

The original publication is available at www.springerlink.com

Abstract

The estimating of large custom software development projects on the basis of an early rough specification is still a challenge for software companies and no satisfactory solution is available yet. Since competition favors the bid with the lowest price, higher estimation accuracy means a competitive advantage (“survival of the fittest estimator”). This paper points out that especially for large projects the commonly used intuitive estimation by experts has to be complemented for validation reasons by an estimation method based on metrics. We present the method UCP 2.0 which has been field-proven. The method is a significant enhancement of the Use Case Point method. The major influencing factors of the effort estimation have been standardized and thus are now reproducible.

Große Softwareprojekte

Aufwandsschätzung mit Use Case Points

Stephan Frohnhoff

Capgemini sd&m AG, Carl-Wery-Str. 42, 81739 München

frohnhoff@sdm.de

Zusammenfassung

Die Aufwandsschätzung von großen Individualsoftware-Entwicklungsprojekten auf Basis einer frühen Grobspezifikation gilt heute weiterhin als eine sehr schwierige Aufgabe für Software-Häuser und ist nicht zufriedenstellend gelöst. Wirtschaftliche Überlegenheit verlangt bessere Schätzgenauigkeit („Survival of the fittest estimator“), da im gestiegenen Wettbewerb bevorzugt der niedrigste Angebotspreis beauftragt wird. Dieser Beitrag zeigt auf, dass insbesondere für Großprojekte die heute weit verbreitete intuitive Expertenschätzung durch eine Metrik-basierte Schätzmethode zur Validierung zu ergänzen ist. Mit UCP 2.0 wird eine solche in der Praxis erprobte Methode vorgestellt, welche die Use Case Points Methode signifikant verbessert hat. Es ist gelungen, die wichtigsten Projekt-Einflussgrößen in der Aufwandsschätzung zu normieren und somit reproduzierbar zu machen.

Einleitung und Problemstellung

Analysten prognostizieren ein kontinuierliches jährliches Wachstum (CAGR) von fast 6% im Bereich der individuellen Software-Entwicklung in Deutschland (siehe Abbildung 1). Die Gründe dafür sind im Zeitalter der mächtigen Standard-Software-Pakete vielschichtig. Unternehmen setzen insbesondere aus folgenden Motiven auf individuelle Lösungen:

- Sie benötigen hochgradig innovative IT-Lösungen, die der Standard-Software Markt (noch) nicht bietet
- Sie wollen sich vom Wettbewerb differenzieren und benötigen dazu eine IT-Lösung, die besser auf ihr Leistungsangebot zugeschnitten ist oder sich vom Standard abhebt
- Sie machen Geschäfte in einem Marktsegment mit wenig Anbietervielfalt; daher gibt es zu wenig Abnehmer für Standard-Lösungen und folglich auch kein Angebot

In diesen Fällen werden Software-Großprojekte aufgesetzt mit dem Ziel, umfangreiche Geschäftslogik neu durch Software zu unterstützen. Die Abschätzung des Entwicklungsaufwandes solcher Projekte ist besonders schwierig, da sie als Unikat kaum miteinander vergleichbar sind, immer individuelle und neue Herausforderungen aufzeigen und daher standardisierte Schätzverfahren nur teilweise alle Komplexitätsaspekte erfassen.

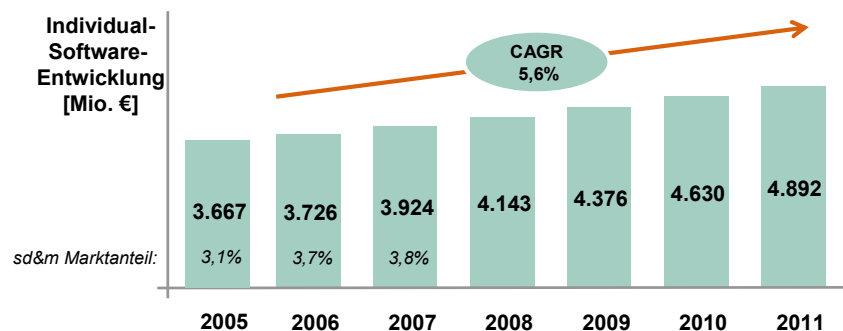


Abb. 1 Analyse der Marktentwicklung für Individualsoftware-Entwicklung im deutschsprachigen Europa [6]. Die durchschnittliche jährliche Wachstumsrate (CAGR) mittelt über die Jahre 2006 bis 2011.

Trotzdem muss zu einem sehr frühen Projektzeitpunkt eine Aufwandsschätzung durchgeführt werden, auch wenn sie auf noch unsicherem Wissen über den Lösungsweg aufsetzt und von daher nur eingeschränkt genau sein kann. Diese Aussage wird untermauert durch die Auswertung von 44 Projekten (500 bis 70.000 Tage Aufwand) aus dem Capgemini sd&m Kontext, die in oben beschriebene Projektkategorie fallen. Die

untersuchten Projekte wurden auf Basis einer Grobspezifikation durch Experten individuell geschätzt und nach Abschluss der Projekte wurde eine Nachkalkulation durchgeführt. Es zeigte sich, dass große Projektvorhaben im **neuen Umfeld** im Mittel um 40 % unterschätzt wurden. Aus informellen Gesprächen mit anderen Software-Häusern wissen wir, dass sich die Abweichungen in ähnlicher Größenordnung bewegen. „Neues Umfeld“ bedeutet dabei, dass eine individuelle Lösung für ein Unternehmen durch ein Software-Haus erstmalig entwickelt wird und folglich keine Erfahrungen aus vorangegangenen Projektphasen oder -Versionen bestehen. Es stellt sich die Frage, wie für den wachsenden Markt der Individual-Software-Entwicklung von betrieblichen Informationssystemen eine zuverlässigere Aufwandsschätzung vor Projektstart erreicht werden kann. Nach der Diskussion der veränderten Rahmenbedingungen durch die Industrialisierung der Software-Branche stellen wir einen Quality Gate Prozess vor, der das bisherige Schätzverfahren bei Capgemini sd&m um eine einfache Schätzmethode auf Basis von Use Case Points (UCP) erweitert. Die originale UCP-Methode stellte sich allerdings als nicht mehr zeitgemäß heraus und konnte deutlich verbessert werden. Daraus ist ein moderner Lösungsansatz für die Aufwandsschätzung von Individualsoftware-Entwicklungsprojekten für betriebliche Informationssystemen entstanden.

Einfluss der Industrialisierung auf die Aufwandsschätzung

Eine Konsequenz bei Capgemini sd&m aus den in der Einleitung beschriebenen unzuverlässigen Aufwandsschätzungen ist eine weitere Industrialisierung der Software Entwicklung. Industrialisierung meint hier, dass die Methoden und Werkzeuge der industriellen Fertigung Einzug in die Software-Entwicklung gehalten haben. Gute Beispiele sind arbeitsteilige Produktion oder das Reifegrad-Management, welches geeigneten Metriken für die Praxis bereit stellt und Qualitäts-Prüfpunkte schon im Angebotsprozess eines Software-Hauses verankert.

Wer als Individualsoftware-Hersteller diese Industrialisierung auch in seinen Fertigungsprozessen der Software-Entwicklung nicht beherrscht, wird wirtschaftlich nicht überlebensfähig sein. Diese allgemeine These wollen wir im Folgenden für den Kontext der Aufwandsschätzung von Software-Projekten begründen. Der Schlüssel zum wirtschaftlichen Erfolg wird initial in einem kaufmännischen Angebot des Software-Hauses an den Kunden (Auftraggeber) gelegt. Grundlage ist in der Regel eine Grobspezifikation des Auftraggebers. Diese erlaubt nicht, den tatsächlichen Aufwand exakt abzuschätzen. Die Analyse von über 500 abgeschlossenen Entwicklungs-Projekten zum Festpreis der letzten 8 Jahre zeigt eine Schätzgenauigkeit von etwa $\pm 30\%$, wobei Projektvorhaben in einem neuen Umfeld eine deutlich höhere Streuung von im Mittel $\pm 40\%$ aufweisen. Dies zeigt, dass insbesondere das Unbekannte und Neue schwerer zu schätzen ist, was keine Überraschung ist.

Es ist normal, dass solche Aufwandsschätzungen im Einzelfall zu hoch oder zu tief ausfallen. Über mehrere Projekte mittelt sich diese Schätzunsicherheit heraus, d.h. mal wird ein Projekt mit Budgetverlust, mal mit Budgetgewinn abgeschlossen. Jedes seriöse Software-Haus kalkuliert dafür in seine Projekte einen Risikopuffer ein. Dies reicht heute aber mit zunehmendem Wettbewerb nicht mehr aus. Ein deutliches Zeichen einer fortgeschrittenen Industrialisierung des Software-Entwicklungs-Marktes ist der gestiegene Wettbewerb. In den 90er Jahren gab es mehr Nachfrage nach IT-Lösungen als Anbieter, die Software-Häuser konnten sich ihre Kunden aussuchen. Heute ist der IT-Anbietermarkt so stark gewachsen, dass die Anwender sich die Anbieter aussuchen – und dabei spielt der Preis eine entscheidende Rolle. Bei vergleichbarer Lösungsqualität bekommt üblicherweise der Anbieter mit dem niedrigsten Preis den Zuschlag, auch wenn er seinen Angebotspreis zu gering geschätzt hat. Damit mitteln sich Schätzungenauigkeiten über mehrere Projekte nicht mehr heraus, der Auftraggeber bekommt bei Überschätzung des Aufwandes mit viel geringerer Wahrscheinlichkeit den Zuschlag, da es mit hoher Wahrscheinlichkeit einen Mitbewerber geben wird, der die Aufwende dieses Mal etwas geringer geschätzt hat. Dies begründet unsere erste These: **Wer genauer schätzen kann, ist wirtschaftlich robuster**, d.h. er macht im Mittel auf lange Zeit gesehen mehr Gewinn und verfügt über bessere finanzielle Reserven.

Wenn es einem Software-Haus gelingt, die Schätzungenauigkeit unter den marktüblichen Risikopuffer zu drücken, dann wird auch weiterhin über viele Projekte gemittelt wirtschaftlich mit Gewinn abgeschlossen. Daher werden mit einer detaillierten Aufwandsschätzung auch möglichst erfahrene Software-Ingenieure betraut. Wer schon viele Projekte selbst durchgeführt hat, der hat ein gutes Gefühl für zu schätzende Aufwende. Jedes Individual-Software-System ist ein Unikat. Dies macht die Aufwandsschätzung so schwierig, da kein Projekt mit einem anderen gut vergleichbar ist. Hier ist die Intuition des Schätzers gefragt.

Als Methode zur Aufwandsermittlung für solche Projekte wird die *Expertenschätzung* verwendet: Zunächst wird eine Stückliste aller für die Projektdurchführung notwendigen Aktivitäten erstellt und der Umsetzungsaufwand je Eintrag der Stückliste in eine Komplexitätsklasse eingeteilt {einfach, mittel, schwer}. Aus Erfahrung oder empirischer Mittelung aus abgeschlossenen Projekten wird für alle Komplexitätsklassen ein Aufwand zugeordnet. Durch Summation über alle Positionen der Stückliste wird dann der Gesamtaufwand ermittelt. Dieses Verfahren ist insbesondere für sehr große Projekte sehr aufwendig und fehleranfällig, da eine Skalierung bei Großprojekten nicht zwangsweise gegeben sein muss. So können die Aufwende für ähnliche Tätigkeiten der Stückliste bei wachsender Zahl durch Effizienzgewinn sinken, andere Effekte wie Kreuzabhängigkeiten

zwischen Systemteilen können überlinear anwachsen und zu steigenden Aufwänden bei wachsenden Stückzahlen führen. Es ist daher wichtig, eine ergänzende empirisch basierte Schätzmethodik zu haben, welche aus der Größe der Anforderungen an das zu erstellende Software-System einen Vergleichswert ermittelt, der zur Plausibilisierung der Expertenschätzung herangezogen werden kann. Weicht die Expertenschätzung von dieser Plausibilisierungsschätzung um einen bestimmten Grad ab, erfolgt keine Freigabe des Angebotes durch den Qualitätsprozessverantwortlichen des Unternehmens. Dieser Freigabeprozess wird als Quality Gate in der Angebotsphase bezeichnet und ist notwendig, um wirtschaftlich bestehen zu können.

Aufwandsschätzung auf Basis von Use Cases

Bei der Abschätzung von sehr großen individuellen Software-Entwicklungs-Projekten kommt eine weitere Schwierigkeit hinzu – *die Grenzen der Intuition sind bei Großprojekten erreicht* – unsere zweite These. Damit Experten den Aufwand eines Vorhabens gut abschätzen können, haben Sie eine vergleichbare Aufgabe in der Vergangenheit selbst schon mindestens einmal, besser dreimal durchgeführt. Großprojekte haben aber üblicherweise eine Laufzeit von mehreren Jahren. Wenn wir z.B. drei Jahre für ein Großprojekt veranschlagen, dann wird, stark vereinfacht angenommen, ein Projektmitarbeiter erst nach dreimal drei = neun Jahren zum Experten. In dieser Zeit ist sein Wissen schon wieder veraltet. Damit ist klar, dass ein Mensch allein aus selbst gemachter Groß-Projekterfahrung nicht ausreichend Wissen über die Schätzung von Großprojekten aufbauen kann, sondern dass dies aus vielen parallel laufenden Projekten entstehen muss. Dazu ist es notwendig, die Projekte hinsichtlich ihrer *Größe, Komplexität* und des *Umfeldes* normiert zu bewerten.

Seit den frühen 80er Jahren wurden hierzu Methoden im Software Engineering entwickelt. Die bekanntesten gehen auf die Function Point Analyse von Albrecht [1] zurück. Im Vordergrund steht hier die Größenbestimmung aufgrund datenorientierter Abstraktion [10]. Die Komplexität und das Umfeld wird gut durch das von Boehm entwickelte Constructive Cost Model (COCOMO) [3] erfasst.

Im Kontext eines Individualsoftware Herstellers wie Capgemini sd&m ist es im Sinne des skizzierten Quality Gates existenziell notwendig, eine Schätzmethode zur Verfügung zu haben, die schon zu einem frühen Projektzeitpunkt unmittelbar auf die in der Grobspezifikation heute übliche **Use Case orientierte Beschreibung** aufsetzt, daraus ein Größenmaß ermittelt und durch Bewertung der technischen Komplexität und des Projektumfeldes die zu erwartenden Aufwände abschätzt. Diese Methode soll eine Expertenschätzung, welche alle möglichen Einflussparameter und Einzelaufwände eines Projektes schätzt, nicht ersetzen, sie dient vielmehr der Plausibilisierung und sollte mit möglichst geringem Aufwand durchführbar sein.

In der Literatur findet sich dazu die Use Case Points Methode (kurz: UCP-Methode), welche auf Karner zurückgeht [13]. Die UCP-Methode wurde speziell für eine UML-basierte Software-Entwicklung konzipiert ([4], [7], [11], [17]) und berücksichtigt die empirischen Grundlagen für eine möglichst genaue Aufwandsschätzung [9]. Dabei hat diese Methode eine ähnliche Struktur, wie die klassischen Function Point Methoden [14]. Als Zählmaß werden bei der UCP-Methode die *Use Cases* und *Aktoren* einer Spezifikation oder Anforderungsdefinition eines Software-Systems herangezogen. Konkret werden die folgenden Aspekte bewertet: *Anzahl Use Cases, Anzahl Aktoren, technische Randbedingungen (nicht funktionale Anforderungen)* sowie *Projektumgebungsfaktoren*.

Auf eine Darstellung der UCP-Methode gemäß Karner wird hier verzichtet, die Methode wurde in einer der letzten Ausgaben des Informatik-Spektrums ausführlich erläutert [12]. Wir haben die Anwendbarkeit der UCP-Methode überprüft und detaillierter analysiert. Dabei fällt auf, dass die Komplexitätsfaktoren, welche in der Original-Methode nach Karner die technischen Anforderungen und Umgebungseinflussgrößen des Projektes berücksichtigen, aus der Sicht heutiger Entwicklungsprojekte die wirklichen Kostenfaktor nicht mehr angemessen erfassen. Dies betrifft konkret den „Technical Complexity Factor“ (TCF) und den „Environmental Factor“ (EF). Ein analytischer Vergleich z.B. mit COCOMO II [3] zeigt, dass hier wesentliche Kostenfaktoren in der UCP-Methode nicht berücksichtigt sind. Auch konnten wir feststellen, dass hinsichtlich des Projektvorgehens unterschiedliche Projekttypen nicht angemessen erfasst werden: Projekte mit „*Standard-Entwicklungsprozess*“, in denen größere Teams zusammenarbeiten und einem formalisierten Vorgehensmodell mit fest definierten Ergebnistypen folgen, und Projekte mit wenig formalem „*schlanken Entwicklungsprozess*“, die in der Regel von kleineren Projektteams (bis zu 5 Mitarbeitern) umgesetzt werden. In [8] wurden diese Effekte ausführlich beschrieben. Es zeigte sich, dass diese unterschiedlichen Entwicklungsprozesse erhebliche Auswirkungen auf die Aufwandsschätzung haben: Der Gesamtaufwand kann dadurch um $\pm 50\%$ schwanken.

Ferner sind die Komplexitätsfaktoren in Karners Methode zu wenig konkret beschrieben. Unterschiedliche Schätzer werden hier für das gleiche Projekt zu unterschiedlichen Einschätzungen gelangen. Ein wesentlicher Schritt ist also, diese Komplexitätsfaktoren durch Beispiele und konkrete Ausprägungen so zu beschreiben, dass eine Normierung möglich ist.

Eine Verbesserung der UCP-Methode muss daher vor allem die Komplexitätsfaktoren TCF und EF weiterentwickeln. Im Folgenden bezeichnen wir als **Kostenfaktor** eine besonders signifikante Einflussgröße für den Aufwand eines Entwicklungsprojektes. Zahlreiche funktionale Schätzmethoden haben hier jeweils einen

eigenen Satz von solchen Kostenfaktoren definiert. In Tabelle 1 sind die Anzahl der Kostenfaktoren von bedeutenden Schätzverfahren thematisch gruppiert gegenüber gestellt. Auf die Kategorisierung in T- und M-Faktoren gehen wir im nächsten Kapitel ein, ebenso erläutern wir später die Spalte *sd&m Umfrage*. Die Tabelle berücksichtigt die bekannteren Methoden wie IFPUG FPA 4.1 [16] als Grundlage einer ISO-Norm, COCOMO II [3], die Original Karner-Methode [13], eine in der Industrie von der Credit Suisse weiterentwickelte UCP-Methode mit Namen MT230 [2] sowie unter der Bezeichnung *sd&m UCP 1.0* eine erste Erweiterung der UCP-Methode von Capgemini *sd&m* [8]. Allein diese Methoden nennen zusammen über 40 verschiedene Kostenfaktoren mit jeweils bis zu 6 Ausprägungen, in Summe also ca. 250 Freiheitsgrade. Dabei sind diese Kostenfaktoren nicht abschließend, es können leicht weitere genannt werden.

T-Faktor		IFPUG FPA 4.1	COCOMO II	UCP (Karner)	Credit Suisse UCP MT230	sd&m UCP 1.0	sd&m Umfrage
Komplexe Berechnungen		1	1	1		1	1
Benutzeroberfläche		2		2	1	2	1
Schnittstellen		1		1	1	1	1
Verteiltes System		1		1	1	1	1
Verfügbarkeitsanford.				1	1	1	1
Wiederverwendbarkeit geford.		1	1	1	1	1	?
Performance		2		1	1	1	?
Flexibilität des Systems		1		1	1	1	?
Installation		1		1		1	
Sicherheitsanforderungen				1	1	1	
Anwender-Schulung				1	1	1	
Anforderung an Portabilität				1	1	1	
Auslastung Zielumgebung		1	2				
Anzahl Clients		1					
Betreibbarkeit		1					
Flexibilität Architektur		1					
Ähnliche Produkte			1				
Ausfallsicherheit (Reliability)			1				?
Datenbankgröße			1				
Tausch Hard-/Software			1				
Flexibilität d. Entwicklung			1				
Summe		14	9	13	7	13	5

M-Faktor		IFPUG FPA 4.1	COCOMO II	UCP (Karner)	Credit Suisse UCP MT 230	sd&m UCP 1.0	sd&m Umfrage
Stabile Anforderungen				1		1	1
Reife der Prozesse			1			1	1
Lstg./Fähigkeit Chefdesigner			1	1		1	1
Verfügbare Zeit			1				1
Teampayer			1				1
Kontinuität Mitarbeiter			1				1
Review und Architektur			1				1
Anzahl Entscheidungsträger					1		1
Integrationsabhängigkeit					1		1
Erfahrung Fachlichkeit			1	1	1	1	?
Motivation				1		1	
Verfügbarkeit Team				1		1	
Effizienz Programm.sprache				1		1	
Erfahrung mit RUP				1			
Erfahrung Objektorientierung				1			
Erfahrung Werkzeuge			1				?
Erfahrung Plattform/Middlew.			1				
Verteiltes Arbeiten			1				?
Dokumentation			1				
Unterstütz. durch Werkzeuge			1				
Lstg./Fähigk. Programmierer			1				?
Summe		0	13	8	3	7	9

Legende: (Bewertung der Kostenfaktoren aus Sicht der sd&m Umfrage)

 relevant
 ? evtl. relevant aber Streuung zu hoch
 fehlt
 überflüssig
 nicht relevant

Tabelle 1 Kostenfaktoren unterschiedlicher Methoden im Vergleich und Ergebnis der sd&m Umfrage

Es wird schnell deutlich, dass die Zahl der Freiheitsgrade in einem Software-Entwicklungsprojekt zu groß ist, um auch aus einer großen Zahl von Projekten durch Korrelationsanalyse die relevanten Kostenfaktoren ermitteln zu können. Für eine Methode zur Plausibilisierung einer Expertenschätzung im Kontext des Quality Gates ist es wichtig, den Aufwand für die Größenbestimmung und die Zahl der berücksichtigten Kostenfaktoren so klein wie möglich zu halten, da sonst der Aufwand für die Plausibilisierung schnell in die Größenordnung des zeitlichen Aufwandes einer Expertenschätzung kommt. Wir halten 10 bis 20 Kostenfaktoren für praktikabel.

Wir haben daher den Ansatz gewählt, zunächst durch eine Expertenurfrage aus den in Tabelle 1 genannten Kostenfaktoren diejenigen zu ermitteln, welche eine besonders hohe Relevanz haben könnten. Daraus resultiert ein Ranking, anhand dessen die Kostenfaktoren hinsichtlich ihres Gewichtes sortiert werden und eine Beschränkung auf wenige Kostenfaktoren erfolgen kann. Im Rahmen einer fragebogenbasierten Umfrage [2] unter 25 Experten (Projektleiter, Chefdesigner) wurde die Einschätzung von Experten hinsichtlich der Relevanz der Kostenfaktoren ermittelt. Das Ergebnis ist in Tabelle 1 in der Spalte *sd&m Umfrage* erfasst. Es wurden nur solche Kostenfaktoren berücksichtigt, die durch die Experten als relevant erachtet wurden. Der Verdacht auf Gleichverteilung und damit nicht einheitliche und zu sehr gestreute Meinung der Experten wurde mit Hilfe eines Chi²-Verteilungstest (0,95 Quantil) ermittelt [5]. Diese Kostenfaktoren sind in Tabelle 1 gelb mit Fragezeichen gekennzeichnet und wurden nicht weiter berücksichtigt. Rot markiert sind die als relevant eingestufteten Kostenfaktoren aus der sd&m Umfrage, die in der jeweiligen anderen Methode nicht berücksichtigt sind.

Verbesserte Methode UCP 2.0

Die vorangehenden Überlegungen führen zu einer verbesserten Methode UCP 2.0. Ziel ist es dabei im Sinne des beschriebenen Quality Gate Prozesses mit möglichst wenig Kostenfaktoren auszukommen. Bei der

Überarbeitung der UCP-Methode, insbesondere der Kostenfaktoren, haben wir zunächst eine saubere Trennung der Einflussgrößen nach unterschiedlichen Quellen der Projektanforderungen getroffen und differenzieren dazu gemäß den nachfolgend definierten A-, T- und M-Faktoren.

A-Faktor

Die funktionalen Anforderungen, d.h. der anwendungsfachliche Umfang, der im Projekt bewältigt werden muss, wird durch Use Cases und Aktoren definiert. In der Original-Methode nach Karner werden Use Cases gemäß ihrer Komplexität {einfach, mittel, hoch} mit {5, 10, 15} Punkten bewertet, Aktoren gemäß dem Typ der zugehörigen Schnittstelle mit {1, 2, 3} Punkten. Wir konnten eine deutlich bessere Schätzgenauigkeit nachweisen, wenn die Aktoren stärker gewichtet werden [2]. In der von uns verbesserten Methode UCP 2.0 werden sowohl für die Aktoren wie für die Use Cases je nach Komplexität bzw. Typ jeweils {5, 10, 15} Punkte vergeben und aufsummiert. Diese Punktesumme nennen wir A-Faktor. In betrieblichen Informationssystemen werden diese Anforderungen in Anwendungssoftware (A-Software) [18] implementiert. Daher bezeichnen wir diesen Aufwandsteil analog als A-Faktor, er ist proportional zu der nach Komplexität gewichteten Summe der Use Cases und Aktoren in der Spezifikation.

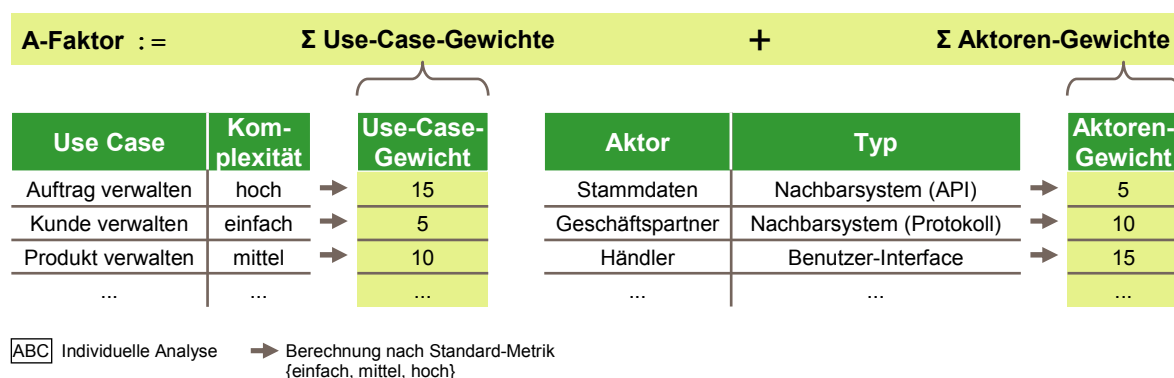


Abb. 2 Beispiel für die Ermittlung des A-Faktors

T-Faktor

Der T-Faktor (Technologie-Faktor) berücksichtigt die technische Komplexität des Entwicklungsprojektes und wird durch die nichtfunktionalen Anforderungen an das System charakterisiert. In Tabelle 1 sind mögliche Kostenfaktoren des T-Faktors aufgeführt. In der UCP-Methode wird der T-Faktor mit Hilfe eines gewichteten Bewertungsschemas der einzelnen Kostenfaktoren ermittelt. Eine normale durchschnittliche Komplexität mündet in einen T-Faktor von Eins, höhere Komplexität entsprechend in einem Faktor größer Eins.

Alle als relevant klassifizierten Kostenfaktoren der beschriebenen sd&m Expertenfrage finden sich im TCF der originalen UCP-Methode nach Karner wieder. Deshalb haben wir für UCP 2.0 die Kostenfaktoren und Gewichte aus dem TCF übernommen. In zukünftigen Arbeiten ist dieser Faktor ggf. weiter anzupassen und die Anzahl der Kostenfaktoren weiter zu reduzieren.

M-Faktor

Der M-Faktor (Management-Faktor) berücksichtigt die Projekteinflüsse auf den Gesamtaufwand des Entwicklungsprojektes. Diese liegen zum größten Teil in der Verantwortung des Auftragnehmers und hängen wesentlich davon ab, wie er die Anforderungen mit welchem Team und welchen Prozessen umsetzt. Wie auch der T-Faktor schwankt der M-Faktor um Eins.

Ein Vergleich der sd&m Expertenfrage mit dem „Environmental Factor“ (EF) der originalen UCP-Methode nach Karner zeigt eine geringe Überdeckung mit nur 2 von 8 Kostenfaktoren (Tabelle 1). Dies bedeutet, dass die originale UCP-Methode die heutigen Belange eines Softwarelieferanten nur noch unzureichend abdeckt. Untermauert wird diese Aussage durch die gute Überdeckung der sd&m Expertenfrage mit der modernen Methode COCOMO II. Deshalb haben wir den M-Faktor neu gebildet und verwendeten hierfür die durch die sd&m Umfrage bestimmten Kostenfaktoren. Entgegen der Original-Methode ermitteln wir den Faktor nicht mehr als Summe über einzeln gewichtete Faktoren sondern als Produkt der Einzelfaktoren, wie in [15] gefordert:

$$M - Faktor := \prod_{i=1}^9 [1 + 0,1 \cdot W_i \cdot (3 - M_i)] \quad (1)$$

M_i sind die Kostenfaktoren gemäß Tabelle 2. Die Gewichte W_i wurden durch numerische Auswertung abgeschlossener Entwicklungsprojekte (siehe Tabelle 3 weiter unten) ermittelt und optimal eingestellt. Für M_2 , M_6 und M_8 wurde numerisch ein Gewicht von 0 ermittelt, d.h. diese Kostenfaktoren könnten entfallen, allerdings ist die Datenbasis mit 18 Projekten zu klein für eine abschließende Bewertung.

M_j	Kostenfaktor	Beispielwerte für die Komplexitätsausprägungen je Kostenfaktor	W_j
M_1	Leistung/ Fähigkeit Chefdesigner	Wie erfahren sind der technische und fachliche Chefdesigner (TCD und FCD) hinsichtlich Aufgabe und Fachlichkeit bzw. Technik)? 0: Wenig erfahren (kein vergleichbares Projekt als FCD oder TCD durchgeführt) 3: Erfahren (ein vergleichbares Projekt als FCD oder TCD durchgeführt) 5: Sehr erfahren (zwei vergleichbare Projekte als FCD oder TCD durchgeführt)	1,4
M_2	Zusammen- arbeit (Teampayer)	Wie gut funktioniert die Zusammenarbeit im gesamten Projekt-Team (sd&m, Kunde und Dritte incl. Sublieferanten von sd&m und Dienstleister des Kunden)? Bewerte das Verständnis für Prozesse, die Leistungsfähigkeit der Zusammenarbeit und die gemeinsamen Ziele. 0: Die genannten Punkte funktionieren nur schlecht bis ausreichend (z. B. der Kunde hat eine hidden Agenda). 3: Die genannten Punkte funktionieren gut. 5: Die genannten Punkte funktionieren ausgesprochen gut (z. B. mit dem Kunden wurde schon einmal ein Projekt durchgeführt, so dass Prozesse vom Kunden gelebt werden).	0,0
M_3	Kontinuität Mitarbeiter	Wie hoch ist die Kontinuität der Mitarbeiter im Projekt? Gibt es Reibungsverluste durch den Wechsel von Mitarbeitern in Teilprojekten? 0: Gering (es wechseln/kündigen mehr als 50 % der Mitarbeiter pro Jahr aus dem Projekt heraus) 3: Normal (es wechseln/kündigen 25 % der Mitarbeiter pro Jahr aus dem Projekt heraus) 5: Hoch (es wechseln/kündigen bis 10 % der Mitarbeiter pro Jahr aus dem Projekt heraus)	0,3
M_4	Qualität der Grob- spezifikation und T- Architektur (Review und Architektur)	Wie nachvollziehbar und detailliert ist die Grobspezifikation, und wie gut sind Risiken bekannt? Müssen z. B. umfangreiche Arbeiten zur Erstellung der T-Architektur durchgeführt (typisch für ein erstes Release) werden oder sind wichtige „Pflöcke“ schon gesetzt (typisch für eine hohe Releasenummer)? 0: Die Grobspezifikation enthält zahlreiche Widersprüche und offene Fragen, für die Klärung sind mehrere Workshops notwendig; eine Risikoanalyse muss durchgeführt werden; es sind umfangreiche Arbeiten notwendig, um eine T-Architektur zu erstellen. 3: Die Grobspezifikation enthält offene Fragen, die mit dem Kunden zu klären sind; eine Risikoanalyse wurde durchgeführt, und es existieren Risiken; die T-Architektur entspricht weitestgehend einer Standardarchitektur oder wurde in einem vorherigen Release bereits so aufgesetzt. 5: Die Grobspezifikation ist ausreichend detailliert und lässt keine oder nur sehr wenige Fragen offen; eine Risikoanalyse wurde durchgeführt und ergab keine nennenswerten Risiken; die T-Architektur existiert.	0,5
M_5	Prozess- Overhead (Reife der Prozesse)	Wie formal sind das Vorgehen und der Entwicklungsprozess im Projekt (bezieht sich auf Aufbau- und Ablauforganisation)? 0: Komplexer Entwicklungsprozess, d.h. sehr formales Vorgehen und Prozesse, hohe Abstimmungs- und Querschnittsaufwände, alle Querschnittsrollen besetzt (typisch für Großprojekt mit mehr als 40 Mitarbeitern) 3: Normaler Entwicklungsprozess mit durchschnittlichen Querschnittsaufwänden (typisch für mittelgroßes Projekt, aber auch für kleine Projekte mit entsprechend formalen Anforderungen des Kunden) 5: Schlanker Entwicklungsprozess, d.h. pragmatisches Vorgehen, wenig Dokumentation, keine formalen Reviews oder Abnahmen, niedrige Querschnittsaufwände (typisch für kleines Projekt mit max. fünf Mitarbeitern)	1,5
M_6	Termindruck	Die optimale Projektlaufzeit wird mit 100 % angesetzt. Wie viel Zeit gesteht uns der Kunde zu? 0: Weniger als 80 % der optimalen Projektlaufzeit, das Team muss sehr steil aufgebaut werden, zusätzlich werden Arbeiten stark parallelisiert 3: Etwa 90 % der optimalen Projektlaufzeit, das Team wird steiler aufgebaut als üblich, Arbeiten werden normal stark parallelisiert 5: 100% der optimalen Projektlaufzeit	0,0
M_7	Stabile Anforderungen	Wie stabil sind die Anforderungen an das System? In welchem Umfang sind Änderungen der Spezifikation zu erwarten? 0: Sehr hohe Änderungsrate, auch grundlegender Anforderungen 3: Normale Änderungsrate, keine grundlegenden Anforderungen geändert 5: Sehr stabile Anforderungen, kaum Änderungen	1,8
M_8	Anzahl Entscheidungs- träger	Wie viele IT- und fachliche Ansprechpartner (=Entscheidungsträger) des Kunden sind involviert, die koordiniert werden müssen? Es sind ggf. auch externe Dienstleister mitzuzählen, die vom Kunden beauftragt wurden. 0: Viele fachliche und technische Ansprechpartner (mehr als 15) 3: Normal viele fachliche und technische Ansprechpartner (6 bis 15) 5: Wenige fachliche und technische Ansprechpartner (bis 5)	0,0
M_9	Integrations- abhängigkeit	Wie viele Abhängigkeiten von oder zu Schnittstellen bestehen, die aktiv gemanagt werden oder neu aufgesetzt werden müssen? 0: Sehr viele (mehr als 12), viele der Schnittstellen sind neu 3: Durchschnittliche viele (5 bis 12) 5: Keine oder wenige (0 bis 4), überwiegend existieren die Schnittstellen bereits	0,7

Tabelle 2 M-Faktor von UCP 2.0

PF (Produktivitätsfaktor)

Dieser Faktor kalibriert die Produktivität des Projektteams und gibt den mittleren Aufwand in Stunden für die Implementierung eines Use Case Points an. Dieser Faktor muss aus Nachkalkulationen empirisch für eine Organisation ermittelt werden und schwankt zwischen 20 und 40 h/UCP. Für die Methode UCP 2.0 haben wir für Capgemini sd&m einen Wert von 17,3 h/UCP gefunden. Der Wert ist geringer als 20, da u.a. die Akteure stärker gewichtet wurden.

Gesamtaufwand

Der Gesamtaufwand des Entwicklungsprojektes wird durch Multiplikation der genannten Faktoren gemäß Abbildung 3 ermittelt. Die funktionalen und nicht funktionalen Anforderungen werden auch gerne als Systemanforderungen bezeichnet.

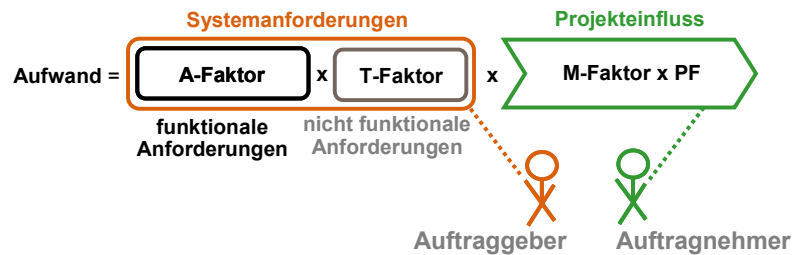


Abb. 3 Zerlegung der Aufwandsteile der Methode UCP 2.0

Bewertung der Methode UCP 2.0

Grundlage der hier vorgestellten Methode UCP 2.0 ist die Auswertung von 18 Individualsoftware-Entwicklungsprojekten im Umfeld betrieblicher Informationssysteme, die Capgemini sd&m als Auftragnehmer zum Festpreis für Auftraggeber aus unterschiedlichen Branchen durchgeführt hat. Verglichen wurde der tatsächlich benötigte Ist-Aufwand in Stunden mit der originalen UCP-Methode nach Karner und der hier vorgestellten Methode UCP 2.0 (Tabelle 3). Durch die geänderte Formel des M-Faktors und die angepassten Gewichte wird der Einfluss des M-Faktors größer. Um diesen Einfluss kontrollieren zu können, haben wir die in Tabelle 2 dargestellte Metrik so ausgearbeitet, dass diese die Projekte möglichst gut abdeckt und bei der Bewertung keinen Interpretationsspielraum zulässt. Damit kommen auch unterschiedliche Schätzer für dasselbe Projekt zum gleichen M-Faktor.

Projekt	Ist-Aufwand [h]	UCP-Methode (nach Karner)			Verbesserte Methode <i>sd&m UCP 2.0</i>				
		UUCP	Aufwand geschätzt [h]	Abweichung	A-Faktor	T-Faktor	M-Faktor	Aufwand geschätzt [h]	Abweichung
Auto 1	4.824	227	6.569	36%	259	0,97	1,14	4.978	3%
Auto 2	7.894	327	9.869	25%	367	1,01	1,36	8.746	11%
Auto 3	7.069	177	5.366	-24%	253	1,02	1,49	6.643	-6%
Bekleidung	728	50	854	17%	70	0,87	0,77	811	11%
Finanz 1	7.825	141	5.208	-33%	205	1,06	2,13	8.012	2%
Finanz 2	3.680	124	3.730	1%	160	1,03	1,14	3.269	-11%
Finanz 3	2.992	71	1.728	-42%	115	0,89	1,49	2.628	-12%
Industrie 1	55.592	1.717	53.702	-3%	1.917	1,05	1,94	67.739	22%
Industrie 2	7.368	221	6.221	-16%	261	1,05	1,14	5.440	-26%
Logistik 1	2.567	61	1.874	-27%	125	1,14	1,04	2.566	0%
Logistik 2	7.250	268	8.234	14%	300	1,14	1,04	6.157	-15%
Logistik 3	944	73	747	-21%	105	0,68	0,81	1.001	6%
Logistik 4	5.362	231	6.617	23%	295	0,96	0,93	4.575	-15%
Logistik 5	2.936	201	5.796	97%	241	0,97	0,74	2.981	2%
Public 1	4.804	182	5.624	17%	198	1,04	1,53	5.463	14%
TelCo 1	65.000	1.395	45.905	-29%	1.503	1,17	2,00	60.638	-7%
TelCo 2	2.456	170	2.088	-15%	210	0,94	0,81	2.748	12%
TelCo 3	2.432	131	1.939	-20%	195	1,04	0,76	2.660	9%
Mittlere Abweichung:				0%					0%
Standardabweichung:				± 34%					± 13%

Tabelle 3 Datenbasis UCP 2.0

Die Güte der Schätzmethode wird durch die Standardabweichung als Maß für die Streuung der Schätzwerte ausgedrückt. Die Methode UCP 2.0 weist mit nur 13 % eine signifikant geringere Streuung auf als die Methode

nach Karner mit 34 % und ist damit eine deutliche Verbesserung im Sinne des beschriebenen Quality Gate Prozesses. Zum Vergleich haben wir für die Projekte aus Tabelle 3 die Standardabweichung der Methode UCP 1.0 von sd&m aus dem Jahr 2006 [8] zu 27 % ermittelt. Ein wesentlicher Teil der Verbesserung ist dabei auf die höhere Gewichtung der Aktoren, wie oben beschrieben, zurückzuführen. Durch den Vergleich der Schätzergebnisse von unterschiedlichen Schätzern der gleichen Projekte konnten wir eine gute Reproduzierbarkeit nachweisen. Dies zeigt, dass **die wichtigsten Kostentreiber (A, T und M) von Individualsoftware-Entwicklungsprojekten normierbar sind** – unsere dritte These.

Die Praxistauglichkeit der Methode konnte damit nachgewiesen werden. Die Methode UCP 2.0 ist sehr schlank und schnell durchzuführen: Der Aufwand für die Durchführung einer UCP-Schätzung eines Projektes mit einem Gesamtaufwand von 10 Jahren (16.000 h) zur Plausibilisierung einer Expertenschätzung wird mit einem Arbeitstag (8 h) angegeben. Trotzdem ist UCP keine Wunderwaffe und das Schätzergebnis hängt empfindlich von der Qualität der fachlichen Grobspezifikation ab. Hierzu laufen weitere Forschungsvorhaben, in welchen die Schätzgüte in Abhängigkeit z.B. des Spezifikationsformates untersucht wird. Auch ist die Heuristik für die Bewertung der Komplexitätsklassen von Use Cases weiter zu untersuchen und ebenfalls sehen wir noch Forschungsbedarf im Bereich des T-Faktors.

Literatur

1. Albrecht, A.J.: Measuring Application Development Productivity. In Proc. IBM Applications Development Symposium. GUIDE Int. and Share Inc., IBM Corp., Monterey, CA, 1979
2. Badent, P.: Projektaufwand in Abhängigkeit von Organisation und Vorgehen. Masterarbeit an der Hochschule München, 2008
3. Boehm, B. W. et al.: Software Cost Estimation with COCOMO II, Prentice Hall, 2000
4. Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley Object Technology Series, 1999
5. Bronstein, I.N.; Semendjajew, K.A.; Musiol, G.: Taschenbuch der Mathematik. Verlag Harri Deutsch, 2005
6. Capgemini: PAC's market segmentation; evolution of the market for custom software development in German-speaking Europe. Interne Marktanalyse, 2007
7. Cockburn, A.: Writing effective use cases. The Agile Software Development Series. 8. Auflage, Addison-Wesley, 2003
8. Frohnhoff, S.; Jung, V.; Engels, G.: Use Case Points in der industriellen Praxis. In: Abran, A. et al. (Hrsg.): Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress, Eds. Shaker Verlag, S. 511-526, 2006
9. Frohnhoff, S.; Kehler, K.; Dumke, R.: Modellbezogene Use-Case-Identifikation für die UCP-basierte Aufwandsschätzung, Preprint Nr. 9, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, 2007
10. Fetcke, T.; Abran, A.; Dumke, R.: Eine verallgemeinerte Repräsentation für ausgewählte Functional Size Measurement Methoden. In: Dumke R.; Rombach D. (Hrsg.): Deutscher Universitäts-Verlag, S. 50-75, 2002
11. Jacobson, I.; Christerson, M.; Jonsson, P. und Overgaard, G.: Object-Oriented Software Engineering. A Use Case Driven Approach. 4. Auflage, Addison-Wesley, 1993
12. Jantzen, K.: Verfahren der Aufwandsschätzung für komplexe Softwareprojekte von heute. Informatik Spektrum 31(1), S. 35-49, 2008
13. Karner, K: Metrics for Objectory. Diploma thesis, University of Linköping, Sweden, No. LiTHIDA-Ex-9344, 1993
14. Lother, M.; Dumke, R.: Points Metrics – Comparison and Analysis. Proc. of the 11th IWSM, Shaker Verlag, S. 228-267, 2001
15. Ouwerkerk, J.; Abran, A.; An Evaluation of the Design of Use Case Points (UCP). Proc. of Mensura2006, Cadiz, Spain, 2006
16. Poensgen, B.; Bock, B.: Funktion-Point-Analyse. dpunkt.verlag, 2005
17. Schneider, G.; Winters, J.: Applying Use Cases - A Practical Guide. Addison Wesley Longman, Inc., 1998
18. Siedersleben, J.: Moderne Software-Architektur, dpunkt-Verlag, 2004