden in ER- oder OOA-Diagrammen so genau beschrieben, daß man die zugehörigen Funktionspunkte automatisch ableiten kann.

Die Varianten der Function-Point-Methode versuchen teilweise, die Anwendbarkeit von Werkzeugen zu verbessern, teilweise wird stärker auf modernere softwaretechnische Methoden und neue Systemklassen abgestellt.

Literatur

[Al79] Albrecht, A.J.: Measuring application development productivity; p.83-92 in: GUIDE/SHARE Procceedings of the IBM application development Symposium (Monterey); 1979

[AlG83] Albrecht, A.J.; Gaffney, J.: Software function, source lines of code, and development effort prediction: a software science validation; IEEE ToSE SE-9, p.639-648; 1983-06

[Ba96] Balzert, H.: Lehrbuch der Software-Technik – Software-Entwicklung; Spektrum Akademischer Verlag; 1996

[NoK86] Noth, T.; Kretzschmar, M.: Aufwandsschätzung von DV-Projekten; Springer; 1986

[NPT] Kelter, U.: Lehrmodul "Netzplantechnik"; 1999/10

[PPL] Kelter, U.: Lehrmodul "Projektplanung"; 1999/10

[VM] Kelter, U.: Lehrmodul "Vorgehensmodelle - Eine Einführung"; 1999/10

Index