Use Case Points in der industriellen Praxis

Stephan Frohnhoff, Volker Jung, Gregor Engels
sd&m AG, Berliner Str. 76, D-63065 Offenbach
frohnhoff@sdm.de

Zusammenfassung:

Im industriellen Umfeld besteht ein hoher Bedarf nach einer Methode zur schnellen und sicheren Abschätzung von zu erwartenden Aufwänden für Software-Entwicklungsprojekte. In einem industriellen Praxistest wurde dazu anhand von zehn Entwicklungsprojekten die Use Case Point Methode mit den tatsächlich eingetretenen Aufwänden nach Projektabschluss verglichen und als praxistauglich bestätigt. Es werden konkrete Vorschläge zur Verbesserung der Methode aufgezeigt, mit denen sich die Schätzgenauigkeit signifikant verbessern ließ.

Schlüsselbegriffe

Projektschätzung, Top-Down-Schätzung, Use Case Points, empirische Daten, UCP

Abstract:

Fast and precise effort estimation of software development projects is crucial in IT industry. Within a case study the Use Case Point method has been applied to 10 commercial software development projects and compared with the incurred project efforts after project close. The method is ready for use in commercial projects. We propose appropriate improvements of the Use Case Point method leading to significantly higher estimation accuracy.

Keywords

project estimation, top down estimation, use case points, empirical data, UCP

1 Einführung

Zu Beginn eines industriellen Software-Entwicklungsprojektes steht die Abschätzung des zu erwartenden Projekt-Aufwandes auf der Basis einer groben Spezifikation, in der die funktionalen und nicht funktionalen Anforderungen beschrieben sind. Dies geschieht in der industriellen Praxis in der Regel anhand einer Projektschätzung [14].

Wir unterscheiden zwei Vorgehensweisen, die zu einer Projektschätzung führen: Die Top-Down-Schätzung kann sehr früh im Projektverlauf eingesetzt werden, ist aber in der Regel für Festpreis-Garantien in der industriellen Auftragsvergabe zu unsicher. Hierfür wird die meist präzisere Bottom-Up-Schätzung verwendet. Sie verlangt aber eine möglichst detaillierte Spezifikation und ist zeitaufwändiger. Die

Blickrichtung Top-Down versus Bottom-Up orientiert sich dabei am Vorgehensmodell, in welchem „oben“ die Spezifikation und „unten“ die Realisierung steht.

Die ***Bottom-Up-Schätzung*** setzt auf den zu programmierenden Software-Komponenten (Realisierungskomponenten) auf. Ausgehend von der Spezifikation wird hierzu ein zukünftiges Software-System entworfen. Dies erfolgt in einem Verfeinerungsschritt auf Basis von Annahmen und Prämissen so detailliert wie möglich. Die Realisierungskomponenten des Software-Systems werden einzeln hinsichtlich des Zeitbedarfs für die Realisierung geschätzt und die Teilaufwände summiert. Projekt-Querschnittsaufwände wie Projektleitung oder Qualitätssicherung werden mit pauschalen Prozentwerten dem Gesamtaufwand hinzu geschlagen.

Bei der ***Top-Down-Schätzung*** klassifiziert man die funktionalen Anforderungen aus der Spezifikation in abzählbare Einheiten und ordnet diesen gemäß ihrer geschätzten Komplexität Punkte zu. Durch Summation der Punkte wird ein Maß für den fachlichen Umfang der Anforderungen ermittelt. Die Einflüsse durch den Projektkontext und die Umsetzungs-Technologie werden in einem Faktor berücksichtigt, der das zunächst technologie-unabhängige Maß weiter justiert. Aufgrund empirischer Erfahrungswerte wird dieses Maß dann in einen Projektaufwand umgerechnet. Weit verbreitet sind die Function-Point-Methode [2] und deren Weiterentwicklungen [13], wie z.B. COSMIC Full Function Point [1].

Diese Schätzmethoden haben zwei Schwächen: Erstens liegt ihr Ursprung in der strukturierten Analyse. Die Methoden berücksichtigen daher nur unzureichend moderne Spezifikationsmethoden wie OOA und OOD [5], welche heute in der industriellen Praxis weit verbreitet sind. Zweitens ist der Schätzprozess möglichst effizient in die frühen Phasen der Spezifikation einzubetten [4], d.h. Berücksichtigung der Unified Modeling Language (UML) gemäß [17] und der Spezifikation der funktionalen Anforderungen mit Hilfe von Use Cases [6].

Den Anforderungen an eine moderne Top-Down-Schätzung wird aus unserer Erfahrung die UCP-Methode [10] am Besten gerecht. Dies ist eine Schätzung basierend auf ***Use Case Points*** (UCP), die zusammen mit der objektorientierten Analyse und insbesondere zusammen mit den Use Cases (Anwendungsfälle) entwickelt wurde. Sie ist zudem leicht, d.h. innerhalb nur eines Tages, für einen erfahrenen Software-Ingenieur erlernbar und erlaubt bei ausreichender Kenntnis der Spezifikation eine sehr effiziente Projekt-Schätzung: Ein Projekt mit 300 Bearbeitertagen (BT) Projektumfang und ca. 25.000 Zeilen Source-Code (SLOC) kann nach unseren Erfahrungen in einer Bearbeiterstunde (Bh) abgeschätzt werden. Diese Schätzmethode aus dem industriellen Kontext ist noch wenig verbreitet und erforscht [3], wenngleich neueste Werkzeuge zur UML-Modellierung bereits UCP-Tools integrieren. Insbesondere fehlt jedoch ein umfassender Praxistest.

Der vorliegende industrielle Praxistest vergleicht die Projekt-Aufwände gemäß UCP-Methode von zehn Entwicklungsprojekten mit der Bottom-Up-Schätzung

sowie den tatsächlich eingetretenen Aufwänden nach Projektabschluss. Zunächst wurde die originale UCP-Methode angewendet und dann aufgrund der Praxisdaten ein Vorschlag für eine erweiterte UCP-Methode entwickelt sowie ein Toolkit aufgebaut. Auch diese erweiterte UCP-Methode bedarf noch der Verbesserung hinsichtlich der Berücksichtigung des gewählten Projektvorgehens. Ist im Folgenden allgemein von der UCP-Methode die Rede, bezieht sich die Aussage sowohl auf die originale wie die erweiterte Methode.

2 Einordnung des industriellen Praxistests in den wissenschaftlichen Kontext

In der originalen UCP-Methode gemäß Karner [11] werden den Use Cases einer Spezifikation, abhängig von der jeweiligen Komplexität, Gewichte zugeordnet (einfach = 5, mittel = 10, komplex = 15) und diese summiert. Die Aktoren werden nach Typ klassifiziert (API = 1, Protokoll-Schnittstelle = 2, Benutzer-Interface = 3) und hinzuaddiert, die Gesamtsumme bilden die Use Case Points (UCP).

Die zu erwartende Komplexität aufgrund des technischen Umfeldes wird anhand von 13 Einflussgrößen wie Verfügbarkeits- und Sicherheitsanforderungen mit jeweils 0 bis 5 Punkten bewertet (0 Punkte bedeutet „kein Einfluss auf das Projekt“, 5 Punkte bedeuten dominanten Einfluss auf das Projekt). Anhand einer einfachen Formel wird hieraus der technische Komplexitätsfaktor (TCF) im Wertebereich [0,6 – 1,3] ermittelt.

Analog wird die organisatorische Komplexität sowie das Umfeld der Projektarbeit anhand von 8 Einflussgrößen abgeschätzt und ebenfalls mit einer einfachen Formel auf den *Environmental Factor* (EF) zusammengefasst. Dieser hat einen Wertebereich von [0,425 – 1,7].

Aus diesen Größen berechnen sich die bereinigten Use Case Points zu

$$\text{bereinigte UCP} = \text{UCP} * \text{TCF} * \text{EF}.$$

Durch Multiplikation mit einem an der jeweiligen Organisation zu eichenden Produktivitätsfaktor (PF) im erwarteten Wertebereich [20 – 35 Bh/UCP] ergibt sich daraus der geschätzte Projekt-Aufwand in Bearbeiter-Stunden.

Die Grundlage für den hier vorliegenden Praxistest stellen zehn kommerzielle Software-Entwicklungsprojekte, welche vom Software-Haus sd&m AG als Dienstleister für Kunden unterschiedlicher Branchen in individueller Auftragsfertigung abgewickelt wurden. Für die Anwendung der UCP-Methode muss die Spezifikation weder Use Case Diagramme enthalten noch müssen Use Cases beschrieben sein: Eine Modellierung in Form von Ereignisgesteuerten Prozessketten (EPK) oder eine rein prosaische Beschreibung reicht aus, wenn daraus die Use Cases abzählbar benannt werden und deren Komplexitäten grob abgeschätzt werden können.

In der Literatur [16] wurde an dieser Methode häufig kritisiert, dass die Definition eines Use Cases nicht eindeutig ist und wesentlich von dem gewählten Abstraktionsgrad abhängt. Dieses Problem wurde hier dadurch gelöst, dass im Rahmen der erweiterten UCP-Methode ein Nutzungskonzept aufgestellt wurde, d.h. welcher Abstraktionsgrad zu wählen ist und welcher Komplexitätsstufe ein Use Case zuzuordnen ist.

3 Darstellung des Vorgehens im Praxistest

Zur Überprüfung der UCP-Methode wurden die Projekte nach Projekt-Ende auf Basis der Grob-Spezifikation geschätzt und mit den tatsächlich benötigten Zeitaufwänden verglichen. Durch dieses Vorgehen wurden jeweils für die UCP-Schätzung mögliche Unschärfen in der Spezifikation oder Veränderungen der Spezifikation im Projektverlauf ausgeschlossen.

Nachfolgend sind die untersuchten Projekte aufgeführt. Dabei bezeichnet *Branche* den Kunden nach Branche, für den das Projekt durchgeführt wurde, *IST-Aufwand* den tatsächlich benötigten Projektaufwand in Bearbeiter-Stunden (Bh) nach Projekt-Ende, *SLOC* die Anzahl der erstellten Zeilen Source-Code, *UCP* die Maßzahl für Use Case Punkte der im Rahmen des Praxistests erweiterten UCP-Methode, *PF* den gewählte Produktivitätsfaktor und *geschätzter Aufwand* den hieraus ermittelte Projektaufwand.

Projektdaten						
	gezählt		erweiterte UCP-Methode			Abweichung
Branche	**A: IST-Aufwand [Bh]**	**SLOC**	**UCP**	**PF**	**B: geschätzer Aufwand [Bh]**	**(B - A)/A**
Automobilhersteller	15.200		406,5	32,5	13.212	-13%
Bekleidungsbranche	728	15.685	42,0	19,5	819	13%
Finanzdienstleister 1	2.992	11.554	51,4	19,5	1.002	-67%
Finanzdienstleister 2	3.680	33.850	115,0	32,5	3.738	2%
Logistiker 1	944	10.722	49,0	19,5	956	1%
Logistiker 2	2.567	22.411	61,0	32,5	1.983	-23%
Logistiker 3	7.250	102.402	308,0	32,5	10.010	38%
Telekom-Branche 1	2.456	26.088	125,0	19,5	2.438	-1%
Telekom-Branche 2	2.432	26.092	111,0	19,5	2.165	-11%
Telekom-Branche 3	1.056		52,9	19,5	1.032	-2%

Bild 1: Kenndaten der untersuchten Projekte im Praxistest

Das Projekt Finanzdienstleister 1 wurde nach Aufwandsverrechnung durchgeführt. Die Ziele haben sich über den Projektverlauf wesentlich verändert und resultierten in erheblichen Mehraufwänden. Alle anderen Projekte wurden zum Festpreis abgewickelt ohne signifikante Change Requests. Dies mag erklären, warum die Messergebnisse für Finanzdienstleister 1 ausreißen.

Als IST-Aufwand wurden einheitlich die Aufwände ab Feinspezifikation bis Bereitstellung zur Abnahme inklusive der zugehörigen QS-Maßnahmen berück-

sichtigt. Im Rational Unified Process (RUP) [12] entspricht dies in etwa den Phasen Elaboration und Construction. Nicht im IST-Aufwand berücksichtigt sind also insbesondere Vorstudien, sowie Inbetriebnahme, Schulung und Gewährleistung. Ebenso sind Projektnebenaufwände ausgegrenzt, wie z.B. generelle Einarbeitung und Teamfindung insbesondere bei steilem Projektaufbau in Großprojekten, Reisezeiten und Overhead durch Arbeiten an mehreren Standorten (auch Offshore). Ebenfalls wird unterstellt, dass die generell notwendige Projekttechnik vorhanden ist, d.h. Software-Entwicklungsumgebung, Konfigurations- und Releasemanagement, technische Infrastruktur (Server, Netzwerk, Basis-Software).

Alle Projektaktivitäten wurden durch die Projekt-Mitarbeiter in der Granularität von ¼ Stunde in einem System während der Projektlaufzeit täglich erfasst und den unternehmensweit einheitlich definierten Aktivitätskonten zugeordnet. Dadurch ist eine klare und einheitliche Abgrenzung möglich, welche Aktivitäten zum Projektaufwand hinzugerechnet und welche ausgegrenzt werden.

4 Eignung von UCP in industriellen Softwareprojekten

Im Rahmen des durchgeführten Praxistests konnte mit der original UCP-Methode für die zehn Software-Entwicklungsprojekte nur eine mäßige Übereinstimmung von Schätzung und IST-Aufwänden mit einer mittleren Abweichung von 14 % bei einer Standardabweichung von 40 % erzielt werden. Gründe sind:

- TCF und EF beruhen auf den ursprünglichen Faktoren der Function Point Methode und sind nicht mehr zeitgemäß; andere Methoden wie COSMIC FFP haben diese besser weiterentwickelt.
- Es wird eine lineare Abhängigkeit von TCF und EF an den UCPs unterstellt, was unserer Erfahrung aus der Praxis widerspricht.
- Der Produktivitätsfaktor ist für jede Organisation zu eichen. Im vorliegenden Praxistest wurden die Projekte von unterschiedlichen Teams an unterschiedlichen Unternehmensstandorten in unterschiedlichen Branchen bearbeitet. Das gewählte Projektvorgehen unterscheidet sich dabei erheblich, der Produktivitätsfaktor kann nicht einheitlich gewählt werden.

Hinsichtlich des Projektvorgehens können zwei Projekttypen identifiziert werden:

Projekte mit **„Standard-Entwicklungsprozess“**, in denen größere Teams (7 Mitarbeiter und mehr) zusammenarbeiten und einem formalisierten Vorgehensmodell mit fest definierten Ergebnistypen folgen, und Projekte mit wenig formalem

„schlanken Entwicklungsprozess“, die in der Regel von kleineren Projektteams umgesetzt werden.

Der Praxistest hat gezeigt, dass diese unterschiedlichen Entwicklungsprozesse erhebliche Auswirkungen auf die Aufwandsschätzung haben. Daher stellen wir detaillierter dar, wodurch der jeweilige Typ charakterisiert ist:

Beispiel für Standard-Entwicklungsprozesse sind RUP [12] oder das V-Modell [8]. Konkret treffen für solche Projekte mehrere oder alle Eigenschaften aus folgender Aufstellung zu:

- Alle Querschnittsrollen (Projektleitung, Chefdesign, Qualitätsmanagement, Konfigurationsmanagement, weitere) sind jeweils voll besetzt, es gibt keine Personalunionen.
- Projektübergreifende Querschnittsaufwände (Programm-Management, übergreifendes Chefdesign) fallen an.
- Alle im Prozess vorgesehenen Dokumente werden erstellt.
- Es herrscht eine ausgesprochene Schriftlichkeit im Projektablauf vor (formale Protokolle werden zu allen Sitzungen angefertigt, ein formales Berichtswesen ist etabliert).
- Formale Entscheidungswege müssen grundsätzlich eingehalten werden.
- Formale Reviews und Abnahmen der Zwischenergebnisse sind erforderlich.

Bei Projekten mit schlankem Entwicklungsprozess ist die volle Aufbau- und Ablauforganisation nicht erforderlich. Dafür ist in der Regel eine Kombination aus folgenden Punkten Voraussetzung:

- Das Projekt hat keine hohe Kritikalität.
- Das Projektumfeld, insbesondere die Anforderungen, sind vergleichsweise stabil.
- Das Auftraggeber/Auftragnehmer-Verhältnis ist etabliert und von gegenseitigem Vertrauen geprägt.
- Das Projektteam, seine Rollen und sein Vorgehen sind bereits eingespielt („jelled team“ [7]).

Dadurch lässt sich das Projekt effizienter organisieren, reduzierte Querschnittsaufwände machen es „schlanker“:

- Das Team steuert sich weitgehend selbst, es benötigt weniger Projekt-Management
- Querschnittsrollen müssen nicht voll besetzt werden, Personalunionen sind möglich (bei kleineren oder mittelgroßen Projekten).

- Zwischenmeilensteine können entfallen und damit organisatorischer Aufwand und Kommunikationsaufwand eingespart werden.
- Einzelne Dokumente zur Dokumentation des Entwicklungsprozesses können entfallen, z.B. Projekthandbuch, QM-Plan in prosa, Projekttagebuch, Qualitätszieledokumente, Testplan. Notwendig bleiben Testfallbeschreibung und grafischer/tabellarischer QM-Plan. Spezifikationsdokumente können schlanker ausfallen, indem auf eine ausführliche IST-Beschreibung verzichtet wird.
- Für einzelne Zwischenergebnisse können formale Reviews entfallen.
- Die Anforderungen an das Berichtswesen sind vergleichsweise niedrig, d.h. z.B. keine oder nur informale Statusberichte, keine oder nur wenige Gesprächsprotokolle.

Im Rahmen dieses Praxistests wurde die UCP-Methode daher erweitert und insbesondere die beiden unterschiedlichen Entwicklungsprozesstypen durch je einen eigenen Produktivitätsfaktor berücksichtigt.

5 Erweiterte UCP-Methode und Darstellung der Ergebnisse

Zunächst führen wir folgende Begrifflichkeit zur Differenzierung der Aufwands-Teile am Projekt ein:

A-Aufwand: Use Cases definieren den funktionalen (anwendungsfachlichen) Umfang, der im Projekt bewältigt werden muss. In betrieblichen Informationssystemen wird dieser in Anwendungssoftware (A-Software) [15] implementiert. Der dazu benötigte Aufwand ist proportional zu den Use Case Points, die Proportionalitätskonstante nennen wir **A-Faktor**.

T-Faktor entspricht dem „Technical Complexity Factor“ (TCF) der originalen UCP-Methode und berücksichtigt die technologischen Randbedingungen auf den Gesamtaufwand.

M-Faktor (Management-Faktor) zur Bestimmung der organisatorischen Komplexität und des Umfeldes entspricht in erster Näherung dem „Environmental Factor“ (EF) der originalen UCP-Methode, allerdings wurde gegenüber der Original-Methode eine Reduktion von 8 auf nur 6 Einflussgrößen vorgenommen.

Eine wesentliche Erweiterung der UCP-Methode besteht in der Normierung der Ausprägungen der Komplexitätswerte für den A-, T- und M-Faktor durch Beispiele und konkretere Vorgaben, wie in Kapitel 5.1 bis 5.3 dargestellt wird. Dadurch konnte eine hohe Reproduzierbarkeit der Schätzung durch unterschiedliche Schätzer erreicht werden. Zum Beispiel wurde die Anforderungsspezifikation einer öffentlichen Ausschreibung von fünf unabhängigen Schätzern bewertet und die gleiche Anzahl Use Case Points ermittelt.

Eine weitere wesentliche Erweiterung der UCP-Methode besteht in der Klassifizierung von Projekten hinsichtlich der gewählten Entwicklungsprozess-Typen (Kapitel 5.4).

5.1 A-Faktor

Um die Granularität und Eindeutigkeit einer Use Case Spezifikation zu verbessern, wurde ein umfangreiches Nutzungskonzept aufgestellt. Die Granularität der Use Case Beschreibung orientiert sich dabei an den Abnahmekriterien des Projektes durch den Auftraggeber. Es gilt das Minimal-Prinzip: Es wird nur das implementiert, was minimal notwendig ist, um die Abnahmekriterien zu erfüllen.

Zur einheitlichen Definition der Komplexitätsstufe wird ein Use Case durch die Zahl seiner Hauptszenarien, Schritte und Dialoge bewertet. Folgende Definition für die Komplexitätsstufen wurde gefunden:

Einfach: maximal 3 Hauptszenarien, Schritte und Dialoge => 5 Punkte

Mittel: maximal 7 Hauptszenarien, Schritte und Dialoge => 10 Punkte

Komplex: mindestens 8 Hauptszenarien, Schritte oder Dialoge => 15 Punkte

Nach unserer Erfahrung ist der Einfluss der Aktoren auf die UCPs zu vernachlässigen. Er wurde im Praxistest zwar beibehalten, könnte aber in einer späteren Weiterentwicklung entfallen.

5.2 T-Faktor

Die Einflussgrößen für den T-Faktor wurden aus der Originalmethode übernommen, aber die Komplexitätswerte (0 bis 5 Punkte) wurden über Analogien und Beispiele normiert. In Abbildung 2 sind je Einflussgröße die Normierungswerte in der dritten Spalte zusammengefasst dargestellt.

T-Faktor ("Technical Complexity Factor")			
Nr.	**Einflussgröße**	**Normierungswerte für Komplexität (0 bis 5 Punkte)**	**Gewicht**
T1	Verteiltes System	Wie stark verteilt ist die Architektur des Systems? 0: Monolithisches System 3: 3-tier mit Randsystemen 5: Hochverteilte Systemarchitektur	2,0
T2	Performanz- und Lastanforderungen	Wie hoch sind Performanz- und Lastanforderungen an das System? 0: keinerlei Anforderungen 3: übliche Performance-Lastanforderungen 5: hohe Performance-Lastanforderungen, z.B. Lastverteilung, Number-Crunching	1,0
T3	Effizienz der Benutzungs-schnittstelle	Wie effizient muss die Benutzungsschnittstelle zu bedienen sein? 0: keine Anforderungen, z.B. Batches 3: normale Benutzungsschnittstelle, z.B. Web-GUI, einfaches Swing-GUI 5: hochintegrierte, effiziente Benutzungsschnittstelle, z.B. Akkord-Anforderungen, Makros	1,0
T4	Komplexität der Geschäftsregeln und Berechnung	Wie komplex sind die Geschäftsregeln / die Berechnungen im System? 0: Nur einfachste Regeln, keine Berechnungen 3: Normale Komplexität 5: Sehr komplexe Regeln / Berechnungen	1,0
T5	Wiederverwend-barkeit	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	1,0
T6	Installations-freundlichkeit	Wie einfach muss die Software zu installieren sein? 0: Keine Installationsanforderungen 3: Normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen) 5: Hohe Installationanforderungen (Eingeständige Istallation durch eine hohe Zahl von Endkunden)	0,5
T7	Benutzer-freundlichkeit	Wie hoch sind die Anforderungen an die Benutzerfreundlichkeit des Systems? 0: keine Anforderungen (keine Benutzer) 3: normale Anforderungen (GUI, Hilfesystem) 5: hohe Anforderungen (z.B. GUI-Varianten für Benutzergruppen, Internationalisierung, Wizards, Fehlertoleranz)	0,5
T8	Portabilität	Wie hoch sind die Anforderungen an die Portabilität des Systems? 0: keine Anforderungen z.B. Software, die nur einmal läuft 3: normale Anforderungen (eine Zielplattforn, üblicher Grad an Abstraktion) 5: hohe Anforderungen (z.B. Cross-Platform, Widows + Unix)	2,0
T9	Variablilität (Änderungs-freundlichkeit)	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-Schnittstelle	1,0
T10	Verfügbarkeit	Wie hoch sind die Verfügbarkeitsanforderungen an das System? 0: keinerlei Anforderungen (keine parallelen Transaktionen) 3: übliche Verfügbarkeitsanforderungen 5: Hohe Verfügbarkeitsanforderungen (24/7-Betrieb, 99,x% Verfügbarkeit, hohe Zahl paralleler Transaktionen)	1,0
T11	Sicherheits-anforderungen	Wie hoch sind die Sicherheitsanforderungen an das System? 0: keinerlei Anforderungen 3: übliche Sicherheitsanforderungen (Authentifizierung und Authorisierung) 5: Hohe Sicherheitsanforderungen (Zertifikate, verschlüsselte Kommunikation, Kryptographie)	1,0
T12	Systemnutzung durch Dritte	Bietet das System direkte Zugänge für Dritte (andere als den Kunden) an? 0: keine Anforderungen 3: Systemnutzung durch Endkunden (B2C-Kunden des Kunden) 5: z.B. externe Serviceschnittstellen, B2B-Systeme, Handelsplattformen	1,0
T13	Trainingsintensiv	Ist die Software so komplex, dass besondere Anwenderschulungen erforderlich sind? 0: keine Schulung erforderlich, instantan nutzbar 3: übliche Anwenderschulung oder Selbststudium 5: Training erforderlich	1,0

Bild 2: Liste der T-Faktoren mit Komplexitätswerten und Gewichtung

5.3 M-Faktor

Für den M-Faktor zur Bestimmung der organisatorischen Komplexität und des Umfeldes wurde gegenüber der Original-Methode eine Reduktion von 8 auf nur 6 Einflussgrößen vorgenommen. Die Faktoren E1: *Kenntnis des Rational Unified Prozesses* und E3: *OO-Erfahrung* haben nach unserer Einschätzung keinen Einfluss mehr auf das Schätzergebnis bei sd&m, da alle Mitarbeiter diesbezüglich gleich gut ausgebildet sind.

Ferner wurden die Komplexitätswerte (0 bis 5 Punkte) über Analogien und Beispiele gemäß der nachfolgenden Abbildung 3 normiert und durch Beispiele konkretisiert (siehe dritte Spalte).

M-Faktor ("Environmental Factor")			
Nr.	**Einflussgröße**	**Normierungswerte für Komplexität (0 bis 5 Punkte)**	**Gewicht**
M1	Kenntnis der Anwendung	Wie gut kennt das Team die Anwendung? 0: kompletes Neuland 3: Anwendung und Umfeld ist dem Team zum Teil bekannt, bzw. einem Teil des Teams bekannt 5: Anwendung und Umfeld ist dem Team vertraut (typisch bei hohen Releasenummern und Minor Releases)	0,5
M2	Fähigkeit des Chefdesigners	Wie erfahren ist der Chefdesigner (FCD) ? 0: relativ unerfahren für die Aufgabe; wenig vertraut mit der Anwendungsdomäne 3: normale Erfahrung für die Aufgabe; Anwendungsdomäne ist bekannt 5: sehr erfahren für die Aufgabe; sehr gut vertraut mit der Anwendungsdomäne	0,5
M3	Motivation	Wie motivaiert ist das Team? 0: unmotiviert 3: motiviert 5: ausgezeichnet motiviert	1,0
M4	Stabilität der Anforderungen	Wie stabil sind die Anforderungen an das System? 0: sehr hohe Änderungsrate auch grundlegender Anforderungen 3: normale Änderungsrate, übliches Änderungsmanagement 5: sehr stabile Anforderungen, kein Änderungsmanagement erforderlich	2,0
M5	Verfügbarkeit der Teammitglieder	Wie ist die Verfügbarkeit der Teammitglieder ? 0: alle Teamitglieder > 90% verfügbar 3: Z.B.: einige Teammitglieder < 70% verfügbar 5: signifikanter Anteil des Teams ist wenig verfügbar	- 1,0
M6	Komplexität der Sprach- / Entwicklungs-umgebung	Wie komplex ist die Sprache / Entwicklungsumgebung des Systems: 1: einfach, z.B. Perl, PHP 3: normal (z.B. Java, Cobol) 5: komplex, exotisch, schlecht verstanden, z.B. Assembler	- 1,0

Bild 3: Liste der M-Faktoren mit Komplexitätswerten und Gewichtung

5.4 Produktivitätsfaktor

Die untersuchten Projekte unterscheiden sich hinsichtlich der gewählten Entwicklungsprozess-Typen (schlank versus Standard). Es konnte nachgewiesen werden, dass für die beiden Typen unterschiedliche Produktivitätsfaktoren zu wählen sind, um eine signifikant bessere Passung zwischen IST-Aufwand und UCP-Schätzung zu erreichen.

Es liegt nahe, dies in Verbindung mit der Projektgröße zu bringen. Bild 1 zeigt jedoch, dass z.B. das Projekt von „Logistiker 2“ einen vergleichbaren IST-Aufwand wie das Projekt von „Telekom-Branche 1“ benötigt hat, die Anzahl der Use Case Points sich aber erheblich unterscheidet. Dies zeigt, dass die Größe des Projektes ausgedrückt in UCP nicht direkt mit dem Produktivitätsfaktor korreliert. Ebenso lässt sich eine direkte Korrelation mit der Größe ausgedrückt in SLOC widerlegen. Unsere Analysen legen nahe, dass vielmehr der gewählte Entwicklungsprozesstyp die Differenzierung im benötigten Aufwand bedingt. Dies ist in der originalen UCP-Methode nicht berücksichtigt und wird von uns zunächst durch einen unterschiedlichen Produktivitätsfaktor in der erweiterten UCP-Methode abgebildet.

In einer Voruntersuchung zu diesem Praxistest wurden dazu verschiedene Entwicklungsprojekte in die Cluster „schlanker Entwicklungsprozess“ und „Standard-Entwicklungsprozess“ aufgeteilt und der jeweils am besten passende Produktivitätsfaktor für dieses Cluster ermittelt zu

- schlankem Entwicklungsprozess: PF = 19,5 Bh/UCP
- Standard-Entwicklungsprozess: PF = 32,5 Bh/UCP

Mit diesen Werten wurden die Projekte dieses vorliegenden Praxistest bewertet (siehe Bild 1). Für die 10 untersuchten betrieblichen Anwendungsentwicklungsprojekte konnte mit Hilfe der so erweiterten UCP-Methode eine sehr gute Passung von geschätztem Aufwand zum IST-Aufwand mit einer mittleren Abweichung von 6 % bei einer Standardabweichung von 27 % erreicht werden. Die originale UCP-Methode erreichte dagegen eine mittlere Abweichung von 14 % bei einer Standardabweichung von 40 %. Damit konnte die Schätzgenauigkeit signifikant verbessert werden.

5.5 Eignung der erweiterten UCP-Methode

In der Literatur gibt es bisher wenig Aussagen, für welche Projektarten die Methode geeignet ist und wo sie noch nicht passt. Für unsere erweiterte Methode können wir folgende Klassifizierung treffen:

Geeignet	**Nicht geeignet**
Individualentwicklung	Produktanpassungen
Neuentwicklung	Wartung, d.h. geringfügige Anpassung bestehender Systeme
Neuentwicklung fachlicher Geschäftsprozesse in betrieblichen Anwendungen	Technikstufen, Steuerungssysteme
Stammdaten-Pflegesysteme	Batch-Prozesse

Nicht geeignet ist die Methode immer dann, wenn der Umfang von System-Anpassungen nur schlecht durch Use Cases erfasst wird. Dies ist z.B. bei technischen Stufen der Fall, in denen sich die Fachlichkeit (A-Faktor) nur wenig ändert.

6 UCP-Toolkit zur Berechnung des Projektaufwandes

Zur Ermittlung des Projektaufwandes mit Hilfe der erweiterten UCP-Methode wurde ein einfaches Excel-Template aus fünf Arbeitsblättern entworfen. Zu jedem Schritt der Methode gehört ein Arbeitsblatt. Damit werden auch mit der Methodik nur wenig vertraute Software-Ingenieure sicher durch den Schätzprozess geführt.

Zunächst werden alle Use-Cases des Projekts erfasst und nach unterschiedlichen Messgrößen („Maße“) charakterisiert (Bild 4):

Use-Case-Points Schätzung *Vorlage V. 1.0 vom 10.04.2006*

Projektname: **Datum:**

PASS-Kürzel: **Autor:**

Use-Case-Beschreibung			Maße			Use-Case-Points			Re-Use in %	Bemerkung
Gruppe	Nr.	Name	Anzahl Szenarien	Anzahl Schritte	Anzahl Dialoge	Berechnet	Intuitiv	Effektiv		
		Summen:	21	43	10	60		47,5		
Dialog	1	Beispiel: einfacher Use Case	2	3	2	5		5,0		
Dialog	2	Beispiel: mittlerer Use Case	2	4	2	10		10,0		
Dialog	3	Beispiel: komplexer Use Case	8	14	2	15		15,0		
Batch	4	Beispiel: Use Case mit Wiederverwendung	8	14	2	15		7,5	50%	Wiederverwendung aus Release 1
Batch	5	Beispiel: Korrektur durch intuitiven Wert	1	8	2	15	10	10,0		Schritte sind alle einfach
	6									

Bild 4: Arbeitsblatt zur Ermittlung der Use Cases

- Anzahl Szenarien: Die Anzahl der unterschiedlichen Erfolgs-Szenarien und nichttrivialen Fehlerszenarien im Use-Case.
  - Ein Erfolgs-Szenario ist ein möglicher fachlicher Ablauf des Use-Cases, der zum Erfolg (Erreichen des Business Goal) führt, z.B.: das Haupt-Szenario („Main Flow“) des Use-Cases oder fachliche Alternativszenarien des Use-Cases (triviale Abweichungen, z.B. „Anzeige einer Meldung, dann Abbruch“ werden nicht mitgezählt)
  - Fehlerszenarien sind solche, die nicht zum Erfolg (Erreichen des Business Goal) führen; fachliche Fehlerszenarien werden gezählt (wenn fachliche Schritte zur Fehlerbehandlung durchlaufen werden), triviale Fehlerszenarien, z.B. „Anzeige einer Meldung, dann Abbruch“ werden nicht gezählt.
- Anzahl Schritte: Die Anzahl der einzelnen unterschiedlichen Schritte im Use-Case in allen Szenarien. Typische Beispiele für Schritte sind: Eingabe eines oder mehrerer Werte in einen Dialog (ohne dass dazwischen ein Server-Roundttrip erfolgt), Server-Aufrufe von Anwendungsfunktionen, Server-Transaktionen.

- Anzahl Dialoge: Die Anzahl der einzelnen unterschiedlichen Dialoge des Use-Cases. Bei Use-Cases auf Serverseite wird hier abweichend die Anzahl der Schnittstellen-Messages gezählt. Dialoge werden wie folgt gezählt: Jeder Reiter eines Dialogs (mit siginifikanten fachlichen Unterschieden) wird als eigener Dialog gezählt, jeder Frame einer Webseite (mit signifikanten Steuerelementen) wird als eigener Dialog gezählt, Pop-Up-Meldungen und Menüs werden nicht gezählt, Anzeigeseiten werden nur gezählt, wenn Daten eingegeben werden können.

Es ist nicht zwingend erforderlich, bei allen Use-Cases immer alle 3 Maße anzugeben. Fehlt ein Maß (da es z.B. nur mit großer Unsicherheit angegeben werden kann), wird die Klassifizierung aus den anderen vorgenommen.

Optional kann ein Wiederverwendungsanteil (Re-Use in %) pro Use-Case angegeben werden. Dies ist z.B. erforderlich, wenn der Use-Case im Projekt nur erweitert oder nicht vollständig neu entwickelt wird, für den Use-Case zum Teil bereits eine Produktsoftware oder ein Framework realisiert ist oder der Use-Case eine Spezialisierung (im OO-Sinne) eines anderen Use-Cases ist.

Im zweiten Schritt werden alle Aktoren des Projekts erfasst und klassifiziert. Liegt ein Use-Case-Modell vor, so werden die Aktoren hieraus übernommen, alternativ müssen die Aktoren rudimentär ermittelt werden. Aktoren werden analog zu den Use-Cases benannt (Gruppe, Nr., Bezeichnung). Jeder Aktor wird schließlich einer von drei Gruppen bzgl. Komplexität zugeordnet, aus der sich direkt die für die Schätzung relevanten Punkte ergeben:

- einfach, z.B. ein Randsystem, das über eine (einfache) Schnittstelle angebunden ist => 1 Punkt
- mittel, z.B. ein Randsystem, das über eine (komplexeres) Protokoll angebunden ist => 2 Punkte
- komplex, z.B. ein Endnutzer, der über ein Frontend mit dem System agiert => 3 Punkte

Endnutzer mit fachlich unterschiedlichen Rollen werden als separate Aktoren gezählt (wie in der OO-Analyse üblich).

Im dritten Schritt wird der T-Faktor des Projekts bestimmt. Dazu wird das Projekt bezüglich einer ganzen Reihe von Einflussgrößen (Bild 2) bewertet. Alle Einflussgrößen gehen zusammen in den T-Faktor ein. Zu jeder Einflussgröße lässt sich eine Komplexität für das Projekt zwischen 0 und 5 angeben. Der mittlere Wert 3 ist voreingestellt und entspricht einem durchschnittlichen sd&m-Projekt. Zu jeder Einflussgröße sind Beispiele und Indizien angegeben, wann verschiedene Komplexitätswerte vergeben werden sollten. Die unterschiedlichen Einflussgrößen gehen mit unterschiedlichen Gewichten in den T-Faktor ein. Der T-Faktor bewegt sich um 1,0 und korrigiert (als Faktor) die Use-Case-Punktzahl des Projekts bezüglich des Aufwands. Ist das Projekt in jeder T-Hinsicht durchschnittlich (alle

Werte bleiben auf 3,0), bleibt der T-Faktor auch bei 1,0. Bei erhöhtem T-Faktor erhöht sich der Aufwand entsprechend, umgekehrt fällt er bei vermindertem T-Faktor.

Im vierten Schritt wird der M-Faktor des Projekts bestimmt. Dies geschieht vollkommen analog zum T-Faktor: Das Projekt wird bezüglich einer ganzen Reihe von Einflussgrößen bewertet (Bild 3), zu jeder Einflussgröße lässt sich eine Komplexität für das Projekt zwischen 0 und 5 angeben. Der mittlere Wert 3 ist voreingestellt und entspricht einem durchschnittlichen sd&m-Projekt. Die unterschiedlichen Einflußgrößen gehen mit unterschiedlichen Gewichten in den M-Faktor ein. Der M-Faktor bewegt sich um 1,0 und korrigiert (als Faktor) die Use-Case-Punktzahl des Projekts bezüglich des Aufwands.

Im fünften Schritt wird das Schätzergebnis ermittelt. Dazu muss noch der Produktivitätsfaktor PF geeignet gewählt werden. PF gibt die Produktivität für den Projekttyp in Bearbeiterstunden pro Use-Case-Punkt (Bh/UCP) an. Das Arbeitsblatt enthält Hinweise, wie PF zu wählen ist. Mit der Wahl von PF steht das Schätzergebnis als Aufwand in Bearbeiterstunden fest.

7 Diskussion der Ergebnisse und Ausblick

Funktionale Schätzmethoden unterliegen einer kontinuierlichen Weiterentwicklung, die mit der technischen Weiterentwicklung der IT-Projekte einhergeht. Eine gute Übersicht der zeitlichen Entwicklung gibt Lother [13] ausgehend von der Function Point Analysis (FPA) [2] über Full Function Point (FFP) und ISO 19761 bis zu COSMIC FFP 2.1 [1]. Alle diese Methoden lassen sich als datenorientierte Abstraktion verallgemeinern [9] und basieren demzufolge auf einem Lösungsentwurf, der Elemente der Konstruktion (z.B. technischer Datenfluss) vorwegnimmt. Die UCP-Methode ist eine Weiterentwicklung von FPA für moderne Use Case basierte Spezifikationen und beschränkt sich auf Elemente der Spezifikation. Anders als die genannten Vorgänger unterstützt sie damit eine saubere Trennung der Projektphasen Spezifikation, Konstruktion und Realisierung, was zu sehr geringen Aufwänden für die Erstellung von Projekt-Schätzungen führt.

Ein quantitativer Vergleich der Prognosegüte mit anderen Verfahren ist nicht Gegenstand dieser Untersuchung.

In der von uns vorgestellten erweiterten UCP-Methode wurde eine Verfeinerung der Klassifizierung von A-, T- und M-Faktor vorgenommen und damit eine einheitliche und normierte Anwendung der Methodik erreicht. Ferner wurden Indikatoren und Kontraindikatoren für die Methode aufgeführt. Unterschiedliche Entwicklungsprozess-Typen (Vorgehensmodelle) werden durch unterschiedliche Produktivitätsfaktoren berücksichtigt. Durch die Normierung der Komplexitätswerte wurde eine Reproduzierbarkeit des Schätzergebnisses durch unabhängige Schätzer erreicht. Die organisationsübergreifende Anwendbarkeit der Methode für kom-

merzielle Entwicklungsprojekte ist damit deutlich erhöht. Ebenso wurde die Prognosegüte signifikant gesteigert.

Allerdings sind die Wirkzusammenhänge von **Organisation und Vorgehen** auf die Projektaufwände noch nicht vollständig verstanden und erfordern weitere Arbeiten: Die Einflussgrößen im gewählten M-Faktor greifen hier zu kurz und müssen auch nicht-lineare Anteile am A-Aufwand berücksichtigen. Ferner sind Management-Aspekte im Projekt wie z.B. das gewählte Vorgehen, Reifegrad der Organisation und Qualifikation des Teams hinsichtlich der Projektaufgaben („eingeschwungener Zustand“ versus „erstes Release“) zu berücksichtigen. Insbesondere sind die verschiedenen Einflüsse aus dem gewählten Entwicklungsprozess („schlank“ versus „Standard“) genauer zu definieren. Wir erwarten, dass in einer weiteren Verbesserung der UCP-Methode diese Unterscheidung der beiden Entwicklungsprozesse durch verbesserte M-Faktoren abgebildet werden kann und ein einziger Produktivitätsfaktor für eine Gesamtorganisation wie ein Softwarehaus ausreichend ist. Dazu sind neue Einflussgrößen in den M-Faktor aufzunehmen, die den gewählten Entwicklungsprozess-Typ charakterisieren und den Grad der Formalisierung im Projektvorgehen berücksichtigen. Eine zusätzliche Abhängigkeit von der durchschnittlichen oder maximalen Teamgröße bzw. der Anzahl der Hierarchie-Ebenen in der Projektorganisation wird ebenfalls vermutet.

Ebenso ist der T-Faktor aufgrund der Erkenntnisse dieses Praxistests möglicherweise zu überarbeiten, wir gehen jedoch weiterhin von einem Beitrag proportional zum A-Aufwand aus. Die Verbesserung des M-Faktors erachten wir als dringlicher.

8 Zusammenfassung und Ausblick

Im industriellen Umfeld besteht ein hoher Bedarf nach einem Top-Down-Ansatz zur schnellen und sicheren Abschätzung von zu erwartenden Aufwänden für Software-Entwicklungsprojekte. In einem industriellen Praxistest wurde dazu anhand von zehn Entwicklungsprojekten für betriebliche Informationssysteme die Use Case Point Methode überprüft und ein geeignetes Toolkit zur Anwendung dieser Methode entwickelt. Es konnte zunächst nur eine zufrieden stellende Übereinstimmung der geschätzten Projektaufwände mit den tatsächlich eingetretenen Aufwänden nachgewiesen werden. Die Übereinstimmung konnte dann nach Anpassung der Methode auf Basis der Analysen des Praxistestes signifikant verbessert werden.

Trotzdem besteht zukünftig weiterer Forschungsbedarf, um die Wirkzusammenhänge der Projektorganisation (M-Faktor) auf den Projektaufwand besser abzuschätzen und damit die Prognosegüte der Methode weiter zu optimieren. Ferner ist geplant, in einer weiteren Verbesserung der UCP-Methode die Unterscheidung der beiden Entwicklungsprozesse (schlank/Standard) durch verbesserte M-Faktoren vollständig abzubilden.

Literaturhinweise

1. Abran, A. et al.: COSMIC-FFP Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003) Version 2.2, Common Software Measurement International Consortium, 2003. http://www.lrgl.uqam.ca/cosmic-ffp
2. Albrecht, A,; Gaffney, J.: „Software function, source lines of code, and development effort predicition: a software science validation“, IEEE Transactions on Software Engineering 9, pp. 639-648, 1983
3. Anda, B. et al: „Estimating Software development Effort based on Use Cases – Experiences from Industry“, 2001. http://www.idi.ntnu.no/grupper/su/publ/parastoo/icse05-ucpeffort-25may05.pdf
4. Azzouz, S.; Abran, A., “A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC-FFP”, in: Software Measurement European Forum - SMEF 2004, Rome, 2004.
5. Booch, G.: „Object-oriented analysis and design with applications”, 2nd ed., Benjamin/Cummings, Redwood City, 1994
6. Cockburn, A.: “Writing Effective Use Cases”, Addison-Wesley, 2001.
7. DeMarco, T.; Lister, T.: “Peopleware”, Dorset House, New York, 1977.
8. Dröschel, W.; Wiemers, M.: „Das V-Modell 97“, Oldenbourg, 1999.
9. Fetcke, Th.; Abran, A.; Dumke, R.: “Eine verallgemeinerte Repräsentation für ausgewählte Functional Size Measurement Methoden” in: Current Trends in Software Measurement, R. Dumke and D. Rombach, Eds. Deutscher Universitäts Verlag, 2002, pp. 50-75.
10. Schneider, G.: “Applying Use Cases: A practical guide”, 2nd ed., Addison-Wesley Object Technology Series, 2005.
11. Karner, K: „Metrics for Objectory“. Diploma thesis, University of Linköping, Sweden, No. LiTHIDA-Ex-9344:21, December 1993.
12. Kruchten, P.: “The Rational Unified Process: An Introduction”, 3rd ed., Addison-Wesley, 2003.
13. Lother, M.; Dumke, R.: “Points Metrics - Comparison and Analysis” in: Dumke et al. (Eds.): Current Trends in Software Measurement - Proceedings of the 11th IWSM, Montréal, Shaker Verlag, Aachen, 2001, pp. 228-267.
14. Siedersleben, J.: “Softwaretechnik – Praxiswissen für Software- Ingenieure” 2. überarbeitete und aktualisierte Auflage, Hanser Verlag, 2003.
15. Siedersleben, J.: „Moderne Software-Architektur“, dpunkt.verlag, 2004.
16. Smith, J.: „The Estimation of Effort Based on Use Cases“, Rational Software, Cupertino, CA.TP-171, October 1999. http://whitepapers.zdnet.co.uk/0,39025945,60071904p-39000629q,00.htm
17. OMG, “UML ™ Resource Page“, http://www.uml.com, Aktualisierungsdatum: 04.01.2006.