ITARDEUM
IT ARCHITECTURE, DESIGN & MANAGEMENT

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN - FH MÜNCHEN

Optimierte Aufwandschätzung in IT-Entwicklungsprojekten

München, Frühjahr 2010

Verfasser/innen: *Patrick Badent, Dr. Stephan Frohnhoff, Prof. Dr. Christian Greiner*
Erschienen in: *Forum Betriebswirtschaft München, Verfahren zur Krisenbewältigung, Vol. 2.2010*
Herausgegeben durch die Hochschule München, Fakultät für Betriebswirtschaft
ISSN 1867-9099, ZDB-ID 24973981., p. 34-43 --- Nachdruck ---
Datum *Frühjahr 2010, überarbeiteter Nachdruck vom 24.11.2017*

- *Experten bemühen sich bereits seit Beginn der Softwareentwicklung um die Verbesserung der Aufwandschätzung.*
- *Dieser Artikel stellt eine in der Praxis erprobte Methode vor, welche die Aufwandschätzung in der Anwendung vereinfacht und die Schätzgenauigkeit verbessert.*

Die Durchführung einer Aufwandschätzung ist zwingender Bestandteil von Softwareprojekten. Hierbei wird geschätzt, welcher Aufwand (in Stunden oder Tagen) bei der Erstellung des Softwaresystems notwendig ist. Je genauer diese Aufwandschätzung durchgeführt wird, desto spezifischer können die Kosten für Ressourcen wie Zeit, Material und Personal fixiert werden. Eine Abschätzung über Anwendungsfälle wird in verschiedenen Aspekten erweitert.

1 Motivation

Wenn Geschäftsprozesse mit der Hilfe von Software automatisiert werden, muss diese Investition wie eine ganz gewöhnliche Anlage betrachtet werden. Bei der Entscheidungsfindung ist deshalb auch die Frage des Umfangs der Investition zu beantworten:

- Bis zu welchem Umfang sollen die Geschäftsprozesse durch ein neues Softwaresystem abgebildet, unterstützt und automatisiert werden?
- Wie groß ist der fachlich-funktionale Umfang, welcher in dem Softwaresystem implementiert werden soll?

Um diese Fragen zu beantworten werden Aufwandschätzungen durchgeführt, welche den Aufwand für einen definierten funktionellen Umfang abschätzen um auf die zu erwarteten Kosten für das Softwaresystem schließen zu können. Da sich Experten seit den Anfängen der Entwicklung von Software um die Verbesserung der Aufwandschätzung bemühen zeigt, dass dies kein leichtes Umfangen ist ([Bundschuh 2004]).

In diesem Artikel stellen wir eine Methode vor, welche die Aufwandschätzung auf Basis von Individualsoftware-Entwicklungsprojekten formalisiert und damit nachvollziehbar macht. Durch Studien konnte die reproduzierbare hohe Schätzgenauigkeit bestätigt werden.

2 Grundlagen

Die Entwicklung von Software erfolgt in überschaubaren, zeitlich und inhaltlich begrenzten (Projekt-)Phasen. Berry W. Boehm geht in [Boehm 2000] davon aus, dass die Schätzgenauigkeit mit dem Fortschritt des Projekts steigt. Abbildung 1 zeigt die erwartete Schätzgenauigkeit am Beispiel der geschätzten Größe in SLOC (Source lines of code) sowie Kosten (Cost), welche in verschiedenen Projektphasen geschätzt wurden.

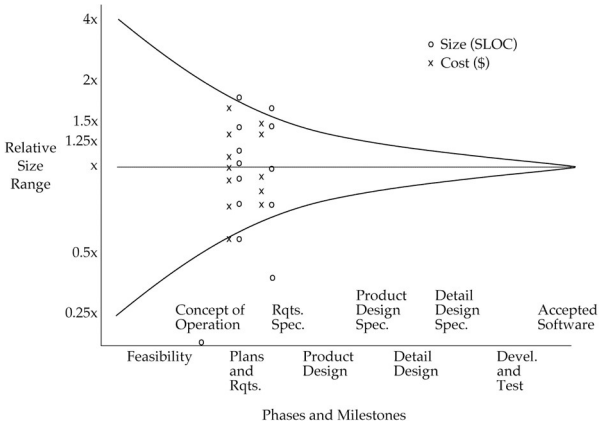

Abbildung 1: Schätzgenauigkeit ([Boehm 2000])

Die hohe Ungenauigkeit in frühen Projektphasen resultiert aus der dürftigen Informationsbasis. Es müssen zu Beginn viele Annahmen gemacht werden. Um aber eine Entscheidungsfindung herbeizuführen, bis zu welchem Umfang es sich lohnt Geschäftsprozesse durch Software abzubilden, muss der zu erwartende Aufwand mit möglichst hoher Genauigkeit zu einem möglichst frühen Zeitpunkt geschätzt werden.

3 Die Use-Case-Points-Methode

Die Use-Case-Points-Methode 2.0 wurde in der Masterarbeit [Badent 2008] in Zusammenarbeit mit der Hochschule München, der Universität Magdeburg und der Firma Capgemini sd&m entwickelt. Sie basiert auf Überlegungen von Gustav Karner, der während seiner Diplomarbeit im Jahre 1993 eine algorithmische Schätzmethode entwickelte [Karner 1993].

Die Schätzmethode nach Karner wie auch die Use-Case-Points-Methode 2.0 (ab hier: UCP-Methode 2.0 genannt) wurde speziell für eine UML-basierte Software-Entwicklung konzipiert ([Booch 1999], [Cockburn 2003], [Jacobson 1993], [Schneider 1998]) und berücksichtigt die empirischen Grundlagen für eine möglichst genaue Aufwandschätzung ([Dumke 2007]).

Um mit der UCP-Methode 2.0 eine Aufwandsschätzung durchzuführen, wird der Aufwand für die Erstellung des Softwaresystems in drei Teile zerlegt:

1. Um den A-Faktor zu bestimmen, werden die Geschäftsprozesse bewertet und deren Umfang mit einer speziellen Zählweise ermittelt.
2. Der T-Faktor wird durch technisch-funktionale und nicht-funktionalen Anforderungen an das System charakterisiert.
3. Die organisatorische Komplexität und das Projektumfeld werden durch den M-Faktor bewertet.

3.1 A-Faktor

Der Anwendungs-Faktor (kurz: A-Faktor) gibt den Umfang der funktionalen Anforderungen der Geschäftsprozesse wieder. Der A-Faktor wird bestimmt, indem die Anforderungen in Aktoren und Use Cases zerlegt, bewertet und in der Einheit Unadjusted Use Case Points (UUCP) gemessen werden.

Unter Aktoren werden alle (menschlichen und technischen) Nutzer verstanden, die das Softwaresystem nutzen.

Ein Use Case (deutsch: Anwendungsfall) beschreibt exakt die Aktivität eines Geschäftsprozesses, mit welchem ein fachliches Ziel erreicht werden soll. Diese Aktivität kann zwischen den Aktoren und dem Softwaresystem stattfinden. Möglich ist aber auch, dass das Softwaresystem diese Aktivität automatisiert durchführt. Im Vergleich zu einem Geschäftsprozess beschreibt ein Use Case, was die Aktoren vom Anwendungssystem erwarten. Dagegen beschreibt ein Geschäftsprozess im Wesentlichen, wie das Softwaresystem intern arbeitet.

Für die erfolgreiche Anwendung der UCP-Methode 2.0 und die Vergleichbarkeit der Ergebnisse ist deshalb vor allem entscheidend, mit welcher Genauigkeit Aktoren und Use Cases aus Konzepten extrahiert, bewertet und gezählt werden können. Deshalb legten wir bei der UCP-Methode 2.0 großen Wert darauf, eine einheitliche Zählweise zu erarbeiten und zu nutzen. Diese Grundlagen sind in [Frohnhoff 2006] und [Frohnhoff 2007] detailliert beschrieben und werden, so weit wie für die Anwendung notwendig, im Folgenden erläutert.

Aktoren werden systemweit nur einmal (nicht pro Use Case) nach folgendem Schema gezählt:

- Einfach (4 UUCP): Über eine einfache Schnittstelle angebundenes System.
- Mittel (8 UUCP): Über eine komplexe Schnittstelle angebundenes System.
- Komplex (12 UUCP): Je Endbenutzer für ein Frontend.

[Karner 1993] zählte Aktoren nur mit einem, zwei oder drei UUCP. Der Grund für die höheren Werte, welche wir annehmen, wird in Kapitel 4 erläutert.

Bei einem Use Case werden Szenarien, Schritte und Dialoge bewertet:

- Ein Szenario beschreibt einen möglichen Ablauf eines Use Cases. Ein Use Case hat ein Erfolgsszenario. Führen mehrere Szenarien zum Erfolg, spricht man von Alternativszenarien. Daneben können Fehlerszenarien zum Abbruch des Use Cases führen, somit ist das fachliche Ziel des Use Cases nicht erreicht. Fehlerszenarien werden dennoch beachtet, da diese auch zum Aufwand beitragen, außer es handelt sich um triviale Fehlerszenarien, wie zum Beispiel ein Abbrechen-Button, ohne weitere Aktion.
- Ein Schritt ist ein kleiner fachlicher Teil eines Use Cases, der ein Zwischenergebnis erzeugt und sich vom davor- und dahinterliegenden Schritt unterscheidet. Bei der Zählung der Schritte in einem Use Case, müssen die Schritte in allen Szenarien dieses Use Cases berücksichtigt werden.
- Als Dialog werden Bildschirmfenster gezählt. Verfügt ein Bildschirmfenster über mehrere Reiter oder Tabs, mit deren Hilfe weitere Informationen dargestellt werden, sind diese mehrfach zu zählen. Daneben werden auch Schnittstellen sowie speziell formatierte Ausgaben über Peripheriegeräte gezählt. Pop-Up-Meldungen, Bestätigungsfenster oder Menus finden keine Beachtung.

Der in Abbildung 2 dargestellte Use Case besitzt zwei Szenarien und sechs Schritte. Die Anzahl der Dialoge ist nicht direkt aus der Abbildung ablesbar. Vorstellbar ist aber, dass ein Dialog für die Eingabe der persönlichen Daten verwendet wird (und ggf. einen neuen Förderer anzulegen) und ein Dialog, um die Vorschlagsliste der Förderer darzustellen.

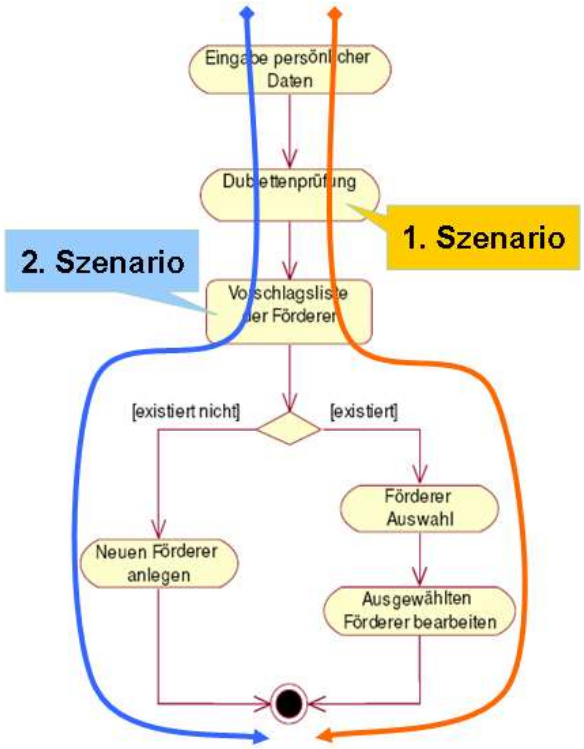

Abbildung 2: Szenarien, Schritten und Dialogen ([Dumke 2007])

Die Einteilung der Use Cases in die Klassen einfach, mittel oder komplex erfolgt nach folgenden Regeln:

- Einfach (5 UUCP): Bei maximal drei Szenarien, Schritten oder Dialogen.
- Mittel (10 UUCP): Bei maximal fünf Szenarien, Schritten oder Dialogen.
- Komplex (15 UUCP): Bei acht oder mehr Szenarien, Schritten oder Dialogen.

Des Weiteren wird der im Bereich 0 bis 1 liegende Faktor „Re-Use“ neu eingeführt. Ist die benötigte Funktionalität schon teilweise verfügbar, kann dieser Faktor mit >0 bewertet werden. Dann gehen die zuvor bestimmten UUCP nur anteilig mit ein.

Der A-Faktor ergibt sich durch die Summe der addierten UUCP aus Aktoren und Use Cases.

3.2 T-Faktor

Der T-Faktor (Technik-Faktor) bewertet die technische Komplexität und wird durch die nichtfunktionalen Anforderungen an das System charakterisiert. Zur Ermittlung des T-Faktors werden die Kostenfaktoren in Tabelle 1 bis einschließlich T_{12} mit einem ganzzahligen Wert aus dem Bereich 0 bis 5 bewertet. Die Bewertung 0 steht hierbei für nicht zutreffend, 5 steht für zutreffend. [Karner 1993] beschreibt die Bewertung 3 mit „not important nor irrelevant“, was für einen durchschnittlichen Einfluss oder alternativ auch für nicht zutreffend gesehen werden kann.

T_K **Kostenfaktor u. Beschreibung**	**Gewicht W_K**
Bewertungsskala für T_K	
T_1 **Verteiltes System**	$W_1 = 2$
Wie stark verteilt ist die Architektur des Systems?	
0: Monolithisches System	
3: 3-tier mit Randsystemen	
5: Hochverteilte Systemarchitektur	
T_2 **Performance- und Lastanforderungen**	$W_2 = 1$
Wie hoch sind Performance- und Lastanforderungen an das System?	
0: Keinerlei (oder geringe) Anforderungen	
3: Übliche Performance-Lastanforderungen	
5: Hohe Anforderung, z. B. Lastverteilung, Number-Crunching	
T_3 **Effizienz Benutzerschnittstelle**	$W_3 = 1$
Wie effizient muss die Benutzerschnittstelle zu bedienen sein?	
0: Keinerlei Anforderungen, z. B. Batches	
3: Normale Benutzungsschnittstelle, z. B. Web-GUI, einfaches Swing GUI	
5: Hochintegrierte, effiziente Benutzungsschnittstelle, z. B. Akkord-Anforderungen, Makros	
T_4 **Komplexität der Geschäftsregeln**	$W_4 = 1$
Wie komplex sind die Geschäftsregeln und/oder die Berechnungen im System?	
0: Nur einfachste Regeln, keine Berechnungen	
3: Normale Komplexität	
5: Sehr komplexe Regeln und/oder Berechnungen	
T_5 **Wiederverwendbarkeit**	$W_5 = 1$
Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes?	
0: Keine Anforderungen, z. B. Software, die nur einmal läuft	
3: Normale Anforderungen	
5: Hohe Anforderungen, z. B. Framework	
T_6 **Installationsfreundlichkeit**	$W_6 = 0,5$
Wie einfach muss die Software zu installieren sein?	
0: Keine Installationsanforderungen	
3: Normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen)	
5: Hohe Installationsanforderungen (Eigenständige Installation, hohe Zahl von Endkunden)	
T_7 **Benutzerfreundlichkeit**	$W_7 = 0,5$
Wie hoch sind die Anforderungen an die Benutzerfreundlichkeit des Systems?	
0: Keine Anforderungen (keine Benutzer)	
3: Normale Anforderungen (z. B. GUI, Hilfssystem)	
5: Hohe Anforderungen (z. B. GUI-Varianten für Gruppen, Internationalisierung, Wizards, Fehlertoleranz)	

T_K Kostenfaktor u. Beschreibung	Gewicht W_K
Bewertungsskala für T_K	
T_8 Portabilität	**W_8 = 2**
Wie hoch sind die Anforderungen an die Portabilität des Systems?	
0: Keine Anforderungen, z. B. Software die nur einmal läuft	
3: Normale Anforderungen (eine Zielplattform, üblicher Grad an Abstraktion)	
5: Hohe Anforderungen (z. B. Cross-Plattform)	
T_9 Variabilität (Änderungsfreundlichkeit)	**W_9 = 1**
Wie variabel (änderungsfreundlich und anpassbar) muss das System sein?	
0: Keine Anforderungen, z. B. Software, die nur einmal läuft	
3: Normale Anforderungen, z. B. Konfigurierbarkeit	
5: Hohe Anforderungen, z. B. Anpassbarkeit über Templates / Skins, Plugin-Schnittstellen	
T_{10} Verfügbarkeitsanforderungen	**W_{10} = 1**
Wie hoch sind die Verfügbarkeitsanforderungen an das System?	
0: Keinerlei Anforderungen (keine parallelen Transaktionen)	
3: Übliche Verfügbarkeitsanforderungen	
5: Hohe Verfügbarkeitsanforderungen (24/7-Betrieb), >99% Verfügbarkeit, hohe Zahl paralleler Transaktionen)	
T_{11} Sicherheitsanforderungen	**W_{11} = 1**
Wie hoch sind die Sicherheitsanforderungen an das System?	
0: Keinerlei Anforderungen.	
3: Übliche Sicherheitsanforderungen (Authentifizierung und Autorisierung)	
5: Hohe Sicherheitsanforderungen (Zertifikate, verschlüsselte Kommunikation, Kryptographie)	
T_{12} Systemnutzung durch Dritte	**W_{12} = 1**
Bietet das System direkte Zugänge für Dritte (d. h. andere als den Kunden) an?	
0: Keine Anforderungen	
3: Systemnutzung durch Endkunden (B2C-Kunden des Kunden)	
5: Z. B. externe Serviceschnittstellen, B2B-Systeme, Handelsplattformen	
T_{13} Programmiersprache	(ohne Gewicht)
Wähle die verwendete Programmiersprache.	
1: C#/.NET	
1: Java	
2,2: Cobol (Host)	

Tabelle 1: T-Faktor

Der in Tabelle 1 dargestellte T-Faktor wurde von Karner übernommen und in zwei Punkten angepasst:

1. Karners Kostenfaktor zur Bewertung der Schulungsaufwände wurde ersatzlos gestrichen. Diese Aufwände wollten wir nicht durch die UCP-Methode 2.0 mit abdecken (vergleiche Kapitel 3.5).
2. Neu aufgenommen wurde ein Kostenfaktor zur Bewertung der Programmiersprache: Wird ein Softwaresystem mit einer hostbasierten Programmiersprache Cobol erstellt, wird mit dem Faktor 2,2 (vergleiche Kapitel 4) der Mehraufwand gegenüber einer modernen Programmiersprache wie Java oder C#/.NET (beide sind mit dem Faktor 1 angegeben) angenommen. Da T_{13} als absoluter Faktor betrachtet wird, verfügt dieser über kein Gewicht. Werden verschiedene Programmiersprachen gleichzeitig genutzt, ist für T_{13} ein Mittelwert zu berechnen, welcher die jeweiligen Anteile berücksichtigt.

In Formel 1 wurde der erste Ausdruck so angepasst, dass bei einer durchschnittlichen Komplexität (alle Kostenfaktoren bis T_{12} mit 3 bewertet) der Wert von 1 erreicht wird. Im zweiten Ausdruck der Formel wird die Programmiersprache berücksichtigt.

$$T_{Faktor} = [0{,}61 + 0{,}01 \cdot \sum_{k=1}^{12} (T_k \cdot W_k)] \cdot [T_{13}] \quad (1)$$

Damit bewegt sich der T-Faktor in einem Bereich zwischen 0,61 und 2,77.

3.3 M-Faktor

Im Gegensatz zum T-Faktor wurde der M-Faktor (Management Faktor) anhand der Auswertung einer Umfrage und der Durchführung einer Regressionsanalyse mit über 40 Kostenfaktoren komplett neu definiert (vergleiche Kapitel 4). Im Vergleich zu dem von [Karner 1993] definierten Environmental Factor betrachten wir den M-Faktor als moderner, welcher die heutigen Gegebenheiten besser abdeckt.

Der M-Faktor bewertet die organisatorische Komplexität und das Projektumfeld mit neun Faktoren. Auch hier wird jeder Faktor M_I mit einem ganzzahligen Wert aus dem Bereich 0 bis 5 bewertet.

Die folgende Tabelle stellt den M-Faktor dar:

M_I Kostenfaktor und Beschreibung	Gewicht W_I
Bewertungsskala für M_K	
M_1 Leistung und Fähigkeit Chefdesigner	**W_1 = 0,8**
Wie erfahren sind der fachliche (FCD) und technische Chefdesigner (TCD) hinsichtlich der Aufgabe?	
0: Wenig erfahren (kein vergleichbares Projekt als FCD oder TCD durchgeführt)	
3: Erfahren, ein vergleichbares Projekt durchgeführt	
5: Sehr erfahren, mind. zwei vergleichbare Projekte durchgeführt	

M_I Kostenfaktor und Beschreibung	Gewicht W_I
Bewertungsskala für M_K	

M_2 Zusammenarbeit **W_2 = 0,2**

Wie gut funktioniert die Zusammenarbeit im Projekt-Team (Auftragnehmer, Kunde und Dritte)? 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).

M_3 Kontinuität Mitarbeiter **W_3 = 0,4**

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)

M_4 Qualität Grobspezifikation und T-Architektur **W_4**

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 nennenswerte Risiken; die T-Architektur existiert.

M_I Kostenfaktor und Beschreibung	Gewicht W_I
Bewertungsskala für M_K	

M_5 Prozess-Overhead **W_5 = 2,5**

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 weniger als fünf Mitarbeitern)

M_6 Termindruck **W_6 = 0,0**

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

M_7 Stabile Anforderungen **W_7 = 1,4**

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

M_8 Anzahl Entscheidungsträger **W_8 = 0,1**

Wie viele IT- und fachliche Ansprechpartner (Entscheidungsträger, inkl. externe Dienstleister) des Kunden sind involviert, die koordiniert werden müssen?

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)

M_9 Integrationsabhängigkeit **W_9 = 0,5**

Wie viele Abhängigkeiten von oder zu Schnittstellen bestehen, die aktiv gemanagt werden müssen?

0: Sehr viele (mehr als 12), viele der Schnittstellen sind neu
3: Durchschnittlich viele (5 bis 12)
5: Keine oder wenige (0 bis 4), überwiegend existieren die Schnittstellen bereits

Tabelle 2: M-Faktor

Zur Berechnung des M-Faktors nutzen wir Formel 2.

$$M_{Faktor} = \prod_{l=1}^{9} \left[(1 + 0{,}1 \cdot W_l)^{3-M_l} \right] \qquad (2)$$

Der M-Faktor kann sich über einen Wert von 0,34 bis 5,01 erstrecken. Zum Vergleich dazu schwankt der Environmental Factor nur im Bereich zwischen 0,425 und 1,7 und damit deutlich geringer.

3.4 Produktivitätsfaktor

Der Produktivitätsfaktor (kurz: PF) kalibriert die Produktivität des Projektteams und gibt den mittleren Aufwand in Stunden für die Implementierung eines Use Case Points an. Dieser Faktor muss durch Nachkalkulationen bereits abgeschlossener Projekte empirisch für eine Organisation ermittelt werden. Für die UCP-Methode 2.0 konnten wir anhand 26 Projekten (siehe Kapitel 4) einen Wert von 16,7 Stunden pro Use Case Point errechnen.

Im Optimalfall ist der PF für jedes Projektteam (oder Organisation) zu bestimmen. Ist dies nicht möglich kann auch mit dem ermitteln Faktor (16,7 Std./UCP) gerechnet werden, es muss jedoch mit einer geringen Schätzgenauigkeit gerechnet werden.

3.5 Ermittlung des Gesamtaufwands

Der erwartete Gesamtaufwand $E_{geschätzt}$ (Effort) zur Durchführung des Projekts in Stunden wird aus den UUCP des A-Faktors ermittelt, indem diese mit dem T-Faktor und dem M-Faktor multipliziert werden. Dieses Ergebnis wird in Use Case Points (UCP) gemessen und spiegelt den relativen Aufwand wieder, welcher benötigt wird, um das Softwaresystem zu erstellen. Um den Gesamtaufwand in Stunden pro UCP zu erhalten, wird dieses Ergebnis mit dem Produktivitätsfaktor multipliziert.

$$E_{geschätzt} = A_{Faktor} \cdot T_{Faktor} \cdot M_{Faktor} \cdot PF \qquad (3)$$

$E_{geschätzt}$ gibt den erwartenden Aufwand über folgende Phasen des verwendeten Aufwandsmodell (vergleiche Fußnote 1) wieder: Spezifikation, Konstruktion, Realisierung, Integration und Projektkoordination.

Ausgenommen sind hierfür Aufwände, welche für folgende Tätigkeiten entstehen:

- Inbetriebnahme: Diese Aufwände hängen stark mit der Architektur der Anwendungslandschaft sowie mit dem zu erstellenden Dokumentationsaufwand für Betrieb, Wartung, Schulung etc. zusammen. Von einer pauschalen Bewertung sehen wir deshalb ab.
- Projekttechnik: Die hieraus resultierenden Aufwände oder Kosten werden durch den ausgewiesenen Aufwand nicht berücksichtigt, da die für das Projekt genutzte Infrastruktur oft zentral bereitgestellt wird.
- Projektnebenaufwände: Darunter fällt zum Beispiel Einarbeitungszeit (für neue junge oder fachfremde Mitarbeiter) oder Reisezeit.

4 Vorgehen zur Entwicklung der UCP-Methode 2.0

Die Grundlage für die UCP-Methode 2.0 bilden folgende Punkte:

- Das in Kapitel 3 beschriebene Verfahren zur Bestimmung des A-Faktor mit einer hohen Genauigkeit sowie die höher gewichteten Aktoren.
- Die Verbesserung des T-Faktors und des M-Faktors.
- Die Möglichkeit, 26 Individualsoftware-Entwicklungsprojekte aus dem Umfeld betrieblicher Informationssysteme mit einer Regressionsanalyse auszuwerten.

Die durchgeführte Regressionsanalyse ist im Folgenden beschrieben:

Die zu minimierende Zielgröße ist die Standardabweichung der UCP-Methode 2.0, welche sich aus dem Ist-Aufwand und dem zu errechnenden Aufwand über alle Projekte ergibt. Als Nebenbedingungen und feste Größen wurden die Daten der Projektdatenbank (siehe Tabelle 3) gesetzt. Als anzupassende Variablen wurden ein veränderbares Gewicht der UUCP aus Aktoren angenommen, der Einfluss der Programmier-sprache (T13 in Tabelle 1) und die einzelnen Gewichte der neun Kostenfaktoren des M-Faktors (W_l in Tabelle 2). Weiterhin prüften wir mit einem beim A-Faktor hinzugefügten Exponent, ob große Projekte im Verhältnis aufwändiger sind. Der Produktivitätsfaktor wurde stets so eingestellt, dass die mittlere Abweichung über alle Projekte bei null liegt.

Damit konnte mit zwölf Freiheitsgraden bei einem Produktivitätsfaktor von 16,7 Stunden pro UCP eine Standardabweichung über alle Projekte von 17% ermittelt werden. Dabei wurden Aktoren mit {4, 8 12} UUCP viermal stärker als bei Karner gewichtet und Host-Anteile mit dem Faktor 2,2 belegt. Ein Unterschied zwischen der Programmiersprache Java und C# konnten wir nicht feststellen, da zu wenige Projekte auf der Basis von .NET vorlagen. Wir nehmen daher die Programmiersprachen als gleichwertig an. Für den Exponenten $E_{Produktivität}$ haben wir einen Wert ermittelt, welcher nahe bei 1 liegt. Eine Abweichung von diesem Wert brachte keine nennenswerte Verbesserung. Die ermittelten Gewichte W_i sind bereits in Tabelle 2 des M-Faktors enthalten, bei M_6 wurde ein Gewicht von 0 ermittelt (keine Korrelation, also auch nicht negativ).

Damit reduziert sich die Anzahl der Freiheitsgrade auf zehn; der ermittelte Produktivitätsfaktor und die Standardabweichung verbleiben unverändert.

Projekt	Ist-Aufwand	A-Faktor	T-Faktor	M-Faktor	errechneter Aufwand	Abweichung
#1	4824 h	251 UUCP	0,97	1,09	4429 h	-8%
#2	7894 h	357 UUCP	1,03	1,38	8522 h	8%
#3	7069 h	234 UUCP	1,57	1,40	8577 h	21%
#4	47069 h	937 UUCP	1,02	2,31	37001 h	-21%
#5	15500 h	418 UUCP	1,82	1,14	14525 h	-6%
#6	8300 h	187 UUCP	1,90	1,45	8595 h	4%
#7	3928 h	120 UUCP	1,64	1,21	3987 h	2%
#8	21696 h	717 UUCP	1,67	1,13	22670 h	4%
#9	42112 h	935 UUCP	1,68	1,37	36063 h	-14%
#10	5368 h	147 UUCP	1,04	1,62	4115 h	-23%
#11	728 h	65 UUCP	0,87	0,86	810 h	11%
#12	7825 h	189 UUCP	1,05	2,12	7038 h	-10%
#13	3680 h	151 UUCP	1,65	0,93	3866 h	5%
#14	2992 h	104 UUCP	0,92	1,28	2037 h	-32%
#15	4804 h	194 UUCP	1,04	1,75	5892 h	23%
#16	73592 h	1705 UUCP	1,07	2,60	79150 h	8%
#17	2936 h	231 UUCP	0,99	1,00	3824 h	30%
#18	55592 h	1867 UUCP	1,05	2,24	73464 h	**32%**
#19	7368 h	251 UUCP	1,08	1,08	4884 h	**-34%**
#20	2567 h	109 UUCP	1,12	1,26	2556 h	0%
#21	7250 h	292 UUCP	1,12	1,26	6846 h	-6%
#22	944 h	97 UUCP	0,71	0,91	1049 h	11%
#23	5362 h	279 UUCP	0,99	0,98	4508 h	-16%
#24	65000 h	1476 UUCP	1,16	2,22	63253 h	-3%
#25	2456 h	200 UUCP	0,97	0,86	2770 h	13%
#26	2432 h	179 UUCP	1,04	0,80	2485 h	2%
Standardabweichung über alle Projekte:						**17%**

Tabelle 3: Projektdatenbank ([Badent 2008])

5 Bewertung der UCP-Methode 2.0

Die Güte einer Schätzmethode wird durch die Standardabweichung als Maß für die Streuung der Schätzwerte ausgedrückt. Die UCP-Methode 2.0 weißt mit nur 17% bei zehn Freiheitsgraden eine sehr geringe Streuung auf. Beim Versuch die Anzahl der Freiheitsgrade durch das Nullsetzten der Gewichte der Kostenfaktoren mit dem kleinsten Gewicht (M_2 und M_8) auf acht zu reduzieren, stellten wir fest, dass die Standardabweichung nur minimal steigt.

Zum Vergleich haben wir die vorliegenden Projekte mit der UCP-Methode nach Karner geschätzt und eine Standardabweichung von 35% errechnet. Hierbei wurde ein Produktivitätsfaktor von 31,8 Stunden pro UCP verwendet, so dass auch hier die mittlere Abweichung über alle Projekte bei null lag.

Die Verbesserung wurde durch die vorliegenden Daten aus der Industrie möglich und beruht im Wesentlichen auf vier Punkten:

1. Die Use-Case-Points werden mit hoher Genauigkeit gezählt,
2. Aktoren werden stärker gewichtet,
3. die Programmiersprache fließt mit einer hohen Gewichtung in den T-Faktor ein und
4. der M-Faktor wurde vollständig überarbeitet.

Aufgrund der einfachen und schnellen Anwendbarkeit und der hohen Schätzgenauigkeit sehen wir damit die verbesserte UCP-Methode als Mittel der Wahl bei Aufwandschätzungen von Informationssystemen in frühen Projektphasen.

Kontakt

- Patrick Badent

  Kontakt:
  http://tinyurl.com/PatBadent

- Dr. Stephan Frohnhoff

  Kontakt:
  http://tinyurl.com/SFrohnhoff

- Prof. Dr. Christian Greiner

  Kontakt:
  http://tinyurl.com/ChrGreiner

Quellenangaben

- **[Badent 2008]** P. Badent: Projektaufwand in Abhängigkeit von Organisation und Vorgehen, Forschungsprojekt zwischen Capgemini Deutschland, der Hochschule München und der Universität Magdeburg
- **[Bundschuh 2004]** M. Bundschuh, A. Fabry: Aufwandschätzung von IT-Projekten; Bonn; 2004
- **[Boehm 1986]** Wirtschaftliche Software-Produktion; Forkel-Verlag, Wiesbaden; 1986
- **[Boehm 2000]** Berry Boehm, et al.: Software Coast Estimation with COCOMO II; Prentice Hall; 2000
- **[Booch 1999]** Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modeling Language User Guide; Addison-Wesley Object Technology Series; 1999
- **[Cockburn 2003]** Cockburn, A.: Writing effec-tive use cases. The Agile Software De-velopment Series. 8. Auflage, Addison-Wesley; 2003
- **[Dumke 2007]** Reiner Dumke, et al.: Modellbezogene Use-Case-Identifikation für die UCP-basierte Aufwandschätzung; Otto-von-Guericke-Universität Magdeburg; 2007
- **[Jacobson 1993]** Jacobson, I.; Christerson, M.; Jonsson, P.und Overgaard, G.: Object-Oriented Software Engineering. A Use Case Driven Approach; 4. Auflage, Addison-Wesley; 1993
- **[Frohnhoff 2006]** Stephan Frohnhoff, et al.: Use Case Points in der industriellen Praxis; IWSM/MetriKon; 2006
- **[Frohnhoff 2007]** Stephan Frohnhoff; Rainer Dumke: Revised Use Case Point Method; IWSM/MetriKon; 2007
- **[Karner 1993]** Gustav Karner.: Metrics for Objectory; University of Linköping Sweden, No. LiTHIDA-Ex-9344; 1993
- **[Schneider 1998]** Schneider, G.; Winters, J.: Applying Use Cases - A Practical Guide; Addison