UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik – Informatik – Mathematik
Institut für Informatik
Fachgebiet Informationssysteme
Warburger Straße 100
33098 Paderborn

Use Case Points 3.0

Implementierung einer Use Case bezogenen Schätzmethode für das Software-Engineering betrieblicher Informationssysteme

Schriftliche Arbeit
zur Erlangung des akademischen Grades
„Doktor der Naturwissenschaften“ (Dr. rer. nat.)

vorgelegt von

Dipl.-Phys. Stephan Frohnhoff
Frankfurter Str. 43 c
63225 Langen

Paderborn, im Februar 2009

The fundamental principle of science, the definition almost, is this: the sole test of the validity of any idea is experiment. (Richard P. Feynman)

Danksagung

Die hier vorliegende Arbeit ist erst durch die intensive Verzahnung mit meinem Arbeitgeber möglich geworden: Die enge Zusammenarbeit mit Mitarbeitern und der Zugriff auf vielfältige Projektdaten und Erfahrungen war der Schlüssel zum vorliegenden Ergebnis. Daher gilt mein Dank dem Unternehmen Capgemini sd&m AG, seinen Mitarbeitern und zuvorderst Dr. Dirk Taubner und Dr. Uwe Dumslaff, die als meine Vorgesetzten diese Arbeit unterstützt haben.

Die wissenschaftliche Betreuung dieser Dissertation hat Prof. Dr. Gregor Engels übernommen, dem ich an dieser Stelle sehr danke. Der intensive, detailgenaue und umfangreiche Dialog insbesondere in der Schlussphase war enorm bereichernd und hat für die notwendige methodische Verankerung in der Welt der Modell-Transformationen gesorgt.

Mein Dank gilt ferner Prof. Dr. Reiner Dumke, dessen Kenntnis im Fachgebiet der Metriken und des Functional Size Measurement (FSM) in vielen wertvollen Diskussionen in den letzten Jahren eingeflossen ist. Ich danke ihm auch für die Übernahme der Aufgabe des Zweitgutachters.

Diese Arbeit wäre nicht ohne die intensive und sehr konstruktive Zusammenarbeit mit Capgemini sd&m Research möglich gewesen. Mein Dank gilt hier insbesondere Ilona Lemmer als Projektleiterin und Thomas Engeroff. Aus dem Kollegen-Kreis möchte ich insbesondere auch Volker Jung, Karsten Kehler und Markus Voß namentlich nennen, die durch viele fruchtbare Diskussionen und fachliche Hinweise zum Gelingen dieser Arbeit beigetragen haben.

Diese Dissertation hat wesentlich von der erfolgreichen Kooperation mit mehreren Universitäten und Hochschulen in Deutschland profitiert. Hierfür bedanke ich mich bei den Lehrstuhlverantwortlichen und Studierenden.

Nicht zuletzt gilt mein herzlicher Dank meinen Eltern, die mir eine akademische Ausbildung zum Diplom-Physiker ermöglicht haben. Zahlreiche Anleihen und Erkenntnisse aus der experimentellen Physik konnte ich so in das Gebiet des Software-Engineerings übertragen.

Abschließend gilt mein innigster Dank meiner Frau Birgit Frohnhoff, die meine Doppelbelastung aus Beruf und Promotion mit getragen, mir jederzeit den nötigen Rückhalt gegeben und die Qualität durch kritische Fragen aus Anwendersicht gesichert hat.

Abstract

Ever increasing industrialisation in software engineering requires improved methods for estimating business application systems at the time of request for proposal. This leads to requirements which cannot entirely be met by the currently known estimating methods. In praxis use case based specification forms are widely spread. Therefore, it would be desirable to be able to directly derive functional size measurements from use cases. An analysis of known estimating methods shows that the Use Case Points (UCP) method fulfils these requirements well, but it reveals some conceptual weaknesses in comparison to the widely spread methods like function points or COCOMO II.

In this thesis, the UCP method has been fundamentally revised and conceptual weaknesses have been removed. Thereby several ideas and findings from experimental physics and stochastics were applied to the field of software engineering during this process.

Firstly, a new conceptual view on the UCP method was developed. With regards to the functional requirements of a specification, a model-based approach has been defined to identify Use Cases and to transform different specification forms into a newly developed UCP language. Herewith, single instances of the UCP language can be mapped to a point metric. This new UCP model has been empirically validated by a field study with a total number of more than 200 estimations.

For capturing non-functional requirements of a specification, a cost driver model has been developed based on COCOMO II and on further models in literature and in industrial praxis. The cost driver model has been empirically validated by an expert survey with 25 participants.

The final solution, called *UCP 3.0* has been proven by 19 commercial software development projects with a total effort of more than 275 man years. For that purpose estimations were performed during an early project phase on the basis of a rough specification and were then compared with the actual project efforts after project close. UCP 3.0 in combination with a counting practice manual, results in a standardised estimating method with high reproducibility in industrial praxis and significantly improved estimating accuracy. In addition, an estimating tool and a best-practice example of its application complete the presented solution to build a sustainable approach for industrial praxis.

The method UCP 3.0 has already proven itself in the software house Capgemini sd&m AG.

Inhaltsverzeichnis

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

1 Einleitung

Analysten prognostizieren ein kontinuierliches jährliches Wachstum (CAGR[1]) von fast 6% im Bereich der Entwicklung von Individualsoftware 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 Schätzmethoden bislang nur teilweise alle Komplexitätsaspekte erfassen.

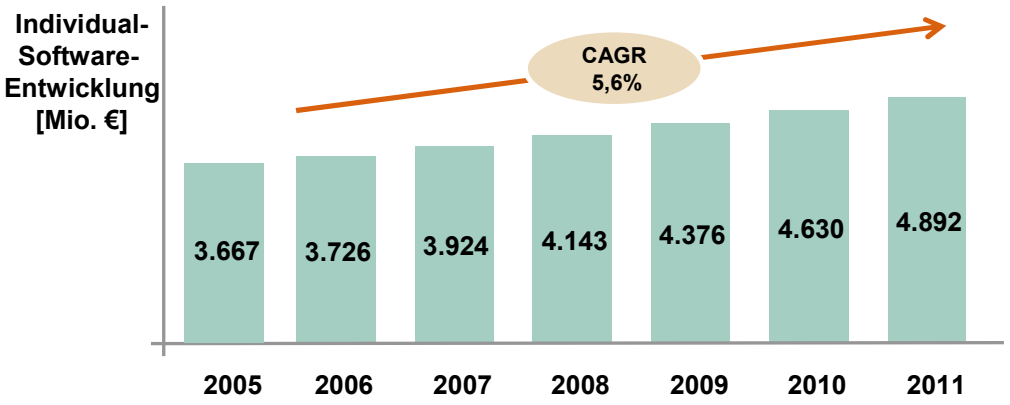


Abbildung 1: Analyse der Marktentwicklung für Individualsoftware-Entwicklung im deutschsprachigen Europa [PAC07]. Die durchschnittliche jährliche Wachstumsrate (CAGR) mittelt über die Jahre 2006 bis 2011.

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.

[1] Compound Annual Growth Rate

Diese Aussage wird untermauert durch die Auswertung von 44 Projekten (500 bis 70.000 Tage Aufwand) [Fro08], 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* (Erläuterung siehe nachfolgend) im Mittel um 40 % unterschätzt wurden

Formal lässt sich die Aufwandsschätzung laut Dumke [Dum03] als Vorhersage von Software- und Projekt-Merkmalen für den Zeitpunkt t_i mit $t_i > t_{aktuell}$ beschreiben. Formal gilt für die Schätzung des Projektaufwandes *PE (Project Effort)*:

$PE := f\ (A, B)$	*Formel 1*

mit $A := \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ als Menge der in die Schätzung eingehenden gegebenen bzw. *gemessenen* Größen und $B := \{\beta_1, \beta_2, \ldots, \beta_m\}$ als Menge der eingehenden *geschätzten* Größen. Die verschiedenen Niveaustufen einer Schätzung sind vereinfachend in Abbildung 2 dargestellt.

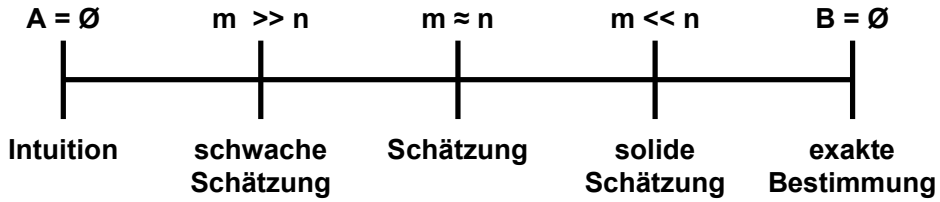


Abbildung 2: Stufen der Schätzung des Projektaufwandes

„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 Projekt-Versionen bestehen. Für ein neues Umfeld gilt folglich $m >> n$.

Aus diesen einleitenden Gedanken können wir bereits zwei Anforderungen an eine Schätzmethode für den geschilderten Kontext festhalten:

A 1:	Die Schätzung muss zu einem sehr frühen Projektzeitpunkt auf Basis einer Grobspezifikation durchführbar sein.
A 2:	Die Schätzmethode muss auch in neuem Umfeld funktionieren.

1.1 Auswirkung der Industrialisierung auf die Schätzmethoden

Gegen Ende der 90er Jahre hat eine deutliche Industrialisierung des Software-Engineerings eingesetzt [Tau05]. Eine Konsequenz ist der gestiegene Wettbewerb hin zu einem Verdrängungsmarkt. Wer als Individualsoftware-Hersteller diese Industrialisierung 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 bei Capgemini sd&m zeigt eine Schätzgenauigkeit von etwa ±30 %, wobei Projektvorhaben in einem neuen Umfeld eine deutlich höhere Streuung von im Mittel ±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. Abbildung 3 illustriert diesen Sachverhalt schematisch: Für ein Individualsoftware-Haus betrachten wir alle Aufwandsschätzungen verschiedener Projekt-Angebote an Auftraggeber über die Zeit und normieren die Aufwände der Einfachheit halber auf gleichen Marktpreis. Im Falle a) bei wenig Wettbewerb werden diese Aufwände um den Marktpreis oszillieren. Über mehrere Projekte mittelt sich diese Schätzunsicherheit heraus, d.h. mal wird ein Projekt mit Budgetverlust, mal mit Budgetgewinn abgeschlossen.

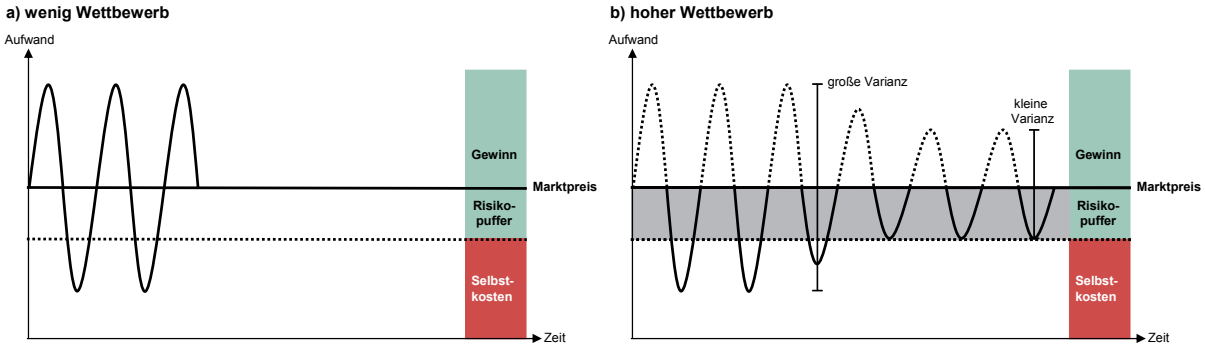


Abbildung 3: Schematische Darstellung der Varianz von Projektschätzungen und ihre Bedeutung für die Profitabilität von Unternehmen

Jedes seriöse Software-Haus kalkuliert dafür in seine Projekte einen Risikopuffer ein. Ein deutliches Zeichen einer fortgeschrittenen Industrialisierung des Software-Entwicklungs-Marktes ist der gestiegene Wettbewerb. Anfang der 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 und damit der kalkulierte Aufwand zur Angebotsabgabe 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 Aufwände dieses Mal etwas geringer geschätzt hat. In der Abbildung 3 b) ist dieser überschätzte Teil nun gestrichelt eingezeichnet, da er in der der Regel nicht beauftragt wird. Daher ist der Auftrag-

nehmer gezwungen, seine Schätzvarianz deutlich zu verringern, um im Mittel über viele Projekte noch wirtschaftlich über den Selbstkosten agieren zu können. 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 Schätzvarianz in 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 die zu schätzenden Aufwände. 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, die bei großen Entwicklungsvorhaben an ihre Grenzen stößt. Bedingt durch die lange Laufzeit von großen Software-Projekten dauert es für einen einzelnen Schätzer zu lange, bis er mehrfach aus eigener Erfahrung die Projektschätzung und das zugehörige Umsetzungsprojekt selbst durchlaufen hat. Daher muss z.B. mit Hilfe von Schätzdatenbanken oder empirischen begründeten mathematischen Modellen das Wissen vieler Software-Ingenieure gesammelt und vergleichbar gemacht werden.

Dies führt unmittelbar zur Fragestellung, wie unterschiedliche Software-Entwicklungsprojekte hinsichtlich ihres zu erwartenden Umfanges bzw. Aufwandes miteinander verglichen werden können.

1.2 Problemstellung in der Aufwandsschätzung

Aufgrund der zuvor beschriebenen wirtschaftlichen Rahmenbedingungen müssen Schätz-Modelle und -Methoden verwendet werden, welche den Aufwand für die Durchführung einer Aufwandsschätzung klein halten, da der Individual-Software Anbieter mit steigendem Wettbewerb nur noch auf wenige Angebote einen Zuschlag erhält und die Kosten für die Aufwandsschätzung selbst zu finanzieren hat.

Weit verbreitet und auf die Anforderungen *A 1* und *A 2* (Seite 2) zugeschnitten ist die sogenannte Expertenschätzung, in welcher sehr erfahrene Software-Ingenieure (=Experten) auf Basis einer zunächst erstellten Stückliste[2] den Gesamtaufwand durch Schätzung aller Einzel-Aktivitäten ermitteln (Bottom-Up Methode). Für ein Software-Projekt z.B. im Umfang von 10 Bearbeiterjahren (BJ) liegt der Aufwand für die Erstellung einer solchen Schätzung im Bereich von einigen Bearbeiterwochen. Die Schätzgenauigkeit lässt sich sicherlich dadurch erhöhen, dass unabhängig voneinander mehrere Experten eine solche Schätzung durchführen und die Ergebnisse verglichen werden. Gängige Methoden sind die Delphi-Methode oder Schätzklausuren [Sne05, Dum03, Boe86] und sind in Abbildung 4 dargestellt. Der Vorteil dieser Methoden ist, dass sie prinzipiell für jedes Softwareerstellungsprojekt anwendbar sind. Eine Qualitätsverbesserung der Schätzung wird durch die Steigerung der Zahl der Experten zur Durchführung der Schätzung erreicht. Der

[2] Der Begriff der *Stückliste* stammt aus der Fertigungsindustrie und hat im Zuge der Industrialisierung Einzug in die industrielle Informatik gehalten. Gemeint ist eine Liste aller zu schätzenden „Stücke“ = Artefakte und Aktivitäten eines Projektes, welche in Summe das gesamte Projektvorhaben („Werk“) ausmachen.

Aufwand steigt dabei linear mit der Zahl der Experten, eine Plausibilisierung durch weitere Experten ist also entsprechend aufwändig. Wünschenswert wäre also eine Methode, welche zusätzlich folgende Anforderung erfüllt:

A 3:	Möglichst schlanke Schätzmethode mit einem deutlich geringeren Durchführungsaufwand als bei einer Bottom-Up Schätzung.

Einzelschätzung	**Delphi-Methode**	**Schätzklausur**
▪ Ein einziger Experte legt die Schätzwerte für einen bestimmten Aufwandsposten fest ▪ Genauigkeit hängt wesentlich von der Erfahrung dieses Experten ab ▪ Hohe Verantwortung für eine Person ▪ Einseitige Beurteilung von Aufwänden und Unsicherheiten	▪ Mehrere Experten führen ihre Schätzung anonym und ohne Abstimmung untereinander durch ▪ Zusammenführung der Schätzung durch den Projektleiter ggf. in Iterationen bei (großen) Abweichungen ▪ Korrektur-Möglichkeiten der Schätzzahl ohne „Gesichtsverlust“ ▪ Keine Gruppenzwänge	▪ Mehrere Experten schätzen „online“ im Rahmen eines gemeinsamen Workshops ▪ Sofortige Zusammenführung von Aufwänden und Plausibilisierung ▪ Inhaltliche Diskussionen bei großen Abweichungen ▪ Gruppe einigt sich auf Aufwand pro Schätzposten ▪ Risiken werden bewusst ▪ Gleichmäßiger Informationsstand im Team
Pragmatisch aber teilweise ungenau	Verlässlich aber sehr zeitaufwändig	Besser als Mittelwerte von Einzelschätzungen aber etwas zeitaufwändiger

Abbildung 4: Gegenüberstellung gebräuchlicher Expertenschätzmethoden

Gemäß Formel 1 kann die Schätzung verbessert werden, wenn die Anzahl der bestimmten bzw. gemessenen Faktoren erhöht wird (d.h. m ist sehr klein). Dies ist dann der Fall, wenn die Schätzung zum Angebotszeitpunkt und der tatsächlich benötigte Aufwand nach Projektabschluss sorgfältig verglichen werden und diese Erfahrung in Form von Metriken und Kennzahlen in Schätzdatenbanken und Algorithmen zur Umfangsbestimmung Eingang finden. Daraus sind die sogenannten Top-Down Methoden entwickelt worden, welche zur Plausibilisierung von Bottom-Up Schätzungen („Experten-Schätzungen“) verwendet werden oder in bekanntem Umfeld sogar an deren Stelle treten können.

Eine Übersicht und zusammenfassende Bewertung der gebräuchlichen Top-Down und Bottom-Up Schätzmethoden zeigt Abbildung 5. Details zu den genannten Methoden finden sich z.B. in [Sne05, Dum03, Boe86], Function Points wird in Abschnitt 2.1, Use Case Points in Abschnitt 2.3 und COCOMO in Abschnitt 2.5 erläutert.

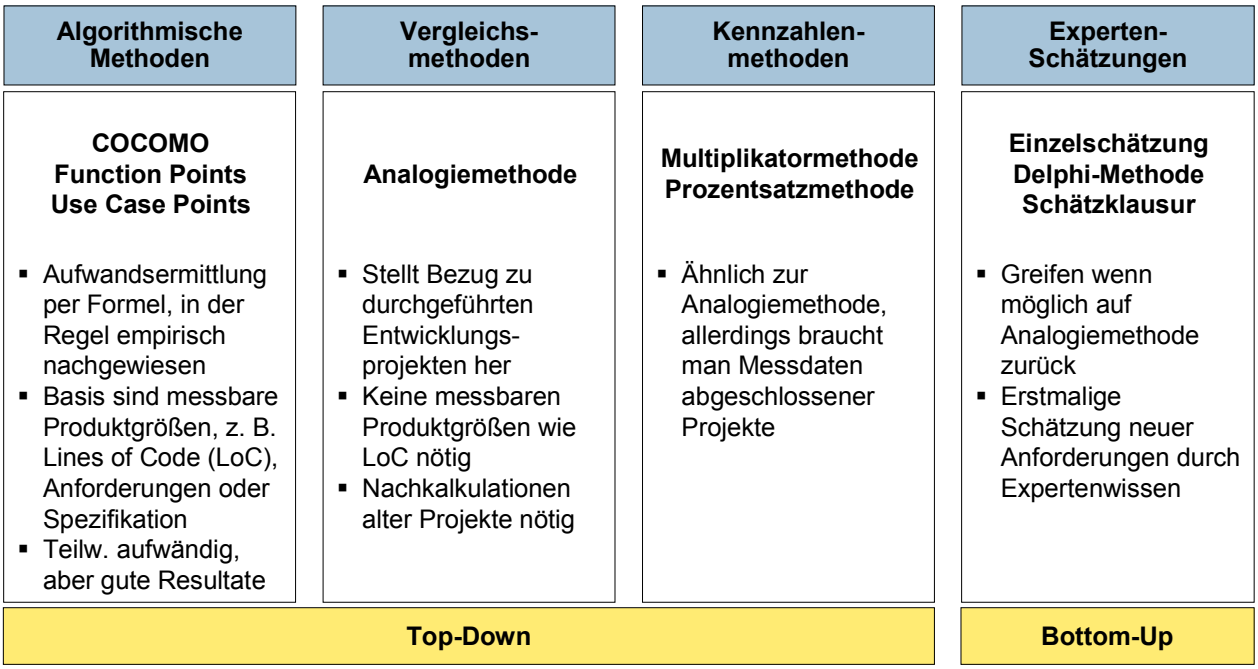


Abbildung 5: Übersicht gebräuchlicher Top-Down und Bottom-Up Schätzmethoden

Anders als bei der Bottom-Up Methode wird bei der Top-Down Methode eine gesamthafte Schätzung des Projektaufwandes mit Hilfe von mathematischen Algorithmen auf Basis der (funktionalen) Anforderungen erzielt, wobei Metriken aufgrund von Größenvergleichen der Anforderungen einbezogen werden. Dies setzt voraus, dass die Größe der fachlichen Anforderungen zum Angebotszeitpunkt für ein Individualsoftware-Haus ermittelbar ist. Sollte dies möglich sein, wären Top-Down Methoden ein vielversprechendes Verfahren zur Plausibilisierung von Expertenschätzungen. Damit könnte die Praxiserfahrung vieler Projekte in Form von Metriken zur Qualitätsverbesserung in die Schätzmethode eingebracht werden, indem die universelle Bottom-Up Schätzung durch eine Top-Down Schätzung validiert wird. Wir halten bereits jetzt die Anforderung fest:

A 4:	Praxiserprobte Methode, die mit möglichst wenig geschätzten Größen gemäß Formel 1 auskommt.

Es bleibt die Frage, wie die oben beschriebene Größe einer fachlichen Anforderung gemessen werden kann. In Top-Down Methoden wird diese Größe in der Regel durch „Punkte“ (Points) einheitenlos ausgedrückt und basiert auf der Annahme der Vergleichbarkeit des Projektaufwands bei gleichem funktionalem Umfang.

In der Literatur finden sich hierzu zahlreiche Ansätze. Seit der ersten Einführung der Function Point Analysis (FPA) Methode durch Albrecht in 1979 [Alb79] wurde das Functional Size Measurement (FSM) verbessert und zahlreiche Variationen und Erweiterungen entwickelt, um auch unterschiedlichsten Anwendungsfällen gerecht zu werden [LD01, BD08].

Der Vorteil dieser Methoden ist die normierte Vorgehensweise. Allerdings berücksichtigen diese Methoden primär nur den funktionalen Umfang, nichtfunktionale Anforderungen oder der Ein-

fluss durch Projektvorgehen und –Organisation werden nicht abgedeckt und bewusst ausgeschlossen. Der ISO/IEC 14143-1 Standard aus dem Jahr 2007 reduziert in seiner Darstellung die Größenbestimmung (FSM) klar auf die sogenannten *Functional User Requirements* (FUR) und grenzt die *Non-Functional User Requirements* (NFUR) aus[3]. In der Konsequenz wurden die Korrekturfaktoren für nichtfunktionale Anforderungen schrittweise aus den FSM-Methoden herausgenommen. Ebenso erfordern diese Methoden hoch qualifizierte Experten mit fundiertem Software-Engineering Wissen und sind nicht „ad-hoc“ schnell durchführbar.

Es stellt sich nun die Frage, welche der genannten Methoden am besten für die Praxis des industriellen Software-Engineerings von betrieblichen Informationssystemen geeignet sind. Dazu werden im nächsten Kapitel zunächst die Methoden analysiert und bewertet. Dann wird die Forschungsfrage verfeinert.

[3] Die Version aus dem Jahr 1998 teilt den Non-Functional Teil noch in Anforderungen nach Qualität und Technik.

2 Grundlagen

In Kapitel 1 wurde bereits herausgearbeitet, dass Top-Down Methoden einen vielversprechenden Ansatz darstellen, um eine Bottom-Up Methode zu ergänzen und damit die Schätzqualität zu verbessern. Dabei wurde das Problemfeld des Functional Size Measurements (FSM) angesprochen. Im Folgenden wird zunächst die Function Point Methode als wichtigster Vertreter des FSM vorgestellt. Sie hat die größte Verbreitung und gilt als „Vater" aller später entwickelten FSM-Methoden [AR94]. Dann werden wir wichtige FSM-Methoden im Überblick vorstellen (Abschnitt 2.2) und in den Kontext der Spezifikation von Anforderungen stellen (Abschnitt.2.3). Es wird sich zeigen, dass die Use Case Points Methode (Abschnitt 2.4) am besten hinsichtlich der gestellten Anforderungen geeignet ist, allerdings auch erhebliche Schwächen aufweist. Einige der Schwächen wurden in COCOMO gelöst, daher werden wir diesen Ansatz zunächst näher analysieren (Abschnitt 2.5). Es folgt ein Abgleich der bekannten Schätzmethoden mit der Problemstellung (Abschnitt 2.6), woraus sich dann eine verfeinerte Aufgabenstellung ableitet (Abschnitt 2.7).

2.1 Function Point Methode

Der Ursprung des Functional Size Measurements (FSM) wurde bei IBM Ende der 70er Jahre gelegt. Mit der Veröffentlichung der Function Point (FP) Methode im Jahr 1979 durch Allen Albrecht [Alb79] wurde eine Methode begründet, deren Grundgedanke bis heute Bestand hat: Die Messung der funktionalen Größe eines Software-Systems aus Sicht des Anwenders durch eine einheitenlose Punktezahl, im Folgenden auch *Points* genannt. 1984 bildete sich die International Function Point User Group (IFPUG), welche Regeln verbindlich festlegte, Standards aufsetzte und die Methode sowie deren Weiterentwicklung förderte. Die IFPUG brachte ab 1986 regelmäßig Weiterentwicklungen der Methode heraus. Bis heute hat sich die Bezeichnung *Function Point Analysis* (FPA) durchgesetzt, man spricht auch von IFPUG FPA.

Eine detaillierte Darstellung der Methode wurde im Rahmen dieser Dissertation unter [Bad08] festgehalten, wir beschränken uns hier auf eine Zusammenfassung der Version der IFPUG FPA 4.1, die als internationaler Standard ISO 20926:2003 anerkannt wurde [ISO-I03]. Im *Function Point Counting Practice Manual* (*CPM*) hat die IFPUG Erkenntnisse, Regeln, Bestimmungen und Definitionen der Methode zusammengefasst [IFP03].

FPA berücksichtigt zur Aufwandsbestimmung den fachlich-funktionalen Umfang, gemessen in Points, sowie weitere qualitative Merkmale. Letztere werden im *Value Adjustment Factor* (*VAF*) abgebildet. Das Produkt aus *VAF* und den *unadjusted Function Points* (*uFP*) ergibt dann die *adjusted Function Points* (*aFP*).

$aFP = uFP \cdot VAF$	*Formel 2*

Die Bestimmung der unadjusted Function Points wird in Abschnitt 2.1.1 beschrieben, die Berechnung des Value Adjustment Factors in Abschnitt 2.1.2.

2.1.1 Unadjusted Function Points

Um die unadjusted Function Points (uFP) zu zählen, werden die Funktionen in solche Elementarprozesse zerlegt, die sich aus Anwendersicht nicht weiter zerlegen lassen. Hierfür werden die Funktionen der Anwendung in zwei Gruppen eingeteilt: Zu der Gruppe der Elementarprozesse zählen Eingabe von Daten, Abfragen sowie Ausgabe von Daten; in der Gruppe der Datenbestände wird unterschieden zwischen Anwendungsdaten und Referenzdaten.

Um die Elementarprozesse nach ihrer Komplexität bewerten zu können, werden je Funktionstyp neben den genutzten Datenelementen auch die referenzierten Datenbestände gezählt. Ein Datenelement ist ein vom Anwender erkennbares Datenfeld oder eine notwendige Schaltfläche, die bedient werden muss, um auf alle Datenfelder zuzugreifen. Die beim Ablauf der Elementarprozesse genutzten Datenbestände werden als referenzierte Datenbestände gezählt. Die unadjusted Function Points werden jeweils gemäß Tabelle 1 definiert, die Punkte (Points) können in der Tabelle in den hell hinterlegten Feldern abhängig von der Anzahl der Datenelemente (Spalten) und der referenzierten Datenbestände (Zeilen) abgelesen werden. Die unterschiedliche Gewichtung in Tabelle 1 bringt bereits Empirie in die uFP ein.

		Anzahl Datenelemente		
Eingaben		< 5	5 – 15	> 15
referenzierte Datenbestände	< 2	3	3	4
	2	3	4	6
	> 2	6	6	6
Ausgabedaten		< 6	6 – 19	> 19
referenzierte Datenbestände	< 2	4	4	5
	2-3	4	5	7
	> 3	5	7	7
Abfragen		< 6	6 – 19	> 19
referenzierte Datenbestände	< 2	3	3	4
	2-3	3	4	6
	> 3	4	6	6

Tabelle 1: Definition der Anzahl Points für Eingabedaten, Ausgabedaten und Abfragen

Eingabedaten sind Daten, die in das System eingegeben und gespeichert werden. Die Eingabe kann durch unterschiedliche Peripheriegeräte erfolgen. Eingabedaten sind mehrfach zu zählen, wenn sich bei mehrfacher Eingabe Datenfelder unterscheiden oder die Eingabedaten in unterschiedlichen Datenbeständen gespeichert werden.

Abfragen lesen Daten aus den Datenbeständen aus und geben diese ohne weitere Bearbeitung aus. Wie bei den Eingabedaten sind auch Abfragen mehrfach zu zählen, wenn diese sich von anderen Abfragen unterscheiden oder sich auf andere Datenbestände beziehen.

Ausgabedaten sind Daten, die vom System ausgegeben werden, hierbei aber zusätzlich abgeleitet oder berechnet werden müssen. Auszugebende Daten zählen auch in die Rubrik Ausgabedaten,

sobald während der Ausgabe Anwendungsdaten gepflegt werden oder sich nach der Ausgabe das Systemverhalten ändert.

Zur Bestimmung der Komplexität der Datenbestände (d.h. Anwendungsdaten und Referenzdaten) wird analog zu den Elementarprozessen die Größe der Datenelemente ermittelt. Hierfür wird neben der Anzahl der Datenelemente auch die Anzahl der möglichen Untergruppen der Datenelemente herangezogen. Die entsprechende Zuordnung zu Function Points definiert Tabelle 2.

		Anzahl Datenelemente		
Anwendungsdaten		< 19	20 - 50	> 50
Anzahl Untergruppen	< 2	7	7	10
	2 - 5	7	10	15
	> 5	10	15	15
Referenzdaten		< 19	20 - 50	> 50
Anzahl Untergruppen	< 2	5	5	7
	2 - 5	5	7	10
	> 5	7	10	10

Tabelle 2: Definition der Anzahl Points für Anwendungsdaten und Referenzdaten

Anwendungsdaten werden durch die Anwendung gespeichert und verwaltet. Nach dem *CPM* müssen hierfür zwei Bedingungen erfüllt sein: Die Daten sind in fachlicher Hinsicht für den Anwender sichtbar und werden durch einen der drei Elementarprozesse genutzt.

Referenzdaten werden im Unterschied zu Anwendungsdaten nur lesend genutzt und sind außerhalb der Anwendung gespeichert, werden also nicht mit der zu schätzenden Anwendung gepflegt.

Die ungewichteten Function Points *uFP* bestimmen sich aus der Summe der Points aus Eingabedaten (*ED*), Abfragen (*AB*), Ausgabedaten (*AU*), Anwendungsdaten (*AD*) und Referenzdaten (*RD*) zu:

	$uFP = ED + AB + AU + AD + RD$	*Formel 3*

2.1.2 Value Adjustment Factor

Der Value Adjustment Factor wird bestimmt zu:

	$VAF = 0{,}65 + 0{,}01 \cdot \sum_{i=1}^{14} C_i$	*Formel 4*

C_i steht für einen Punkt-Wert der *General System Characteristics* (kurz *GSC*) gemäß Tabelle 3 und kann ganzzahlige Werte zwischen 0 (kein Einfluss) und 5 (hoher Einfluss) annehmen.

C_i	Name des GSC (Englisch) / Name des GSC (Deutsch) / Beschreibung
C_1	**Data Communications** / Datenkommunikation Die Daten- und Kontrollinformationen, die im Anwendungssystem genutzt werden, werden über Kommunikationseinrichtungen gesendet und empfangen.
C_2	**Distributed Data Processing** / Verteilte Datenverarbeitung Verteilte Daten- oder Verarbeitungsfunktionen charakterisieren das Anwendungssystem.
C_3	**Performance** / Leistungsanforderung Performance-Ziele bezüglich Antwortzeiten oder Durchsatz, die vom Anwender entweder vorgegeben werden, beeinflussen Entwurf, Entwicklung und Installation des Anwendungssystems.
C_4	**Heavily Used Configuration** / Ressourcennutzung Das Anwendungssystem soll auf einer stark belasteten Rechnerkonfiguration laufen. Dieser Tatbestand beeinflusst die Entwurfsüberlegungen für das Anwendungssystem.
C_5	**Transaction Rate** / Transaktionsrate Die Transaktionsrate ist hoch und beeinflusst Entwurf, Entwicklung, Installation und Betrieb des Anwendungssystems.
C_6	**Online User Interface** / Dateneingabe Das Anwendungssystem sieht eine Online-Dateneingabe und Kontrollfunktion vor.
C_7	**End User Efficiency** / Bedienerfreundlichkeit Die Online-Funktion des Anwendungssystems soll betont unter Endanwender-Effizienzaspekten entworfen werden.
C_8	**Online Update** / Onlineverarbeitung Das Anwendungssystem sieht ein Online-Update für die internen logischen Datenbestände vor.
C_9	**Complex Processing** / Komplexe Prozesslogik Das Anwendungssystem wird durch komplexe Verarbeitungsprozeduren gekennzeichnet.
C_{10}	**Reusability** / Wiederverwendbarkeit Die Anwendungs-Software wird in anderen Anwendungssystemen wiederverwendet.
C_{11}	**Installation Ease** / Migration- und Installationshilfen Es bestehen hohe Anforderungen, um die Daten des Anwendungssystems leicht zu migrieren und das Anwendungssystem leicht installieren zu können.
C_{12}	**Operational Ease** / Betriebshilfen Das Anwendungssystem soll leicht betrieben und bedient werden können.
C_{13}	**Multiple Sites** / Mehrfachinstallationen Entwurf, Entwicklung und Entwicklungsunterstützung sind darauf auszurichten, dass das Anwendungssystem bei vielen Anwendern in vielen Organisationen benutzt werden kann.
C_{14}	**Facilitate Change** / Anpassbarkeit Entwurf, Entwicklung und Entwicklungsunterstützung sind darauf auszurichten, dass das Anwendungssystem auch in Zukunft flexibel verändert werden kann.

Tabelle 3: General System Characteristics der FP-Methode [Alb79] und [PB05]

Rechnerisch kann der *VAF* somit zwischen 0,65 (alle *GSC* sind mit 0 bewertet) und 1,35 (alle *GSC* sind mit 5 bewertet) variieren. Jeder *GSC* geht hierbei mit dem gleichen Gewicht in die Berechnung ein.

2.1.3 Bestimmung des Aufwands

Die adjusted Function Points (aFP) gemäß Formel 2 lassen noch keine direkte Aussage über den zu erwartenden Aufwand zu. In der Praxis werden mehrere Methoden angewandt, um die errechneten Function Points in Aufwand umzurechnen. Das *CPM* beschreibt diese Methoden jedoch nicht.

Aufwandsbestimmung mit gesammelten Vergleichswerten

Für die Umrechnung werden die adjusted Function Points mit Vergleichswerten aus bereits abgeschlossenen Projekten verglichen, im Optimalfall mit Vergleichswerten von derselben Organisation aus abgeschlossen Projekten. Nach Abschluss eines Projektes wird der Ist-Aufwand berechnet und den Vergleichswerten hinzugefügt, ältere Vergleichswerte verblassen mit der Zeit oder werden gelöscht.

Die Daten abgeschlossener Projekte werden zum Beispiel als Wertepaare (Function Points/Aufwand in Personenmonaten) angegeben oder in [Bal00] mit Formel 5 berechnet.

$PE := 0{,}015216 \cdot FP^{1{,}29}$	*Formel 5*

Eine Übersicht zeigt die nachfolgende Tabelle 4 mit PE-Werten aus unterschiedlichen Quellen:

Function Points:	**100**	**250**	**500**	**750**	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**	**4000**	**4500**
Balzert [Bal00]:	5,8	18,9	46,1	77,8	113	190	276	368	465	568	674	785
Firma IBM [BF04]:	5,6	18,6	46,1	78,4	114	194	284	380	483	591	704	821
Firma VW [BF04]:	N/a	19,3	27,1	105	157	319	-	-	-	-	-	-

Tabelle 4: Zuordnung Function Points zu Projektaufwand (PE) in Bearbeitermonaten

Die Zuverlässigkeit der Methode wird erhöht, wenn die Vergleichswerte aus einer vergleichbaren Anwendungsdomäne stammen und das Projekt unter vergleichbaren Rahmenbedingungen durchgeführt wurde.

Aufwandsbestimmung mit COCOMO

Alternativ zur Aufwandsbestimmung mit gesammelten Vergleichswerten wird gerne zur Aufwandsbestimmung das COCOMO-Modell von Boehm [Boe+00] herangezogen. COCOMO wird in Abschnitt 2.5 detailliert beschrieben. Es ist ein Modell, welches den Einfluss der nichtfunktionalen Anforderungen und der Projektorganisation berücksichtigt und zur Größenbestimmung auf *Quellcodezeilen* (Source Lines of Code, kurz *SLOC*) aufsetzt. Die Umrechnung von SLOC in Function Points kann mit Hilfe von Formel 6 erfolgen.

$SLOC = GF * uFP$	*Formel 6*

uFP steht hierbei für den unadjusted Function Point Wert.

GF ist der *Gearingfaktor* und hängt in erster Linie von der verwendeten Programmiersprache ab. Tabelle 5 nennt den Gearingfaktor in SLOC pro FP Wert nach den Erfahrungen von [Boe+00] und [QSM08].

Programmiersprache:	Assembler	C	Cobol	C++	Java
GF [Boe++00]:	320	128	91	55	53
GF [QSM08]:	157	104	77	53	59

Tabelle 5: Tabelle häufiger Gearingfaktoren

Die Werte sind jedoch mit Vorsicht zu verwenden, da weitere Faktoren wie Programmierstil, Bibliotheken und Eignung der Programmiersprache eine wichtige Rolle spielen. So wurden laut [Bal00] und [QSM08] für Javaprojekte eine Streuung zwischen ca. 15 und 100 beobachtet.

2.1.4 Zusammenfassung

Die Zählung der Function Points ist keine triviale Angelegenheit und erfordert technisches Wissen und viel Übung.

Bei der *Aufwandsbestimmung mit gesammelten Vergleichswerten* wird nur der fachlich-funktionale Aufwand berücksichtigt. Die gesammelten Vergleichswerte lassen sich damit nicht einfach auf andere Gegebenheiten übertragen, da sich diese zum Teil deutlich unterscheiden. So weichen z.B. die Werte in Tabelle 4 der Firma VW erheblich von denen der Firma IBM ab. Um eine eigene, firmenspezifische Justierung zu ermöglichen, ist eine größere Anzahl von Projekten notwendig.

Wird der Aufwand mithilfe von COCOMO errechnet, ist im Vorfeld zu überlegen, ob als Eingangsgröße die adjusted oder besser die unadjusted Function Points herangezogen werden, da z.B. COCOMO eigene Metriken liefert, welche die General System Characteristics gemäß Tabelle 3 ersetzen.

2.2 Bewertung der aktuellen standardisierten FSM-Methoden

Die in 2.1 vorgestellte FPA-Methode hat bis heute zahlreiche Erweiterungen und Verbesserungen erfahren. Viele Variationen wurden entwickelt, um auch unterschiedlichsten Anwendungsfällen gerecht zu werden. Einen guten Überblick gibt Abbildung 6, welche auf der Literaturrecherche von Lother [LD01] aufsetzt und ab dem Jahr 2001 durch die Recherchen dieser Dissertation erweitert wurden.

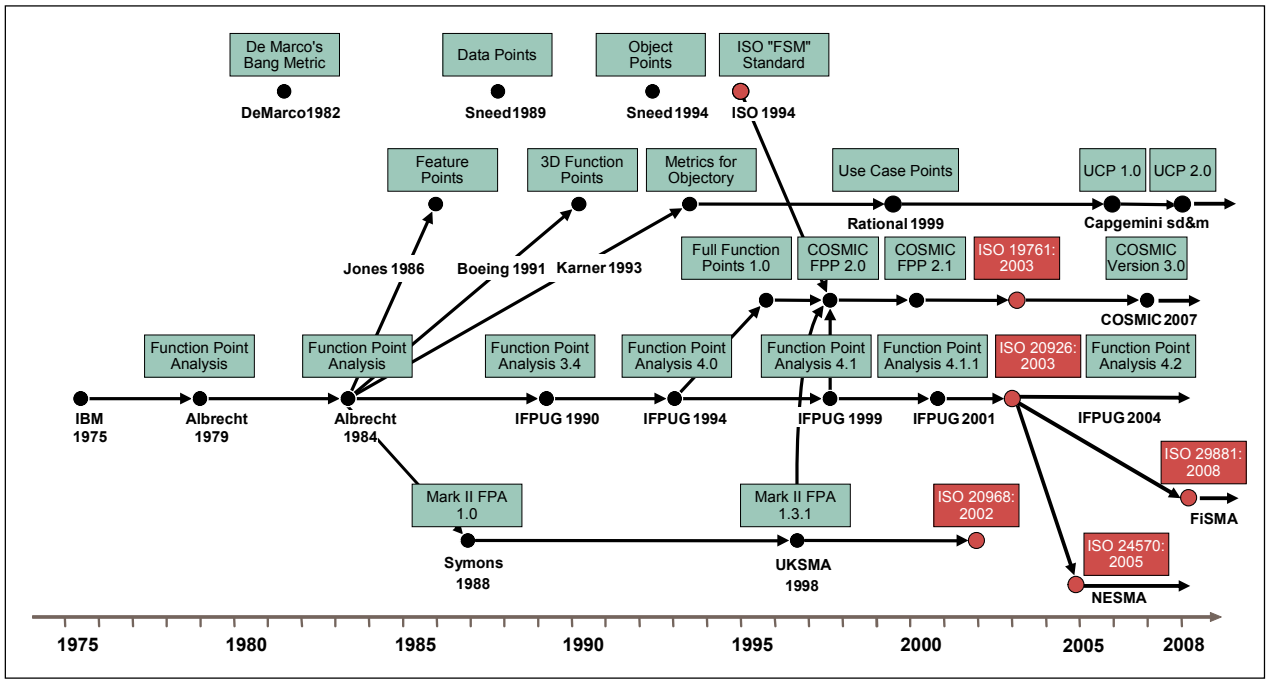


Abbildung 6: Entwicklung der Methoden zur funktionalen Größenmessung (FSM)

Es ist für diese Arbeit nicht notwendig, alle Varianten des FSM im Detail vorzustellen. Wir beschränken uns zunächst auf die fünf Methoden, welche eine hohe Reife und Anerkennung erfahren haben und zu einem international anerkannten Standard gemäß ISO/IEC 14143-1 entwickelt wurden (in Abbildung 6 rot mit weißer Schrift markiert). Trotzdem halten wir bereits hier die Anforderung nach einer ausgereiften Methode fest:

A 5: Ausgereifte Methode, die dem aktuellen Stand der Technik entspricht.

Gemäß des Standards ISO/IEC 14143-1 erfolgt die Größenbestimmung durch Messung der funktionalen Anforderungen des Anwenders[4]. Die bisher standardisierten Methoden sind (in chronologischer Abfolge):

- ISO 20968:2002. Mark II FPA wurde von Symons 1988 entwickelt, um die original FPA Methode zu verbessern. Diese Methode brachte einige Vorschläge für die Berücksichtigung der inneren Komplexität eines Software-Systems ein. Aktuell wird die Methode vom Metrics Practices Committee (MPC) der UK Software Metrics Association (UKSMA) federführend weiterentwickelt [ISO-M02].
- ISO 19761:2003. Die Full Function Point (FFP) Methode wurde ausgehend von der FPA der IFPUG entwickelt, beschränkt sich aber auf die reine Größenbestimmung von funktionalen Anforderungen, also berücksichtigt keine Gewichts- oder Komplexitätsfaktoren

[4] ISO/IEC 14143-1: "The concepts of Functional Size Measurement (FSM) are designed to overcome the limitations of earlier methods of sizing software by shifting the focus away from measuring how the software is implemented to measuring size in terms of the functions required by the user"

der System- oder Projektanforderungen. Federführend für die Weiterentwicklung ist das Software Measurement International Consortium (COSMIC) [ISO-C03].

- ISO 20926:2004. Die International Function Point User Group (IFPUG) hat die Ideen von Albrecht in der Methode IFPUG FPA fortgeführt. Die Methode ist weit verbreitet und sehr populär [ISO-I03].
- ISO 24570:2006. In Holland hat die Netherlands Software Metrics Association (NESMA) die Arbeiten der IFPUG FPA aufgegriffen und um eigene Anleitungen, Hinweise und „best practices“ ergänzt. Der Aufbau der Methode ist fast identisch mit IFPUG FPA [ISO-N05].
- ISO 29881:2008. In Finnland hat die Finnish Software Measurement Association (FiSMA[5]) aufbauend auf der FPA die Erfahrung aus über 600 Projekten in der Zeit von 1997 bis 2003 ausgewertet und daraus die Methode FiSMA 1.1 entwickelt. Sie berücksichtigt ausschließlich funktionale Anwenderanforderungen und definiert sich in Abgrenzung zur prozessorientierten FPA als *serviceorientierte* Methode für alle Arten von Software [ISO-F08].

Eine strukturelle Analyse der FSM-Methoden findet sich in der Literatur z.B. bei Fetcke in [FAD02]. Er verwendet einen axiomatischen Ansatz basierend auf der Messtheorie, um ein Modell für bekannte FSM-Methoden zu entwickeln (Abbildung 7). Daraus konnte er eine verallgemeinerte Repräsentation für die Methoden IFPUG FPA, Mark II FPA und COSMIC FFP entwickeln. Die Grundlage des Modells ist eine daten-orientierte Abstraktion der FSM-Methoden. Diese Aussage ist aufgrund der methodischen Ähnlichkeit auch auf die Methode der NESMA übertragbar.

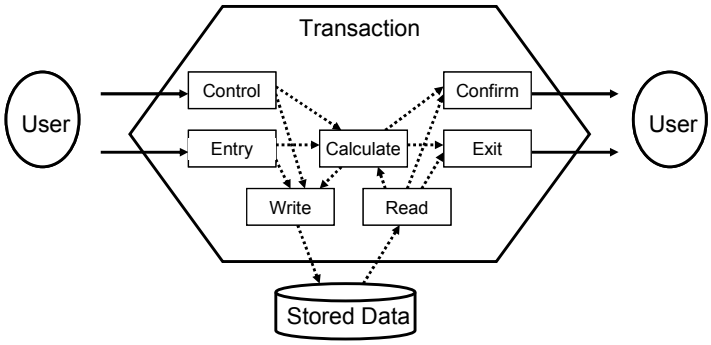


Abbildung 7: Transaktionen werden mit daten-orientierten Aktionen dargestellt [FAD02]

Die daten-orientierte Sicht ist allerdings *nicht* die dominierende Sicht in heute üblichen Spezifikationen in der Industrie. Hier werden überwiegend funktionale Beschreibungsformen in Anlehnung an die UML verwendet. In der Regel wird der Auftraggeber seine Anforderungen in Form einer Grobspezifikation oder eines Lastenheftes an verschiedene potenzielle Auftragnehmer ausschreiben. Auf dieser Basis führt der potenzielle Auftragnehmer eine Aufwandsschätzung durch.

[5] FiSMA stand früher für Finnish Software Metrics Association (1997-2002)

Eine Analyse von ca. 50 Spezifikationen, welche das Software-Haus Capgemini sd&m von unterschiedlichsten Auftraggebern in den letzten 5 Jahren erhalten hat, zeigt einen nicht einheitlichen Aufbau der Dokumente aus formloser textueller Beschreibung und Elementen der UML. Im Kontext der betrieblichen Informationssysteme überwiegt klar eine Use Case orientierte Sicht auf die Spezifikation, die UML als Beschreibungssprache ist weit verbreitet und anerkannt [JCJO93, BRJ99, Coc95, Coc03, AN05].

Damit wird deutlich, dass im Rahmen der genannten FSM-Methoden ein Transformationsschritt in eine daten-orientierte Abstraktion erfolgen muss. Diese Abstraktion basiert demzufolge auf einem Lösungsentwurf, der Elemente der Konstruktion (z.B. technischer Datenfluss) vorwegnimmt. Dies zeigt, dass bereits sehr viel technisches Verständnis und Erfahrung verlangt wird.

Es wäre also wünschenswert, eine Methode zur funktionalen Größenbestimmung zu haben, welche unmittelbar auf der Use Case orientierten Sicht aufsetzt. Diese Anforderung halten wir fest:

A 6:	Direkte Ableitung der zu schätzenden (funktionalen) Größen unmittelbar aus einer Grobspezifikation mit einer Use Case orientierten Sicht.

Für die Bewertung der zuvor dargestellten fünf standardisierten FSM-Methoden wurde eine detaillierte Analyse durchgeführt. Sie zeigt im Wesentlichen drei Schwächen hinsichtlich der bisher formulierten Anforderungen. Diese Schwächen können wie folgt zusammengefasst werden:

- Hohe Anforderungen an die Qualifikation des Schätzers: Diese skizzierten Methoden setzen alle ein gutes technisches Verständnis voraus und verlangen viel Übung und Erfahrung im Umgang mit der Methode, um gute reproduzierbare Ergebnisse zu erreichen. Der Aufwand für die Durchführung einer Schätzung kann in der Größenordnung einer Expertenschätzung liegen, wird bei sehr erfahrenen FSM-Schätzern aber als geringer angegeben.
- FiSMA und COSMIC FFP: Ausschließliche Ausrichtung auf den funktionalen Teil der Anforderungen, d.h. nichtfunktionale Anforderungen und Einfluss von Projekt und Organisation bleiben unberücksichtigt (siehe Erläuterung dazu am Ende von Abschnitt 1.2).
- Keine direkte Umsetzung einer Use Case orientierten Sicht.

Damit scheiden die fünf standardisierten FSM-Methoden für die weitere Betrachtung aus. Daher wird die Betrachtung auch auf nicht standardisierte Methoden ausgedehnt. Zunächst sei bemerkt, dass die ursprüngliche Function Point Methode von Albrecht (ebenso wie IFPUG FPA) sowie einige ihrer Weiterentwicklungen sehr wohl nicht funktionale Anforderungen berücksichtigen. Anstelle des Zählmaßes mit einer daten-orientierten Sicht wurden zudem Varianten der Function Point Methode mit anderen Sichten und Zählmaßen entwickelt, z.B. Object Points, Feature Points oder Use Case Points (siehe Abbildung 6). Diese Methoden sind allerdings bisher nicht gemäß ISO standardisiert und haben bis heute keine hohe Verbreitung erlangt.

Ein besonderes Augenmerk gilt hierbei der Use Case Points (UCP) Methode, die unmittelbar auf einer Use Case orientierten Sicht aufsetzt und auch nichtfunktionale Anforderungen berücksichtigt. Die UCP-Methode wurde 1993 von Karner in Zusammenarbeit mit der Firma Objectory in

seiner Diplomarbeit entwickelt [Kar93]. Allerdings weist die Methode einige konzeptionelle Schwächen auf [OA06] und bis 2005 beschränkt sich die Praxiserfahrung auf wenige Unternehmen und vergleichsweise wenige Veröffentlichungen [MAC05] (im Gegensatz zur Function Point Methode).

Daher wurde im Rahmen dieser Dissertation zunächst eine erste Überprüfung anhand von Software-Entwicklungsprojekten aus der Praxis der Firma Capgemini sd&m im Jahr 2005 durchgeführt. Es konnte eine recht gute Anwendbarkeit der Methode für betriebliche Informationssysteme festgestellt werden [FJ06]. Ein Vergleich z.B. mit dem weit verbreiteten kommerziellen Top-Down Werkzeug SLIM, basierend auf Arbeiten von Putnam [PM92], zeigte eine deutlich bessere Passung der UCP-Methode: Die Aufwände von 10 repräsentativ ausgewählten Industrie-Projekten wurden nach Projektabschluss mit Nachschätzungen basierend auf SLIM und UCP verglichen. Im firmeninternen Abschlussbericht heißt es zusammenfassend: „Bei dem direkten quantitativen Vergleich der Schätzergebnisse beider Methoden ist UCP (Karner) mit 14,3% durchschnittlicher absoluter Abweichung SLIM mit 32,8% Abweichung klar überlegen. Beide Methoden produzieren Ausreißer, die Ausreißer bei SLIM sind jedoch zahlreicher und weichen stärker vom realen Wert ab.“

Diese Studie motivierte eine eingehende Beschäftigung mit UCP als FSM-Methode, auch wenn sie bis heute noch nicht gemäß ISO standardisiert ist. Nachfolgend wird zunächst die Spezifikation mit Use Cases kurz skizziert und dann die Use Case Points Methode detailliert dargestellt und bewertet.

2.3 Grundlagen zur Spezifikation mit Use Cases

Bevor das Konzept der Use Cases erläutert wird, wird das Verständnis des in dieser Arbeit verwendeten Spezifikationsbegriffes zusammengefasst. Wir beschränken uns dabei auf den Kontext der Erstellung betrieblicher Informationssysteme, die im Auftrag eines Anwenders (=Auftraggeber) individuell gefertigt werden. Die zu erstellenden Anwendungen unterstützen bestimmte Geschäftsabläufe des Auftraggebers und enthalten z.B. Anteile der Kategorien Datenverwaltungsanwendung, algorithmische Anwendung und Auswertungsanwendung.

2.3.1 Spezifikationsmethodik

Eine *Spezifikation* nach dem hier vorliegenden Verständnis beschreibt ein Modell einer zu erstellenden Anwendung [Den91, Sie03]. Das bedeutet, dass eine *fachliche Lösung* und nicht eine fachliche Problemdefinition spezifiziert wird. Eine Spezifikation beschreibt, „Was“ eine Anwendung leistet *und* „Wie sie dies fachlich tut“. Das „Wie“ der technischen Umsetzung ist nicht Bestandteil der Spezifikation. Das bedeutet für Inhalt und Vorgehen: In der Spezifikation wird ein betriebliches Informationssystem fachlich gestaltet, das eine fachliche Problemstellung anhand einer fachlichen Lösungsidee (z. B. ein betriebswirtschaftliches Konzept) löst.

Das Modell der Anwendung wird fachlich gestaltet, da es nicht durch Analyse eines Ist-Zustandes entsteht (die Anwendung, die das Modell beschreibt, existiert in der Regel noch nicht), sondern schrittweise durch die Entwicklung eines Soll-Zustandes. Selbst bei einer Altsystem-Ablösung werden in der Regel zwar die funktionalen Anforderungen übernommen, jedoch wer-

den die fachliche Struktur der Anwendung und damit das „Wie“ der fachlichen Umsetzung meist neu gestaltet. Im Idealfall ist das Modell der Anwendung fachlich vollständig und legt keine technische Umsetzung fest. Pragmatisches Vorgehen führt dazu, dass vorhersehbare technische Restriktionen bereits in die Spezifikation einfließen werden und Einfluss auf die Gestaltung der fachlichen Lösung nehmen.

Neben dem Begriff *Spezifikation* findet man in der Praxis auch Bezeichnungen wie Fachspezifikation, fachliche Spezifikation, Fachfeinkonzept, Sollkonzept, Funktionelle Spezifikation, oder Feature Specification. Da diese Bezeichnungen in der Regel nicht standardisiert sind, können damit durchaus Dokumente im Sinne der hier beschriebenen Spezifikation gemeint sein. Häufig wird auch der normierte Begriff des *Pflichtenheftes* laut DIN 69905[6] benutzt, was die „vom Auftragnehmer erarbeiteten Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenheftes“ meint. Damit umfasst das Pflichtenheft über die Spezifikation hinaus bereits eine Beschreibung der technischen Umsetzung.

Als Beispiel für die Struktur einer Spezifikation dient Abbildung 8. Die *Anforderungsbausteine* spiegeln hier die Anforderungsanalyse („Was muss die Lösung können?“) und die *Beschreibungsbausteine* definieren die fachliche Gestaltung („Wie sieht die Lösung *fachlich* aus?“). Weiß hinterlegt und fett umrandet sind die für die UCP-Methode unmittelbar relevanten Bausteine.

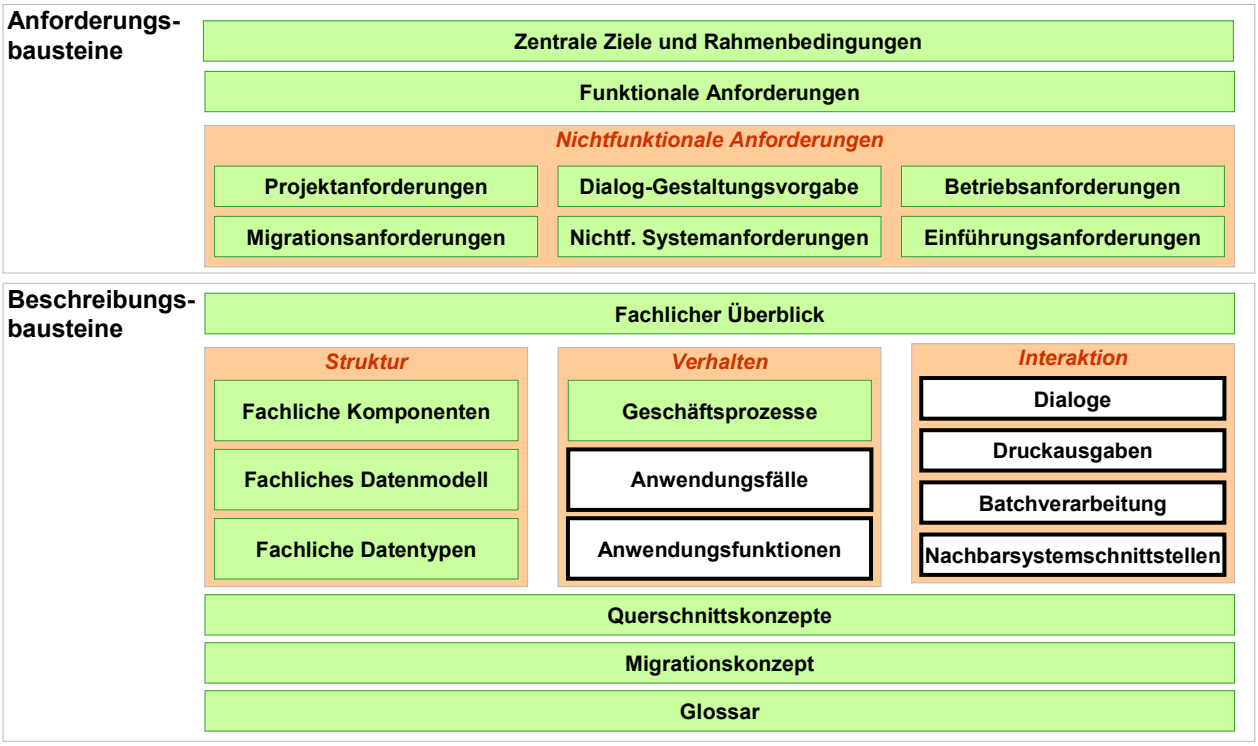


Abbildung 8: Spezifikationsbausteine für betriebliche Informationssysteme [DS06]

Nachfolgend werden nur die Elemente aus Abbildung 8 kurz beschrieben, die für das weitere Verständnis im Kontext der UCP-Methode relevant sind [DS06].

---

[6] siehe auch VDI Richtlinie 2519 Blatt 1 „Vorgehensweise bei der Erstellung von Lasten-/Pflichtenheften“

Der Baustein …

Anwendungsfälle spezifiziert die Funktionalität einer Anwendung. Ein Anwendungsfall beschreibt das Verhalten und die Interaktion eines Systems als Reaktion auf die zielgerichtete Anfrage oder Aktion eines Akteurs (Definition siehe später in Abschnitt 2.3.2).

Anwendungsfunktionen beschreibt Ausschnitte von Anwendungsfällen, die ohne Unterbrechung vom System, ggf. unter Benutzung von Schnittstellen zum Nachbarsystem, ausgeführt werden. Die Beschreibung der Anwendungsfunktionen ähnelt der Beschreibung eines Anwendungsfalls. Dabei wird wesentlich stärker auf den Aspekt "Wie soll die Anwendung eine Verarbeitung durchführen" eingegangen.

Dialoge beschreibt, wie der Anwender mit dem System kommuniziert, d.h. welche Funktionen er wie in welchen Dialogen aufruft und welche Daten er pflegen kann. Die Beschreibung umfasst das statische Aussehen (Layout) und das dynamische Verhalten der Dialoge.

Druckausgaben beschreibt das Layout und den Inhalt von Drucklisten, Serienbriefen, etc. Für komplexe Auswertungen werden mögliche Selektions- und Verdichtungskriterien beschrieben. Spezifikationen von Druckausgaben entsprechen im Aufbau weitestgehend den Dialogspezifikationen. Allerdings entfällt die dynamische Komponente. Druckausgaben werden durch das Druck-Layout sowie durch die Referenz auf die Entitäten und Attribute des Datenmodells spezifiziert.

Batchverarbeitung dokumentiert Anforderungen an die batchverarbeitenden Programme des Systems (Konfiguration, Abhängigkeiten, Datenvolumen, etc.). Ein Batchprogramm realisiert eine eigenständige Verarbeitung ohne direkten Benutzereingriff während des Ablaufes. Die Fachlichkeit wird im Baustein *Anwendungsfunktionen* beschrieben; die Definition und das Layout der Ausgabelisten findet man im Baustein *Druckausgaben*.

Nachbarsystem-Schnittstellen beschreibt detailliert die Anforderungen an die Verbindung zwischen dem spezifizierten System und den externen Nachbarsystemen, mit denen das System kommunizieren soll. Die Spezifikation umfasst folgende Punkte: Nennung und Charakterisierung der Nachbarsysteme, Ablauf der Kommunikation (grob), Nennung der Geschäftsprozesse und Anwendungsfälle, die die Schnittstelle nutzen. Bei Stammdatenschnittstellen: Nennung und Charakterisierung der Stammdaten, Fehlersituationen und Fehlerbehandlung aus fachlicher Sicht, Transaktionsunterstützung. Ein Übersichtsbild aus fachlicher Sicht über die Schnittstellen zu den Nachbarsystemen ist Bestandteil des Bausteines *Fachlicher Überblick*. Zur Beurteilung der Komplexität einer Schnittstelle dient die hier in diesem Baustein vorgestellte Detailbeschreibung.

Das zugehörige Metamodell als Kompositionsstrukturdiagram gemäß UML zeigt Abbildung 9.

Die Anwendungsfälle (engl. Use Cases) detaillieren die systemunterstützten Arbeitsschritte aus den Geschäftsprozessen. Durch die Summe aller Anwendungsfälle wird die Funktionalität eines Systems im Detail vollständig gestaltet. Im Normalfall erfolgt in den Anwendungsfällen die wesentliche Beschreibung des fachlichen Verhaltens der Anwendung. In zwei Fällen setzen wir den Schwerpunkt anders:

- Bei Datenverwaltungsanwendungen mit komplexer Interaktion auf der Benutzungsoberfläche wird der Schwerpunkt in die Dialogspezifikation verlagert. Komplex bedeutet hier,

dass es verschiedene Maskenbereiche und -elemente mit gegenseitigen Wechselwirkungen gibt. Anwendungsfälle beschreiben hier das fachliche Verhalten grob und mit dem Schwerpunkt, das fachliche Ziel des Anwendungsfalls herauszuarbeiten. Das Detailverhalten der Anwendung wird in der Dialogspezifikation ausgearbeitet (z.B. was passiert, wenn in einem bestimmten Maskenelement eine Eingabe erfolgt).

- Bei algorithmischen Anwendungen mit komplexen Algorithmen werden (Teil-) Abläufe ohne Interaktion mit einem Akteur als Anwendungsfunktionen beschrieben. In den Anwendungsfällen wird dann auf diese Anwendungsfunktionen verwiesen.

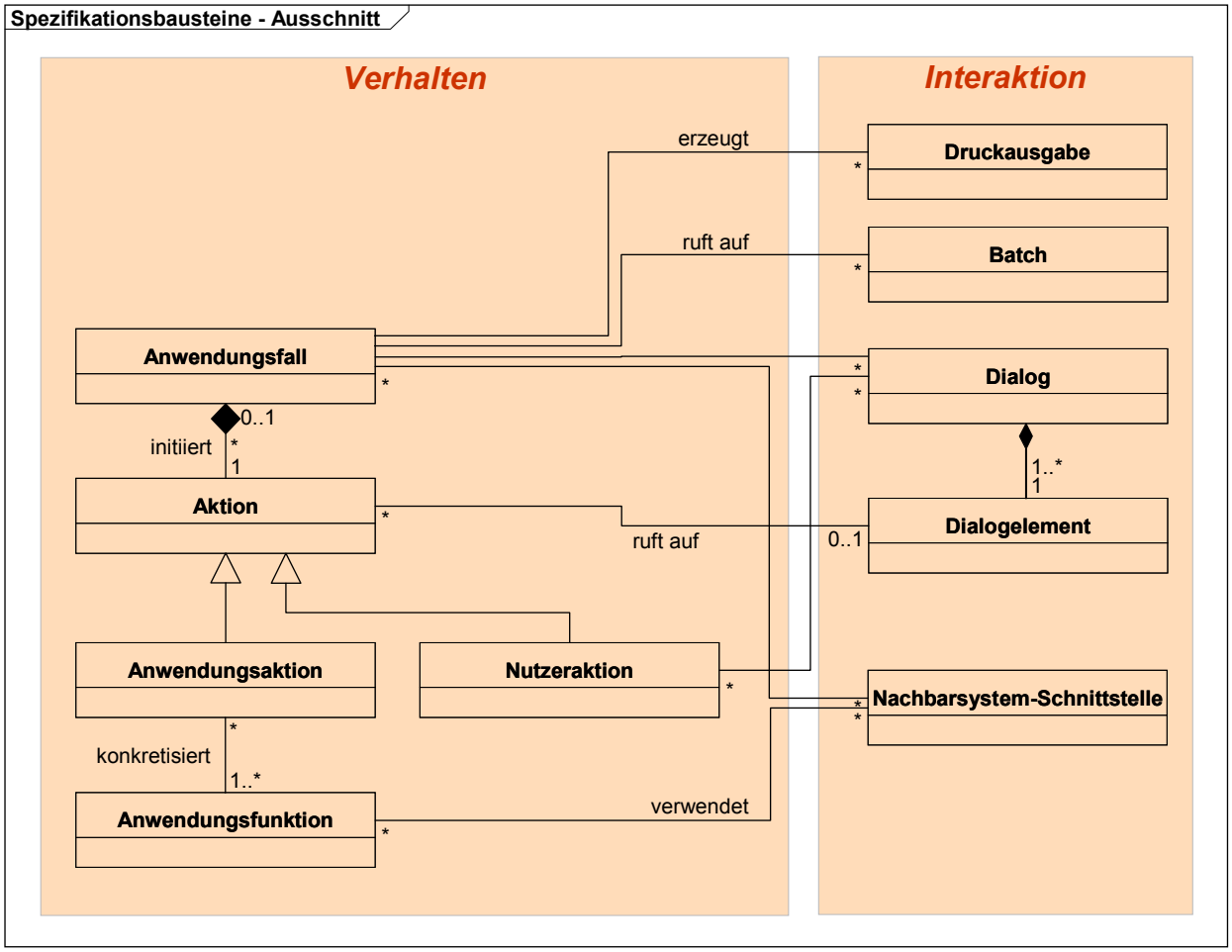


Abbildung 9: Das Metamodell zur Spezifikationsmethodik gemäß Abbildung 8

Mit den Anwendungsfunktionen steht bei Bedarf eine dritte Detaillierungsstufe für Abläufe zur Verfügung. Anwendungsfunktionen werden genutzt, um Teilabläufe eines Anwendungsfalls zu beschreiben, die ohne Unterbrechung von der Anwendung, ggf. unter Benutzung von Schnittstellen zu Nachbarsystemen, ausgeführt werden. In der Regel spezifizieren wir solche Teilabläufe als Anwendungsfunktion,

- die in vielen Anwendungsfällen verwendet werden, und daher als Anwendungsfunktion herausgezogen und nur einmal beschrieben werden,
- die fachlich komplex sind und daher eine eigene, detaillierte Darstellung erfordern, oder

- für die eine Gruppierung aus fachlicher Sicht gefordert ist, um fachlich bereits die Bildung einer Komponente festzulegen. Das ist sinnvoll, wenn man z.B. Festlegungen, für die fachlich häufig Änderungen erwartet werden, an einer Stelle bündeln will (z.B. in einer Komponente mit Kapselung und definierter Schnittstelle).

Eine Spezifikation kann in die Phasen eines Software-Entwicklungsprojektes eingeordnet werden (Abbildung 10). Dazu unterscheiden wir die Phasen *Grobspezifikation* (oftmals auch als Grobkonzept oder Studie bezeichnet), *Feinspezifikation* (oftmals nur als Spezifikation oder Fachfeinkonzept bezeichnet), *Konstruktion* und *Realisierung*. Die hier gemeinte *Spezifikation* beginnt bereits während der Phase *Grobspezifikation* und unterscheidet sich von der Spezifikation während der Phase *Feinspezifikation* nur hinsichtlich ihres Detaillierungsgrades und Umfanges.

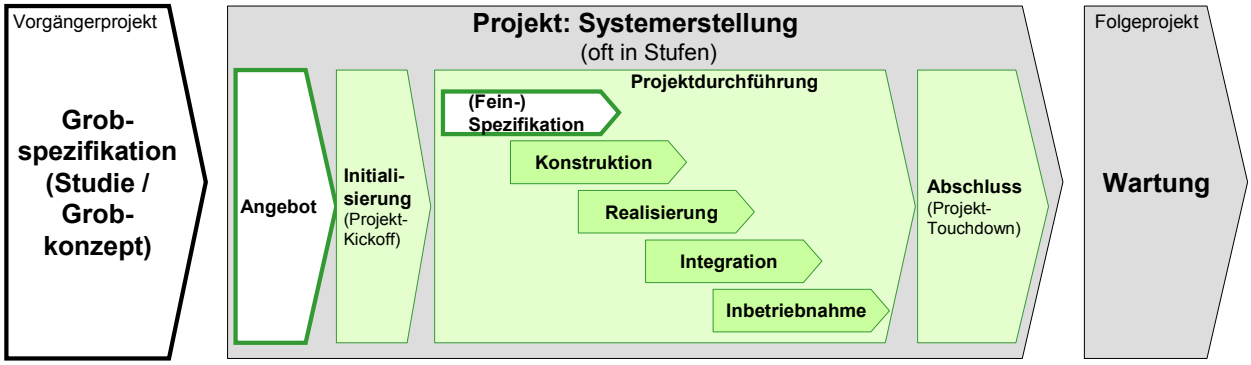


Abbildung 10: Einbettung der Spezifikation in die Phasen eines Software-Entwicklungsprojektes

Im Rahmen einer Grobspezifikation werden häufig nur die in Abbildung 8 weiß hinterlegten Bausteine ausgearbeitet, also zentrale Ziele und Rahmenbedingungen, Funktionale Anforderungen, fachlicher Überblick, Geschäftsprozesse und Anwendungsfälle. Die nichtfunktionalen Anforderungen werden sehr grob spezifiziert, eine vollständige Ausarbeitung, ebenso wie die der übrigen Spezifikationsbausteine, erfolgt dann in der Phase *Feinspezifikation*.

Im Rahmen des in 1.2 beschriebenen Angebotsprozesses wird dann auf Basis des Ergebnisses der Grobspezifikation der Projekt-Aufwand *PE* geschätzt und einem Angebot des Auftragnehmers an den Auftraggeber zugrunde gelegt. Es ist intuitiv klar, dass die Schätzgenauigkeit zunimmt (und damit die Streuung abnimmt), je detaillierter und präziser eine Spezifikation ist (Abbildung 11). In der Konstruktionsphase kann dann weitere Schätzsicherheit z.B. durch Prototypen gewonnen werden. Dieser Zusammenhang wurde bereits von Boehm [Boe81] beschrieben. Boehm gibt in [Boe+00] eine Schätzunsicherheit von einem Faktor vier für sehr frühe Phasen vor der Grobspezifikation an und einen Faktor von 1,5 für den Zeitpunkt zur Angebotsabgabe (Grobspezifikation). aus dem Kontext von Capgemini sd&m ist bekannt, dass eine Streuung zum Zeitpunkt der Grobspezifikation von 40% und weniger erwartet wird und nach Abschluss der Feinspezifikation von 20% und weniger.

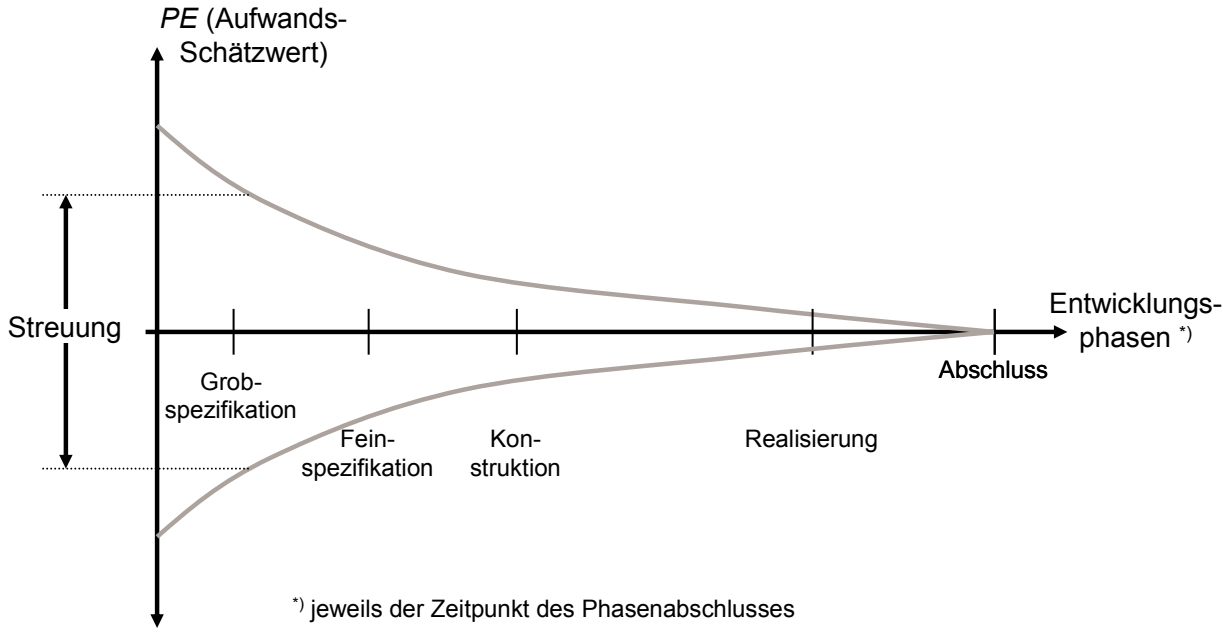


Abbildung 11: Schematische Darstellung der Aufwandsschätzgenauigkeit im zeitlichen Verlauf

Innerhalb der Entwicklungsphasen erfolgt die Erarbeitung der Spezifikationsbausteine gemäß Abbildung 8 nicht strikt sequentiell. In der Praxis erfolgt die Erarbeitung der verschiedenen *Anforderungsbausteine* und der *Beschreibungsbausteine* zeitlich teilweise bereits überlappend, beginnend mit den Anforderungsbausteinen. Dies deckt sich mit den Analysen von Jacobson [JBR99, AN05], welche im Rational Unified Prozess (RUP) Eingang gefunden haben. In Abbildung 12 ist RUP [Kru03] schematisch dargestellt, die Disciplines beschreiben die Tätigkeiten während der Phasen *Inception*, *Elaboration*, *Construction* and *Transition* und überlappen sich in den Phasen. Der oben definierte Begriff der Grobspezifikation kann am ehesten durch die Phase *Inception* gespiegelt werden, allerdings sind die Modelle nicht vollständig aufeinander abbildbar. In der Praxis verschwimmen die Phasen *Grobspezifikation* und *Feinspezifikation* teilweise ineinander.

Für die hier relevante Themenstellung der Aufwandsschätzung ist es nicht notwendig, die Abgrenzung zwischen Grob- und Feinspezifikation weiter zu vertiefen, da es nur darauf ankommt, aus der Spezifikation die Anwendungsfälle abzählbar zu identifizieren und hinsichtlich ihrer Komplexität grob zu klassifizieren. Diese Fragestellung der Identifikation wird in Kapitel 3 dann weiter vertieft werden. Zunächst soll in Abschnitt 2.3.2 definiert werden, was im Rahmen dieser Arbeit unter einem *Anwendungsfall* (engl.: Use Case) verstanden wird.

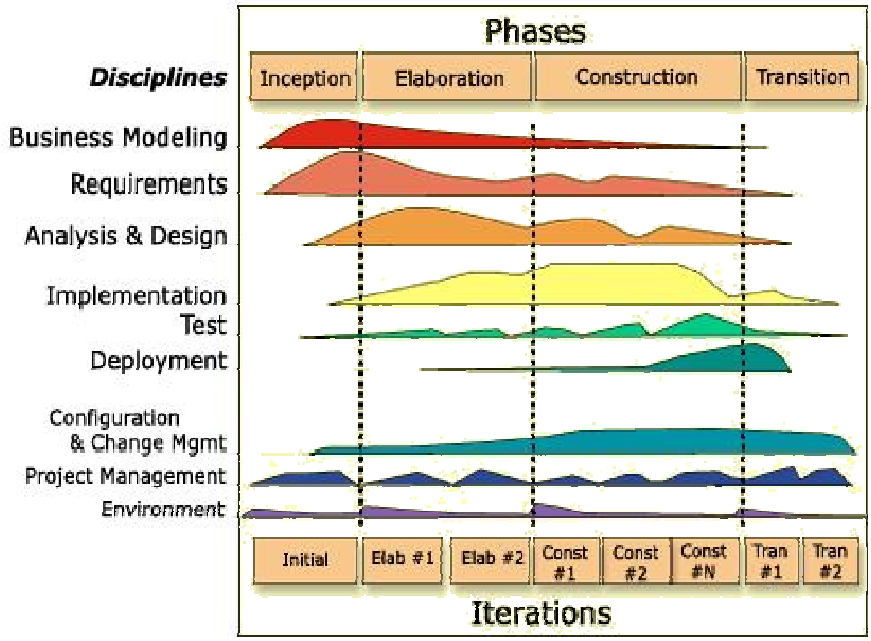


Abbildung 12: Der Rational Unified Process (RUP)[JBR99]

2.3.2 Use Case Beschreibung

In der Spezifikations-Welt gibt es viele Möglichkeiten, funktionale Anforderungen „aufzuschreiben“. Eine weite Verbreitung hat die UML 2 erreicht [AN05, RQZ07]. Weite konzeptionelle Verbreitung aber wenig praktischen Nutzen (siehe dazu weiter unten) haben in diesem Kontext die Use-Case-Diagramme (Abbildung 13). Für das Verständnis der UCP-Methode ist zu beachten, dass Use-Case-Diagramme *keine* Voraussetzung für die Anwendung der UCP-Methode sind – sie sind nur eine von vielen möglichen Notationselementen.

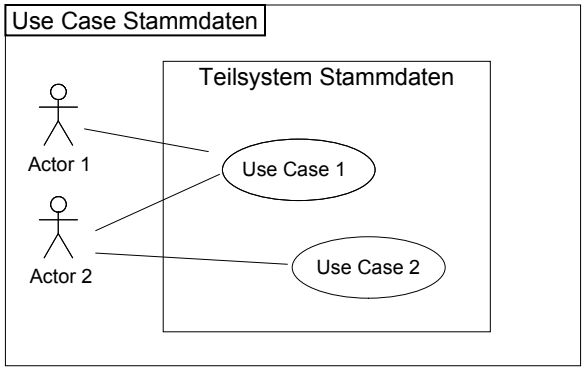


Abbildung 13: Beispiel für ein Use-Case-Diagramm

Als Use Case (deutsch: *Anwendungsfall*) definieren wir in Anlehnung an Cockburn [Coc95, Coc98, Coc03]:

Ein Use Case erfasst einen Vertrag zwischen den Stakeholdern[7] eines Systems hinsichtlich des Systemverhaltens. Der Use Case beschreibt das Systemverhalten unter verschiedenen Bedingungen als Antwort auf eine Anfrage von einem der Stakeholder, genannt *Actor*. Der Actor löst eine Interaktion mit dem System aus, um ein bestimmtes Ziel zu erreichen. Das System antwortet unter Berücksichtigung der Interessen aller Stakeholders. Abhängig von der speziellen Systemanfrage und dem Anfragekontext können sich eine unterschiedliche Reihenfolge des Systemverhaltens oder unterschiedliche Szenarien ergeben. Der Use Case fasst alle möglichen unterschiedlichen Szenarien zusammen.

In der Praxis hat sich für die Beschreibung der Fachlichkeit großer betrieblicher Informationssysteme eine Beschreibung der Use Cases in strukturierter textueller Form anstelle der Use-Case-Diagramme etabliert. Leider ist dafür bisher kein grundsätzlicher Standard vorhanden. Die nachfolgende Definition ist an [DS06] angelehnt. Um einen Use Case vollständig zu beschreiben, werden verschiedene Bestandteile benötigt:

- Use Case Beschreibung
- Ablaufdiagramm (z.B. als Aktivitätsdiagramm der UML)
- CRUD-Diagramm[8] (optional)
- Zustandsdiagramm (optional)
- Sequenzdiagramm (optional)

Nicht alle Bestandteile sind zwingend, sondern können teilweise optional verwendet werden, um eine bessere Verständlichkeit für den Leser zu erzeugen. Einige sind daher als optional gekennzeichnet. Die Verwendung hängt dabei wesentlich von der notwendigen Detailbeschreibung des Use Cases ab. Wir beschränken uns nachfolgend auf die Darstellung der Use Case Beschreibung, für die anderen Diagramm-Typen sei z.B. auf Rupp [RQZ07] verwiesen.

Tabelle 6 zeigt, wie eine tabellarische Use Case Beschreibung aussehen kann. Die einzelnen Elemente (Attribute) bedeuten:

Titel ist ein eindeutiger Bezeichner des Use Cases. Er sollte so formuliert sein, dass er möglichst prägnant Hinweise auf das Ziel des Use Cases gibt.

Kurzbeschreibung fasst den Use Case in wenigen Sätzen kurz zusammen. Dieser Text dient in Übersichten, Tabellen, Auswertungen etc. zur Klärung des Inhalts.

Ziel beschreibt aus fachlicher Sicht die Absicht und den Grund, weshalb der Actor den Use Case überhaupt durchführt und bildet somit eine Art „Management Summary“ für die Beschreibung, was dieser Use Case leisten soll. Jeder Use Case hat genau ein Ziel. Die Er-

---

[7] Für den Begriff Stakeholder gibt es im Deutschen keine gut passende Entsprechung, daher wird der englische Begriff weiterhin auch in dieser Arbeit benutzt. Eine sinngemäße Entsprechung wäre Interessenvertreter, „Geschäftsinteressent“, Akteur, oder Anspruchsberechtigter. In der Wirtschaft hat sich allerdings auch der Begriff „stakeholder management“ fest verankert, was eine weitere Verwendung des englischen Begriffes unterstützt.

[8] CRUD := Create, Read, Update, Delete. In einem CRUD-Diagramm wird dokumentiert, welche Lese- und Schreibzugriffe ein Anwendungsfall direkt auf die Entitäten des fachlichen Datenmodells ausführt.

gebnisse werden dann im Abschnitt „Ergebnis“ definiert und die Auswirkungen auf die Objekte erläutert.

Titel	**Dient der eindeutigen Identifikation eines Use Cases.**
Kurzbeschreibung	Worum geht es?
Ziel	Was soll mit dem Use Case erreicht werden?
Actor	Wer löst den Use Case aus?
Auslöser	Wie kommt es zur Ausführung des Use Cases? Was ist der Grund?
Vorbedingung	Welche Bedingungen müssen zu Beginn herrschen?
Szenarien	Was passiert während des Ablaufs? Ein Erfolgsszenario und möglicherweise weitere Alternativszenarien
Ergebnis	Welche Informationen werden vom Use Case geliefert?
Ausführungshäufigkeit	Wie oft wird dieser Use Case im System ausgeführt?
NFA	Was sind die nicht-funktionalen Anforderungen an diesen Use Case?
Bemerkungen	Was gibt es sonst noch zu berichten?

Tabelle 6: Beispiel für die Strukturierung einer textuellen Use Case Beschreibung

Actor eines Use Cases charakterisiert die menschlichen oder technischen Nutzer der Anwendung, die einen Use Case auslösen oder nur beteiligt sind. Anwendungsfälle beschreiben nur die Teile eines Geschäftsprozesses, die durch die Anwendung unterstützt werden. Daher werden hier keine Actors dokumentiert, die zwar an den zugehörigen Aktivitäten der Geschäftsprozesse beteiligt sind, aber nicht mit der Anwendung interagieren. Diese werden nur in der Geschäftsprozessmodellierung erfasst (siehe Baustein Geschäftsprozesse in Abbildung 8).

Auslöser ist die fachliche Ursache für das Ausführen eines Use Cases. Er ist der Startpunkt der Interaktion mit dem System. Zum Beispiel könnte der Auslöser für einen Use Case „Adressdaten erfassen“ der Besuch eines Neukunden im Unternehmen sein. Sind mehrere Auslöser für den Start eines Use Cases möglich, so werden diese hier einzeln erläutert.

Vorbedingung beschreibt alle Voraussetzungen die erfüllt sein müssen, damit der Use Case durchgeführt werden kann.

Szenarien eines Use Cases sind der Kernbestandteil einer Use Case Beschreibung. Sie beschreiben den Ablauf von Aktionen des Actors und des Systems, der notwendig ist, das Ziel des Use Cases zu erreichen. Die Beschreibung erfolgt in textueller Form und ist in einzelne Schritte („Aktionen“) aufgegliedert. Jeder Schritt ist nummeriert und trägt die Information, wer oder was diesen Schritt initiiert bzw. ausführt, z.B. „Der Anwender…“ oder „Das System…“. Ein Szenario besitzt üblicherweise keine oder nur wenige eigene Verzweigungen, um einen guten Lesefluss zu ermöglichen und das Verständnis zu erhöhen. Es wird zwischen Erfolgs- und Alternativszenarien unterschieden.

Erfolgsszenario des Use Cases ist der fehlerfreie Erfolgsfall, der im Rahmen des Ziels des Use Cases genannt ist. Jeder Use Case hat genau ein Erfolgszenario.

Alternativszenarien können spezifiziert werden. Diese beschreiben separat *Spezialfälle*, z.B. beim Vorliegen besonderer Randbedingungen, und *Fehlerfälle* in textueller oder tabellarischer Form. Fehlerfälle sind auf fachliche Fehler beschränkt, technische Fehler wie z.B. der Ausfall einer Netzwerkverbindung werden nicht beschrieben.

Ergebnis dokumentiert alle Veränderungen aller vom Use Case betroffenen Entitäten. Ermittelt der Use Case Daten, sind die Rückgabewerte grob zu umschreiben. So könnte z.B. ein Ergebnis lauten: „Der Use Case gibt eine Liste von Objekten des Entitätstyps *Partner* zurück“. Die Ergebnisse sind hinsichtlich aller vorhandenen Szenarien pro Szenario zu beschreiben, solange nicht alle Szenarien das gleiche Ergebnis liefern.

Darüber hinaus werden mit dem Ergebnis auch fachliche Fehler zurückgemeldet, die durch fachliches Fehlverhalten z.B. des Actors entstehen (z.B. Eingabe einer nicht vorhandenen Kundennummer). Diese Fehler müssen vorhersehbar sein, sonst könnten sie hier nicht dokumentiert werden. Kann so ein Fehler auftreten, so wird er in einem Alternativszenario beschrieben und das Ergebnis, z.B. die Meldung, hier in diesem Abschnitt dokumentiert.

Ausführungshäufigkeit gibt an, wie oft der Use Case durchgeführt wird. Entweder bezieht sich diese Häufigkeit auf bestimmte Ereignisse (z.B. jeweils einmal nach Neustart des Systems) oder auf Zeitintervalle (z.B. zehnmal pro Sachbearbeiter und Monat).

NFA (Nicht-funktionale Anforderungen) werden hier festgehalten, wenn sie speziell für diesen Use Case vorliegen. Andernfalls werden NFAs (z.B. Antwortzeiten, Sicherheit etc.) in einem eigenen Spezifikationsbaustein allgemeingültig dokumentiert (siehe Abbildung 8).

Bemerkungen dienen der Hinterlegung von sonstigen Informationen, die in den bisherigen Attributen nicht dokumentiert wurden oder für die im Projekt kein eigenes Attribut definiert wurde, z.B. Review-Anmerkungen oder Historienbeschreibung.

Zur begrifflichen Trennung für die Aufwandsschätzung führen wir ein spezielles Metamodell der Use Cases ein und verwenden hierfür auch bewusst den englischen Begriff *Use Case*. In der Spezifikationswelt benutzen wir weiterhin den deutschen Begriff *Anwendungsfall.*

Für die Aufwandsschätzung im Sinne der UCP-Methode sind Anwendungsfälle auf Use Cases abzubilden. Dies bezeichnen wir als *Mapping* (Abbildung 14).

Wie genau dieses Mapping funktioniert und wie das Meta-Modell aussieht, wird in Kapitel 3 beschrieben werden. Karner [Kar93] hat z.B. ein einfaches Mapping für die Anwendungsfälle durch zählen der Transaktionen je Anwendungsfall vorgeschlagen (siehe Abschnitt 2.4.1). Der Begriff der Transaktion wurde dabei aber nicht sauber definiert.

Wir werden in Abschnitt 3.1.3 ein anderes Mapping vorschlagen, welches auf der Zählung der Anzahl Szenarien, Schritte und Interaktionselementen (Dialoge, Schnittstellen, Berichte) beruht.

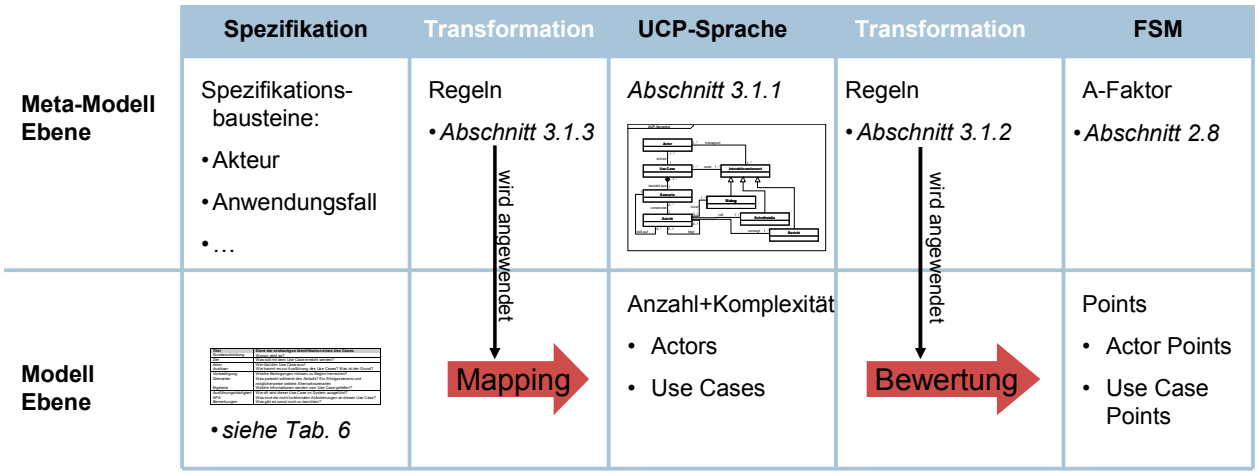


Abbildung 14: UCP-Methode – Schematische Darstellung der Transformation von einer Spezifikation über die UCP-Sprache zum Functional Size Measurement (FSM)

Der fachliche Umfang, den ein Use Case beschreibt, kann erheblich variieren. Cockburn hat dafür drei unterschiedliche Abstraktions-Ebenen definiert, auf denen Use Cases definiert werden können [Coc03]:

Summary Goal: Dies beschreibt die höchste Abstraktionsebene, die Anwendungsfälle sind an eher strategischen Zielen ausgerichtet und umfassen gewöhnlich mehrere Anwenderziele. Ein Use Case repräsentiert dann für gewöhnlich ein komplettes Anwendungssystem.

User Goal: Diese Ebene deckt sich weitgehend mit der in Abschnitt 2.3.1 beschriebenen Abstraktionsebene von Anwendungsfällen. Die Summe aller Anwendungsfälle beschreibt fachlich vollständig ein Anwendungssystem.

Subfunctions: Diese Abstraktionsebene ist deutlich feiner als die der User Goals und adressiert Funktionalität der Anwendungsfälle auf der Ebene von Schritten innerhalb von Szenarien.

Im Kontext der Aufwandsschätzung fokussieren wir im Folgenden immer auf den *User Goal* Abstraktionslevel. Die unterschiedliche Granularität von Anwendungsfällen ist damit noch nicht grundlegend gelöst, denn insbesondere in sehr großen Anwendungssystemen kann die Zahl der Schritte und Szenarien in einem Anwendungsfall sehr umfangreich werden. Dann bietet es sich an, in der Modellierung einen Teil der Fachlichkeit in einen anderen Anwendungsfall auszulagern. Die UML unterstützt diese Form der Abhängigkeit durch die Möglichkeit des Imports eines anderen Anwendungsfalls mittels der <<include>> Beziehung (siehe Abbildung 15).

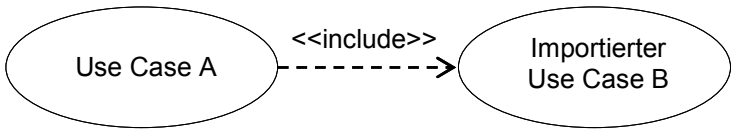


Abbildung 15: Inklusion von Use Cases

Eine <<include>> Beziehung ist nicht optional, das Verhalten wird immer vollständig inkludiert. Der Anwendungsfall (Use Case A), der das Verhalten einbindet, ist somit abhängig vom Anwendungsfall (Use Case B), den er inkludiert. Dies symbolisiert der Pfeil in Abbildung 15.

Eine optionale Beziehung ist die <<extend>> Beziehung, siehe Abbildung 16. Sie zeigt an, dass das Verhalten eines Use Cases A abhängig von bestimmten Bedingungen (Conditions) an der Stelle des Erweiterungspunktes (engl. Extension Point) durch einen anderen Use Case B erweitert werden kann.

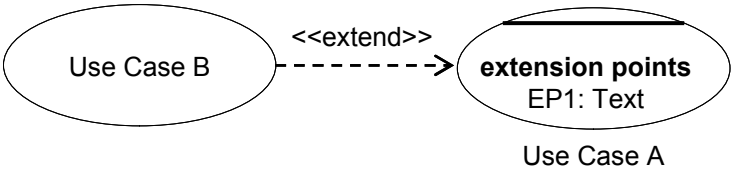


Abbildung 16: Erweiterung von Use Cases

Der Vollständigkeit halber sei noch die Generalisierungsbeziehung erwähnt (Abbildung 17): Hierbei erbt ein Use Case B das gesamte Verhalten eines Use Case A und kann dieses überschreiben und ergänzen. In der Spezifikationspraxis spielt die Generalisierung kaum eine Bedeutung.

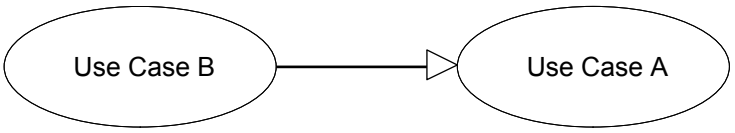


Abbildung 17: Generalisierungsbeziehung zwischen Use Cases

Diese Möglichkeiten der Modellierung werfen die Frage nach dem „richtigen“ Schnitt der Anwendungsfälle auf. Ist es besser, mehrere kleine Anwendungsfälle zu modellieren oder einen größeren Anwendungsfall ohne <<include>> Beziehung? Hier gibt es kein „richtig“ oder „falsch“, in Kapitel 3 werden wir auf diesen Punkt zurück kommen.

2.3.3 Zielgruppen und Granularität

Zielgruppen für eine Anwendungsfall-Modellierung sind der Auftraggeber eines Software-Projektes, der Konstrukteur, der Tester und der Schätzer! Diese unterschiedlichen Zielgruppen erfordern eine unterschiedliche Detaillierung der Anwendungsfälle, das gilt natürlich insbesondere für den Schätzer. Günstigerweise erfordert die Schätzung keine zusätzliche Detaillierung, sie kann und sollte auf einem rein fachlichen Level erfolgen.

Eine Anwendungsfall-Beschreibung von bzw. für einen Fachbereich (Auftraggeber) sollte grundsätzlich ausreichend Informationen für eine Schätzung des fachlichen Umfanges nach UCP enthalten. Entscheidend ist hierbei nicht der Detaillierungsgrad der Beschreibung eines einzelnen Schrittes oder auch der Dialogoberfläche, sondern die Trennung und Benennung der verschiedenen Anwendungsfälle inklusive ihrer Schritte, Szenarien, Dialoge und Druckausgaben.

Damit kann eine Grobspezifikation mit bloßer Aufzählung und kurzer Benennung dieser Komponenten sogar eine bessere (weil schneller zu erfassende) Basis für eine UCP-Schätzung als eine detaillierte Beschreibung sein.

2.4 Use Case Points (UCP)

Die Use Case Points Methode ist eine Schätzmethode, die von Gustav Karner Anfang der Neunziger Jahre in einer Diplomarbeit [Kar93] unter Betreuung von Ivar Jacobsen bei der Firma Objectory AB entwickelt wurde [Kar93a, Smi99]. Sie geht aus der Function Points Methode hervor und überträgt diese in die OO-Welt. Use Case Points und Function Points sind nicht direkt aufeinander abbildbar [Pro02]. Use Case Points wurden zunächst u.a. von Rational, Sun und IBM in einer leicht modifizierten Form eingesetzt [ADS+01].

Die UCP-Methode wurde erst in den letzten Jahren von mehreren Industrieunternehmen unabhängig voneinander aufgegriffen und weiterentwickelt [ADS+01]. Hierzu zählen unter anderem die Capgemini sd&m [FJE06], Credit Suisse [Dut05], Ericsson [MAC05] und IBM [And02]. In Abbildung 6 sind exemplarisch die im Rahmen dieser Dissertation bei Capgemini sd&m weiterentwickelten Methoden UCP 1.0 und UCP 2.0 aufgeführt. Einige kommerzielle Tool-Hersteller haben bereits die Zählweise der Use Case Points in ihren Werkzeugen aufgegriffen[9].

Zur Abgrenzung der Original-Methode von Karner von anderen UCP Weiterentwicklungen sprechen wir später auch kurz von der *Karner-Methode*. Sie wird nun detailliert vorgestellt.

2.4.1 UCP-Methode nach Karner

Die UCP-Methode wurde speziell für eine UML-basierte Software-Entwicklung konzipiert [BRJ99, Coc03, Coc95, JCJO93, McC06] und berücksichtigt die empirischen Grundlagen für eine möglichst genaue Aufwandsschätzung [Alb79, BZ07, Boe+00, ED07, Koi04, Sne05]. Dabei hat diese Methode eine ähnliche Struktur, wie die klassischen Function Point Methoden [BF04, Fra07, GH96, ISO-I03]. Als Zählmaß werden bei der UCP-Methode die *Use Cases* und *Actors* aus einer Spezifikation bestimmt. Konkret werden die folgenden Aspekte bewertet: *Anzahl Use Cases, Anzahl Actors, technische Randbedingungen (nicht funktionale Anforderungen)* sowie *Projektumgebungsfaktoren.*

Für jeden Use Case wird die Komplexität anhand der einzelnen Transaktionen klassifiziert. Die Transaktionen ähneln den Schritten in den Anwendungsfällen (siehe Abschnitt 2.3.2), eine genaue Definition des Transaktionsbegriffes gibt Karner aber leider nicht. Für die Bewertung der Komplexität wird von Karner [Kar93] eine Intervallskala vorgeschlagen, um den Umfang eines Use Cases in einzelne Komplexitäts-Klassen einzuteilen.

Ein Use Case mit einem Umfang aus ein bis drei Transaktionen heißt „einfach“ und bekommt ein Gewicht W_i von 5 Use Case Points. Ein Use Case mit einem Umfang von vier bis sieben Transaktionen im Standardablauf ist „mittel“ und bekommt 10 Punkte und ein „komplexer“ Use Case, der mehr als sieben Transaktionen hat, bekommt ein Gewicht W_i von 15 Use Case Points. Tabelle 7 fasst dies zusammen.

[9] Zum Beispiel: Sparx Enterprise Architect (www.sparxsystems.com), Cost Xpert Suite 3.5 (www.costxpert.eu), Duvessa Estimate Easy Use Case (www.duvessa.com)

Komplexität	Beschreibung	Gewicht W_i
einfach	A use case is simple if it has 3 or less transactions including alternative courses. You should be able to realise the use case with less than 5 analysis objects.	5
mittel	A use case is average if it has 3 to 7 transactions including alternative courses. You should be able to realise the use case with 5 to 10 analysis objects.	10
komplex	A use case is complex if it has more than 7 transactions including alternative courses. The use case should at least need 10 analysis objects to be realised.	15

Tabelle 7: Gewichtung der Use Case Kategorien nach Karner

Bei der Bewertung der Use Cases schlägt Karner vor, <<include>> oder <<extend>> Beziehungen von Anwendungsfällen unberücksichtigt zu lassen. Die Gründe dafür sind unklar [MAC05].

Die Gewichte W_i aller Use Cases werden summiert und ergeben die ***Unadjusted Use Case Weights (UUCW)***.

Im zweiten Schritt werden die Actors bewertet, die den Use Cases zugeordnet sind [Coc95, SW98]. Die Actors werden in drei Komplexitäts-Klassen eingeteilt:

- Actors, die über Schnittstellen mit anderen Systemfunktionen (API) kommunizieren, werden als *einfach* betrachtet und bekommen einen Use Case Point.
- Actors, die über ein Protokoll, (z.B. TCP/IP) mit einem anderen System angebunden sind, werden der Kategorie *mittel* zugeordnet und bekommen zwei Use Case Points.
- Actors, die eine Benutzerschnittstelle (GUI) nutzen, um mit dem System zu kommunizieren, werden als *komplex* eingestuft und bekommen drei Use Case Points als Gewichtungsfaktor.

Komplexität	Beschreibung	Gewicht W_i
einfach	An actor is simple if it represents another system with a defined application programming interface.	1
mittel	An actor is average if it is: 1. An interaction with another system through a protocol. 2. A human interaction with a line terminal.	2
komplex	An actor is complex if it interacts through a graphical user interface.	3

Tabelle 8: Gewichtung der Actors nach Karner

Tabelle 8 fasst die Gewichtungen zusammen. Die Gewichte W_i aller Actors werden dann summiert und ergeben die ***Unadjusted Actor Weights (UAW).*** Die Summe der Unadjusted Use Case Weights und der Unadjusted Actor Weights ergibt die ***Unadjusted Use Case Points (UUCP).***

$UUCP = UUCW + UAW$	*Formel 7*

Als letzter Schritt der Aufwandsabschätzung werden nun die *UUCP* durch einen technischen Faktor genannt *Technical Complexity Factor* (TCF) und einen Umgebungsfaktor genannt *Environmental Factor* (EF) korrigiert.

T_i	Einflussgröße	Gewicht W_i
T_1	Distributed systems	2
T_2	Application performance objectives, in either response or throughput	1
T_3	End user efficiency (on-line)	1
T_4	Complex internal processing	1
T_5	Reusability, the code must be able to be reused in other applications	1
T_6	Installation ease	0,5
T_7	Operational ease, usability	0,5
T_8	Portability	2
T_9	Changeability	1
T_{10}	Concurrency	1
T_{11}	Special security features	1
T_{12}	Provide direct access for third parties	1
T_{13}	Special user training facilities	1

Tabelle 9: Einflussgrößen des TCF mit Gewichtung nach Karner

Um TCF zu ermitteln sind einzelne Einflussgrößen wie Effizienz, Sicherheitsanforderungen usw. einzeln zu bewerten. Die jeweilige Komplexität wird anhand einer Ordinalskala von null bis fünf Punkten bewertet. Null bedeutet, dass die Einflussgröße irrelevant ist. Fünf Punkte bedeuten, dass die Einflussgröße sehr wichtig ist. Außerdem werden für die einzelnen Einflussgrößen noch unterschiedliche Gewichte W_i definiert. Tabelle 9 gibt einen Überblick über die Einflussgrößen für den TCF und deren Gewichtung. Durch die Bewertung der einzelnen technischen Einflussgrößen T_i mit [0..5] Punkten erhält man den *TCF* über folgende Formel, wobei W_i der Tabelle 9 entnommen wird:

$TCF := 0{,}58 + 0{,}01 \cdot \sum_{i=1}^{13} (T_i \cdot W_i)$	*Formel 8*

Der *TCF* liegt im Wertebereich [0,58 – 1,28].

Analog zum *TCF* wird der *EF* durch acht Einflussgrößen E_i bestimmt (siehe Tabelle 10). Der *EF* bewertet damit bestimmte Aspekte der Projektorganisation und der Projektrahmenbedingungen.

Durch die Bewertung der einzelnen Einflussgrößen E_i mit [0..5] Punkten erhält man den *TCF* über folgende Formel, wobei W_i der Tabelle 10 entnommen wird:

$EF := 1{,}405 - 0{,}03 \cdot \sum_{i=1}^{8} (E_i \cdot W_i)$	*Formel 9*

E_i	Einflussgröße	Gewicht W_i
E_1	Familiar with Rational Unified Process	1,5
E_2	Part time workers	-1
E_3	Analyst Capability	0,5
E_4	Application experience	0,5
E_5	Object oriented experience	1
E_6	Motivation	1
E_7	Difficult programming language	-1
E_8	Stable requirements	2

Tabelle 10: Einflussgrößen des EF mit Gewichtung nach Karner

Man beachte das negative Vorzeichen vor dem Summenterm in Formel 9. Dadurch verkleinern Einflussgrößen mit fünf Punkten (= sehr wichtig) den *EF*, wohingegen null Punkte zu einem größeren *EF* führen. Allerdings sind die Gewichte W_2 und W_7 negativ gewählt, wodurch diese Werteskala für diese beiden Einflussgrößen wieder herumgedreht wird. Aufgrund dieser Effekte liegt *EF* im Wertebereich [0,430 – 1,705].

Die ***Adjusted Use Case Points (AUCP)*** berechnen sich damit zu:

AUCP = UUCP TCF* EF*	*Formel 10*

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

*PE : = AUCP * PF*	*Formel 11*

2.4.2 Bewertung der UCP-Methode nach Karner

Das Zählmaß der UCP-Methode setzt mit den Actors und Use Cases unmittelbar auf den Zählgrößen auf, die in einer Use Case basierten Spezifikation verfügbar sind. Damit ist es nicht notwendig, für die fachliche Größenbestimmung (UUCP) einen technischen Lösungsentwurf vorweg zu nehmen, wie dies in Teilen für die Function Point Methode in Abschnitt 2.2 analysiert wurde. Die UCP-Methode ohne TCF und EF ist damit leicht anwendbar für Personen, die die fachliche Beschreibung des zu entwerfenden Software-Systems gut kennen. Aus ersten Erfahrungen im Praxis-Einsatz konnte diese Aufgabe auch von Nicht-Informatikern gut bewältigt werden [MAC05, FJE06]. Der zeitliche Aufwand für die Durchführung einer UCP-Schätzung kann mit

einem Tag für ein Projekt in der Größenordnung von 10 Bearbeiterjahren (16.000 Stunden) angesetzt werden, wenn die Fachlichkeit bzw. Spezifikation gut bekannt ist [And02, FJE06]. Die Schätzung wurde dabei durch einen mit der UCP-Methode vertrautem Schätzer und dem Projektleiter durchgeführt. Für noch größere Projekte steigt der Schätzaufwand unterproportional, da die Komplexitätsbewertung der Use Cases dann von Teilen der Spezifikation auf das Gesamtvorhaben extrapoliert werden kann. Für sehr große Vorhaben (85 Bearbeiterjahre) stieg der Aufwand auf gut zwei Tage für die Durchführung der Schätzung [FE08a].

Für die Bestimmung des *TCF* ist technisches Verständnis notwendig. In einem gegeben Projektkontext mit etablierter Technologie und wenig Änderungen hinsichtlich der Kriterien in Tabelle 9 wird sich dieser Faktor allerdings kaum ändern und kann einmalig von Technologie-Experten festgelegt werden. Gleiches gilt für den *EF* mit Bezug auf die Umgebungsfaktoren. Diese könnten einmal von einem Experten festgelegt werden. Dann ist es durchaus möglich, dass für Folgestufen oder fachliche Erweiterungen an einem bestehenden System eine Aufwandsschätzung durch den Fachbereich eines Unternehmens durchgeführt wird.

In der Literatur findet sich eine Reihe von Kritikpunkten an der UCP-Methode. Sie werden nachfolgend zusammengefasst und als *Problempunkte* zwecks besserer Referenzierbarkeit nummeriert:

P 1 Unterschiedlicher Schnitt der Use Cases

Ein wesentlicher Kritikpunkt an der UCP-Methode ist, dass Use Cases in unterschiedlicher Granularität beschrieben werden können [Boe81, Coc03, Smi99, And02] und dies unmittelbar Einfluss auf das Schätzergebnis hat [ADS+01, FJE06]. Damit droht die Gefahr, dass Systeme gleicher fachlicher Größe je nach Spezifikation in unterschiedlichen Use Case Points Werten resultieren.

Smith [Smi99] schlägt die Levels 0 – *Class*, 1 – *Subsystem*, 2 – *SubsystemGroup*, 3 – *System*, 4 – *SystemOfSystems* vor. Für jeden Level wird die Gesamtzahl der „nach außen sichtbaren“ (external) Use Cases auf ca. 10 festgelegt. Dies soll helfen, die Spezifikation übersichtlich zu halten. Für jeden Level schlägt Smith folglich einen anderen Produktivitätsfaktor vor, um die unterschiedliche Granularität der Use Cases je Level zu berücksichtigen. Diese Vorgehensweise schafft zwar übersichtliche Spezifikationen, schafft aber keine Klarheit hinsichtlich der Größe eines einzelnen Use Cases und ist aus dem Blickwinkel der Aufwandsschätzung nicht praktikabel.

Diese Einschätzung teilen Ouwerkerk und Abran [OA06] und fordern eine exakte Definition der Größe eines Use Cases bzw. für das verwendete Punktmaß. Damit wird ein Standard gefordert, wie Use Cases beschrieben werden, um ein einheitliches Größenverständnis für Use Cases zu entwickeln.

P 2 Reuse Schätzfehler

Karner schlägt in der Original-Methode vor, <<include>> Beziehungen und <<extend>> Beziehungen zu ignorieren. Die Gründe dafür bleiben unklar [MAC05]. Dies birgt die Gefahr, dass große Teile an Funktionalität nicht mit berücksichtigt werden [HGS+05]. Insgesamt fehlt ein Konzept, wie die Wiederverwendung von Use Cases zu berücksichtigen ist.

P 3 Actor-Gewichte zu gering

Die Gewichtung der Actors erfolgt mit [1..3] Punkten und hat damit nur ein Fünftel des Gewichts eines Use Cases. Analysen größerer Software-Entwicklungsprojekte haben gezeigt, dass damit der Einfluss des Actors im Sinne einer Metrik auf das Schätzergebnis vernachlässigbar ist. Dies steht im Widerspruch zur Erfahrung von Software-Ingenieuren aus der Praxis [MAC05, FJE06]. Das Unternehmen Credit Suisse z.B. bewertet Actors dreieinhalb Mal stärker als die Karner-Methode [Bad08].

P 4 Kaum relevante vergleichbare Praxiserfahrung verfügbar

In der Literatur sind bisher nur wenige Projektdaten zu Use Case Points Schätzungen veröffentlicht worden. Meistens handelt es sich dabei um kleinere Projekte mit bis zu zwei Bearbeiterjahren Aufwand [ADS+01]. Neuere Studien greifen erstmals auch große Projekte mit entsprechenden Release-Zyklen auf [MAC05], verfügen aber nur über die Daten eines einzigen sehr großen Projektes, womit die Vergleichbarkeit zu anderen Studien schwierig wird, da die Methode ggü. Karner modifiziert wurde. Andere Studien greifen auf Projektdaten zurück, die an Universitäten durch Studierende durchgeführt wurden [Coe08]. Die größte öffentlich verfügbare unabhängige Projektdatenbank ist die der ISBSG[10] mit knapp 5.000 erfassten Projekten. Davon sind aber nur ca. 1.500 Neuentwicklungen und es dominiert IFPUG FPA für die Größenmessung, Werte für die UCP-Methode sind kaum vorhanden, Angaben zu den konkret verwendeten Zählregeln sind nicht verfügbar.

Die Fachliteratur liefert bisher wenig Hinweise auf den industriellen Einsatz der UCP-Methode. Die Industrieunternehmen lassen sich nicht gerne „in die Karten schauen“. Aufgrund der Zusammenarbeit mit dem Software- und Beratungshaus Capgemini sd&m besteht allerdings für den deutschsprachigen Markt ein guter Überblick hinsichtlich der 100 größten Industrieunternehmen. Bekannt ist daher, dass die UCP-Methode in abgewandelter Form in folgenden Unternehmen eingesetzt wird: Capgemini Gruppe weltweit [FJE06], Credit Suisse [Dut05, Bad08], Ericsson [MAC05], IBM [And02], Sparkassen Informatik GmbH & Co. KG seit 2002 in Verbindung mit der Methode COCOMO II [Jer03], O2 Germany seit 2002 vereinzelt. Weitere Details sind nicht bekannt. Eine Internetrecherche von Ende 2007 erbrachte keine weiteren Erkenntnisse darüber hinaus. Projektdaten zu Forschungszwecken konnten für diese Dissertation nur von Capgemini sd&m einbezogen werden.

P 5 UCP Schätzmodell verwendet unzulässige Skalen-Transformationen

Habra [HAL+08] erinnert daran, dass es unterschiedliche Typen von mathematischen Skalen gibt und Skalen aus algebraischen Gründen nicht beliebig von einer Skala in eine andere transformiert werden dürfen. Die gebräuchlichen Skalentypen sind *Nominal* (Werte, die in keiner Reihenfolge zueinander stehen), *Ordinal* (Werte stehen in Reihenfolge zueinander), *Intervall* (es gibt einen definierten relativen Abstand zwischen den Werten) und *Verhältnis* (absolute Intervalle der Werte; der Null-Wert ist definiert).

[10] www.isbsg.org, Non-Profit-Organisation, Stand November 2008

Diese konzeptionelle Diskussion ist Mitte der 90er Jahre bereits in die Function Points Methode eingebracht worden und steht für die UCP-Methode noch aus. Jeder Komplexitätsfaktor im TCF und EF der Karner-Methode wird durch einen Punktwert [0..5] bestimmt, ohne das klar ist, um welchen Skalentyp es sich handelt. Ferner wird der TCF (Formel 8) wie auch der EF (Formel 9) durch Summation (Ordinal-Skalentyp) gebildet. Ouwerkerk und Abran [OA06] fordern hierfür aber aus algebraischen Gründen eine Produktformel (Verhältnis-Skalentyp). Damit würde vermieden, dass z.B. die Use Case Komplexität (=Verhältnis-Skalentyp) durch den TCF bzw. EF in einen Ordinal-Skalentyp transformiert wird, wohingegen der Projektaufwand PE wieder als Verhältnis-Skalentyp verstanden wird.

P 6 Definition von TCF und EF sind nicht mehr zeitgemäß

Die Faktoren des EF und TCF wurden seit der Entstehung der Methode nicht weiterentwickelt. Viele Erkenntnisse, die insbesondere in COCOMO II eingeflossen sind, blieben bisher unberücksichtigt. So liegt z.B. der Komplexitätsfaktor *EF* im Wertebereich [1,2 – 1,7] und ist damit um ein Vielfaches geringer als entsprechende Komplexitätsfaktoren in COCOMO II (siehe später in Abschnitt 2.5). Damit werden Einflussgrößen unter Umständen nur unzureichend berücksichtigt. Erste eigene Untersuchungen zeigen zudem, dass die Wirkzusammenhänge von **Organisation und Vorgehen** auf die Projektaufwände noch nicht ausreichend berücksichtigt sind und weitere Analysen erforderlich sind [FJE06]. Der TCF ist dem technischen Fortschritt anzupassen: So wird zum Beispiel die *Portabilität* (bei TCF T_8) mit einem Gewicht von 2 bewertet und ist damit der Faktor, der im TCF am stärksten gewichtet ist. Durch Plattform-unabhängige Programmiersprachen, wie zum Beispiel Java, scheint die hohe Gewichtung in der heutigen Zeit fraglich. Ein weiteres Beispiel aus dem EF belegt den Überarbeitungsbedarf: In *Object oriented experience* (F_5) wird der Anfang der 90er Jahre geringen Verbreitung der Objektorientierten Methoden Rechnung getragen, was heute state of the art der Grundausbildung jedes Software-Ingenieurs ist.

P 7 Komplexitätsstufen der TCF und EF sind zu vage definiert

Die Kostenfaktoren TCF und EF sind nicht nur veraltet, es ist zudem nicht präzise definiert, wie viel Punkte bei welcher Ausprägung je Einzelfaktor zu vergeben sind. Damit wird das Ergebnis subjektiv und führt in der Konsequenz zu einem Mangel an Reproduzierbarkeit der UCP-Schätzungen. Dies wurde auch bereits von Ouwerkerk in [OA06] kritisiert. IFPUG hat hier z.B. detaillierte Evaluationskriterien im Counting Practise Manual (*CPM*) festgelegt und auch COCOMO II liefert präzisere Vorgaben.

P 8 Interpretationsspielraum von Use Cases

Es ist unklar, wie viel Erfahrung für die Durchführung einer UCP-Schätzung tatsächlich benötigt wird. Die Use Cases geben eine fachliche Beschreibung der Anforderungen und der fachlichen Lösungsidee (Abschnitt 2.3.2). Diese wird erst in einer späteren Projektphase in ein technisches Lösungsmodell transformiert. Es ist unklar, ob bei dieser Transformation nicht erhebliche Aufwandsunterschiede resultieren, die möglicherweise auch von der technischen Erfahrung der Person abhängen, welche die Schätzung durchführt. Ein erfahrener Software-Entwickler ahnt möglicherweise eine andere Komplexität in der späteren Umsetzung eines Use Cases und bewertet diesen daher anders als ein Nicht-

Informatiker, der nur die Fachlichkeit bewertet. Ouwerkerk [OA06] kritisiert in diesem Zusammenhang z.B. auch, dass eine graphische Benutzeroberfläche dreimal so stark gewichtet wird wie eine Anwendungsschnittstelle (API), ohne dass es hierfür dokumentierte empirische Daten gibt.

P 9 Umfang der durch UCP abgedeckten Projektaufwände

Die UCP-Methode liefert im Ergebnis einen Aufwand in Bearbeiterstunden. Was genau ist aus dem Kontext eines Entwicklungsprojektes damit geschätzt? Gehört z.B. die Inbetriebnahme und Gewährleistungsverpflichtung mit zum geschätzten Aufwand oder nicht? Es fehlt ein klarer Aufgabenrahmen der mit der Schätzung abgedeckten Projektaufwände.

Entgegen dieser Problempunkte konnten in der Literatur [ADS+01, MAC05] als auch durch eigene erste Studien [FJ06, FJE06] eine teilweise vielversprechende Übereinstimmung von UCP-Schätzungen mit den tatsächlich eingetretenen Projektaufwänden nach Projektabschluss aufgezeigt werden. Anda zeigte zudem an einem singulären Beispiel, dass die UCP-Methode sogar genauer sein kann als eine Expertenschätzung [And02].

Es erscheint daher lohnend, hinsichtlich der zuvor genannten Problempunkte zu prüfen, ob und durch welche Veränderung der UCP-Methode diese Punkte ausgeräumt werden könnten.

Als besonders vielversprechend wird dabei eine Berücksichtigung der neueren Entwicklungen im Bereich der Einflussgrößen sein, wie sie in den Arbeiten von Boehm in der Entwicklung von COCOMO (veröffentlicht in 1981) zu COCOMO II (veröffentlicht in 2000) vollzogen wurden. Daher wird zunächst COCOMO im Detail in Abschnitt 2.5 vorgestellt, dann ein Abgleich der bekannten Schätzmethoden mit der Problemstellung durchgeführt (Abschnitt 2.6) und darauf aufbauend eine verfeinerte Aufgabenstellung für diese Dissertation in Abschnitt 2.7 beschrieben.

2.5 COCOMO

Das COCOMO (COnstructiveCOstMOdel) wurde von Barry W. Boehm, einem Software-Ingenieur bei Boeing, entwickelt und 1981 veröffentlicht [Boe81]. Ausgangsgröße des COCOMO sind Codezeilen gemessen in KDSI[11] (Kilo Delivered Source Instructions). Ein KDSI entspricht 1.000 Anweisungen geschriebenen Codes im fertigen Produkt. Kommentare und Hilfs- oder Testprogramme werden hierbei nicht gezählt.

$PE = m \cdot KDSI^{n}$	*Formel 12*

Boehm berechnete mit der Formel 12 den Projektaufwand *PE* in Personenmonaten. Der Faktor *m* und die Metrikzahl *n* beruhen auf Erfahrungswerten einfacher („organic mode“), mittelschwerer („semi-detached“) oder komplexer („embedded“) Projekte und sind in Tabelle 11 aufgelistet.

[11] Bezeichnung des Autors in [Boe81]. Teilweise werden auch andere Bezeichnungen genutzt.

Projektart	*m*	*n*	Definition
Einfach	2,4	1,05	kleines Team; geringer Zeitdruck; bekannte Werkzeuge, Arbeitsumgebung und Hardware; bis 50 KDSI
mittelschwer	3	1,12	nicht alle unter einfach genannten Punkte treffen zu; bis 300 KDSI
komplex	3,6	1,2	unbekannte Umgebung und Hardware, Echtzeitanforderungen; komplexe Rahmenbedingungen, > 300 KDSI

Tabelle 11: Prognose oder Messung des Funktionsumfangs [Boe81]

Bis heute wurde COCOMO mehrmals überarbeitet und verbessert. Das neueste Modell COCOMO II ist weit verbreitet und wird in den verschiedensten Branchen und Industriezweigen genutzt. Es wird nachfolgend detailliert vorgestellt.

COCOMO II wurde von Boehm im Sommer 2000 veröffentlicht [Boe+00] und dient zur Aufwandsschätzung von Projekten des Typs Neuentwicklung als auch Wartung. Ziel des neuen Modells ist es, die Schätzgenauigkeit zu verbessern und die aufgekommenen Trends, wie objektorientierte Programmierung, grafische Oberflächen (die mit grafischen Editoren halbautomatisch erzeugt werden), Wiederverwendung von Software durch Frameworks und die Nutzung von Middleware angemessen zu berücksichtigen.

Zusätzlich zur Aufwandsabschätzung ist es mit COCOMO II möglich, Kosten- und Zeitpläne zu erstellen. Außerdem sind Modelle verfügbar, die spezielle Projektmodelle unterstützen. Diese Optionen sind für diese Arbeit nicht relevant und werden daher nicht weiter betrachtet.

COCOMO II basiert wie das ältere COCOMO auf der Programmgröße, z.B. gemessen in *KSLOC*[12] (Kilo Source Lines of Code), die identisch zu KDSI gezählt wird. Weiterhin kommen Faktoren zum Einsatz, die exponentiell oder linear in die Berechnungen mit eingehen. Die exponentiellen Faktoren heißen *Scaling Drivers* und die linearen Kostenfaktoren werden *Effort Multipliers* genannt.

$$PE := A \cdot Size^{E} \cdot \prod_{i=1}^{17} EM_i \qquad \textit{Formel 13}$$

Ausgangsbasis zur Berechnung des Projektaufwandes (*PE*) in Bearbeitermonaten ist die Formel 13, wobei *A* auf Erfahrungswerten beruht und von Boehm mit 2,94 vorgeschlagen wird, *Size* die Größe des Programms in KSLOC oder *uFP* (ungewichtete Function Points, 2.1.1) ist, der Exponent *E* sich durch die Scaling Drivers ergibt und EM_i für die Effort Multipliers steht.

[12] Bezeichnung des Autors in [Boe++00]. Teilweise werden auch andere Bezeichnungen genutzt.

Scaling Drivers

Die Scaling Drivers gehen exponentiell in den Aufwand ein und werden mit der Formel 14 errechnet. Dabei ist *B* eine Konstante, die empirisch zu 0,91 bestimmt wurde und SF_j sind fünf *Scale Factors*.

$$E = B + 0{,}01 \cdot \sum_{i=1}^{5} SF_i \qquad \textit{Formel 14}$$

Die Scale Factors beschreiben die Vor- und Nachteile, die sich durch die Größe der Systemlösung ergeben. Jeder Scale Factor kann mit *VL* (very low), *L* (low), *N* (normal), *H* (high), *VH* (very high) und *XH* (extreme high) sechs mögliche Ausprägungen annehmen. Jeder Ausprägung ist ein Zahlenwert zugeordnet. Dieser Zahlenwert ist Tabelle 12 zu entnehmen und als SF_j in Formel 14 einzusetzen, um den Exponenten *E* zu berechnen.

SF_i	COCOMO-Kürzel	Scale Factor / Beschreibung	Gewichtung VL	L	N	H	VH	XH
SF_1	PREC	Precedentedness / Vertrautheit Ähnlichkeit der Fachlichkeit zu bereits existierenden Systemen? Faktoren, die zu beachten sind: Generelles Verständnis der Zielsetzung, Erfahrung mit der Funktion ähnlicher Systeme, involvierte Hardware oder betriebliche Funktionalität ist noch in Entwicklung, innovative Ablaufsteuerungen oder Algorithmen.	6,2	4,96	3,72	2,48	1,24	0
SF_2	FLEX	Development Flexibility / Flexibilität der Entwicklung Wie flexibel kann entwickelt werden bezüglich technischer Anforderungen? Faktoren, die zu beachten sind: Konformität der Software mit technischen Anforderungen und Spezifikationen von Schnittstellen.	5,07	4,05	3,04	2,03	1,01	0
SF_3	RESL	Architecture/Risk Resolution / Architektur und Risikoausschluss Wie gründlich war das Review der Grobspezifikation, und wie gut sind Punkte bekannt, die mit einem Risiko belastet sind? Faktoren, die zu beachten sind: Identifikation mögliche Risiken, stimmiger Budget- und Zeitplan, Aufwand zur Erstellung einer technische Architektur, Unterstützung durch Tools für Design, Sicherheitsfaktor für Technologie und Hardware.	7,07	5,65	4,24	2,83	1,41	0
SF_4	TEAM	Team Cohesion / Teaminteraktion Wie gut funktioniert das Team? Bewertet den Zusammenhalt und die Kooperation aller beteiligten Stakeholder (Auftraggeber, Anwender, Entwickler, Projektmanagement, usw.).	5,48	4,38	3,29	2,19	1,1	0
SF_5	PMAT	Process Maturity / Prozessreife Bewertet die Reife des Entwicklungsprozesses, die Laufzeiten der Prozesse und die Produktivität.	7,8	6,24	4,68	3,12	1,56	0

Tabelle 12: Scaling Drivers, Übersetzung aus [Bad08]

Bedingt durch die vorgegebenen Werte der Scale Factors liegt *E* im Bereich zwischen 0,91 (alle Scale Factors sind mit „extreme high“ bewertet) und 1,226 (alle Scale Factors sind mit „very low“ bewertet).

Auffallend ist, wie die einzelnen Zahlenwerte eines Scale Factors gebildet sind: Der Zahlenwert für *XH* ist stets Null und somit in der Formel 14 neutral. Der Wert für *VH* schwankt zwischen

1,01 und 1,56 und kann als Ausgangswert betrachtet werden, da die Werte für *H* stets genau doppelt so groß sind. Die Werte für *N* entsprechen dem dreifachen, *L*-Werte sind viermal so groß und die *VL*-Werte betragen das Fünffache des Ausgangswertes *VH*. Je größer dieser Zahlenwert ist, desto größer ist die Auswirkung im Ergebnis der Formel: Der Scale Factor *FLEX* ist der am niedrigsten gewichtete Faktor, *PMAT* geht am stärksten in das Ergebnis ein.

Effort Multipliers

Die Effort Multipliers beschreiben die Produktivität, mit der gearbeitet wird. COCOMO II definiert 17 Effort Multipliers, die linear in den zu errechnenden Aufwand eingehen und in Tabelle 13 gelistet sind. Jeder Effort Multiplier kann mit *VL, L, N, H, VH* und *XH* bis zu sechs mögliche Ausprägungen annehmen, teilweise wird dieser Bereich aber nur mit vier oder fünf Kombinationen ausgeschöpft. Auch hier wird jeder möglichen Ausprägung ein Zahlenwert zugeordnet, um diesen als EM_i in die Formel 13 einzusetzen.

Die möglichen Werte der Effort Multipliers bewegen sich zwischen 0,71 und 1,74. Niedrige Werte stehen für eine hohe Produktivität und somit für einen geringen Aufwand. Abhängig von der Fragestellung finden sich die niedrigsten Werte auf der Seite von *VL* oder *XH*. In Formel 13 gehen die Effort Multipliers als Produkte ein: Nimmt jeder Effort Multiplier den niedrigsten Wert an, ergibt sich der Wert 0,0569. Als höchster Wert kann maximal 115,58 erreicht werden [Bad08].

Zusammenfassung

Als Zählgröße werden in COCOMO *KDSI* bzw. *KSLOC* vorgeschlagen. Eine Ermittlung dieser Maße durch eine Expertenschätzung ist als praktisch zu aufwändig einzuschätzen, vielmehr verwendet man in der Praxis andere Metrik-basierte Methoden wie z.B. eine FSM-Methode (Abschnitt 2.2). Boehm schlägt *uFP* (ungewichtete Function Points gemäß IFPUG FPA) vor. Mit Hilfe von sogenannten *Gearingfaktoren* werden die gezählten Function Points (siehe z.B. Tabelle 5) in *KDSI* bzw. *KSLOC* umgerechnet.

Boehm unterteilt die Effort Multipliers in die vier Bereiche Produkt (*Product Factors*), Plattform (*Plattform Factors*), Mitarbeiter (*Personell Factors*) und Projekt (*Project Factors*) - wie in Tabelle 13 dargestellt. Bei den Scale Factors hat Boehm diese Gruppierung nicht vorgenommen. Anders dagegen Dumke: Er ordnet die Scale Factors *PREC*, *FLEX*, *RESL* dem Bereich *Project Factors* zu, *TEAM* den *Personell Factors* [Dum01].

Mit seinen fünf exponentiellen und 17 linearen Kostenfaktoren ist COCOMO II ein umfangreiches Modell zur Berücksichtigung ausschließlich nichtfunktionaler Anforderungen. Es wurde auf Basis der Daten von 161 Projekten kalibriert. Die Kostenfaktoren können ein Vielfaches des ursprünglichen Aufwandes bewirken. Ohne Weiteres ist aus der Praxis nicht nachvollziehbar, warum durch die linearen Kostenfaktoren der Aufwand bei entsprechender Bewertung auf nur 5,69 % reduziert bzw. um das rund 116fache gesteigert werden kann. Bei COCOMO kann die exponentielle Einflussgröße aus den Klassen *einfach* (1,05), *mittelschwer* (1,12) und *komplex* (1,2) gewählt werden. Bei COCOMO II berechnet sich die exponentielle Einflussgröße durch fünf Einzelwerte, die jeweils in sechs Stufen bewertet werden können. Unter der Annahme, dass

niemals der untere Extremwert (0,91) oder der obere Extremwert (1,226) erreicht wird, bewegen sich die exponentiellen Größen bei COCOMO und COCOMO II etwa im selben Bereich.

Bereich	EM_i COCO-MO-Kürzel	**Effort Multiplier** Beschreibung	Gewichtung VL	L	N	H	VH	XH
Produkt	EM_1 RELY	Required Software Reliability / Sicherheit Wie zuverlässig muss die Software sein? Wenn die Auswirkung eines Fehlers nur leichte Unannehmlichkeiten bedeutet, ist RELY niedrig, bei Gefahr für Menschen hoch.	0,82	0,92	1	1,1	1,26	-
	EM_2 DATA	Database Size / Datenbankgröße Wie groß ist die Datenbank in Bytes/Programm Lines of Code? Ein größerer Datenbestand verursacht aufwändigere Testdaten.	-	0,9	1	1,14	1,28	-
	EM_3 CPLX	Product Complexity / Komplexität der Anwendung Gewichteter Mittelwert aus der Bewertung der Funktionen: Anwenderschnittstellen-, Berechnungs-, Kontroll-, Datenmanagement- und hardwareabhängige Funktionen.	0,73	0,87	1	1,17	1,34	1,74
	EM_4 RUSE	Developed for Reusability / Wiederverwendbarkeit Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes im selben System oder als Framework in anderen Projekten?	-	0,95	1	1,07	1,15	1,24
	EM_5 DOCU	Documentation Match to Life-Cycle Needs / Anforderungen an Dokumentation Wie aufwändig ist die Dokumentation (auch innerhalb der einzelnen Projektphasen) für uns bzw. als Lieferleistung für den Kunden?	0,81	0,91	1	1,11	1,23	-
Plattform	EM_6 TIME	Execution Time Constraint / Auslastung d. Rechenzeit Wie stark ist die Rechenleistung auf der Zielumgebung ausgelastet?	-	-	1	1,11	1,29	1,63
	EM_7 STOR	Main Storage Constraint / Auslastung der Speicherkapazität Wie stark ist das Storage-System auf der Zielumgebung ausgelastet?	-	-	1	1,05	1,17	1,46
	EM_8 PVOL	Platform Volatility / Häufigkeit der Plattformwechsel Wie oft wird die Hard- und Software (auch Datenbank oder Middleware) ausgetauscht?	-	0,87	1	1,15	1,3	-
Mitarbeiter	EM_9 ACAP	Analyst Capability / Leistung des Chefdesigners Wie hoch ist das Leistungsvermögen der Analysten (hierzu zählt nicht die Erfahrung der Chefdesigner, diese ist in APEX enthalten)?	1,42	1,19	1	0,85	0,71	-
	EM_{10} PCAP	Programmer Capability / Leistung der Programmierer Wie hoch ist das Leistungsvermögen der Programmierer im Team (hierzu zählt nicht die Erfahrung der Programmierer, diese ist in APEX enthalten)?	1,34	1,15	1	0,88	0,76	-
	EM_{11} PCON	Personal Continuity / Kontinuität der Mitarbeiter Wie hoch ist die Kontinuität der Mitarbeiter im Projekt?	1,29	1,12	1	0,9	0,81	-
	EM_{12} APEX	Application Experience / Erfahrung über die Anwendung Wie viel Erfahrung hat das Team (Analysten und Programmierer) mit der Fachlichkeit der zu entwickelnden Software?	1,22	1,1	1	0,88	0,81	-
	EM_{13} PLEX	Platform Experience / Erfahrung mit der Plattform Wie viel Erfahrung hat das Team über die zu nutzende Plattform und deren Middleware?	1,19	1,09	1	0,91	0,85	-
	EM_{14} LTEX	Language and Tool Experience / Erfahrung mit Programmiersprache und Tools Wie viel Erfahrung hat das Team mit der Programmiersprache und den Werkzeugen für Analyse und Darstellung von Anforderungen und Entwürfen, Konfigurationsmanagement, Dokumentationsgenerierung, Bibliotheksverwaltung, Programmstil und Formatierung?	1,2	1,09	1	0,91	0,84	-
Projekt	EM_{15} TOOL	Use of Software Tools / Tool-Einsatz Wie gut ist die Unterstützung der Entwickler durch Tools, und wie gut sind diese in die Prozesse eingebunden?	1,17	1,09	1	0,9	0,78	-
	EM_{16} SITE	Multisite Development / Verteiltes Arbeiten Wird standortübergreifend entwickelt? Welche Möglichkeiten zur Kommunikation stehen zur Verfügung? Bei verteilten Teams mit geringen oder aufwändigen Möglichkeiten zur Kommunikation wird SITE mit niedrig eingestuft, bei Arbeiten an einem Standort mit hoch.	1,22	1,09	1	0,93	0,86	0,8
	EM_{17} SCED	Required Development Schedule / Zeitplan Steht ausreichend Zeit für die Projektabwicklung zur Verfügung, wird SCED mit hoch eingestuft, bei Zeitdruck mit niedrig.	1,43	1,14	1	1	1	-

Tabelle 13: Effort Multipiers, deutsche Übersetzung aus [Bad08]

2.6 Abgleich der bekannten Schätzmethoden mit der Problemstellung

Als Erstes werden die bisher in den Kapiteln 1 und 2 entwickelten Anforderungen an eine Schätzmethode für betriebliche Informationssysteme um einen weiteren strukturellen Aspekt ergänzt. Dieser Aspekt ergibt sich auf dem Hintergrund der in den vorherigen Abschnitten vorgestellten Schätzmethoden (FSM, COCOMO). Anschließend werden alle Anforderungen zusammengefasst und die bekannten Schätzmethoden hinsichtlich ihrer Eignung diskutiert werden.

Bei der Entwicklung von betrieblichen Informationssystemen hat es sich bewährt, die Geschäftslogik von der technologischen Sicht zu trennen. Die Methode *Quasar* beschreibt dies: Dazu werden aus der fachlichen Anforderungsbeschreibung die Use Cases extrahiert und daraus eine A-Architektur abgeleitet [Sie04, Hum04]. Getrennt davon werden die technologischen Anforderungen in der T-Architektur und TI-Architektur modelliert. Darauf aufbauend führen Experten wie in Abschnitt 1.2 beschrieben eine Bottom-Up Schätzung durch.

Es stellt sich die Frage, wie die Qualität dieser Schätzungen auf Management-Ebene überprüft und nachvollziehbar gemacht werden kann, ohne den Aufwand für die Schätzung deutlich zu steigern. Hier wird das Instrument der Angebotsfreigabe vorgeschlagen: In Anlehnung an das Reifegradmanagement der Automobilindustrie wird ein Reifegradmanagement im Softwarebau [BEHS07] vorgeschlagen und damit *Quality Gates* (QG) im Software-Entwicklungsprozess etabliert. Abbildung 18 erweitert Abbildung 10 um solche Quality Gates. Diese Quality Gates haben Review-Charakter und eine Freigabe wird nur bei Erreichen eines bestimmten Benchmarks erteilt. Innerhalb des Quality Gates *Angebotsabgabe* (*QG Angebot* in Abbildung 18) ist die Expertenschätzung durch eine separate „Kontroll-“ Schätzung zu überprüfen. Für diese Kontroll-Schätzung wird eine geeignete Top-Down Methode inklusive Schwellwertdefinition (Benchmark) für die tolerierte Abweichung benötigt. Diese Top-Down Methode stellt damit Metriken für die Praxis bereit und verankert Qualitäts-Prüfpunkte schon im Angebotsprozess eines Software-Hauses.

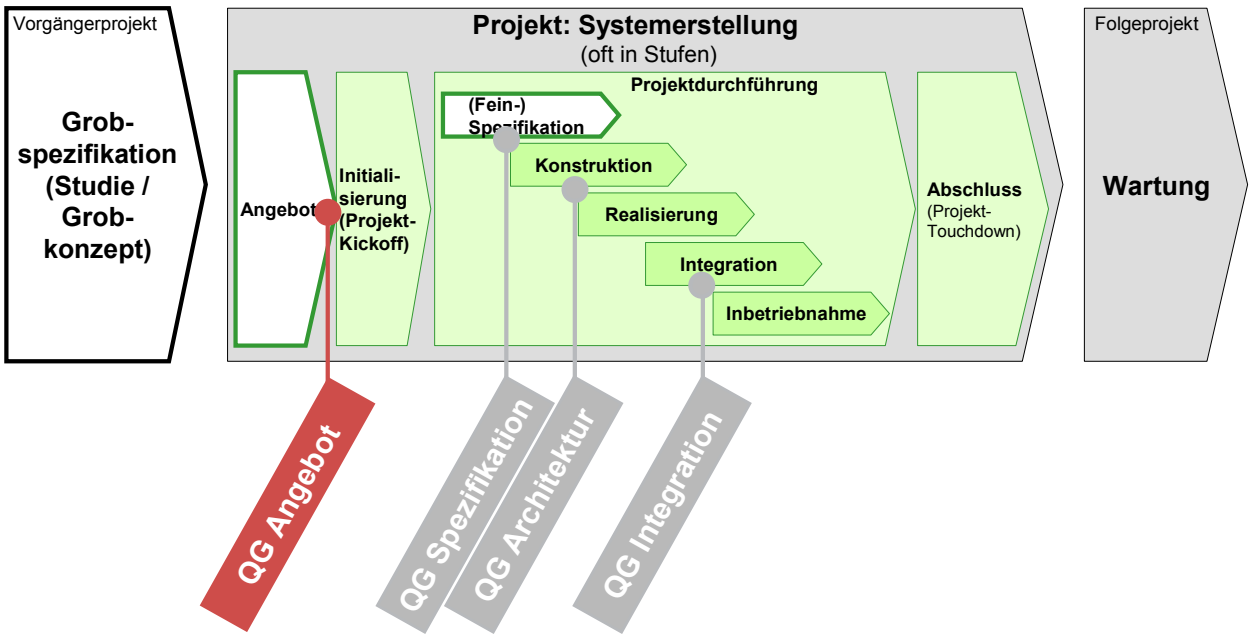


Abbildung 18: Quality Gate Prozess für den Softwarebau

Da der Aufwand eines Softwareentwicklungsprojektes durch sehr viele Faktoren bestimmt ist, muss für die Aufwandsschätzung (Kontroll-Schätzung) eine wirtschaftlich sinnvolle Vorgehensweise und Beschränkung auf die dominierenden Einflussgrößen gefunden werden. Aufgrund der zuvor dargestellten Aspekte leiten wir folgende Anforderung an die Schätzmethode aus dem Quality Gate Prozess ab:

A 7:	Die Schätzmethode trennt zwischen fachlicher Größenbestimmung und technischer Komplexitätsbewertung. Die Größenbestimmung stellt damit keine technischen Anforderungen an den fachlichen Schätzer und ist damit auf Management-Ebene bzw. durch Nicht-Informatiker anwendbar.

Hinzu kommen die sechs bereits formulierten Anforderungen:

A 1: Die Schätzung muss zu einem sehr frühen Projektzeitpunkt auf Basis einer Grobspezifikation durchführbar sein. (Seite 2)

A 2: Die Schätzmethode muss auch in neuem Umfeld funktionieren. (Seite 2)

A 3: Möglichst schlanke Schätzmethode mit einem deutlich geringeren Durchführungsaufwand als bei einer Bottom-Up Schätzung. (Seite 5)

A 4: Praxiserprobte Methode, die mit möglichst wenig geschätzten Größen gemäß Formel 1 auskommt. (Seite 6)

A 5: Ausgereifte Methode, die dem aktuellen Stand der Technik entspricht. (Seite 14)

A 6: Direkte Ableitung der zu schätzenden (funktionalen) Größen unmittelbar aus einer Grobspezifikation mit einer Use Case orientierten Sicht. (Seite 16)

Es stellt sich nun die Frage, welche Schätzmethode oder welche Kombination an Methoden für den wachsenden Markt der Individual-Software-Entwicklung von betrieblichen Informationssystemen am besten geeignet ist.

Wir hatten bereits dargelegt, dass prinzipiell im neuen Umfeld eine Expertenschätzung den Ansatz der Wahl repräsentiert, der in jedem Umfeld und unter allen erdenklichen Randbedingungen einsetzbar ist. Zur Erhöhung der Schätzqualität dieser Expertenschätzung sollte nun eine Top-Down Methode ausgewählt werden, welche deutlich weniger Zeitaufwand für die Durchführung im Vergleich mit einer Expertenschätzung benötigt. Eine Bewertung der Schätzmethoden hinsichtlich der Anforderungen ist in Tabelle 14 zusammengefasst.

Die in Kapitel 2 vorgestellten Methoden wurden bereits so ausgewählt, dass sie die Anforderungen *A 1* und *A 2* erfüllen. Prinzipiell wird auch Anforderung *A 3* erfüllt, allerdings gibt es graduelle Unterschiede hinsichtlich des Zeitbedarfes für die Durchführung der Methode (siehe Kapitel 2.2 die Anmerkungen zum Zeitbedarf). In Tabelle 14 sind jene Methoden in der Zeile zu *A 3* mit 0 bewertet, die einen deutlich höheren Zeitbedarf wegen höherer Komplexität der Schätzmethode benötigen als zum Beispiel COCOMO.

Anforderung *A 4* wird insbesondere von den gemäß ISO/IEC standardisierten FSM-Methoden[13] gut hinsichtlich der Praxiserprobung erfüllt, die anderen FSM-Methoden sind aufgrund der fehlenden Standardisierung abgewertet. Eine leichte Abwertung (0) wird vom Autor auch für die Methode Mark II FPA vorgenommen, weil sie eine geringere Verbreitung hat und der Methode in den letzten Jahren in der Function Point Community weniger Beachtung geschenkt wurde.

Anforderung *A 6* wird von den standardisierten FSM-Methoden nicht unterstützt, wohl aber von der nicht standardisierten Use Case Points Methode.

Wie bereits in Abschnitt 2.2 gezeigt lassen sich die standardisierten FSM-Methoden als datenorientierte Abstraktion verallgemeinern und basieren demzufolge auf einem Lösungsentwurf, der Elemente der Konstruktion (z.B. technischer Datenfluss) vorwegnimmt. Damit wird die Anforderung *A 7* von diesen Methoden nicht erfüllt, denn die Methoden erfordern hoch qualifizierte Experten mit fundiertem Software-Engineering Wissen und sind nicht „ad-hoc“ schnell durchführbar.

Aufgrund der bisherigen Bewertung scheiden die standardisierten FSM-Methoden (d.h. Mark II FPA, COSMIC FFP, IFPUG FPA NESMA FPA und FiSMA 1.1) aus der weiteren Betrachtung aus, da sie die Anforderung in Summe nur unzureichend abbilden. Es bleibt also noch die nicht standardisierten FSM-Methoden und COCOMO zu vergleichen.

Alle bekannten nicht standardisierten FSM-Methoden erfüllen grundsätzlich die Anforderungen *A 2*, *A 3* und *A 4*, wenn auch unterschiedlich gut. Für die weitere Auswahl kann die Entscheidung aber aufgrund von Anforderung *A 6* auf die Use Case Points Methode unter den „anderen FSM-Methoden“ eingeschränkt werden.

Betrachten wir die COCOMO-Methode: COCOMO II basiert wie die ältere COCOMO Methode auf der Software-Programmgröße, gemessen in KSLOC (Kilo Source Lines of Code) und ist damit für die Größenbestimmung von Software-Projekten auf Basis einer fachlichen Spezifikation *isoliert* nicht anwendbar, Anforderung *A 6* ist damit nur in Kombination mit einer ergänzenden FSM-Methode erfüllbar.

Doch auch diese Kombination von separaten Methoden ist mit Einschränkungen verbunden, was folgende Analyse zeigen wird: Wie bereits in Kapitel 1.2 dargestellt, ist der Projektaufwand *PE* (Project Effort) abhängig sowohl von den *Functional User Requirements* (*FUR*) als auch den *Non-Functional User Requirement* (*NFUR*). Formel 15 fasst dies zusammen, sie wird später in Kapitel 3 hinsichtlich *FUR* und in Kapitel 5 hinsichtlich *NFUR* weiter ausgearbeitet:

$PE := f\ (FUR, NFUR)$	*Formel 15*

In der Praxis wird z.B. COCOMO aufgrund der Zusammenhänge gemäß Formel 15 gerne mit COSMIC FFP kombiniert: *FUR* kann mit Hilfe von COSMIC FFP ermittelt und darauf aufbauend mit COCOMO die Anteile für *NFUR* berücksichtigt werden. Anders als die Use Case Points

[13] dies sind die in Abschnitt 2.2 vorgestellten Schätzmethoden: Mark II FPA, COSMIC FFP, IFPUG FPA, NESMA FPA und FiSMA 1.1. Wir bezeichnen sie im Folgenden kurz als „standardisierte FSM-Methoden“.

Methode oder IFPUG FPA ist COSMIC FFP alleine auf die funktionale Größenbestimmung ausgerichtet, also den *FUR*-Teil. Erst durch sequentielle Anwendung der Methoden wird der Gesamtaufwand *PE* gemäß Formel 15 ermittelt.

Aus der Projektmanagement Domäne erwächst jedoch unter dem Begriff des *Scope Management* [PMI04] die Forderung nach Berücksichtigung aller Aufwandsteile an einem Projekt in einer ganzheitliche Methode, da sich diese Aufwandsteile möglicherweise gegenseitig bedingen und getrennte bzw. sequentiell angewendete Methoden dies möglicherweise nicht ausreichend berücksichtigen könnten [Bug+07]. Daraus folgern wir eine weitere Anforderung, welche in gewisser Weise mit *A 7* bereits vorbereitet wurde:

A 8: Ganzheitliche Methode, welche alle Aufwandsteile gemäß Formel 15 berücksichtigt.

Damit scheidet COCOMO als Methode für die weitere Betrachtung aus, da sie immer mit einer anderen Methode kombiniert werden muss.

Anforderung	Beschreibung der Anforderung	Mark II FPA	COSMIC FFP	IFPUG FPA	NESMA FPA	FiSMA 1.1	Use Case Points	Andere FSM	COCOMO II
A1	Die Schätzung muss zu einem sehr frühen Projektzeitpunkt auf Basis einer Grobspezifikation durchführbar sein.	+	+	+	+	+	+	+	+
A2	Die Schätzmethode muss auch in neuem Umfeld funktionieren.	+	+	0	+	+	+	+	+
A3	Möglichst schlanke Schätzmethode mit einem deutlich geringeren Durchführungsaufwand als bei einer Bottom-Up Schätzung.	0	0	0	0	0	+		+
A4	Praxiserprobte Methode, die mit möglichst wenig geschätzten Größen gemäß Formel 1 auskommt.	0	+	+	+	+	0	-	+
A5	Ausgereifte Methode, die dem aktuellen Stand der Technik entspricht.	0	+	+	+	+	0	-	+
A6	Direkte Ableitung der zu schätzenden (funktionalen) Größen unmittelbar aus einer Grobspezifikation mit einer Use Case orientierten Sicht.	-	-	-	-	0	+	-	-
A7	Die Schätzmethode trennt zwischen fachlicher Größenbestimmung und technischer Komplexitätsbewertung. Die Größenbestimmung stellt damit keine technischen Anforderungen an den fachlichen Schätzer und ist damit auf Management-Ebene bzw. durch Nicht-Informatiker anwendbar.		-	-	-	0	+		+
A8	Ganzheitliche Methode, welche alle Aufwandsteile gemäß Formel 15 berücksichtigt.	+	-	+	+	-	+		-

Tabelle 14: Zusammenfassung der Bewertung der vorgestellten Schätzmethoden hinsichtlich der formulierten Anforderungen.
Es bedeutet +: gut erfüllt; 0: mit Einschränkung erfüllt; -: nicht erfüllt

Hinsichtlich der gestellten Anforderungen ist die Use Case Points Methode am besten geeignet. Hier kommt insbesondere die gute Einbettung in eine Use Case orientierte Spezifikation sowie die Berücksichtigung von funktionalen und nicht funktionalen Anteilen zum Tragen. Allerdings weißt die UCP-Methode zahlreiche Schwächen auf (siehe Abschnitt 2.4.2) und insbesondere könnten die Anforderungen *A 4* und *A 5* deutlich besser erfüllt werden, wenn mit Hilfe von mehr Praxiserfahrung die Methode hinsichtlich ihrer Reife verbessert und auf den aktuellen Stand der

Technik gebracht würde. Es gilt nachzuweisen, ob die UCP-Methode derart weiter entwickelt werden kann, dass die in Abschnitt 2.4.2 genannten Schwächen aufgelöst werden können. Dabei könnte es sich empfehlen, die guten Erfahrungen aus der Entwicklung von COCOMO einzubringen. Damit kann nun die Aufgabenstellung dieser Dissertation feiner gefasst werden.

2.7 Verfeinerte Aufgabenstellung und Struktur dieser Arbeit

Wie in Abschnitt 2.6 dargelegt, erfüllt die Use Case Points (UCP) Methode die gestellten Anforderungen A 1 – 8 an eine Top-Down Schätzmethode am besten. Trotzdem sind neun wesentliche *Problempunkte* an der UCP-Methode gegeben (siehe Abschnitt 2.4.2). Nachfolgend sind diese Punkte nochmals aufgelistet:

P 1 Unterschiedlicher Schnitt der Use Cases

P 2 Reuse Schätzfehler

P 3 Actor-Gewichte zu gering

P 4 Kaum relevante vergleichbare Praxiserfahrung verfügbar

P 5 UCP Schätzmodell verwendet unzulässige Skalen-Transformationen

P 6 Definition von TCF und EF sind nicht mehr zeitgemäß

P 7 Komplexitätsstufen der TCF und EF sind zu vage definiert

P 8 Interpretationsspielraum von Use Cases

P 9 Umfang der durch UCP abgedeckten Projektaufwände

Die Entwicklung der Lösung für diese Problempunkte erfolgt in mehreren Schritten. Dazu wird diese Dissertation strukturiert, wie in Abbildung 19 dargestellt.

Zunächst wird die UCP-Methode als Ganzes untersucht und eine neue strukturelle Sicht präsentiert (Abschnitt 2.8). Darauf aufbauend erfolgt die Entwicklung von Lösungen getrennt nach *Functional User Requirements* (FUR) und den *Non-Functional User Requirements* (NFUR). Zur Definition und Bedeutung siehe Abschnitt 2.6 und Formel 15 (Seite 43).

In Kapitel 3 wird ein Modell für die Identifikation von Use Cases entwickelt: Ein wesentlicher Kritikpunkt an der Karner-Methode ist, dass Use Cases in unterschiedlicher Granularität beschrieben werden können (siehe P 1). Dies hat unmittelbar Einfluss auf das Schätzergebnis. Ferner sind die Ausgangsformen der Grobspezifikationen uneinheitlich und es bedarf eines Regelwerkes, wie Use Cases zu identifizieren sind. In der vorliegenden Arbeit wird dazu auf Basis von zahlreichen Projektbeispielen ein Leitfaden zum "Schneiden" von Use Cases für die Abschätzung des funktionalen Umfangs von Software bei der Anwendung der UCP-Methode geben. Damit wird auch eine Lösung für P 1, P 2 und P 3 vorgestellt. Dieser Leitfaden ist Bestandteil einer neu entwickelten Methode UCP 3.0, welche in mehreren Iterationen aus einer Version 1.0 und 2.0 entwickelt und erprobt wurde. In der UCP-Methode entspricht UUCP dem *FUR*-Teil aus Formel

15, in UCP 3.0 werden wir diesen Teil als *A-Faktor* bezeichnen. Er stellt sozusagen das „Fundament“ der neuen UCP-Methode dar.

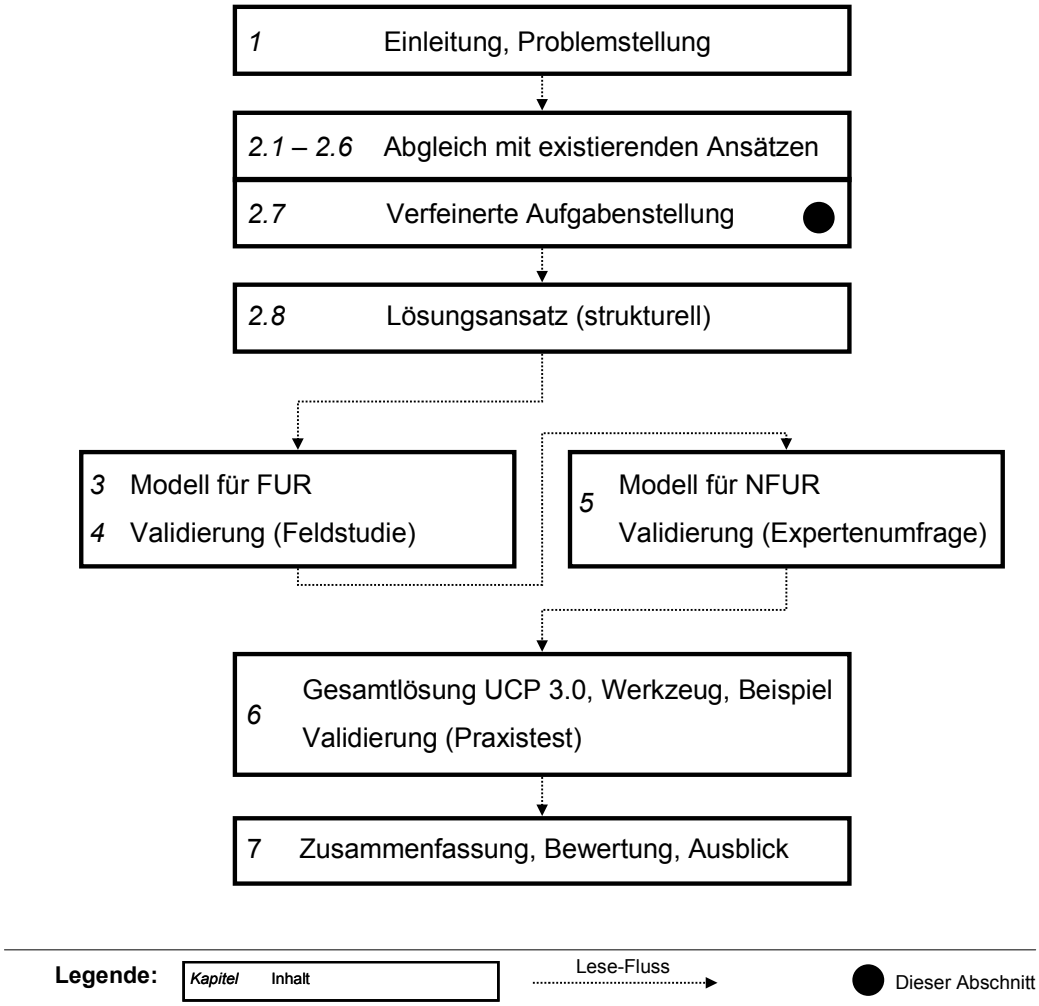


Abbildung 19: Aufbau dieser Arbeit

In Kapitel 4 erfolgt die Validierung der modellbezogenen Use Case Identifikation mit Hilfe einer Feldstudie: Zunächst ist zu zeigen, dass die UCP-Sprache und der entwickelte A-Faktor aus Kapitel 3 zu einer guten Reproduzierbarkeit des geschätzten Aufwandes führt. Dabei sind unterschiedliche Spezifikationsformen einzubeziehen. Dazu werden experimentelle Vorgehensweisen in der Informatik betrachtet und eine Feldstudie zur Überprüfung entwickelt. Es wird sich zeigen, dass die Forderungen von Ouwerkerk und Abran [OA06] hinsichtlich Reproduzierbarkeit gut erfüllt werden (siehe hierzu auch die Erläuterungen zu P 8). Die in Anforderung *A 7* enthaltene Forderung nach Anwendbarkeit auch durch Nicht-Techniker ist bisher in der Literatur nicht überprüft worden. Mit Hilfe der Feldstudie kann diese ebenfalls überprüft werden.

In Kapitel 5 wird dann ein neues Kostenfaktor-Modell für die UCP-Methode abgeleitet und durch eine Expertenumfrage validiert: Bei der Definition des neuen Kostenfaktor-Modells sind die Forderungen von Ouwerkerk und Abran [OA06] hinsichtlich korrekter algebraischer Skalentransformation (siehe P 5) der UCP Einflussgrößen zu berücksichtigen. Ferner erlaubt die Definition der Einflussgrößen des *TCF* und *EF* der Karner-Methode erheblichen Interpretationsspiel-

raum. Hier ist eine präzisere Fassung zu erreichen. Im Bereich der betrieblichen Informationssysteme sind bei der Bewertung nicht-funktionaler Anforderungen erhebliche Fortschritte und Weiterentwicklungen gemacht worden, siehe Abschnitt 1.2. Diese Erkenntnisse sind in die neue erweiterte UCP-Methode für den NFUR-Teil einzubringen. Die dargestellten Entwicklungen z.B. von COCOMO hin zu COCOMO II sind bisher in der UCP-Methode nicht eingeflossen und sollten für die UCP-Methode überprüft werden.

Zunächst ist ein Verfahren für die UCP-Methode zur Bestimmung der Kostenfaktoren aufzusetzen. Die Auswahl geeigneter Einflussgrößen wird dabei durch eine Expertenumfrage validiert und mit Hilfe von empirischen Daten aus der Praxis wird das Kostenfaktormodell dann kalibriert. Damit wird eine Lösung für P 6 und P 7 formuliert werden.

In Kapitel 6 erfolgt die Synthese aus FUR und NFUR-Teil zur neuen Gesamtlösung *UCP 3.0*. In Abschnitt 6.1 wird zunächst die Gesamtlösung nochmals zusammengefasst und der Umfang der durch UCP 3.0 abgedeckten Projektaufwände definiert (P 9). In Abschnitt 6.2 wird dann ein Schätzwerkzeug zur Methode vorgestellt und in Abschnitt 6.3 die Anwendung der Methode an einem Beispiel erläutert. UCP 3.0 erfüllt die gestellten Anforderungen an die gesuchte Schätzmethode und löst zudem die Problempunkte P 1 – 9. Hinsichtlich der Anforderung *A 4* ist festzuhalten, dass bisher wenig UCP-Praxiserfahrung [OA06] existiert. Diese ist zunächst in Form einer geeigneten Projektdatenbank aufzubauen und damit eine Aussage zur Schätzgenauigkeit der Methode UCP 3.0 zu erreichen. Mit diesem Praxistest ist die Methode als Ganzes zu validieren (Abschnitt 6.4).

In Kapitel 7 wird das wissenschaftliche Ergebnis dieser Dissertation zusammengefasst (Abschnitt 7.1) und die Methode UCP 3.0 abschließend bewertet (Abschnitt 7.2) und dabei die Vorteile gegenüber der Karner-Methode herausgestellt als auch die Grenzen von UCP 3.0 aufgezeigt. Die Dissertation schließt mit einem Ausblick in Abschnitt 7.3 auf möglichen weiterführenden Forschungsschwerpunkte, die sich aus dieser Arbeit ableiten lassen.

Insgesamt bauten die vorliegenden Untersuchungen wesentlich auf den Erfahrungen und auf der Datenbasis des Unternehmens Capgemini sd&m auf. Vergleiche mit anderen Unternehmen und mit in der Literatur beschriebenen Schätzmethoden stellen die Ansätze und Ergebnisse auf eine breite Basis.

2.8 Lösungsansatz zur UCP-Methode

In einem ersten Schritt wird die UCP-Methode in ihrer Gesamtheit untersucht mit dem Ziel, die Einflussgrößen besser zu verstehen und eine klarere Struktur für eine Weiterentwicklung und Verbesserung zu erhalten. Aus Sicht der industriellen Nutzung (siehe Abschnitt 1.2) der Methode wäre es hilfreich, den Einfluss auf den Projektaufwand *PE* durch die Anforderungsdefinition und durch die Art der Projektdurchführung zu separieren. Dies führt zu einer neuen Interpretation der UCP-Terme hin zu den folgenden drei Aufwandstypen:

A-Faktor: Use Cases definieren die funktionalen Anforderungen in Bezug auf die Zielsetzung des Projektes. In betrieblichen Informationssystemen werden diese Anforderungen als **A**nwendungssoftware (A-Software) implementiert [Sie04]. Der Implementierungsaufwand ist proportio-

nal zu den Unweighted Use Case Points (UUCP, Formel 7, Seite 31), wir nennen dies den A-Faktor.

T-Faktor: Dieser entspricht dem *Technical Complexity Factor* (TCF) in der UCP-Methode nach Karner. Allerdings werden die Einflussgrößen im Detail überarbeitet und die Formel zur Bildung des Faktors wird neu definiert (Abschnitt 5.7)

M-Faktor: Der Management (M-) Faktor definiert die Komplexität, die durch die Projektorganisation bedingt wird und wurde von der UCP-Methode nach Karner aus dem *Environmental Factor* (EF) abgeleitet. Der EF wird durch eine neue Formel ersetzt und es wurden neue Einflussgrößen identifiziert. Damit ist der M-Faktor für heutige Software-Entwicklungsprojekte deutlich besser geeignet.

T-Faktor und M-Faktor werden in dieser Dissertation generell als ***Kostenfaktor*** bezeichnet. Ein Kostenfaktor kann sich wiederum aus mehreren ***Einflussgrößen*** zusammensetzen. Die Komplexität der Einflussgrößen ist im Projektkontext durch Punkte zu bewerten und aus dieser Bewertung wird dann mittels einer mathematischen Formel der Kostenfaktor bestimmt.

Der A-Faktor repräsentiert die anwendungsspezifischen funktionalen Anforderungen des Projektes. Der T-Faktor steht für die nichtfunktionalen Anforderungen. Beide zusammen definieren die Systemanforderungen, die wesentlich durch den Auftraggeber eines Software-Entwicklungsprojektes bestimmt werden. Der M-Faktor bildet den Einfluss des Projektprozesses auf den Gesamtaufwand des Projektes ab. Ein konstanter Produktivitätsfaktor (PF) kalibriert die Effizienz der Projektumsetzung. M-Faktor und PF sind wesentlich durch den Auftragnehmer eines Software-Entwicklungsprojektes bestimmt. Das Produkt aller dieser Terme bestimmt den Gesamtprojektaufwand PE der neuen UCP-Methode 3.0. Dieser Zusammenhang wird in Abbildung 20 zusammengefasst.

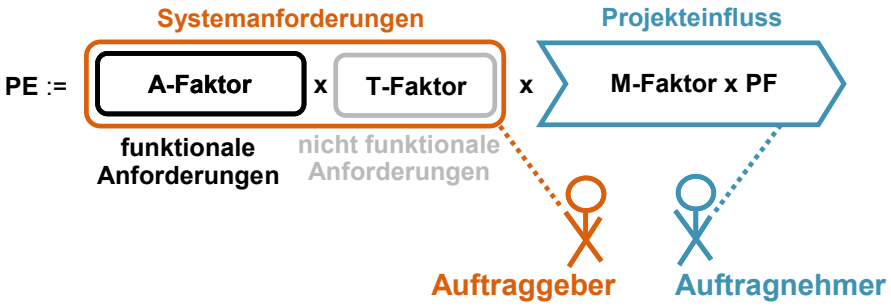


Abbildung 20: Strukturelle Aufteilung des Projektaufwandes (PE) in UCP 3.0

Diese konzeptionelle Trennung in A-, T- und M-Faktor schafft insbesondere Klarheit in Bezug auf die Quellen bzw. Stakeholder für die Aufwandsfaktoren der Anforderungen. Damit werden die Anforderungen A 6 und A 7 aus Abschnitt 2.7 erfüllt:

A 6: Direkte Ableitung der zu schätzenden (funktionalen) Größen unmittelbar aus einer Grobspezifikation mit einer Use Case orientierten Sicht.

A 7 Die Schätzmethode trennt zwischen fachlicher Größenbestimmung und technischer Komplexitätsbewertung. Die Größenbestimmung stellt damit keine technischen Anforderungen

an den fachlichen Schätzer und ist damit auf Management-Ebene bzw. durch Nicht-Informatiker anwendbar.

Im folgenden Kapitel 3 werden zunächst nur die Regeln für den A-Faktor entwickelt. Die Kostenfaktoren (T-Faktor, M-Faktor) werden in Kapitel 5 entwickelt.

3 Modellbezogene Use Case Identifikation (A-Faktor)

In diesem Kapitel wird eine modellbezogene Vorgehensweise zur Identifikation und Komplexitätsbewertung von Use Cases aus unterschiedlichen Spezifikationen beschrieben. Damit kann die Use Case Points Methode zur Aufwandsschätzung von Software-Entwicklungsprojekten einheitlich und standardisiert zu einem frühen Zeitpunkt angewendet werden. Wesentliche Teile dieses Kapitels wurden als Preprint [FKD07] bereits zur Diskussion gestellt.

Die Use Case Point Methode (UCP-Methode) erlaubt eine schnelle Schätzung von zu erwartenden Aufwänden für Software-Entwicklungsprojekte. Basis für eine solche Schätzung sind Grob-Spezifikationen unterschiedlicher Formen und unterschiedlicher Granularität (siehe Abschnitt 2.3). Entscheidend für den Erfolg der UCP-Methode und die Vergleichbarkeit der Ergebnisse ist vor allem, ob und wie es gelingt, eine vorliegende Spezifikation auf Use Cases im Sinne der UCP-Methode abzubilden. Dafür beschreibt dieses Kapitel einen Vorschlag für einen Leitfaden, der aus der Spezifikations-Praxis des Softwarehauses Capgemini sd&m abgeleitet wurde. Damit werden die Problempunkte P 1, P 2 und P 3 (Abschnitt 2.7) gelöst. Der Nachweis dafür wird dann zum Einen durch die empirische Analyse der Reproduzierbarkeit von Aufwandsschätzergebnisse in Kapitel 4 und zum Anderen in Kapitel 6 durch den gesamthaften Vergleich der original Methode nach Karner mit der hier weiterentwickelten Methode erbracht.

Im Folgenden wird versucht, eine einheitliche Sicht auf Use Cases aus Sicht der UCP Schätzmethodik zu beschreiben. Sie basiert auf Definitionen, die in der Industrie übliche Praxis sind [Coc03, DAS05, Dum03, LD01, Sie04]. Wir wollen dabei bewusst keine neuen Begriffe und Ebenen einführen, sondern die Ebene und damit auch Granularität finden, die für eine Schätzung nach UCP erforderlich ist.

Zunächst definieren wir ein Use Case Metamodell für die UCP-Methode (kurz *UCP-Sprache*). Das Metamodell unterteilt sich in Sprachelemente (Abschnitt 3.1.1) mit seinen „Zählregeln“ und der Bestimmung der Komplexität von Use Cases und Actors (Abschnitt 3.1.2). Dieses Metamodell wird für die neue Methode UCP 3.0 in dieser Dissertation zugrunde liegen. In Abschnitt 3.1.3 wird die Anwendung dieses Metamodells für die UCP-Schätzung beschrieben. Es folgt in Abschnitt 3.2 ein Leitfaden zur Use Case Identifikation mit allgemeinen Hinweisen, die für alle Spezifikationsformen gelten. In den beiden nachfolgenden Abschnitten werden Anhand von Regeln und Beispielen dann dargestellt, wie das Mapping von UML-basierten (Abschnitt 3.3) und weiteren (Abschnitt 3.4) Spezifikationsformen auf die UCP-Sprache konkret erfolgt und welche Zählung zur Bestimmung der Use Case Komplexität dabei vorgenommen wird. In der Praxis kommen diese Spezifikationsformen nie in Reinform vor, sondern verschiedene Formen sind um textuelle Erklärungen ergänzt oder mehrere Formen werden in einer Spezifikation verwendet. Für eine sorgfältige Analyse der Spezifikationsformen ist zunächst die isolierte Betrachtung der Einzelformen aber unumgänglich.

3.1 Metamodell zur Use Case Points Schätzung (UCP-Sprache)

Karner hat für die Bestimmung der Komplexität von Anwendungsfällen die Zahl der Transaktionen als Maß genommen, eine exakte Definition des Transaktionsbegriffes fehlte aber und die Regeln zur Komplexitätsbewertung sind wenig ausgefeilt (siehe Abschnitt 2.4.1). In der Literatur gibt es unterschiedliche, teils widersprüchliche Ansätze, diese Regeln anzuwenden oder zu verfeinern [Coh05, Cle05, Koi04]. Als praxistaugliche Synthese dieser Ansätze hat sich zunächst ein Klassifikationsschema unter Berücksichtigung von Schritten, Szenarien und Dialogen eines Use Cases herausgestellt. Hiermit konnte bereits eine deutliche Verbesserung der Methode erreicht werden [FJE06]. Dieses Schema ist Grundlage einer Methode UCP 1.0, worauf wir später noch zurückkommen werden.

Die unzulängliche Definition in der Karner-Methode ist für uns Anlass, zunächst ein Use Case Metamodell zu definieren, auf welches die Anwendungsfälle einer Spezifikation abgebildet werden. Dieses Abbilden bezeichnen wir als *Mapping* (siehe auch Abbildung 14 in Abschnitt 2.3.2). Die Entwicklung des Metamodells und die Regeln für die Komplexitätsbestimmung von Use Cases erfolgten schrittweise von einer Version 1.0 über hin zu einer Version UCP 3.0. Diese Weiterentwicklung wurde durch die detaillierte Auswertung von Projektdaten sowie eine Expertenumfrage hinsichtlich der Einschätzung von komplexitätsrelevanten Faktoren [Hel08] abgesichert.

In der Version UCP 3.0 wird gegenüber UCP 1.0 eine verfeinerte Bestimmung der Komplexität von Use Cases vorgenommen, indem an die Stelle der *Dialoge* die *Interaktionselemente* treten. Eine Rechtfertigung dafür kann später in Abschnitt 4.6 gegeben werden. *Interaktionselement* unterscheidet dann in *Dialoge*, *Schnittstellen* und *Berichte*.

3.1.1 UCP-Sprachelemente des A-Faktors

Zur Einordnung der UCP-Sprache in den zweistufigen Transformationsprozess verweisen wir nochmals auf Abbildung 14 (Seite 27).

Nachfolgend definieren wir die Elemente des Metamodells für die UCP-Methode. Für die Begriffe der Spezifikationsmethode verweisen wir auf Abschnitt 2.3.2. Abbildung 21 zeigt die Elemente des Metamodells (UCP-Sprache) für die UCP-Methode.

Dabei benutzen wir in der gesamten Arbeit den Begriff der *Zählung* und meinen damit, dass das entsprechende Element aus der Spezifikation 1:1 auf das entsprechende Element des UCP Metamodells abgebildet werden kann und daher relevant ist. Wenn ein Element „nicht gezählt“ wird bedeutet dies, dass es für die UCP-Methode unberücksichtigt bleibt.

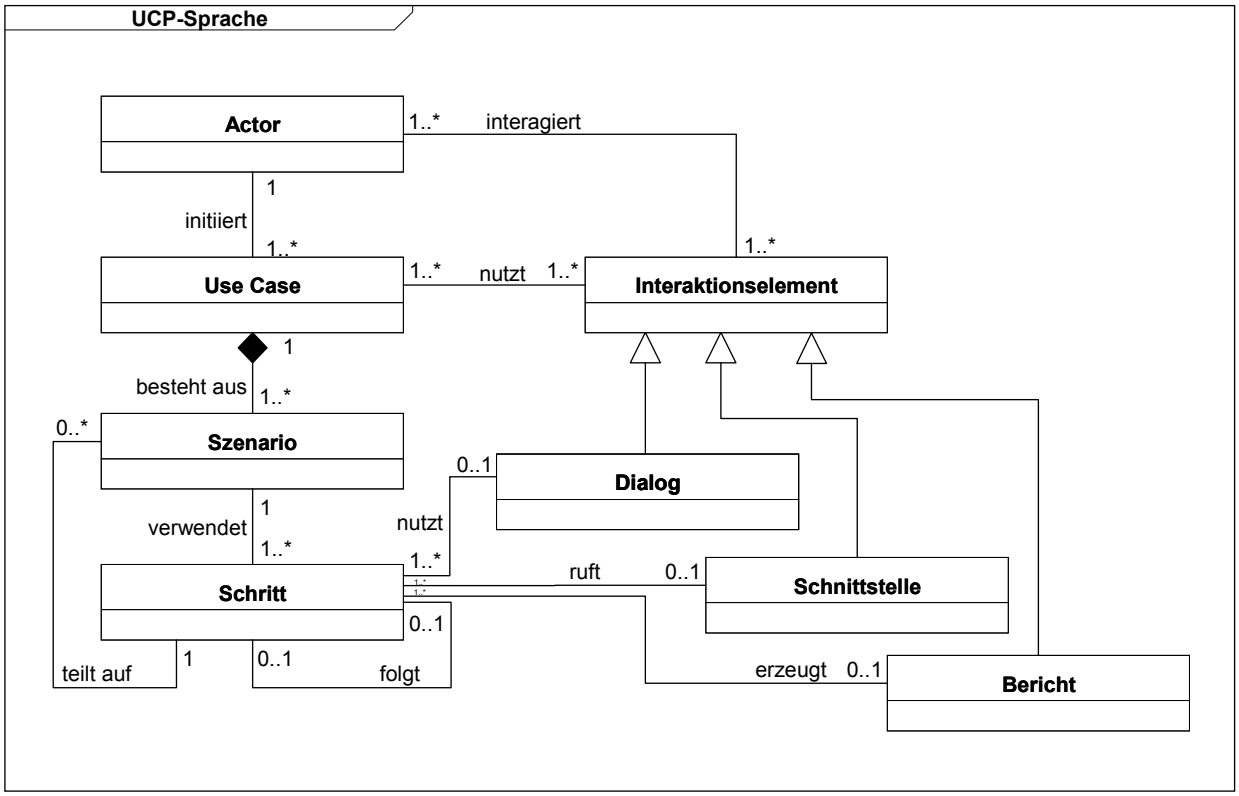


Abbildung 21: Metamodell zur UCP-Methode (UCP-Sprache)

Actor: Damit sind alle (menschlichen und technischen) Nutzer des Systems gemeint, die Use Cases auslösen oder im weiteren Ablauf mit ihnen interagieren. Endnutzer mit fachlich unterschiedlichen Rollen (z.B. Benutzer/Administrator) werden als separate Actors gezählt (wie in der Anwendungsfall-Modellierung üblich). Falls verschiedene Actors bezüglich eines Use Cases fachlich verschiedene Abläufe benötigen oder unterschiedlich mit der Anwendung interagieren, sollten in diesem Fall auch neue Use Cases dafür verwendet werden oder zumindest innerhalb eines Use Cases neue Szenarien eingeführt werden. Eine weitere Besonderheit bei der Erfassung der Actors für UCP ist die einmalige Zählung dieser über alle Use Cases hinweg. Ein Actor wird also nicht pro Use Case, sondern über die gesamte Anwendung nur einmal erfasst. Damit ist es besonders wichtig, Actors als Rollen bzw. als in Gruppen kategorisiert zu betrachten.

Use Case: Ein Use Case spezifiziert exakt eine systemunterstützenden Aktivität eines Geschäftsprozesses. Er beschreibt das Verhalten und die Interaktion eines Systems als Reaktion auf die zielgerichtete Anfrage oder Aktion eines Actors. Die Beschreibung des Use Cases erklärt sowohl das extern sichtbare als auch das detaillierte interne Systemverhalten in der Sprache und aus der Sicht der Anwender. Mit einem Use Case wird durch das System ein für den Anwender sinnvoller Dienst erbracht oder ein benutzbares Ergebnis erreicht. Ganz entscheidend ist dabei der Hinweis, dass mit einem Use Case immer ein bestimmtes *fachliches Ziel* erreicht werden soll [Coc95]. Ohne das Erreichen dieses Zieles werden wir in der Regel nur von Anwendungsfunktionen oder Schritten eines Use Cases sprechen.

Beispiele dafür sind alle funktional orientierten Beschreibungen oder auch dialogorientierten Beschreibungen. So ist z.B. das Verwenden eines Suchdialoges häufig kein eigener abgeschlossener Use Case, da das eigentliche fachliche Ziel mit dem Finden und Anzei-

gen der Treffermenge noch nicht erreicht ist, weil z.B. mit einem bestimmten Datensatz nach dem Suchen weitere fachliche Schritte durchgeführt werden müssen, um das eigentliche fachliche Ziel zu erreichen.

Use Cases sind dadurch charakterisiert, dass sie immer eine Interaktion des spezifizierten Systems mit mindestens einem externen Actor enthalten. Dieser Actor kann entweder ein menschlicher Nutzer, ein anderer Use Case (wie z.B. ein Batch) oder ein Nachbar- oder Partnersystem sein. Die Interaktion selbst kann dabei sowohl von dem Actor als auch von dem System selbst ausgehen. Der Auslöser kann demzufolge sowohl ein externes Ereignis (z.B. Aufruf durch den Anwender) als auch ein internes Ereignis im spezifizierten System (u.a. Zeitereignis) sein.

Wir unterscheiden keine verschiedenen Arten von Use Cases, wie z.B. Business Use Case, Background Use Case (Batch), Technical Use Case, Geschäftsvorfall oder Systemfunktion. Damit wird stellenweise eine Kategorisierung nach der unterschiedlichen Interaktion, dem Detaillierungsgrad der Beschreibung oder der Einordnung in die Geschäftsprozesse vorgenommen. Für die UCP-Methode ist diese Unterscheidung bei der Bestimmung und dem Finden von Use Cases nicht erforderlich und wird nicht verwendet. Allerdings wird später auf Besonderheiten einiger dieser Arten hingewiesen, wenn es um das Abschätzen der Komplexität dieser Use Cases geht.

In einer Spezifikation können Anwendungsfälle durch Verwenden von Beziehungen eine *Hierarchie* bilden (z.B. Unteranwendungsfälle). Für die UCP-Methode ist die Hierarchie als Strukturinformation nicht relevant sondern nur die Anzahl der Use Cases, sofern sicher gestellt wird, dass ein Use Case nicht mehrfach berücksichtigt wird. Darauf kommen wir in Abschnitt 3.2.6 detaillierter zurück.

Szenario: Ein Szenario beschreibt den Ablauf von Aktionen der Actors und des Systems, der notwendig ist, das Ziel des Use Cases zu erreichen. Dazu zählen auch nichttriviale Fehlerszenarien. In der Spezifikation wird zwischen Erfolgs- und Alternativszenarien unterschieden. Für die UCP-Methode ist diese Unterscheidung nicht relevant, da sie hinsichtlich der Gewichtung des Use Cases gleichberechtigt behandelt werden. Zu beachten ist nur, dass es im UCP-Modell mindestens ein Szenario (i.d.R. das Erfolgsszenario in der Spezifikation) geben muss, dass mindestens zwei Schritte hat, alle anderen Szenarien können auch nur einen Schritt als Untergrenze haben. In Abbildung 21 wird aus Gründen der Übersichtlichkeit nur der generalisierte Fall modelliert, dass ein Szenario [1..*] Schritte hat.

Bei sehr komplexen Use Cases kann es vorkommen, dass es auch mehrere gleichberechtigte Erfolgsszenarien gibt (im Sinne mehrerer Hauptszenarien). Im Allgemeinen handelt es sich aber um untergeordnete Alternativszenarien.

Fehlerszenarien sind solche, die nicht zum Erfolg (= Erreichen des fachlichen Ziels) führen, wobei wir für die Zählung bei der UCP-Schätzung zwischen trivialen und nichttrivialen Fehlerszenarien unterscheiden. Fachliche Fehlerszenarien werden berücksichtigt, wenn fachliche Schritte zur Fehlerbehandlung durchlaufen werden (z.B. Autokorrekturen, Änderung statt Neuanlage, ...), triviale Fehlerszenarien werden nicht berücksichtigt (z.B. „Anzeige einer Meldung, dann Abbruch“).

Schritt: Entscheidend für die Gewichtung eines Use Cases ist die Anzahl der für die Erreichung des fachlichen Zieles erforderlichen Schritte. Dafür gibt es leider keine allgemein anerkannte Definition. In verschiedenen Quellen wird auch von Aktionen, Operationen bzw. Folge von Operationen oder Aktivitäten gesprochen. Wir verwenden im Metamodell der UCP-Methode den Begriff *Schritt*. Systemseitige Aktionen werden häufig als Anwendungsfunktionen bezeichnet und als Schritte gewertet, wenn die Beschreibung bereits eine gewisse funktionale Zerlegung der Use Cases darstellt.

Ein Schritt im Ablauf eines Use Cases ist ein in sich geschlossener fachlicher Teil des Use Cases, der vom folgenden Schritt und davorliegenden Schritt eindeutig getrennt ist durch z.B.

- den Wechsel des Actors, oder der verarbeitenden "Schicht" (z.B. Eingabe im Dialog durch den Nutzer => Verarbeitung der Eingabe am Server => Anzeige des Ergebnisses an der Oberfläche)
- Erzeugen eines fachlich nicht trivialen (Zwischen-) Ergebnisses (z.B. Erzeugen von Druckausgaben oder mehrere fachliche Prüfungen nacheinander)
- Verzweigen in neue Szenarios

Entscheidend für die Use Case Modellierung ist hierbei immer die Betrachtung der obersten Ebene der Zerlegung aus Sicht des Use Cases. Jede weitere Verfeinerung ist für die Schätzung nach UCP dann nicht mehr relevant. Werden in einer Spezifikation also z.B. in einer Beschreibung eines Anwendungsfalles Aktionen genannt, die wiederum aus Aktionen bestehen, so werden hier nur die Aktionen der obersten Ebene als Schritte gezählt.

Als Schritte werden gezählt:

- die Anzahl aller Aktionen in allen Szenarien; es handelt sich also um ein Aufsummieren über alle Szenarien, wobei Aktionen, die in mehreren Szenarien verwendet werden, nur einmal in die Zählung eingehen
- gleichwertig sowohl Anwenderaktionen als auch systemseitige Aktionen
- der Aufruf bzw. die Initialisierung eines Dialogs als ein Schritt
- der Aufruf einer Schnittstelle als ein Schritt

Typische Beispiele für Schritte sind:

- Eingabe eines oder mehrerer Werte in einen Dialog (ohne dass dazwischen ein Server-Roundtrip erfolgt)
- Aufruf von Anwendungsfunktionen
- Datentransport (Lesen, Schreiben)
- Durchführung fachlicher Prüfungen
- Erzeugung von Ausgaben
- triviale Auswahlvorgänge durch einen Actor aus einer Anzeige heraus werden nicht als eigener fachlicher Schritt gezählt (z.B. Adresse auswählen), es können aber je nach Auswahl neue Szenarien entstehen

Eine Besonderheit bei den Schritten stellen fachlich sehr komplexe Abläufe dar, die nach obigen Kriterien in einem einzigen Schritt abgearbeitet werden. Diese würden mit der genannten Definition potentiell unterschätzt werden. Darauf wird speziell im Abschnitt 3.2.4 eingegangen.

Dialog: Als weiterer Parameter geht die Anzahl der unterschiedlichen Dialoge eines Anwendungsfalls in die UCP-Methode ein. Dialoge werden wie folgt gezählt:

- Jeder Reiter bzw. jede Maske eines Dialoges (mit signifikanten fachlichen Unterschieden) wird als eigener Dialog gezählt
- jeder Frame einer Webseite (mit signifikanten Steuerelementen) wird als eigener Dialog gezählt
- Triviale Pop-Up-Meldungen, Bestätigungen und Menüs werden nicht gezählt

Der Aufruf bzw. die Initialisierung eines Dialogs wird immer auch als Schritt gezählt.

In der Methode UCP 1.0 wurde unter einem Dialog in einem erweiterten Sinn jede Interaktionsschnittstelle verstanden, für UCP 3.0 differenzieren wir explizit in Schnittstellen und Berichte. Daher wurden für UCP 1.0 neben den klassischen Dialogen auch noch folgende Interaktionsschnittstellen als Dialog gezählt und damit bei der UCP-Methode berücksichtigt:

- Komplexe Schnittstellen, die Nachbarsystemen (zusätzlich zur Verarbeitung) zur Verfügung gestellt werden
- Druckstücke bzw. sonstige speziell erzeugte formatierte Ausgaben (Berichte)

Schnittstelle: Dieses Konzept wurde erst mit der Methode UCP 3.0 eingeführt. Zusätzlich zur einmaligen Erfassung von Nachbarsystemen als Actors (was den generellen Aufwand zur Anbindung abdeckt) werden in den Use Cases die jeweils relevanten Schnittstellen gezählt. Eine Schnittstelle ist als Interaktionselement zum Ansprechen einer fachlichen Funktion zu verstehen. Auch wenn mehrere fachliche Funktionen (z.B. lesen/schreiben) über eine aus technischer Sicht gemeinsame (generische) Schnittstelle abgewickelt werden, wird für jede Funktion eine separate Schnittstelle gezählt. Die Zählung der Schnittstellen erfolgt in dem Use Case, in dem sie verwendet werden. Nicht verwendete Schnittstellen werden auch nicht gezählt.

Es werden sowohl angebotene Schnittstellen berücksichtigt, die das zu schätzende System seinen Nachbarn zur Verfügung stellt, als auch benötigte Schnittstellen, über die das zu schätzende System auf Nachbarsysteme im Ablauf des Use Cases zugreift. Wird dieselbe Schnittstelle in mehreren Use Cases verwendet, so wird diese nur im ersten Use Case mitgezählt. Der Aufruf einer Schnittstelle wird im Use Case immer auch als Schritt gezählt.

Bericht: Die statische Druckausgabe ist die klassische Form eines Berichts. Für die Methode UCP 3.0 umfasst ein Bericht aber auch dynamische (filterbare, verlinkte, etc.) formatierte Präsentationen von Daten und berechneten Ergebnissen. Es werden Berichte gezählt, die im Ablauf eines Use Cases (typischerweise mit Hilfe der üblichen Reporting-Werkzeuge) erstellt werden. Kann für die Berichterstellung allerdings nicht auf ein Werkzeug zurückgegriffen werden, sondern enthält die Spezifikation dazu einen eigenen Anwendungsfall, ist dies als Use Case abzubilden und kann nicht durch einen Bericht allein repräsentiert

werden (Gefahr der Unterschätzung!). Dumps von Dialog-Eingaben zählen nicht als Berichte, können aber z.B. mit einem zusätzlichen Schritt erfasst werden. Die Erstellung eines Berichts wird immer auch als Schritt gezählt.

3.1.2 Bewertung der Komplexität von Use Cases und Actors

Zur Einordnung der Bewertung von Use Cases und Actors in den zweistufigen Transformationsprozess verweisen wir nochmals auf Abbildung 14 (Seite 27), hier geht es jetzt um die zweite (rechte) Transformation von der UCP-Sprache in die Points des FSM.

Wie Eingangs in Abschnitt 3.1 angerissen hatte Karner für die Bestimmung der Komplexität von Use Cases die Zahl der Transaktionen als Maß genommen. <<include>> und <<extend>> Beziehungen von Use Cases blieben unberücksichtigt. Als praxistaugliche Weiterentwicklung werden in der Methode UCP 1.0 folgendes Klassifikationsschema unter Berücksichtigung von Schritten, Szenarien und Dialogen eines Use Cases verwendet, <<include>> und <<extend>> Beziehungen werden dabei einmalig mit berücksichtigt. Für Use Cases wird jeweils das Maximum aus der Anzahl der Szenarien, Schritte oder Dialoge je Use Case genommen:

- einfach: bei 1-3 Szenarien, Schritten oder Dialoge
- mittel: bei 4-7 Szenarien, Schritten oder Dialoge
- hoch: bei 8 oder mehr Szenarien, Schritten oder Dialoge

Actors wurden nach dem Typ der Schnittstelle klassifiziert:

- einfach (System über Schnittstelle) : 5 UCP[14]
- mittel (System über Protokoll) : 10 UCP
- komplex (Dialogschnittstelle): 15 UCP

Dieses Klassifikationsschema findet in dieser Dissertation Anwendung für die Methode UCP 1.0 und Folgeversionen UCP 2.x. Die Praxis zeigte jedoch, dass

- die Actors im Vergleich zu den Use Cases um einen Faktor fünf zu schwach bewertet wurden [Bad08]
- für die Use Cases meistens die Zahl der Schritte für die Komplexitätsbestimmung ausschlaggebend war [Hel08].

In der finalen Fassung der Methode UCP 3.0 liegt ein verfeinertes Klassifikationsschema zugrunde, die Rechtfertigung dafür wird in Kapitel 4 hinsichtlich der Reproduzierbarkeit von Schätzergebnissen und in Kapitel 6 hinsichtlich des Vergleiches zwischen den Methoden gegeben.

An der Stelle der Dialoge werden nun Interaktionselemente bewertet, die sich wiederum unterteilen in Dialoge, Schnittstellen und Berichte. Wenn möglich, wird die Komplexität der Interakti-

[14] UCP steht hier für Use Case *Points*. Zur Unterscheidung wird in der Methode UCP 3.0 nur noch von *Points* statt *Use Case Points* die Rede sein.

onselemente in „einfach“, „mittel“ und „komplex“ unterschieden. Sind nicht genügend Informationen verfügbar, wird im Regelfall „mittel“ als Standardwert für die Interaktionselemente angenommen. Alle Kenngrößen gehen gleichberechtigt in die Bewertung ein[15]. Actors und Use Cases werden einheitlich in der neuen Bewertungsskala „Points“ bewertet, nicht mehr in UCP.

Als neues Konzept wird der Gedanke des *elementaren* Use Case eingeführt. Damit ist der einfachste bzw. „kleinste“ Use Case gemeint, der vorstellbar ist. Er besteht aus nur einem Szenario, einem Interaktionselement und zwei bis vier Schritten (d.h. im Mittel also drei Schritte). Dieser elementaren Use Case wird mit einem Punkt (1 Point) bewertet. Abbildung 22 zeigt ein Beispiel für einen solchen einfachen Use Case anhand eines Ablaufdiagrammes und eines Dialoges. Man beachte, dass die triviale Fehlerschleife (*BLZ nicht korrekt*) nicht als Szenario gewertet wird.

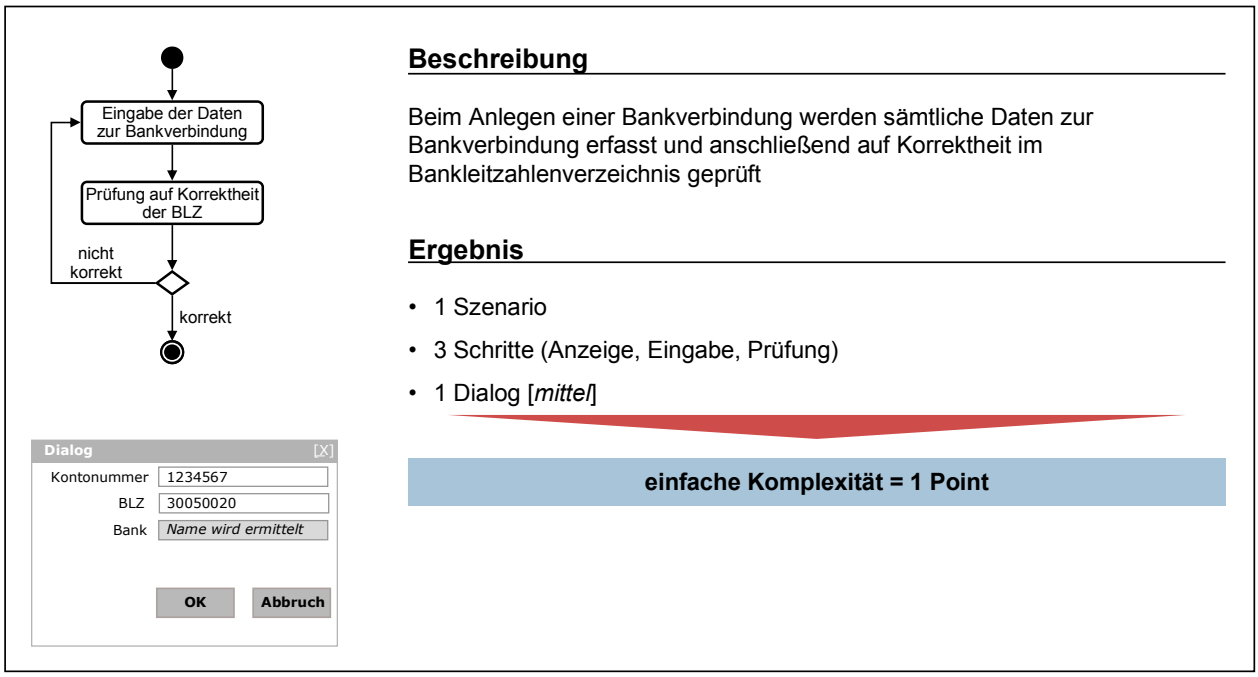


Abbildung 22: Beispiel für einen elementaren Use Case

Damit bekommt die Größe „1 Point“ eine anschauliche Bedeutung. Komplexere Use Cases werden dann als Vielfaches dieses elementaren Use Cases verstanden. Dabei ist die alte dreistufige Skala der Komplexitätsklassen {*einfach, mittel, komplex*} nun nach oben geöffnet, womit *sehr* komplexe Use Cases ohne Zerlegung bewertet werden können (Tabelle 15). Die Einteilung in Komplexitätsklassen bleibt erhalten und ist nun intuitiver zu verstehen.

Use Cases größer als 5 Points sind zwar konzeptionell möglich, deuten aber auf einen falschen Use Case „Schnitt“ hin. Hier ist die Granularität der Use Cases in Frage zu stellen, evtl. ist der Anwendungsfall auf einem zu hohen Abstraktionsniveau gefasst (siehe Abschnitt 2.4.2, Anmerkungen zu P 1). Es empfiehlt sich, eine Zerlegung in kleinere Use Cases vorzunehmen, wenn dies

[15] Liegen zukünftig neue Erfahrungswerte vor, die zeigen, dass die Kenngrößen nicht gleichviel Aufwand erzeugen, so kann die Methode in einer späteren Version leicht angepasst werden.

für mehrere Use Cases einer Anwendung der Fall ist. Ist es eine „Ausnahme“, so kann das seine Berechtigung haben und mit UCP 3.0 ist dann eine Zerlegung nicht mehr notwendig.

UCP 3.0 [Points]	**UCP 1.0** [UCP]	**Komplexität**
1	5	einfach
2	10	mittel
3	15	hoch
4	-	sehr hoch
5	-	extrem hoch
>5	-	- nicht definiert -

Tabelle 15: Bewertungsskala der UCP-Methode

Ebenfalls ist es mit dem Konzept des elementaren Use Cases möglich, das Äquivalent in der Komplexitätsermittlung von Szenarien zu Schritten und von Interaktionselementen zu Schritten zu definieren. Ein Szenario ist vom Gewicht äquivalent zu drei Schritten definiert und entspricht einem Point. Ein mittlerer Dialog, eine mittlerer Schnittstelle oder ein mittlerer Bericht ist vom Gewicht äquivalent zu jeweils 2,25 Schritten oder 0,75 Points. Diese Äquivalente wurden empirisch aus der Auswertung von umfangreichen Projektdaten gewonnen [Hel08].

Ferner entspricht nun ein einfacher Actor ebenso einem Point und ist damit gegenüber der Methode UCP 1.0 um einen Faktor fünf stärker bewertet.

Der A-Faktor berechnet sich aus der Summe aller Actors A, bewertet in Points und der Summe aller Use Cases U, bewertet in Points. Sei *#A* die Anzahl aller Actors A_i und *#UC* die Anzahl aller Use Cases U_i der UCP-Sprache, dann gilt:

$$A\text{-}Faktor := \left(\sum_{i=1}^{\#A} A_i\right) + \left(\sum_{i=1}^{\#UC} U_i\right) \qquad \textit{Formel 16}$$

Im Detail wird folgendes Bewertungsschema für die Elemente des A-Faktors, also die Actors und Use Cases, zugrunde gelegt:

Actors: Ist der Akteur vom Typ *menschlicher Benutzer* bzw. *menschliche Rolle*, werden drei Points angesetzt. Handelt es sich um ein Nachbarsystem, so wird eine Bewertung gemäß Tabelle 16 vorgenommen.

Einfach (1 Point)	Mittel (2 Points)	Komplex (3 Points)
• Kommunikation über zustandslose Schnittstellen • keine Versionierung der Schnittstellen notwendig • es muss kein fachliches Transaktionskonzept beachtet werden	• **Mindestens eines** der drei Merkmale der Klasse „komplex“	• Kommunikation über zustandsbehaftete Schnittstellen • es müssen evtl. mehrere Versionen von Schnittstellen angeboten werden • Beachtung von langen fachlichen Transaktionen (über mehrere Aufrufe hinweg) erforderlich

Tabelle 16: Bewertungsskala für Akteur vom Typ Nachbarsystem

Use Cases: Die Bewertung eines Use Cases kann intuitiv durch den Schätzer gemäß der Komplexitätsklassen in Tabelle 15 erfolgen. Diese *intuitive Bewertung* macht nur Sinn, wenn ein Analogieschluss zu Use Cases möglich ist, die zuvor gemäß *exakter Zählung* bestimmt wurden. Aus Erfahrung ist zu empfehlen, dass mehr als 30% der Use Cases eines Gesamtsystems durch exakte Zählung ermittelt werden sollten.

Die *exakte Zählung* eines Use Cases erfolgt durch das getrennte Zählen seiner Szenarien, Schritte und Interaktionselemente. Diese ermittelten Anzahlen werden jeweils gemäß separater Heuristik in eine Bewertung in Points umgerechnet und aus dem Median[16] dieser drei Werte die Gesamtbewertung des Use Cases ermittelt. Abbildung 23 verdeutlicht dieses Vorgehen, es entspricht der Transformation eines Use Cases aus der UCP-Sprache in Points als FSM-Maß (siehe Abbildung 14, Seite 27).

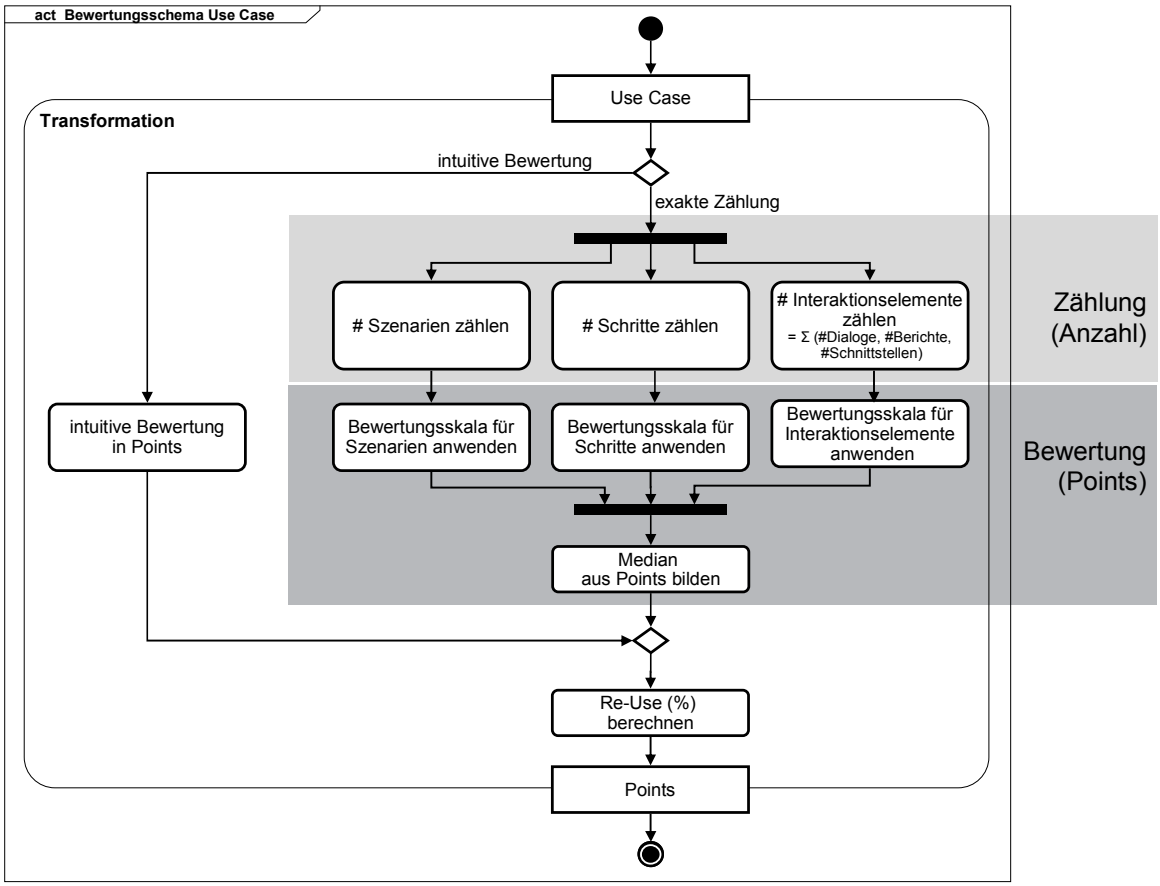


Abbildung 23: Aktivitätsdiagramm zum Bewertungsschema für Use Cases (#: = Anzahl)

Bei der Durchführung einer Use Case Bewertung kann das Ergebnis der exakten Zählung durch eine *intuitive Bewertung* überstimmt werden.

Werden Use Cases wiederverwendet (Re-Use), kann dies in der UCP-Methode auf zwei Arten berücksichtigt werden: Schritte und Interaktionselemente, die in mehreren Use Cases (nahezu) unverändert verwendet werden (d.h. bei vollständiger Wiederverwendung) sollten nur im ersten Use Case gezählt werden. Dies ist die empfohlene Vorgehensweise.

16 Für den Median wird hier die allgemein übliche Definition benutzt, siehe später auch Abschnitt 4.3.

Alternativ kann der Grad der Wiederverwendung (Re-Use in %) auf Ebene eines Use Case geschätzt werden. Dabei bezieht sich der Wiederverwendungsanteil auf alle Phasen, nicht nur die Realisierung! Wenn eigenständig spezifiziert und/oder getestet werden muss, ist der Re-Use Anteil entsprechend herabzusetzen. Hinweis: Es darf also nicht der Fehler begangen werden, als Re-Use einfach den Anteil anzusetzen, der durch die Wiederverwendung bei der Realisierung gespart wird. Re-Use in % ist besonders nützlich, wenn

- ein Use Case nur erweitert und nicht neu entwickelt wird
- ein Use Case teilweise durch ein Produkt oder Framework realisiert wird
- ein Use Case eine Spezialisierung (gemäß UML) eines anderen Use Case ist

Szenarios: Es gehen alle nach den in Abschnitt 3.1.1 angegebenen Regeln gefundenen Szenarien gleichwertig in die Bewertung ein. Die so gezählte Anzahl an Szenarien wird gemäß Tabelle 17 in Points umgerechnet, d.h. jedes Szenario wird mit genau einem Point bewertet.

# Szenarien	**Bewertung *isoliert für Szenarien*** [Points]
1	1
2	2
3	3
4	4
…	…

Tabelle 17: Bewertungsskala für Szenarien

Schritte: Es gehen alle nach den in Abschnitt 3.1.1 angegebenen Regeln gefundenen Schritte gleichwertig in die Bewertung ein. Die so gezählte Anzahl an Schritten wird gemäß Tabelle 18 in Points umgerechnet, d.h. jeder Schritt wird mit 1/3 Point bewertet und die Summe auf einen ganzen Points-Wert gerundet. Die Skala beginnt mit 2 Schritten, da der kleinste vorstellbare Use Case im Sinne des UCP Metamodells bereits mindestens 2 Schritte hat.

# Schritte	**Bewertung *isoliert für Schritte*** [Points]
2 - 4	1
5 - 7	2
8 - 10	3
11 - 13	4
…	...

Tabelle 18: Bewertungsskala für Schritte

Interaktionselemente: Es werden die Komplexitätsklassen *einfach*, *mittel* und *komplex* unterschieden und je Komplexitätsklasse die Summe der Anzahl aller Interaktionselemente (Dialoge, Schnittstellen und Berichte kumuliert) eines Use Cases gebildet. Ein einfaches

Interaktionselement zählt das ½fache eines mittleren, ein komplexes wie ein doppeltes mittleres. Damit kann die Anzahl Interaktionselemente als Äquivalente der mittleren Komplexitätsklasse bestimmt werden, die Umrechnung erfolgt gemäß Tabelle 19, d.h. jedes mittlere Interaktionselement-Äquivalent wird mit 0,75 Points bewertet und die Summe auf einen ganzen Points-Wert gerundet.

# Interaktions-elemente [mittel]	**Bewertung *isoliert für Interaktionse.*** [Points]
1	1
2 - 3	2
4	3
5	4
6 - 7	5
8	6
…	…

Tabelle 19: Bewertungsskala für Interaktionselemente

Dialoge: Es gehen alle nach den in Abschnitt 3.1.1 angegebenen Regeln gefundenen Dialoge in die Bewertung ein. Die Dialoge eines Use Cases werden entsprechend der Komplexitätsklassen für Interaktionselemente in *einfach*, *mittel* und *komplex* unterschieden. Die Tabelle 20 gibt konkrete Hilfestellung für die Festlegung der Komplexitätsklasse je Dialog.

Kriterium	**Einfach**	**Mittel**	**Komplex**
Funktionalität	Nur Anzeige	Einfaches Anzeigen, Anlegen/Ändern	Komplexes Anlegen/Ändern
Inhalte	Standard-Widgets, keine komplexeren Widgets als Listen statisches Layout	Standard-Widgets, Listen, Tabellen	Alle Arten von Widgets, inkl. Baumstrukturen und selbst zu entwickelnden Widgets adaptives Layout und/oder mehrere Layouts hochgradig interaktiv and dynamisch

Tabelle 20: Bewertungsskala für Dialoge

Abbildung 24 veranschaulicht durch Beispiele die Komplexitätsklassen von Dialogen. Um einen möglichst schnellen Überblick zu ermöglichen, sind die Beispiele bewusst „klein“ gehalten. In der Praxis wird die Anzahl der verwendeten Dialogelement (Widgets) typischerweise deutlich höher sein, was alleine die Klassifikation nicht verändert.

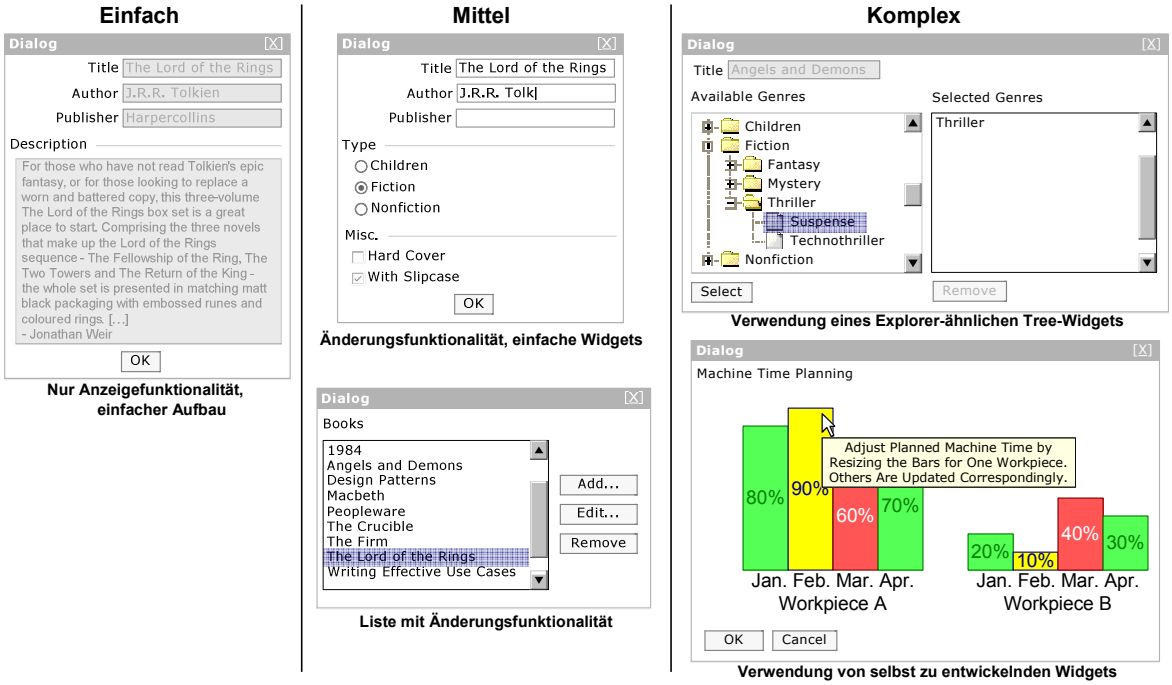


Abbildung 24: Beispiele für die Komplexitätsklassen von Dialogen

Schnittstellen: Es gehen alle nach den in Abschnitt 3.1.1 angegebenen Regeln gefundenen Schnittstellen in die Bewertung ein. Die Schnittstellen eines Use Cases werden entsprechend der Komplexitätsklassen für Interaktionselemente in *einfach*, *mittel* und *komplex* unterschieden. Die Tabelle 21 gibt konkrete Hilfestellung für die Festlegung der Komplexitätsklasse je Schnittstelle. Asynchrone Schnittstellen sind meist einfacher als synchrone, da die beiden Kommunikationsrichtungen als unabhängige Schnittstellen isoliert voneinander aufgelistet und bewertet werden können.

Kriterium	Einfach	Mittel	Komplex
Datenmodell und Mapping	Einfache 1:1-Abbildung zwischen dem internen Datenmodell des zu schätzenden Systems und dem Datenmodell der Schnittstelle	Relativ einfache Abbildung zwischen den Datenmodellen (inkl. Umverpacken und einfacher Parameterkonversion, falls nötig)	Komplizierte Abbildung zwischen den Datenmodellen nötig (die Modelle unterscheiden sich hinsichtlich der Semantik der zu übertragenden Daten)
Übertragene Daten	Flache Attribute	Hauptsächlich flache Attribute, einige Objekte	Komplexe Objekthierarchien sind beteiligt
Kommunikation (fachliche Sicht)	asynchron		synchron

Tabelle 21: Bewertungsskala für Schnittstellen

Berichte: Es gehen alle nach den in Abschnitt 3.1.1 angegebenen Regeln gefundenen Berichte in die Bewertung ein. Die Berichte eines Use Cases werden entsprechend der Komplexitätsklassen für Interaktionselemente in *einfach*, *mittel* und *komplex* unterschieden. Die Festlegung der Komplexitätsklasse je Bericht erfolgt dann in Abhängigkeit davon, wie viele der folgenden Aussagen zutreffen. Für *einfache* Berichte darf nur eine Aussage zutreffen, für *mittel* mindestens zwei Aussagen und ab drei Aussagen wird der Bericht als *komplex* eingestuft:

- Hohe Anzahl an berechneten Feldern und/oder (Teil-) Summen
- Komplizierte Anforderungen an die Formatierung (z.B. zusätzliche Statusanzeige mittels "Ampeln", pixelgenaue Positionierung, PowerPoint-Export)
- Hohe Anzahl an Parametern
- Hohe Anzahl an Gruppen (inkl. Formatierungsanforderungen pro Gruppe)
- Hohe Anzahl an Verknüpfungen (Links) zwischen den einzelnen Elementen des Berichts
- Es muss auf elementare Daten (z.B. auf Transaktionsebene) zugegriffen werden (typischerweise als Teil einer "drill-down"-Funktionalität)

Berichte, deren Komplexität über das hier beschriebene Maß hinausgeht, sollten separat geschätzt werden.

3.1.3 Mapping der Spezifikationsbausteine (Anwendungsfälle) auf den A-Faktor

Wie im vorherigen Abschnitt beschrieben, wird zur Bestimmung des A-Faktors die Anzahl der Actors und Anzahl der Use Cases in den Komplexitätsklassen *einfach*, *mittel* und *komplex* benötigt. Die Komplexität des Use Cases wird durch das Zählen der Schritte, Szenarien und Interaktionselemente festgelegt. Liegt eine Spezifikation in Anwendungsfällen vor (siehe Abschnitt 2.3), kann die Anzahl in der Regel direkt abgelesen werden, in dem ein Akteur aus der Spezifikation 1:1 einem Actor der UCP-Methode zugeordnet wird und ein Anwendungsfall der Spezifikation 1:1 einem Use Case. Die Komplexität eines Actors oder eines Use Case wird z.B. aus der Anzahl seiner Schritte, Szenarien etc. bestimmt oder aufgrund der Erfahrung des Schätzers „intuitiv" bewertet.

Die konkrete Abbildung („Mapping") von Elementen der Spezifikation aus Kapitel 2.3 auf das Use Case Modell für die UCP-Schätzung beschreibt Tabelle 22.

Liegt innerhalb der Spezifikation kein Anwendungsfall-Modell vor, muss zunächst aus der vorhandenen Grobspezifikation ein (rudimentäres) Modell erstellt werden. In Abschnitt 3.4 wird dies detailliert beschrieben. Die Use Case Modellierung muss dabei nur so weit vorgenommen werden, dass die Kenngrößen (Actors; Szenarien, Schritte, Interaktionselemente) feststehen. Dies ist häufig viel einfacher im Rahmen der „Zählung" zu bewerkstelligen als ein Anwendungsfall-Modell vollständig zu erstellen.

Die schematische Darstellung der Transformation von einer Spezifikation auf die UCP-Sprache (Abbildung 14, Seite 27) wird in Abbildung 25 nun ergänzt um die Transformationsvorschriften für Anwendungsfall-basierte und weitere Spezifikationsformen. Für die Anwendungsfälle der Spezifikationsbausteine aus Abschnitt 2.3 wurde mit Tabelle 22 bereits eine formale Zuordnungstabelle angegeben. Für die Anwendungsfall-basierten Spezifikationsformen können nur Regeln anstelle einer eindeutigen Zuordnungstabelle formuliert werden. Dies erfolgt in den folgenden Abschnitten, die Abschnittsnummern sind in Abbildung 25 jeweils kursiv dargestellt. Für die weiteren Spezifikationsformen können keine konkreten Regeln angegeben werden, da die Spezifikationsformen zu unspezifisch sind. Hier werden wir uns auf Hinweise beschränken.

Begriff aus:		Besonderheiten
Spezifikation	**UCP-Sprache**	
Akteur	Actor	Grundsätzlich bezeichnen beide Begriffe dasselbe. Für die UCP-Sprache wird zur Unterscheidung der englische Begriff verwendet. Es brauchen Primär- und Sekundärakteure nicht unterschieden zu werden.
Anwendungsfall	Use Case	Grundsätzlich bezeichnen beide Begriffe dasselbe. Für die UCP-Sprache wird zur Unterscheidung der englische Begriff verwendet.
Aktion	Schritt	Grundsätzlich bezeichnen beide Begriffe dasselbe. Zum Zählen von Schritten ist allerdings die Definition der UCP-Methode maßgeblich. Nutzer- und Anwendungsschritte (bzw. –aktionen) brauchen nicht unterschieden zu werden.
Anwendungsfunktion	Schritt	
Szenario	Szenario	(Fachliche) Fehlerszenarien sind für die UCP-Methode nur dann relevant, wenn auch fachliche Schritte zur Fehlerbehandlung durchlaufen werden.
Interaktion	Interaktionselement	„Interaktionselement“ ist der Oberbegriff für Dialoge, Schnittstellen und Berichte, umfasst aber im Gegensatz zum Begriff Interaktionen keine Batchverarbeitung.
(Teil-) Dialog	Dialog	Bei UCP bezieht sich der Begriff Dialog immer auf einen abgeschlossenen und eigenständigen Bildschirmbereich (Reiter, Gruppe, Frame, Fenster, etc.), also auf einen Teildialog im Sinne der Spezifikationsmethodik.
Nachbarsystem-schnittstelle	Schnittstelle	Jeder fachliche Dienst einer Nachbarsystemschnittstelle entspricht genau einer Schnittstelle im Sinne der UCP-Methode.
Druckausgabe	Bericht	Die statische Druckausgabe ist die klassische Form eines Berichts, die UCP-Sprache umfasst aber auch dynamische (filterbare, verlinkte, etc.) formatierte Präsentationen von Daten und berechneten Ergebnissen.
Batch	Use Cases	Batches werden für die UCP-Methode als Use Cases interpretiert und bewertet.

Tabelle 22: Mapping-Tabelle für Anwendungsfälle einer Spezifikation

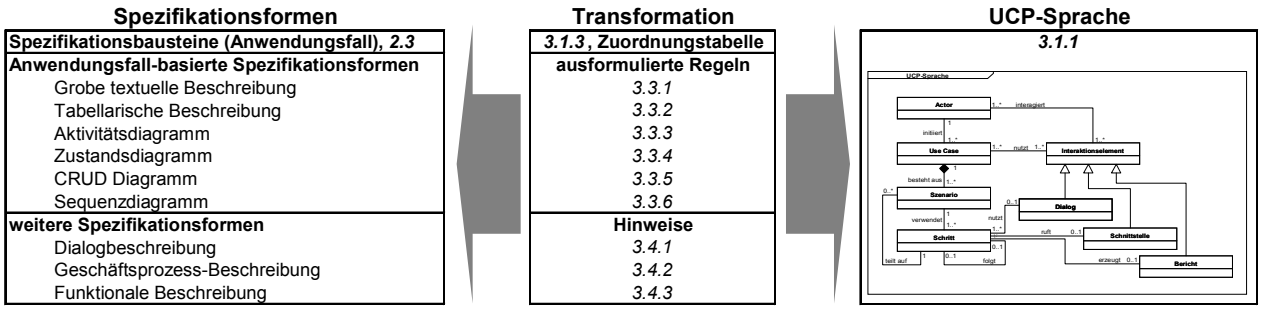


Abbildung 25: Mapping unterschiedlicher Spezifikationsformen auf die UCP-Sprache, Verweise auf Abschnitte dieser Arbeit sind kursiv gedruckt

3.2 Leitfaden zur Use Case Identifikation

Ein wesentlicher Kritikpunkt an der UCP-Methode nach Karner ist, dass Use Cases in unterschiedlicher Granularität beschrieben werden können und dies unmittelbar Einfluss auf das Schätzergebnis hat (siehe Seite 33, P 1). Nachfolgend wird auf Basis von zahlreichen Projektbeispielen ein Leitfaden zum einheitlichen "Schneiden" von Use Cases für die Anwendung der UCP-Methode gegeben. Zunächst werden in diesem Abschnitt allgemeine Hinweise gegeben. In Abschnitt 3.3 wird dann auf Spezifikationsformen beispielhaft eingegangen, welche sich nahe an einer Anwendungsfall-Beschreibung orientieren. In Abschnitt 3.4 werden andere Spezifikationsbeispiele betrachtet, die nicht auf Anwendungsfällen basieren. Die Auswahl der behandelten Spezifikationsformen fokussiert dabei auf jene Formen, die in der Praxis der Software-Entwicklung betrieblicher Informationssysteme häufig[17] angetroffen werden.

Basis für eine solche Abschätzung ist immer mindestens eine vorliegende (Grob-) Spezifikation der Anwendung. Dabei ist es unerheblich, in welcher Form und Granularität diese Spezifikation vorliegt. Entscheidend für den Erfolg der Schätzung und die Belastbarkeit des Ergebnisses ist vor allem, ob und wie es gelingt, die vorliegende Dokumentation auf die hier beschriebenen Use Cases abzubilden. Im Idealfall entspricht die Spezifikation den aufgeführten Anforderungen und es ist nur noch eine quantitative Bewertung erforderlich.

Es nicht Ziel dieses Abschnittes, ein einheitliches Vorgehen bei der Beschreibung von Anwendungsfällen innerhalb der Spezifikation zu definieren. Es wird davon ausgegangen, dass im Zuge der Schätzung keine Anwendungsfall-Beschreibung neu erstellt wird. Daher werden auch formale Aspekte einer Anwendungsfall-Beschreibung nicht weiter betrachtet (siehe hierzu [Boe07, Cle05, HRZ06, JFB03]).

Eine konkrete Anwendungsfall-Beschreibung kann und darf sowohl in Form als auch in der Granularität durchaus von den hier genannten Kriterien abweichen, ohne dass die Anwendbarkeit der UCP-Methode eingeschränkt wird.

Die Abschätzung des funktionalen Umfanges ist ein wesentlicher Schritt zur Bestimmung des Aufwandes zur Umsetzung eines Softwareprojektes. Daneben spielen andere Kostenfaktoren eine Rolle, auf die hier nicht eingegangen wird. Verschiedene dieser Einflussgrößen können gegen eine Anwendbarkeit der UCP-Methode sprechen oder zumindest das Ergebnis stark verfälschen oder zu einer mit großer Unsicherheit behafteten Schätzung führen. Dies trifft auch für einige später genannte Kriterien im Bereich der Use Cases zu.

Ganz allgemein kann an dieser Stelle bereits gesagt werden, dass Schwierigkeiten beim Finden der Use Cases oder ihrer Bewertung darauf hindeuten, dass nicht genügend Informationen zur Abschätzung des funktionalen Umfanges vorhanden sind. In diesen Fällen wird auch die UCP-Methode keine zuverlässigen Werte liefern können.

---

[17] Dazu wurden ca. 50 verschiedene Software-Entwicklungsprojekte bei Capgemini sd&m ausgewertet und die häufigsten Formen ausgewählt. Die Spezifikationen stammen i.d.R. aus Projekten bzw. Ausschreibungen, die große deutsche Unternehmen aus allen Branchen ausgeschrieben haben. Aus Gründen des Datenschutzes können keine Kundennamen genannt werden. In Abschnitt 4.2 wird die Auswahl der Spezifikationsformen wieder aufgegriffen werden.

3.2.1 Zweckmäßige Use Case Granularität

Das Finden des richtigen Schnittes der Use Cases ist für die Schätzung ein ganz entscheidender Punkt. Ein genaues Maß für eine ausgewogene Use Case Beschreibung kann aber nicht angegeben werden, daher hier einige Richtwerte.

Ein Zeichen von zu komplexen (oder auch zu einfachen) Use Cases kann die Größe der betreffenden Beschreibungen im Spezifikationsdokument liefern. Wir haben verschiedene Kriterien, um an der Beschreibung festzustellen, ob der Umfang eines Szenarios zu groß gewählt ist:

- Die textuelle Beschreibung eines Szenarios umfasst mehr als eine DIN-A4-Seite oder
- Ein Szenario enthält mehr als 16 Schritte oder
- Ein Use Case beinhaltet mehr als fünf Szenarien. In diesem Fall sind die einzelnen Szenarien von der Gewichtung her eher als eigene Use Cases zu betrachten.
- Ein Use Case enthält mehr als 7 Interaktionselemente

Übersteigt ein Use Case diese Werte wesentlich, ist er für die Schätzung zu umfangreich und sollte zerlegt werden. Dies sind Erfahrungen aus nachgeschätzten Projekten. Die angegebenen Grenzwerte resultieren einzeln bereits in 5 Points für den Use Case. Die Grenzwerte, wie alle angegeben Grenzwerte, sollten im konkreten Fall nicht blind befolgt, sondern anhand eigener Erfahrungen oder anderer Use Cases überprüft werden (siehe später Abschnitt 6.4).

Ebenso sind sehr kleine Use Cases, die sich vor allem in einer geringen Anzahl von Schritten, Szenarien und Dialogen zeigen, ein Anzeichen für einen potentiell falschen Schnitt. Zeichen dafür sind:

- Die Beschreibung eines Anwendungsfalls umfasst nur wenige Zeilen (Vorsicht, in einer Grobspezifikation kann dieses Kriterium sehr häufig zutreffen, wenn in einem Satz viele Aktionen einfach nur aufgezählt werden, in diesem Fall kann der Schnitt trotzdem korrekt sein)
- Der Anwendungsfall enthält kein Interaktionselement und
- der Anwendungsfall hat nur ein Szenario und
- der Anwendungsfall hat nur einen oder zwei Schritte

Hier sollte noch einmal überlegt werden, ob es sich eventuell nicht eher um eine Anwendungsfunktion oder einen Teilschritt eines übergeordneten Anwendungsfalles handelt (z.B. "Suchen").

Wichtig ist aber auch, dass grundsätzlich „Ausreißer“ erlaubt sind. Auch hier eine hilfreiche Maßzahl: Es sollten sich ca. 80%-90% aller vorhandenen Use Cases in diesen Grenzen bewegen.

Die restlichen Fälle sind Ausnahmen, die sehr groß oder sehr klein sind, insgesamt aber vom Schnitt her zu den anderen "passen". Insgesamt ist eine ausgewogene Verteilung von kleinen, mittleren oder großen Use Cases ein Zeichen für einen "guten" Schnitt.

Es sei hier noch angemerkt, dass die Kriterien für die Größe der textuellen Beschreibung nur für eine schon als detaillierte Anwendungsfall-Beschreibung vorliegende Spezifikation gelten. Die

Angaben bezüglich der Anzahlen gelten natürlich auch nach einem aus einer anderen Form der Spezifikation erfolgten Mapping (siehe Abschnitt 3.4).

Die Wiederholbarkeit und Konsistenz bzgl. einer konkreten Schätzung ist dann sehr hoch, wenn die in diesem Leitfaden vorgestellten Kriterien zum Finden und Zählen von Use Cases bereits bei der Modellierung der Anwendungsfälle in der Spezifikation berücksichtigt wurden und alle Anwendungsfälle nach einheitlichen Regeln „geschnitten“ sind. Eine Gefahr unterschiedlicher Granularität besteht vor allem

- wenn mehrere (Teil-) Spezifikationen aus unterschiedlichen Projektphasen vorliegen
- wenn unterschiedliche Autoren an (Teil-) Spezifikationen gearbeitet haben
- Dokumente unter Zeitdruck erstellt wurden und gegen Ende der Grad der Detaillierung nachlässt

Es empfiehlt sich daher, im Verlaufe einer UCP-Schätzung immer wieder den "Schnitt" von Use Cases und Schritten zu vergleichen.

Besteht zumindest Konsens oder eine gewisse Sicherheit, dass alle Use Cases gefunden und damit benannt und gezählt werden können und die Szenarien, Schritte und Interaktionselemente können nicht benannt werden, so kann ein erster grober Entwurf durchaus in einer pauschalen Bewertung der Komplexität *aller* Use Cases mit einem festen Wert von 2 Points (= mittlerer Komplexität) liegen. In der Regel hat man damit eine grobe Abschätzung. Dieser Wert hat sich vor allem bei größeren Projekten, d.h. größer 5.000 Bearbeitertage (BT) als relativ treffend erwiesen. Bei mittleren und kleineren Projekten sind die Use Cases in der Regel einfacher geschnitten, hier ist ein Durchschnittswert von ca. 1,5 Points angemessen. Tabelle 23 fasst die entsprechende Auswertung von 17 Projekten zusammen.

Eine Einführung in die Statistik wird in Abschnitt 4.3 gegeben werden, hier reicht zunächst das Verständnis, dass der Variationskoeffizient ein Maß für die relative Streuung des Mittelwertes ist. 18,9 Bearbeitertage entsprechen einem Point (UUCP). Man beachte jedoch, dass die Schwankung über die Projekte mit 35% relativ hoch ist und von daher dies nur sehr grobe Angaben sein können. Bei starker Wiederverwendung von fachlicher Funktionalität (Abschnitt 3.2.6) in unterschiedlichen Use Cases sind z.B. die einzelnen Use Cases im Durchschnitt eher als leicht (d.h. 1 Point) einzustufen.

Projektgröße	Größe [BT]	# Projekte	*μ* UC [Points]	*V* UC [%]	*μ* Actor [Points]	*V* Actors [%]
Klein	625 – 2.500	9	1,4	26%	2,1	15%
Mittel	2.500 – 5.000	2	1,2	19%	2,0	27%
Groß	> 5.000	5	1,9	37%	2,6	9%
Alle		**16**	**1,5**	**35%**	**2,3**	**17%**

Tabelle 23: Mittelwert μ in Points und Variationskoeffizient V als relatives Streumaß in Prozent von Use Cases (UC) und Actors in Abhängigkeit der tatsächlichen Projektgröße in Bearbeitertagen (BT)

Analog können Actors für kleine und mittlere Projekte mit 2 Points und für größere Projekte mit 2,6 Points veranschlagt werden. In großen Projekten dominieren i.d.R. doch komplexe Schnittstellen zu Nachbarsystemen (= 3 Points).

3.2.2 Berücksichtigung impliziter Funktionalität

Häufig kommt es vor, dass in einer Spezifikation neben der reinen fachlichen Funktionalität in gesonderten Kapiteln weitere "Nebenfunktionalität" beschrieben wird. Dazu gehören z.B.

- Bereinigungsläufe, Migrationsprogramme
- administratorische Funktionalität, Verwaltungs- und Konfigurationsoberflächen

Diese Funktionalität muss selbstverständlich auch berücksichtigt werden. Es gibt auch keinen Grund, warum sich solche Funktionen nicht als Use Cases formulieren lassen sollten. In diesem Fall sind das z.B. ein Use Case „Nächtliche Bereinigung der Kontodifferenzen“, „Migration der Adressdaten“ oder „Konfiguration der Metadaten“. Entscheidend ist die vollständige Abdeckung der gesamten spezifizierten (und implizit angenommenen) Anwendungsfunktionalität mittels der erfassten Use Cases.

3.2.3 Use Case Beziehungen

Im Rahmen der Anwendungsfall-Modellierung sind <<include>> Beziehungen möglich (siehe Abschnitt 2.3.2). Verboten im Sinne einer guten und lesbaren Spezifikation sind Verzweigungen auf einzelne Szenarien oder Schritte eines importierten Anwendungsfalls – der importierte Anwendungsfall wird immer vollständig durchlaufen.

Für die UCP-Methode ist der Umgang mit solchen importierten Anwendungsfällen relativ einfach folgendermaßen geregelt:

- der importierte Anwendungsfall wird als eigener Use Case gezählt
- der Aufruf des importierten Anwendungsfalles und die Schaffung der Vorbedingungen für diesen zählt als ein Schritt im aufrufenden Anwendungsfall
- die Szenarien, Aktionen und Interaktionselemente aus dem importierten Anwendungsfall werden nur dort und nicht im aufrufenden Anwendungsfall gezählt (und damit über die gesamte Anwendung hinweg betrachtet auch nur einmal)
- eine Wiederverwendung (siehe auch später Abschnitt 3.2.6) kommt bei dem Use Case des aufrufenden Anwendungsfalles nicht zur Anwendung

Diese Regeln sind auch anzuwenden, wenn möglicherweise die Spezifikation nicht der Regel des vollständigen Importes eines Anwendungsfalles folgt oder eine grundsätzlich andere nicht UML-konforme Spezifikationsform vorliegt, die dann auf Use Cases für die UCP-Methode abgebildet werden muss (*Mapping*).

Eine intensive Verwendung von <<include>> Beziehungen von Anwendungsfällen in einer Spezifikation kann ein Hinweis auf einen noch nicht passenden "Schnitt" der Use Cases für die Schätzung sein (muss es aber nicht!). Wir unterscheiden zwei Fälle:

- Es liegt eigentlich eine Beschreibung auf Ebene *Summary Goal* vor [Coc03], siehe auch Abschnitt 2.3.2. In diesem Fall können bei der Anwendung der UCP-Methode die Anwendungsfälle der Summary Goal Ebene weggelassen werden und die eigentlichen Anwendungsfälle der User Goal Ebene betrachtet werden. Allerdings sind dann auch die Hinweise bzgl. des Umganges mit Grobbeschreibungen aus 3.3.1 und 3.4.2 zu beachten.
- Der zweite (entgegen gesetzte) Fall ist eine bereits vorgenommene starke Zerlegung der Anwendungsfälle auf der Ebene *Subfunctions*. Die Anwendungsfälle werden dann also eher auf Schritte (in diesem Fall Anwendungsfunktionen) und nicht auf Use Cases abgebildet. Hier entsteht die Hierarchie und Verzahnung durch die Wiederverwendung von Funktionalität. In diesem Fall gelten die Hinweise in Abschnitt 3.2.6.

Abbildung 26 zeigt ein typisches Beispiel aus einer realen Spezifikation mit einer oft anzutreffenden Konstellation mit einem zentralen Anwendungsfall zum „Suchen“, von dem aus wiederum andere Anwendungsfälle aufgerufen werden. Dieser aufrufende Anwendungsfall führt selbst noch zu keinem fachlichen Ziel. Man könnte diesen ersten Teil auch als Schritt jedes einzelnen aufgerufenen Anwendungsfalls auffassen. Dann hätte man aber das Problem der Zählung der Wiederverwendung dieses ersten Schrittes und außerdem entspricht bei solchen komplexen Suchfunktionen die Gewichtung als eigener Use Case eher dem wirklichen Aufwand. In diesem Fall besteht der Use Case „Förderer Suchen“ neben den hier noch nicht ersichtlichen Schritten zum eigentlichen Suchen dann letztendlich aus den fünf Szenarien zum Aufrufen der importierten Anwendungsfälle und somit auch aus fünf zusätzlichen Schritten (neben dem Suchen).

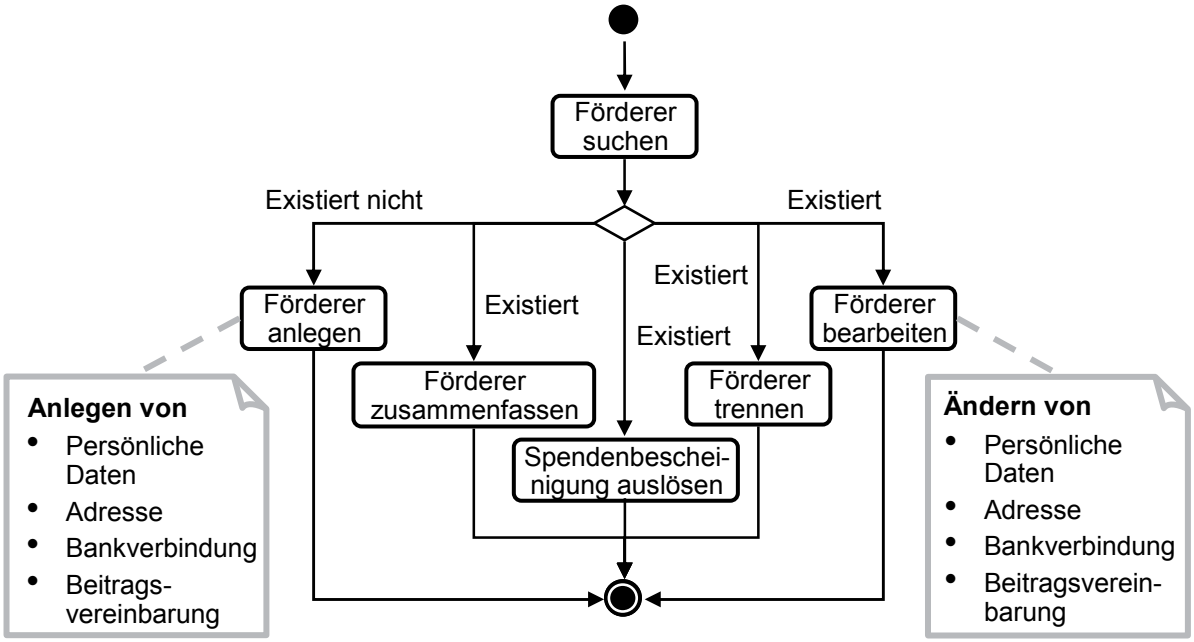


Abbildung 26: Beispiel für einen Anwendungsfall „Suchen“

3.2.4 Verwendung von Anwendungsfunktionen

Anwendungsfunktionen beschreiben Ausschnitte von Anwendungsfällen, die ohne Unterbrechung vom System, ggf. unter Benutzung von Schnittstellen zum Nachbarsystem, ausgeführt werden. Grundsätzlich wird eine Anwendungsfunktion als Schritt eines Use Cases in der UCP-Methode interpretiert. Eine Anwendungsfunktion hat also ein nicht so stark definiertes fachlich abgeschlossenes Ziel. Nichtsdestotrotz können solche Anwendungsfunktionen eine sehr hohe Komplexität haben, daher wollen wir hier den Umgang mit ihnen erläutern.

Wir unterscheiden dabei zwei Fälle:

1. **Unabhängig von Anwendungsfällen spezifizierte Anwendungsfunktionen**

   Komplexere systemseitige Funktionen, vor allem wenn sie häufig wiederverwendet werden oder wenn die vorliegende Beschreibung bereits funktional zerlegt ist, sind in der Regel separat beschriebene Abläufe von fachlichen Berechnungen oder z.B. Prüfungen und Batchabläufe (siehe auch nächster Abschnitt).

   Hier kann es sein, dass es sich um eine eigentlich in anderen Anwendungsfällen bereits verwendete aber nicht explizit referenzierte Funktionalität handelt. Bei komplexen Anwendungsfunktionen ist dann zu überlegen, ob es sich nicht eher um einen eigenen Use Case handelt, der via <<include>> Beziehung entsprechend dem vorherigen Abschnitt modelliert werden sollte.

   Falls die beschriebene Anwendungsfunktion für sich alleine steht, sollte sie als eigener Use Case in die Schätzung aufgenommen werden. Grundsätzlich soll die gesamte Funktionalität der Anwendung mit der UCP-Methode erfasst werden. In der Regel ist eine einzelne, fachlich komplexe Anwendungsfunktion immer auch als Use Case darstellbar.

2. **In Anwendungsfällen aufgerufene komplexe Anwendungsfunktion**

   Der zweite Fall sind bereits in Anwendungsfällen verwendete oder referenzierte Anwendungsfunktionen, die aufgrund ihrer Komplexität eigentlich zu groß für einen Schritt eines Use Cases sind. Dazu zählen z.B. in einem einzigen Schritt ausgeführte Prüfungen, danach komplexe Berechnungen und evtl. noch die Erzeugung bzw. Veränderung einer größeren Anzahl von Entitäten. In diese Kategorie fallen ebenso umfangreiche Regelwerke oder auch Funktions- und Entscheidungsmatrizen. In solchen Fällen schlagen wir eine Vergleichsgewichtung gegenüber anderen ("klassischen") Use Cases aus dem Projekt vor. Das bedeutet, dass die komplexen Schritte noch einmal aufgeteilt werden können in einzelne kleinere Schritte. Diese sollten vom fachlichen Umfang her den anderen (serverseitigen) Anwendungsfunktionen entsprechen und zumindest ein fachlich abgrenzbares Teilergebnis produzieren. Bei sehr komplexen Anwendungsfunktionen sollte überlegt werden, ob es sich nicht eher um einen eigenen Anwendungsfall handelt, der ebenfalls via <<include>> Beziehung eingebunden wird.

Anwendungsfunktionen werden häufig dann beschrieben, wenn es um Wiederverwendung geht. Es soll eine mehrfache Beschreibung gleicher Funktionalität in mehreren Use Cases vermieden werden. In solchen Fällen wird eine Anwendungsfunktion als einzelner Schritt im Use Case nur

einmal gezählt. Wird eine solche Anwendungsfunktion in mehreren Anwendungsfälle referenziert, soll sie nur bei einem dieser Anwendungsfälle (üblicherweise dem ersten) als Schritt gezählt werden. In diesem Fall kann die Auslagerung in einen eigenen (Unter-) Anwendungsfall eine einfache Lösung sein.

Die Schwierigkeit liegt hierbei vor allem im Erkennen, dass eine Anwendungsfunktion bereits als Schritt in einem anderen Use Case gezählt wurde. Eine Möglichkeit der Berücksichtigung von Anwendungsfunktionen ist der Einsatz des Parameters "Wiederverwendung" in der UCP-Schätzmethodik. Darauf wird im Abschnitt 3.2.6 noch detaillierter eingegangen werden.

3.2.5 Berücksichtigung der Batch-Funktionalität

Häufig werden serverseitig komplexe Funktionen oder auch Batchabläufe, die ohne Interaktion mit anderen Abläufen oder Akteuren durchgeführt werden, nicht als „klassische“ Anwendungsfälle spezifiziert. Trotzdem handelt es sich oft um Anwendungsfälle, sie haben einen definierten Start mit Vorbedingungen (das ist häufig sogar einfacher zu bestimmen als bei „klassischen“ Anwendungsfällen), es gibt Ablaufszenarien und ein bestimmtes fachliches Ziel.

Die Herausforderung besteht in diesen Fällen eher darin, den Ablauf für die Schätzung in einzelne Schritte der richtigen Größe zu zerlegen. Im Prinzip gelten hier alle Hinweise über die Schritte wie auch bei anderen Use Cases:

- Wenn sehr komplex, dann Zerlegung in Use Cases entsprechend der fachlichen Blöcke.
- Schritte können ebenfalls anhand fachlicher Blöcke gefunden werden, anhand des Zugriffs auf Schnittstellen oder anhand der verarbeiteten Entitäten.
- Ein einzelner Schritt kann hier eine geringfügig größere Mächtigkeit (=abgedeckte Funktionalität) haben als ein Schritt eines Nicht-Batch Use Cases, da es in der Regel keinen Actor-Eingriff und damit keine komplexe Schnittstelle oder z.B. einen Schichtenwechsel innerhalb der Anwendungsarchitektur geben wird.
- Generell gilt, dass Schritte innerhalb eines Use Cases über alle Use Cases einer Schätzung hinweg und (idealerweise) über alle UCP-Schätzungen hinweg etwa gleich groß sein sollten.

Ist für einen Batch eine solche Zerlegung in Use Cases gefunden worden, wird er genauso gezählt wie ein "normaler" Use Case, d.h. es werden Szenarien, Schritte und Interaktionselemente (hier i.d.R. keine Dialoge, sondern eher Berichte oder komplexe Ausgabedateien über Schnittstellen) gezählt.

Fazit: Das in den letzten beiden Abschnitten beschriebene Vorgehen bezüglich des Umganges mit komplexen Anwendungsfunktionen weicht zum Teil die Definition des Use Cases (fachliches Ziel wird erreicht) oder aber des Schrittes im Use Case auf (Abgrenzung zum nächsten Schritt ist nicht so deutlich vorhanden).

3.2.6 Re-Use

Die UCP-Methode sieht die Bewertung von Re-Use vor, also die Erfassung von wiederverwendeter Funktionalität vor. Hier kann der Schätzer den prozentualen Anteil des Use Cases angeben, dessen Funktionalität bereits anderweitig in der Schätzung abgedeckt ist und damit nicht mehr umgesetzt und getestet werden muss.

Damit ist insbesondere gemeint:

- Der Anwendungsfall wird im Projekt nur erweitert, nicht vollständig neu entwickelt, dieser Fall tritt insbesondere bei Folge- oder Wartungsprojekten häufig ein.
- Der Anwendungsfall wird fachlich zum Teil bereits durch eine Produktsoftware oder ein Framework realisiert.
- Der Anwendungsfall ist eine Spezialisierung (im OO-Sinne) eines anderen Anwendungsfalles.
- Bestimmte Aktionen des Anwendungsfalles (speziell wenn es sich um Anwendungsfunktionen handelt) werden bereits in anderen Anwendungsfällen verwendet.

Die Abschätzung des Grades der Wiederverwendung ist in der Regel nicht einfach. Ein hoher Anteil an mit Re-Use geschätzten Use Cases ist ebenfalls ein Hinweis auf eine potentiell kritische Schätzung. Zu berücksichtigen ist ebenfalls, dass sich Re-Use auf alle Phasen beziehen muss. Falls es nur um die reine Realisierung geht, die wiederverwendete Funktionalität aber getrennt spezifiziert und getestet werden muss, ist der Grad der anzusetzenden Wiederverwendung geringer.

Speziell das letzte angeführte Kriterium kann in Schätzungen problematisch werden, wenn die Spezifikation bereits einen hohen Detaillierungsgrad aufweist. Im Falle einer starken "Verzahnung" der Anwendungsfälle und damit auch einer gehäuften Wiederverwendung von Anwendungsfunktionalität empfehlen wir, diese Aktionen dann nur einfach beim ersten verwendenden Use Case zu zählen und nicht über den Parameter Re-Use zu steuern.

Beispiel: Eine Anwendungsfunktion "Kunde suchen" wird in vielen verschiedenen Anwendungsfällen als Aktion verwendet. Für die Schätzung gibt es jetzt verschiedene Möglichkeiten:

- Es handelt sich eigentlich um einen eigenen Anwendungsfall (eigener Dialog, mehrere Szenarien, Aktionen). In diesem Fall wird er auch als eigener Use Case geschätzt und in den anderen als importierter Anwendungsfall berücksichtigt (siehe auch Abschnitt 3.2.3).
- Es ist kein eigener Anwendungsfall und die Aktion "Kunde suchen" wird beim ersten verwendenden Anwendungsfall als Schritt gezählt, in den weiteren wird er als Re-Use prozentual abgezogen (z.B. ein Use Case mit 5 Schritten, von denen dann einer die Suche ist, hat dann 20% Re-Use, ein anderer Use Case mit 4 Schritten entsprechend 25% Re-Use). Dieses Verfahren kann für einzelne Fälle schnell angewendet werden, bei häufigem Auftreten ist es aber sehr mühsam und bei weiteren wieder verwendeten Anwendungsfunktionen besteht die Gefahr der Unübersichtlichkeit.
- Es ist kein eigener Anwendungsfall und die Aktion "Kunde suchen" wird beim ersten verwendenden Anwendungsfall als Schritt gezählt, in den weiteren wird dieser Schritt nicht

mehr gezählt (z.B. ein Use Case mit 5 Schritten, von denen dann einer die Suche ist, hat dann noch 4 gezählte Schritte). Dieses Verfahren hat sich in der Praxis bewährt und wird bei Wiederverwendung einzelner Schritte empfohlen. Es entspricht auch dem Vorgehen, wenn innerhalb eines Use Cases Schritte mehrfach verwendet werden.

Der prozentuale Faktor Re-Use wird immer zum Schluss der Bewertung eines Use Cases mit dem Points-Wert multipliziert (siehe Abbildung 23, Seite 59). Dies geschieht bei Bestimmung der Use Case Komplexität sowohl durch *exakte Zählung* als auch bei *intuitiver Bewertung*. Ein nach bereits genannter Formel berechneter Use Case mit zum Beispiel 2 Points und einer Wiederverwendung von 20% wird also mit 1,6 Points in die Schätzung eingehen. Werden anstelle der exakt gezählten 2 Points intuitiv 1 Point bestimmt, bleiben bei weiterhin angenommenen 20% Wiederverwendung also noch 0,8 Points übrig. Das bedeutet: Wiederverwendung geht grundsätzlich in die Schätzung ein, die intuitive Bestimmung bezieht sich auf den Use Case ohne Wiederverwendung!

3.2.7 Erfassung von Actors für die Schätzung

Bei der Erfassung der Actors kommen im Prinzip folgende Varianten vor:

- Es liegt bereits eine Liste aller Akteure als Zusammenfassung der Spezifikation bei. In diesem Fall kann diese Liste direkt für die Actors übernommen werden.
- Für jeden einzelnen Anwendungsfall sind Akteure angegeben. In diesem Fall muss darauf geachtet werden, dass gleichartige Akteure zu einem Actor zusammengefasst werden. Es kann allerdings vorkommen, dass gleich bezeichnete Akteure in unterschiedlichen Anwendungsfällen als getrennte Akteure betrachtet werden müssen. So kann z.B. ein Administrator im Kontext eines bestimmten Anwendungsfalles eine andere Rolle wahrnehmen als in einem anderen Anwendungsfall.
- Akteure sind nicht angegeben. In diesem Fall sind sie im Laufe der Use Case Zählung "aufzusammeln" und analog wie bei der Variante mit der einzelnen Beschreibung einzuordnen.

3.3 Mapping von Anwendungsfall-basierten Spezifikationsformen

Für die Dokumentation von Anwendungsfällen gibt es in der industriellen Praxis keine einheitliche Verwendung bestimmter Spezifikationsformen. Es wurden die häufigsten Formen aus der Praxis ausgewählt. Im Folgenden wird erläutert, wie aus den verschiedenen Spezifikationsformen, die auf Anwendungsfällen basieren, eine Zählung für die UCP-Methode gefunden werden kann. Andere Spezifikationsformen werden dann in Abschnitt 3.4 behandelt. Einen guten Überblick zu den nachfolgenden Abschnitten gibt Abbildung 25 auf Seite 64.

3.3.1 Grobe textuelle (Anwendungsfall-)Beschreibung

In der Praxis kann es häufig vorkommen, dass eine Anwendungsfall-Beschreibung erst in einer sehr groben Form vorliegt und (noch) keiner formalen Sprache folgt. Das wird in der Regel in

frühen Projektphasen der Fall sein. Dann kann z.B. eine Übersicht der Benutzer-Rollen und eine weitere Übersicht der Anwendungsfälle mit einer kurzen textuellen Beschreibung oder eine Geschäftsprozessbeschreibung inklusive der Benennung der Anwendungsfälle vorhanden sein.

Die so benannten Akteure und Anwendungsfälle geben bereits eine gute Basis für die Schätzung nach UCP. Sie können als erster Schritt direkt als Use Cases übernommen werden. Tabelle 24 fasst dafür die Mapping-Regeln zusammen. Im zweiten Schritt ist dann je nach Detaillierung der Schätzung eine Gewichtung der Komplexität nur nach intuitiver Einschätzung möglich oder es können aus der textuellen Beschreibung Hinweise auf die Schritte entnommen werden. In diesem Fall sind die Regeln aus Abschnitt 3.2.1 zu beachten.

Spezifikation: Grobe textuelle Beschreibung	UCP-Sprache
Beschreibung von Nutzern	Actor
Textuelle Liste der Anwendungsfälle	Use Case
-	Szenario
-	Schritt
-	Dialog
-	Schnittstelle
-	Bericht

Tabelle 24: Mapping-Tabelle für Grobe textuelle Beschreibungen

Liegt eine Grobspezifikation ohne Benennung von Anwendungsfällen vor, ist der erste Schritt auf jeden Fall das Finden und Aufzählen der Use Cases selbst. Je nach Detaillierung der Spezifikation kann dabei eine der im Abschnitt 3.4 beschriebenen Mapping-Methoden helfen. Häufig wird der zweite Schritt aber in der Durchführung einer intuitiven Bewertung der Komplexität bestehen.

Ähnlich wie bei einer Expertenschätzung gibt es auch hier den Fall, dass zu wenig Information für eine Schätzung vorliegt und somit eine Schätzung nicht möglich ist.

Besteht zumindest Konsens oder eine gewisse Sicherheit, dass alle Use Cases gefunden und damit benannt und gezählt werden können und die Szenarien, Schritte und Interaktionselemente können nicht benannt werden, so kann für eine erste groben Schätzung durchaus eine pauschale Bewertung der Komplexität *aller* Use Cases mit einem fixen Points-Wert gemäß Abschnitt 3.2.1 vorgenommen werden.

3.3.2 Tabellarische Beschreibung

Eine textuelle vollständige Beschreibung von Anwendungsfällen wurde bereits in Abschnitt 2.3.2, Tabelle 6 vorgestellt und ist in der Regel in Form einer Tabelle aufgeschrieben, wir nennen sie daher *Tabellarische Beschreibung*. Sie enthält in der Regel die folgenden Informationen bzw. mindestens eine Untermenge davon:

Titel, Kurzbeschreibung, Ziel, Akteur, Auslöser, Vorbedingung, Szenarien, Ergebnis, Ausführungshäufigkeit, Nichtfunktionale Anforderungen (NFA), Bemerkungen

Diese Form der Beschreibung in mehr oder weniger starker Anlehnung an die genannte Strukturierung der Informationen wird zum Zeitpunkt einer Aufwandsschätzung nach UCP am häufigsten anzutreffen sein bzw. als Ergänzung zu einer der anderen später genannten Spezifikationsformen auftreten.

Aus Sicht der UCP-Methode liegt damit schon eine sehr gute Basis für eine Schätzung vor. In diesem Fall kann ein sehr einfaches Mapping erfolgen. Tabelle 25 fasst dafür die Mapping-Regeln zusammen.

- Titel oder Kurzbeschreibung liefert direkt die Use Case Bezeichner.
- Die Anzahl der Szenarien kann aus der Beschreibung der Szenarien entnommen werden.
- Die Anzahl der Schritte ist als Summe der Aktionen aus den einzelnen Szenarien zu betrachten und kann damit aus der Beschreibung der Szenarien entnommen werden.

Spezifikation: Tabellarische Beschreibung	UCP-Sprache
Akteur	Actor
Ganze Tabelle (Anwendungsfall)	Use Case
Szenario	Szenario
Aktion aus Auflistung/Beschreibung innerhalb der Szenario-Beschreibung	Schritt
Semantischer Kontext	Dialog
Semantischer Kontext	Schnittstelle
Semantischer Kontext	Bericht

Tabelle 25: Mapping-Tabelle Tabellarische Beschreibungen

Es wird allerdings empfohlen, den richtigen "Schnitt" stichprobenartig anhand der zusätzlichen Beschreibung zu überprüfen.

Zur Evaluierung kann bei einem angegeben Ziel in der Szenariobeschreibung überprüft werde, ob zumindest das Hauptszenario komplett beschrieben ist. Eventuell aufgeführte Hinweise auf Anwendungsfunktionen oder Referenzen auf andere Use Cases helfen bei der Erkennung von Wiederverwendung. Sollten diese Referenzen nicht explizit vorhanden sein, liegt es an den Fähigkeiten des Schätzers, sich an diese zu erinnern und sie zu erkennen.

Auch hier sei mit Tabelle 26 ein Beispiel aus einer realen Spezifikation angegeben. Es liegt eine Tabellarische Beschreibung des Anwendungsfalles (AF) *LeistungBuchen* vor. Interessant ist in diesem Beispiel, dass der beschriebene Anwendungsfall einen Unter-Anwendungsfall *Reservierungsliste suchen* aufruft. Dies ist dann sinnvoll, wenn mit der gefundenen Reservierungsliste außer der Buchung auch noch verschiedene andere Anwendungsfälle durchgeführt werden können. Ansonsten wäre das Suchen der Reservierungsliste besser als erster Schritt für den hier beschriebenen Anwendungsfall modelliert und damit auch gezählt worden. Bei der hier vorgegebenen Modellierung würde man das Auswählen einer Reservierungsliste noch als Schritt im Use Case *Buchung* und nicht im Use Case *LeistungBuchen* zählen.

Damit ist der erste Schritt die Auswahl der zu buchenden Leistungen, der zweite Schritt ist die Prüfung auf ein vorliegendes Kontingent und der dritte die eigentliche Buchung. Das (Haupt-) Erfolgsszenario besteht also aus 3 Schritten. Alternativ (2. Szenario) erfolgt bei nicht vorliegendem Kontingent eine direkte Buchung über das Buchungssystem des Leistungsanbieters (4. Schritt). Außerdem gibt es ein 3. Szenario, bei dem ein an sich vorhandenes Kontingent ausgeschöpft ist und eine Abfrage erfolgt, ob eine direkte Buchung erfolgen soll (5. Schritt). Die darauf folgende Buchung ist bereits gezählt worden, ebenso werden die Fehlermeldungen über nicht buchbare Leistungen als trivial eingestuft. Wir zählen also insgesamt 3 Szenarien mit 5 Schritten und 1 Dialog (Auswahl der Leistungen; das Pop-Up für die Abfrage, ob direkt gebucht werden soll, zählt nicht).

Titel	**AF_LeistungBuchen**	
Kurzbeschreibung	Der Anwendungsfall " AF_LeistungBuchen" der Komponente Buchung führt für eine Reservierungsliste die Buchungen durch. Im Anwendungsfall "Buchung::AF_LeistungReservieren" wurde durch den Sachbearbeiter eine Reservierungsliste mit Reisen erstellt, die gebucht werden können, also vakant sind. Der Use Case "AF_LeistungBuchen" führt jetzt für alle Reservierungen eine Buchung durch.	
Ziel	Durchführen einer Buchung	
Szenarien	Leistung buchen (Erfolgsszenario)	1. Der Anwender wählt eine Reservierungsliste aus (Anwendungsfall "Buchung::Reservierungsliste suchen") 2. Der Anwender wählt alle Leistungen aus, die gebucht werden sollen. 3. Das System führt für alle gewählten Leistungen eine Buchung durch. 3.1 Zuerst prüft das System, ob für die Leistung ein Kontingent vorliegt. 3.1.1 Liegt ein Kontingent vor, so bucht das System von diesem Kontingent die Leistung. 3.1.2 Wenn kein Kontingent vorliegt, bucht das System die Leistung direkt über das Buchungssystem des Leistungsanbieters.
	Kontingent erschöpft	1. Ist das Kontingent erschöpft, wird der Anwender gefragt, ob eine direkte Buchung durchführen möchte. 1.1 Will der Anwender eine direkte Buchung durchführen, wird mit Punkt 3.1.2 fortgefahren. 1.2 Will der Anwender keine direkte Buchung durchführen, wird die Buchung mit einer Fehlermeldung abgebrochen.
	Buchung nicht möglich	1. Wenn eine Buchung nicht möglich ist, weil die Leistung nicht verfügbar ist, wird die Buchung mit einer entsprechenden Meldung abgebrochen.
Ergebnis	Wurde die Buchung erfolgreich durchgeführt, ist die Buchung am Ende geschlossen. Es werden keine Daten zurückgegeben.	

Tabelle 26: Beispiel einer tabellarischen Use Case Beschreibung

3.3.3 Aktivitätsdiagramm

Zur Modellierung von Abläufen eignet sich das Aktivitätsdiagramm gemäß UML. An dieser Stelle betrachten wir ausschließlich Aktivitätsdiagramme für Anwendungsfälle. Auf die Beschreibung von Geschäftsprozessen wird in Abschnitt 3.4.2 noch eingegangen.

Aus den Aktivitätsdiagrammen können sehr schnell und einfach direkt die erforderlichen Informationen für die UCP-Methode entnommen werden. Es wird allerdings empfohlen, den richtigen "Schnitt" stichprobenartig anhand der zusätzlichen Beschreibung zu überprüfen. In diesen Dia-

grammen werden normalerweise Schritte und Dialoganzeigen in unterschiedlichen Symbolen dargestellt und gerichtete Kanten kennzeichnen den fachlichen Ablauf des Use Cases. Tabelle 27 fasst dafür die Mapping-Regeln zusammen.

Spezifikation: Aktivitätsdiagramm	UCP-Sprache
Semantischer Gesamt-Kontext	Actor
Ganzes Diagramm oder Teildiagramm	Use Case
Verzweigungsknoten	Szenario
Aktion	Schritt
Semantischer Kontext der Aktion	Dialog
Semantischer Kontext der Aktion	Schnittstelle
Semantischer Kontext der Aktion	Bericht

Tabelle 27: Mapping-Tabelle für Aktivitätsdiagramme

Dabei sind folgende Hinweise zu beachten:

- Dialoge (und damit auch Anzeigen = einfache Dialoge) sind auch als Schritt zu zählen.
- Entscheidungsalternativen werden manchmal nicht gesondert als Schritt dargestellt, wenn sie systemseitig erfolgen, können aber durchaus fachlich komplexe Abläufe enthalten, die eine Zählung als eigenen Schritt rechtfertigen.
- Jede Aufspaltung bei einer Alternative führt zu neuen Szenarien entsprechend der Anzahl der neuen Verzweigungen (keine Kombinationen mit vorherigen Szenarien).
- Wenn ein Algorithmus (z.B. ein Zuordnungsalgorithmus) als Aktivitätsdiagramm dargestellt ist, handelt es sich bereits um eine starke funktionale Zerlegung. In diesem Fall sind die Verzweigungen und Schritte möglichst weitgehend zusammenzufassen, bevor die Zählung beginnt.
- Triviale Fehlerszenarien können als eigene Szenarien mit Schritten dargestellt sein, sollten aber nicht als solche gezählt werden.

In einem Beispiel soll dieses Vorgehen veranschaulicht werden. Wir nehmen den Anwendungsfall *Adresse anlegen* (Abbildung 27). Die gezählten Szenarien, Dialoge und Schritte sind dabei farblich unterschiedlich dargestellt und nummeriert.

Dialoganzeigen werden als eigene Schritte mitgezählt. Ebenso ist die Prüfung der Adresse auf postalische Korrektheit ein eigener Schritt, da ein eigener fachlicher Ablauf damit verbunden ist: Bei nicht korrekter Adresse wird ein neues Szenario mit Anwenderinteraktion ausgelöst. Im Gegensatz dazu wird die Entscheidung, das eine neue Adresse angelegt oder eine bestehende bearbeitet wird, nicht als eigener Schritt gezählt. Die dadurch ausgelöst Verzweigung in das 3. Szenario führt zum Abschluss des Schrittes *Auswahl der Adresse*. Die Vorschlagsliste (2. und 3. Dialog) ist eigentlich nur eine Anzeige, wird aber trotzdem statt *einfach* als *mittel* bewertet, da eine umfangreiche Blätter- und Auswahlfunktion vermutet wird. Der Aufruf des Unter-Anwendungsfalls *Adresse bearbeiten* wird als einzelner eigener Schritt gezählt wird, der eigentliche Ablauf des Unter-Anwendungsfalls geht in die Zählung aber nicht ein.

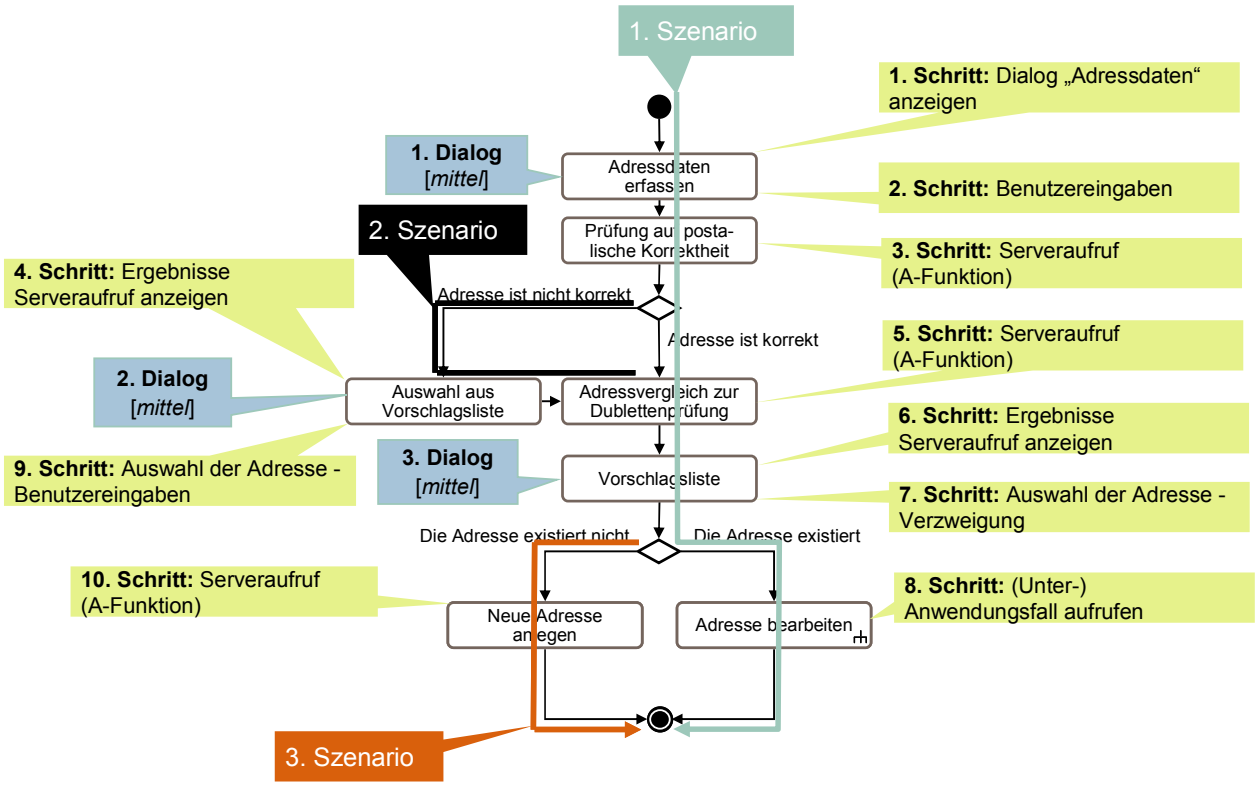


Abbildung 27: Beispiel für Aktivitätsdiagramm „Adresse anlegen“

Die Anzahl der Szenarien ergibt sich durch Aufsummieren aller Alternativen zum Hauptablauf über den Gesamtablauf hinweg plus den Hauptablauf. In diesem Fall gibt es eine Alternative nach dem Schritt der Prüfung auf postalische Korrektheit und eine Alternative bei der Entscheidung, ob eine Adresse existiert oder nicht. Damit haben wir insgesamt zweimal Alternativen und mit dem Hauptszenario insgesamt drei Szenarien.

3.3.4 Zustandsdiagramm

Bei bestimmten Anwendungen spiegelt sich ein großer Teil der fachlichen Komplexität im Lebenszyklus eines Objektes wider. In diesem Fall durchlaufen diese Entitäten im Laufe ihrer Existenz im System verschiedene Zustände und Zustandswechsel. Handelt es sich dabei um fachlich relevante Vorgänge, kann man sie mit Hilfe von Zustandsdiagrammen beschreiben und dokumentieren.

Spezifikation: Zustandsdiagramm	UCP-Sprache
-	Actor
Kantenbeschriftung oder ganzes Diagramm	Use Case
Mehrere Pfade	Szenario
Kantenbeschriftung	Schritt
Semantischer Kontext der Kantenbeschriftung	Dialog
Semantischer Kontext der Kantenbeschriftung	Schnittstelle
Semantischer Kontext der Kantenbeschriftung	Bericht

Tabelle 28: Mapping-Tabelle für Zustandsdiagramme

In der Regel werden Zustandsdiagramme nicht isoliert vorkommen. Trotzdem können auch aus ihnen relevante Informationen für die UCP-Schätzung gefunden werden. Tabelle 28 fasst dafür die Mapping-Regeln zusammen.

Zustandsdiagramme decken üblicherweise mehrere Anwendungsfälle, aber nicht zwangsläufig vollständig, ab. Sie stellen also einen anderen "Schnitt" durch die Anwendung dar. Bei einer vollständigen Abdeckung aller Entitäten und ihrer Zustandsübergänge (insbesondere auch der trivialen) sollten auch alle Anwendungsfälle abgedeckt sein. Dies soll hier am Beispiel eines Zustandsdiagramms in Abbildung 28 erläutert werden.

Die Zustände sind hierbei durch abgerundete Rechtecke und die Zustandsübergange durch Kanten (Pfeile) zwischen diesen dargestellt. Die Kanten des Diagramms sind entweder beschriftet oder textuell beschrieben. Hier finden sich dann die Use Cases oder Schritte von Use Cases.

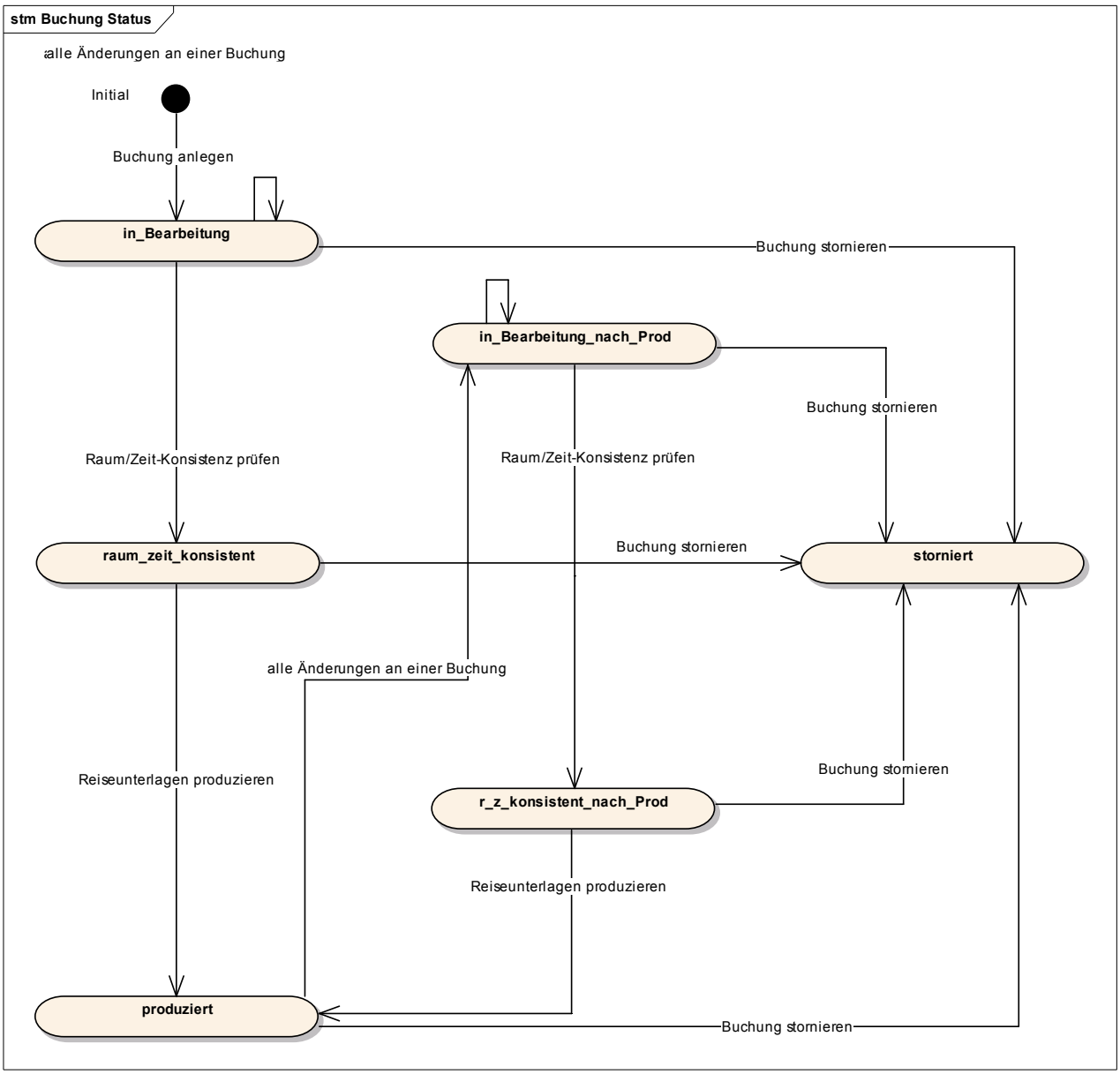


Abbildung 28: Beispiel für Use Case bezogenes Zustandsdiagramm

Im Beispiel in Abbildung 28 ergeben sich so mindestens die Use Cases *Buchung anlegen*, *Buchung ändern* und *Buchung stornieren*. Der Use Case *Buchung anlegen* enthält mit Sicherheit die Eingabe von Daten über einen Dialog und in diesem Beispiel laut Diagramm auch den Schritt *Raum/Zeit Konsistenz prüfen*. Aus dem Diagramm alleine ist nicht ersichtlich, ob *Reiseunterlagen produzieren* ein eigener Use Case oder auch ein Schritt von *Buchung anlegen* ist. In obigem Beispiel liefert das Zustandsdiagramm außerdem einen Hinweis darauf, das es für den Use Case *Buchung stornieren* in Abhängigkeit vom Status mehrere Erfolgsszenarien gibt (in diesem Fall fünf verschiedene aufgrund fünf verschiedener möglicher Ausgangszustände). Je nach Komplexität der durchzuführenden fachlichen Abläufe sind das dann auch unterschiedliche Schritte im Use Case.

Eine UCP-Schätzung ausschließlich auf Basis von Zustandsdiagrammen wird häufig nur einen groben Hinweis auf die Aufteilung der Anwendung in Use Cases ermöglichen. Die zu zählenden Schritte können und müssen dann ergänzend aus dem Text gefunden werden oder es erfolgt eine intuitive Bewertung der Use Case Komplexität.

Bei einer UCP-Schätzung anhand von Zustandsdiagrammen sollte auf jeden Fall darauf geachtet werden, das triviale Zustände und ihre Übergänge nicht übersehen werden, da diese in der Regel nicht in der Form von Diagrammen spezifiziert sind, durchaus aber eigene Use Cases sein können. Dazu gehören z.B. das simple "Erschaffen" eines Objektes und dessen spätere Löschung, d.h. es existiert nur ein Zustand.

3.3.5 CRUD Diagramm

Auch CRUD (Create/Read/Update/Delete) Diagramme als eine spezielle Form der Beschreibung von Anwendungsfällen liefern eine Basis für eine UCP-Schätzung. In einem solchen Diagramm ist dokumentiert, welche Lese- und Schreibzugriffe ein Anwendungsfall direkt auf die Entitäten des fachlichen Datenmodells ausführt. Damit gehen die Use Cases selbst direkt aus diesen Diagrammen hervor (sie stehen in der Regel im Mittelpunkt). Tabelle 29 fasst dafür die Mapping-Regeln zusammen.

Spezifikation: CRUD Diagramm	UCP-Sprache
-	Actor
Anwendungsfall	Use Case
-	Szenario
Semantischer Kontext der Abhängigkeitskanten	Schritt
-	Dialog
Semantischer Kontext der beteiligten Entitäten (generische Schnittstelle)	Schnittstelle
-	Bericht

Tabelle 29: Mapping-Tabelle für CRUD Diagramme

Auch hier soll die Verwendung für die UCP-Methode an einem Beispiel in Abbildung 29 erläutert werden. Die Kanten geben Auskunft darüber, welche Entitäten im Zusammenhang mit diesem Anwendungsfall auf welche Art und Weise verändert werden.

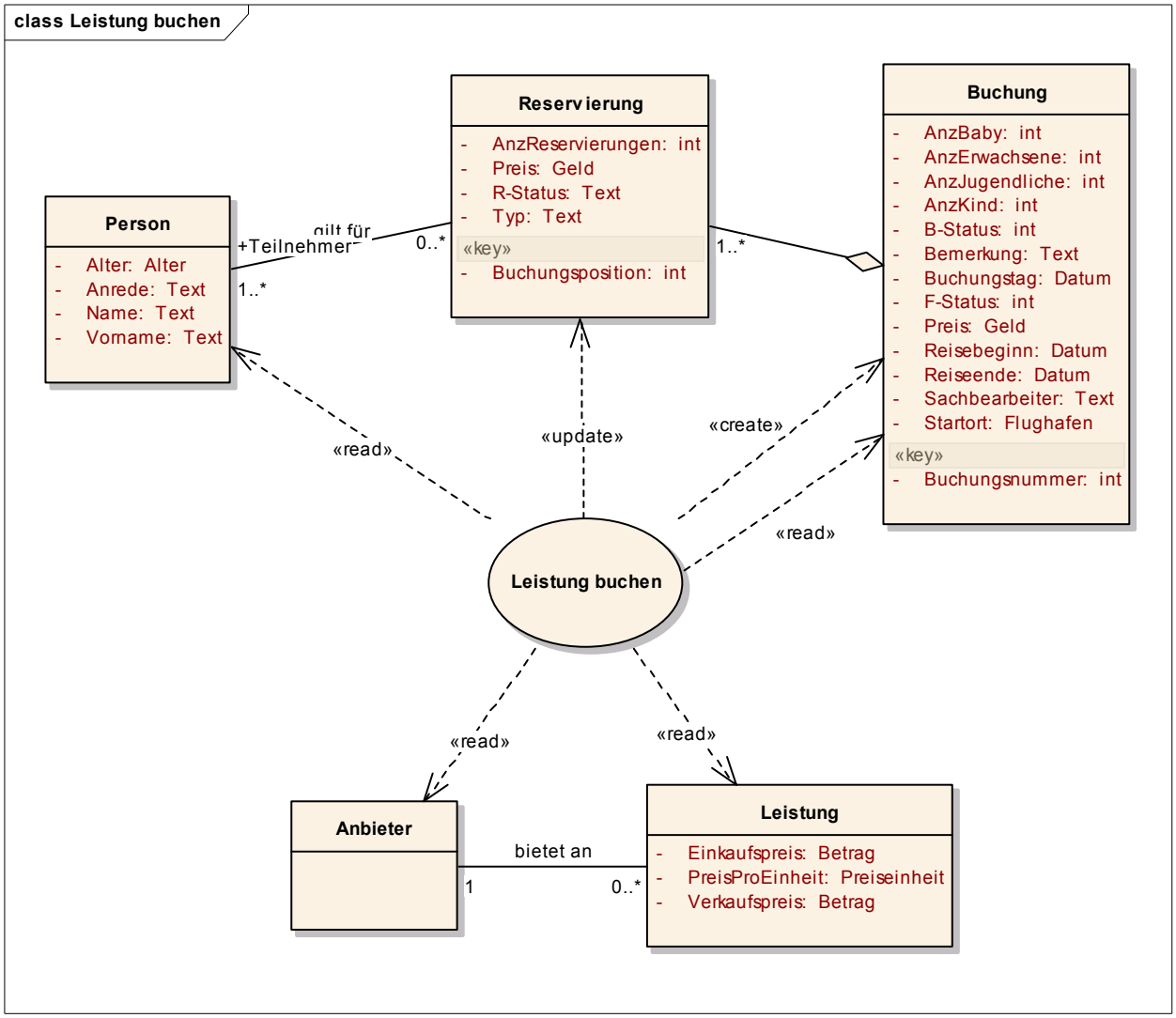


Abbildung 29: Beispiel für ein CRUD-Diagramm

Leider kann in diesem Fall nicht direkt aus der Anzahl der Kanten auf die Anzahl der Schritte des Use Cases geschlossen werden! Es fehlen die Hinweise auf Interaktionen mit den Akteuren zwischen den einzelnen CRUD Operationen und damit der eigentliche "innere Schnitt" des Anwendungsfalles.

In unserem Beispiel für den Anwendungsfall *Leistung buchen* werden z.B. das Lesen (<<*read*>>) der *Leistung* und der *Anbieter* keine getrennten Schritte sein. Wird der Anwendungsfall wie dargestellt als eigener Anwendungsfall modelliert (das ist durchaus berechtigt, da eine sofortige Buchung nicht immer Bestandteil einer Reservierung ist), so werden wahrscheinlich in einem ersten Schritt Personen- oder Reservierungsdaten vom Anwender eingegeben, anschließend werden im System diese Informationen gesucht/gelesen und dann dem Anwender angezeigt. Für die UCP-Methode würden bis dahin 3 Schritte anfallen.

Der 4. und letzte Schritt im (Haupt-) Erfolgsszenario ist dann die Durchführung der eigentlichen Buchung. Ob und inwieweit das Lesen der Leistungen und Anbieter in diesem Erfolgsszenario überhaupt zu berücksichtigen sind oder ob sie Bestandteil eines Alternativszenarios oder eines

anderen Schrittes des Use Cases sind (Suchen und Lesen der Reservierung oder eigentliches Buchen), kann aus diesem Diagramm nicht entnommen werden.

CRUD Diagramme alleine werden uns höchstens den Schnitt der Anwendung in Use Cases liefern, nicht aber ausreichende Hinweise auf die einzelnen Schritte, hier muss grundsätzlich die Beschreibung oder eine der anderen Formen der Darstellung von Anwendungsfällen zusätzlich berücksichtigt werden. Generell besteht bei CRUD Diagrammen die Gefahr einer zu feinen Granularität und damit einer zu starken Zerlegung der Use Cases in Funktionen.

3.3.6 Sequenzdiagramm

Sequenzdiagramme zur Veranschaulichung des Austausches von Nachrichten zwischen Akteuren und Objekten können ebenfalls als Basis für die UCP-Methode dienen. In der Praxis wird dies nicht allzu häufig vorkommen, da der Grad der erforderlichen Detaillierung stärker ist als es in der Regel in einer solch frühen Phase möglich ist. An dieser Stelle sollen trotzdem Hinweise gegeben werden, wie mit einer solchen Spezifikation umgegangen werden kann. Tabelle 30 fasst dafür die Mapping-Regeln zusammen.

Spezifikation: Sequenzdiagramm	UCP-Sprache
Akteur / Kommunikationspartner	Actor
Sequenzdiagramm oder Menge von Sequenzdiagrammen	Use Case
Alternativelemente	Szenario
Sequenzdiagramm oder gerichtete Kanten	Schritt
Aufruf eines Objektes	Dialog
Aufruf eines Objektes	Schnittstelle
Aufruf eines Objektes	Bericht

Tabelle 30: Mapping-Tabelle für Sequenzdiagramme

Falls in der ersten Spalte die Akteure (Kommunikationspartner) des Systems die Auslöser der Nachrichten sind, ist eine Strukturierung nach Use Cases direkt möglich. Falls hier schon Objekte genannt werden, ist die Granularität bereits zu fein (auf Ebene von Schritten oder noch detaillierter). Dann müssen die übergeordneten Anwendungsfälle vor der Zählung erst wieder zusammengesetzt werden. Gezählt werden dann als Schritte nur die Nachrichten zwischen den Kommunikationspartnern (1. Spalte) und den Objekten der 1. Hierarchie (2. Spalte). Ausnahmen können Prüfungen (tiefere Ebenen) sein, die Alternativszenarien auslösen, aber auch diese sind in der Regel dann wieder als Interaktion mit dem Akteur zu erkennen.

Im Ausschnitt eines Sequenzdiagramms in Abbildung 30 ist z.B. nur ein einzelner, im Sinne der UCP-Schätzung gezählter, Schritt eines Use Cases dargestellt, nämlich *buche Reise*! Die weiteren Schritte sind dann im Ablauf des Diagramms weiter unten dargestellt oder werden sogar in gesonderten Diagrammen beschrieben (häufiger Fall). Im letzteren Fall ist dann also ein einzelner Use Case auf mehrere Diagramme verteilt beschrieben.

Ähnlich wie bei CRUD Diagrammen besteht hier grundsätzlich die Gefahr einer zu starken Zerlegung des Use Cases für die UCP-Methode.

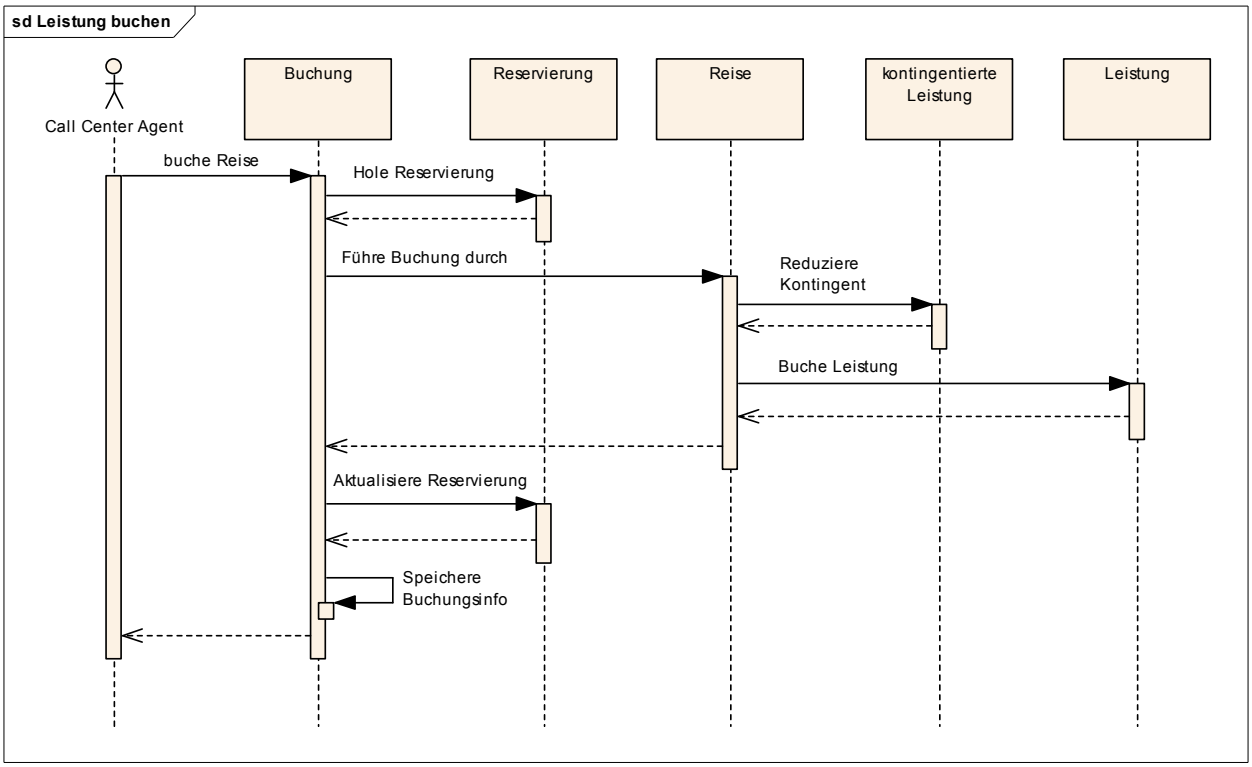


Abbildung 30: Beispiel für ein Sequenzdiagramm

3.3.7 Fazit

Spezifikationsform	Vorteile	Risiken
Grobe (textuelle) Beschreibung	Aufzählung kann direkt übernommen werden; Szenarien und Schritte sind grob beschrieben	Unterschätzung durch zu wenig Informationen über die Szenarien/Schritte, Unterschätzung durch vergessene ganze Use Cases
Tabellarische Beschreibung	Aufzählung kann direkt übernommen werden; Szenarien und Schritte sind bereits beschrieben	
Aktivitätsdiagramm	Direkte Übernahme von Use Cases und Zählung der Szenerien und Schritte; einfach und schnell	Gefahr der Vernachlässigung der Überprüfung gegen textuelle Beschreibung, Aktivitätsdiagramme können zu fein sein.
Zustandsdiagramm		Anderer "Schnitt" durch die Anwendung; Übersehen von Use Cases; Nur im Zusammenhang mit textuellen Ergänzungen verwendbar!
CRUD Diagramm	Die meisten Use Cases sind bereits gefunden	Schritt müssen erst gefunden werden; Übersehen von Use Cases; Nur im Zusammenhang mit textuellen Ergänzungen empfohlen
Sequenzdiagramm	Schnitt der Use Cases bereits vorgegeben; Schritte und Szenarien einfach ersichtlich	Gefahr zu feiner Granularität; In der Regel keine vollständige Abdeckung der Anwendung; Überprüfung der textuellen Ergänzung empfohlen

Tabelle 31: Eignung der Anwendungsfall-basierten Spezifikationsformen für die UCP-Methode

Eine vorliegende Anwendungsfall-Beschreibung ist grundsätzlich in jedem Fall geeignet, eine UCP-Schätzung für den fachlichen Umfang einer Anwendung durchzuführen und belastbare Ergebnisse zu erhalten. Es wird hier nicht betrachtet, ob dann auch genug Information für die Bewertung des T-Faktors und M-Faktors vorliegt.

Je nach Detaillierung und Art der Anwendungsfall-Beschreibung ist unterschiedlich viel Aufwand in die Schätzung selbst zu investieren, um eine gleiche Qualität der Schätzergebnisse zu erreichen. Die Tabelle 31 fasst die dazu in den letzten Abschnitten gemachten Aussagen noch einmal zusammen und basiert auf der Annahme, dass mit den betrachteten Methoden eine die Anwendung vollständig abdeckende UCP-Schätzung erreicht werden soll.

3.4 Mapping von weiteren Spezifikationsformen

Liegt eine (Grob-)Spezifikation nicht als Anwendungsfall-Beschreibung in der bisher dargestellten Form vor, muss im Vorfeld der Schätzung abgewogen werden, ob und mit welchem Aufwand das Finden geeigneter Use Cases möglich ist. Insbesondere wenn das Erstellen der Use Cases ausschließlich für die Schätzung (also keine Weiterverwendung im Projekt absehbar) erfolgt, kann dieser initiale Teil der Schätzung einen potentiell hohen Aufwand bedeuten. Dieser Aufwand sollte aber auch unter dem Aspekt der Qualitätssicherung sowohl für das Verständnis der Fachlichkeit als auch für das Ergebnis und die Belastbarkeit der Schätzung betrachtet werden. Ein mit der Fachlichkeit vertrauter oder/und erfahrener Use Case Modellierer kann diesen Schritt durchaus in vertretbarer Zeit durchführen (wenige Stunden bis einige Tage je nach Umfang der Anwendung).

Können Use Cases aufgrund mangelnder zur Verfügung stehender Informationen nicht oder nur sehr schwer gefunden werden, wird auch eine Schätzung auf Basis dieser Use Cases nicht belastbar sein. In diesem Fall also besser: Finger davon lassen oder zumindest die Ergebnisse mit wohlwollender Toleranz betrachten! Das muss nicht zwangsläufig bedeuten, dass eine Expertenschätzung nicht möglich ist, aber auch diese dürfte dann nicht wesentlich belastbarer sein. Ein gutes Kriterium ist auch die Überprüfung, ob sich Testfälle relativ einfach bestimmen lassen.

Speziell im Fall einer Spezifikation, welche nicht auf einer Anwendungsfall-Beschreibung beruht, wird empfohlen, eine Schätzung nach UCP ebenfalls als eine unabhängige Schätzung mindestens zweier verschiedener Personen durchzuführen. Dies gilt insbesondere für das Finden der Use Cases selbst (den "Schnitt"), die eigentliche Gewichtung ist dann nicht mehr ganz so schwierig.

Einen guten Überblick zu den nachfolgenden Abschnitten gibt Abbildung 25 auf Seite 64.

3.4.1 Dialogbeschreibung

Häufig wird in den frühen Phasen von besonders dialoglastigen Projekten nicht mit Anwendungsfällen oder textuellen Beschreibungen von Funktionalität gearbeitet. Fachbereichsmitarbeiter kommen häufig mit einer visuellen Darstellung besser zurecht. Daher treffen wir dann zum Zeit-

punkt der Schätzung auf Screenshots[18], Mock-Ups[19] oder ähnliche Beschreibungen von Dialogen. Wir nennen eine solche Spezifikationsform kurz *Dialogbeschreibung*. Tabelle 32 fasst dafür die Mapping-Regeln zusammen.

Spezifikation: Dialogbeschreibung	UCP-Sprache
Semantischer Kontext	Actor
Teile des Dialogsystems oder Menusystems	Use Case
Semantischer Kontext: Dialogablauf	Szenario
Semantischer Kontext: finale Aktionen im Dialogablauf, Buttons	Schritt
Dialog (Reiter, Frame)	Dialog
Semantischer Kontext: Buttons	Schnittstelle
Semantischer Kontext: Aktion Drucken	Bericht

Tabelle 32: Mapping-Tabelle für Dialogbeschreibungen

Aus diesen Darstellungen lassen sich in der Regel Use Cases gut finden. Wir möchten dazu an dieser Stelle einige Hinweise geben:

- Über den Einstieg des Dialogsystems und das Menusystem lassen sich meist direkt Use Cases oder zumindest fachliche Blöcke ableiten.
- Den Workflow im Dialog dann soweit verfolgen, bis "finale" Aktionen ausgeführt werden (z.B. speichern, prüfen, senden, ...).
- Ein Dialog kann von mehreren Use Cases verwendet werden, entscheidend ist der Workflow durch das Dialogsystem bis zum Erreichen eines bestimmten fachlichen Ziels.
- Querbezüge aus der Aufrufhierarchie heraus in eine andere (parallele) Ebene der Hierarchie heraus sind Hinweise auf die Verwendung eines anderen Use Cases.
- Buttons bzw. durch andere Interaktionselemente mit der GUI ausgelöste Aktionen sind ein guter Indikator für einen neuen Schritt.
- Weitere (auch "verdeckte") Schritte gehen dann entweder aus dem Gesamtkontext oder aus zusätzlichen textuellen oder graphischen Darstellungen hervor oder müssen erfragt werden. Ein Screenshot alleine reicht für eine UCP-Schätzung nicht aus.

Bei der Zählung können die Dialoge (Reiter, Frames beachten!) direkt übernommen werden. Szenarien und Schritte ergeben sich wie beschrieben aus dem Workflow und der Folge von verwendeten Dialogen für ein bestimmtes fachliches Ziel.

Abbildung 31 zeigt beispielhaft die Menüstruktur für ein Dialogteilsystem, aus dem auf jeden Fall eine grobe fachliche Strukturierung ersichtlich ist. Hier liefert die Hauptmenüstruktur nur zum Teil Hinweise auf Use Cases (fehlende Verben in den Menupunkten!). Ausnahmen sind die Dialoge *Category Sort*, *Manual Upload* und *Auto Upload*, die sich im Übrigen später auch als echte Use Cases heraus gestellt haben.

---

[18] *Screenshot* bezeichnet das statische Abbild eines Dialoges, wie er auf dem Bildschirm angezeigt wird

[19] *Mock-Up* (engl. „Attrappe“) bezeichnet einen Vorführ-Prototypen der Benutzeroberfläche ohne weitere Funktionalität

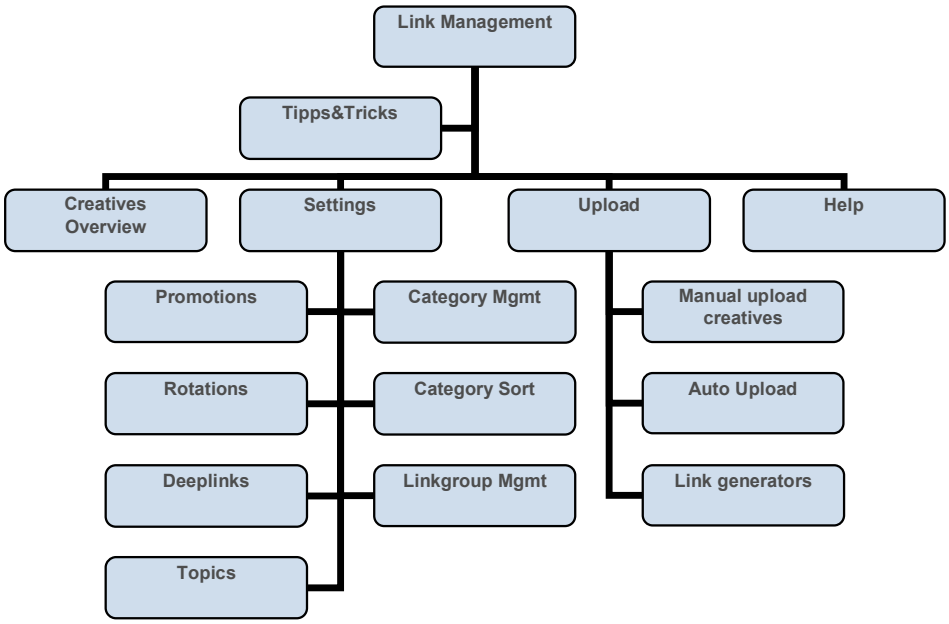


Abbildung 31: Beispiel für eine Menüstruktur eines Dialogteilsystems

Als weitere Vertiefung wird in Abbildung 32 der Punkt *Category Mgmt* aus der vorherigen Menüstruktur anhand eines Mock-Ups weiter spezifiziert.

Aus den Buttons können hier direkt die Use Cases *Create Category*, *Activate Category*, *Deactivate Category*, *Delete Category*, *Update Category* und *Download as csv* abgeleitet werden. Ob sich hinter den Buttons weitere Schritte und Dialoge verbergen oder der Use Case damit abgeschlossen ist, muss dann aus weiteren Mock-Ups und den textuellen Erläuterungen hervorgehen.

Details sind z.B. hier ein spezielles Szenario von *Update*, was allerdings noch nicht alleine und direkt aus dem Mock-Up hervorgeht. Die Aktion *Delete* enthält auch mehrere fachliche Schritte, die erst aus der ergänzenden Beschreibung hervorgehen. Man kann auch vermuten, dass allen diesen Use Cases ein gemeinsamer Schritt *Category suchen* bzw. *lesen* vorausgeht, da kein spezieller Button dafür auf diesem Mock-Up zu ersehen ist.

Zusätzlich können Interaktionsdiagramme vorliegen, aus denen ebenfalls Use Cases und Hinweise auf Schritte entnommen werden können (siehe Abbildung 33). Hierbei sind die Dialoge (Screens) als Ellipsen dargestellt und die Aktionen (Buttons) als Kästchen. Es lassen sich wieder die Use Cases anhand der Buttons erkennen: *Create Rotation*, *Update Rotation*, *Deactivate Rotations*, *Activate Rotations* und *Assign Rotation Creatives*, wobei *Assign Rotation Creatives* noch mit einem weiteren Dialog fortgesetzt wird.

Es sei hier noch einmal darauf verwiesen, zusätzlich zu den graphisch vorliegenden Informationen auch die Beschreibungen und Erläuterungen einzubeziehen.

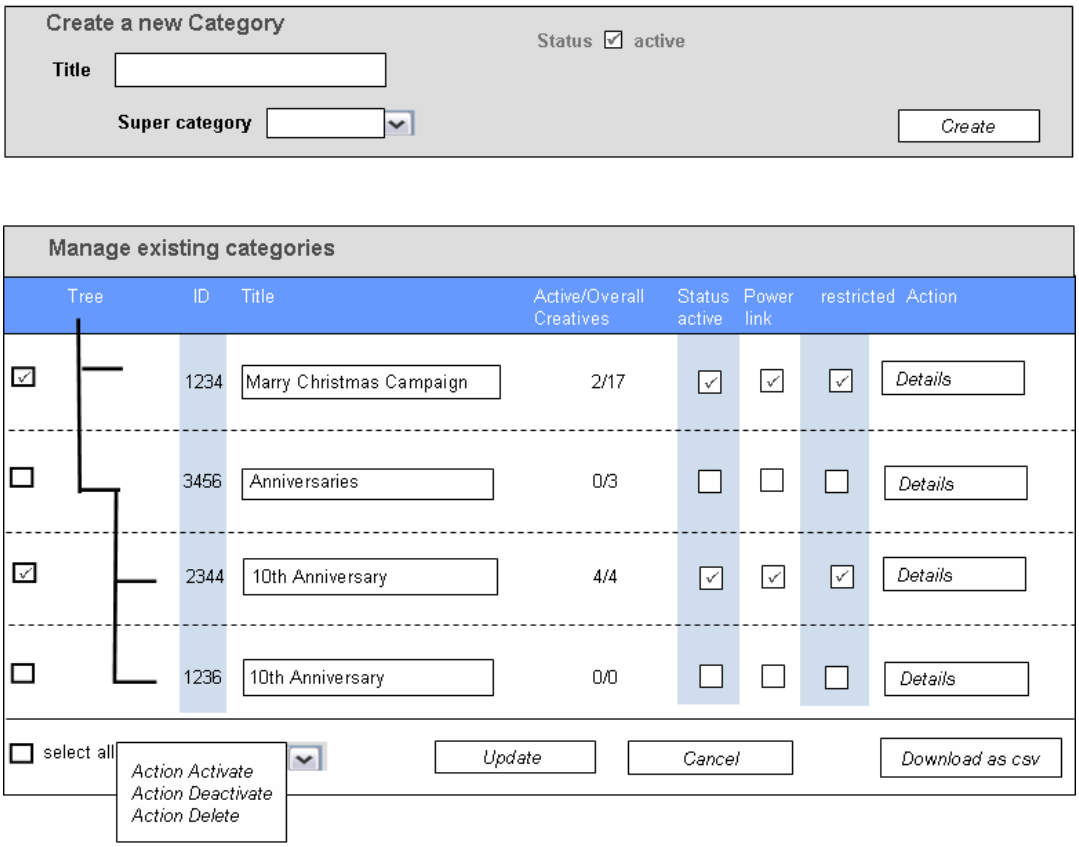


Abbildung 32: Beispiel für ein Mock-Up zum Category Managements

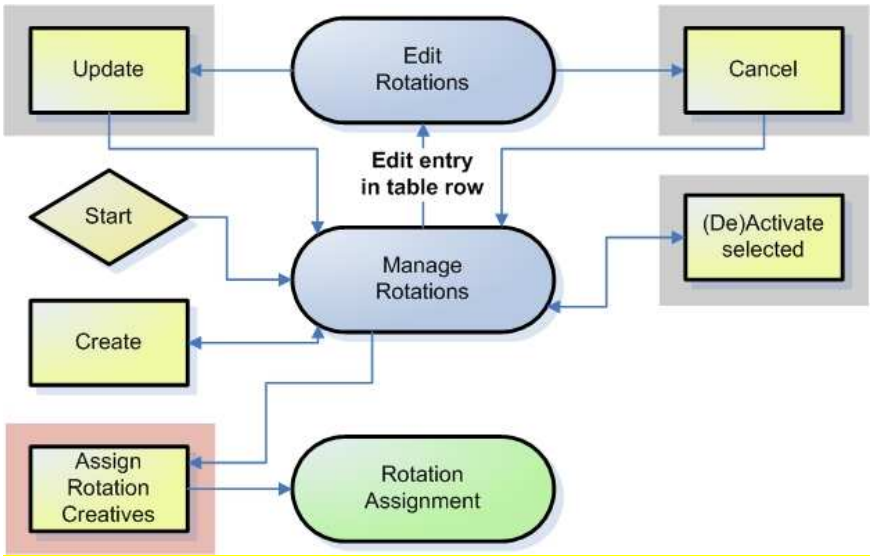


Abbildung 33: Beispiel für ein Interaktionsdiagramm eines Use Cases

3.4.2 Geschäftsprozess-Beschreibung

Die Beschreibung von Geschäftsprozessen erfolgt in der Regel in sehr frühen Phasen im Rahmen der Grobspezifikation und bildet damit auch die Basis für Schätzungen. In der Praxis existieren verschiedene Formen der Modellierung von Geschäftsprozessen, angefangen bei nicht formal definierten textuellen Beschreibungen, über semiformale grafische Darstellungen (z.B. BPMN[20], UML) zur Visualisierung, bis hin zu formalsprachlichen Darstellungen (z. B. BPEL[21]) zur Unterstützung von Simulation und Übertragung in ausführbaren Code.

Es überwiegen jedoch die nicht formalisierten textuellen Beschreibungen, häufiger auch unter Zuhilfenahme von UML-basierten Aktivitätsdiagrammen. Wir werden uns auf diese Formen beschränken. Der Grund dafür liegt darin, dass formale grafische Geschäftsprozessdarstellungen, die beispielsweise mit BPMN oder UML-Diagrammen (z.B. Anwendungsfall-Diagramm) spezifiziert sind, oft in zu grober Granularität vorliegen. Im Ebenen-Modell von Cockburn sind die darin enthaltenen Aktivitäten dann am ehesten der 1. Ebene (*Summary Goal,* Abschnitt 3.1.1) zuzuordnen, womit es nicht oder zumindest nur schwer möglich ist, die für eine UCP-Schätzung notwendigen Informationen zu extrahieren. In solchen Fällen muss dann ebenfalls auf die ergänzenden textuellen Beschreibungen zurückgegriffen werden.

Andererseits ist auch häufig der Fall anzutreffen, dass eine als Geschäftsprozess-Beschreibung deklarierte Spezifikation eigentlich eine Anwendungsfall-Spezifikation im hier betrachteten Sinne ist. Hier ist noch einmal der Hinweis wichtig, dass wir nicht zwischen Business Use Cases und "normalen“ Use Cases unterscheiden (Abschnitt 3.1.1, Anmerkungen zu Use Case Definition). Auch wenn nur der Begriff (Geschäfts-)*Prozess* verwendet wird, kann es sich durchaus um Anwendungsfälle handeln. Diese können dann einfach, wie bereits beschrieben, auf Use Cases abgebildet und geschätzt werden.

Falls eine „echte“ Geschäftsprozessbeschreibung vorliegt, hängt es wiederum vom Grad der Detaillierung ab, ob eine belastbare Bewertung der Use Cases möglich ist. Folgende Schritte sind dazu durchzuführen:

- Finden und Erfassen der Use Cases. Ein Use Case ist normalerweise ein (oder wenige) Schritte in einer Geschäftsprozessbeschreibung. Vorsicht, in Prozessbeschreibungen tauchen auch manuelle Aktionen auf, die nicht zu Use Cases gehören.
- Use Cases in Prozessbeschreibungen können durch einen Wechsel des Akteurs oder Prozessverantwortlichen, durch Erzeugung von permanenten Zwischenergebnissen oder auch durch einen Medien- oder Systemwechsel voneinander getrennt und damit zu finden sein.
- Falls zusätzliche Informationen zu den Prozess-Schritten vorliegen, daraus die Use Cases bewerten; im Notfall intuitiv ohne Szenarien und Schritte, evtl. vergleichbare Use Cases aus ähnlichem fachlichen oder technischen Umfeld heranziehen.

---

[20] **Business Process Modeling Notation** (**BPMN**) ist eine grafische Spezifikationssprache und stellt Symbole zur Verfügung, mit denen Fach- und Informatikspezialisten Geschäftsprozesse und Arbeitsabläufe (Workflows) modellieren können.

[21] **Business Process Execution Language (BPEL)**, ist eine XML-basierte Sprache zur Beschreibung von Geschäftsprozessen, deren einzelne Aktivitäten durch Webservices implementiert sind.

- Falls keine weiteren Informationen vorliegen, hat man außer der Auflistung der Use Cases nicht viel. Daraus kann dann maximal eine grobe Abschätzung gemacht werden, die dann aber nur einen Richtwert darstellt.

Tabelle 33 fasst die Mapping-Regeln zusammen.

Spezifikation: Geschäftsprozess-Beschreibung	UCP-Sprache
Prozessverantwortlicher	Actor
Prozess-Schritt	Use Case
Semantischer Kontext der Prozessbeschreibung	Szenario
-	Schritt
Semantischer Kontext	Dialog
Semantischer Kontext	Schnittstelle
Semantischer Kontext	Bericht

Tabelle 33: Mapping-Tabelle für Geschäftsprozessbeschreibungen

Beispiel: Im Prozess in Abbildung 34 werden die einzelnen Use Cases durch den (wahrscheinlich manuell durchgeführten) Versand von E-Mails voneinander getrennt. Außerdem wechselt der Actor von *HR* zu *FK*. In diesem Fall haben wir zwei Use Cases, die Vorbereitung der *TA* durch *HR* und den Abschluss der *TA* durch *FK*. Es gibt auch Hinweise auf mindestens durchzuführende Schritte, wie das Ausfüllen bestimmter Felder und eine Evaluierungsfunktion.

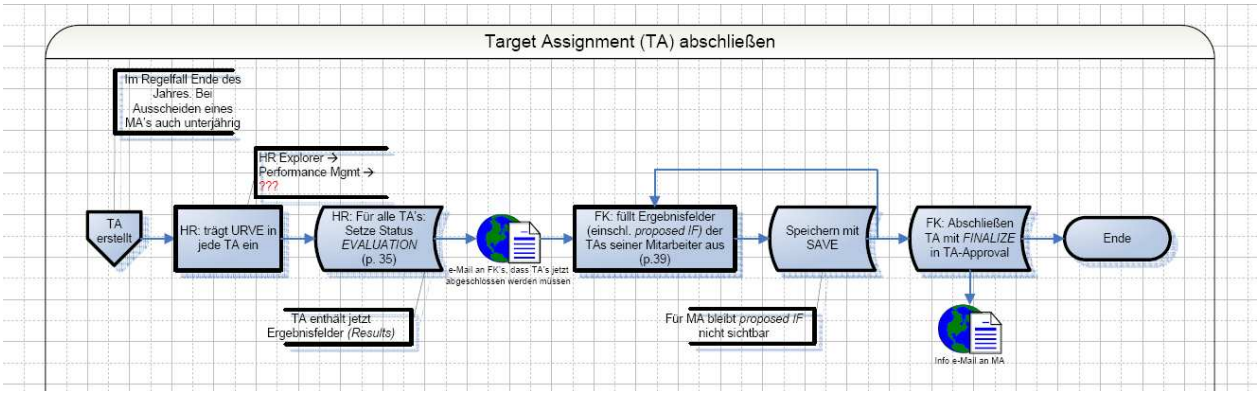


Abbildung 34: Beispiel einer Prozessbeschreibung

Ob das Versenden der E-Mail systemseitig unterstützt wird und damit noch zum Use Case gehört, geht aus der Darstellung nicht hervor. Es liegt die Vermutung nahe, dass beide Use Cases in die Kategorie *einfach* fallen.

Abbildung 35 zeigt den Ausschnitt eines Geschäftsprozesses auf einem gröberen Level. Die einzelnen Schritte des Prozesses sind eigene Use Cases und sind auch nicht mehr weiter erläutert. Nur auf Basis dieser Informationen ist eine Schätzung sehr schwer möglich, hier helfen nur fachlich ähnliche und bereits bekannte Fälle. Im konkreten Beispiel liegen noch die Informationen gemäß Tabelle 34 vor.

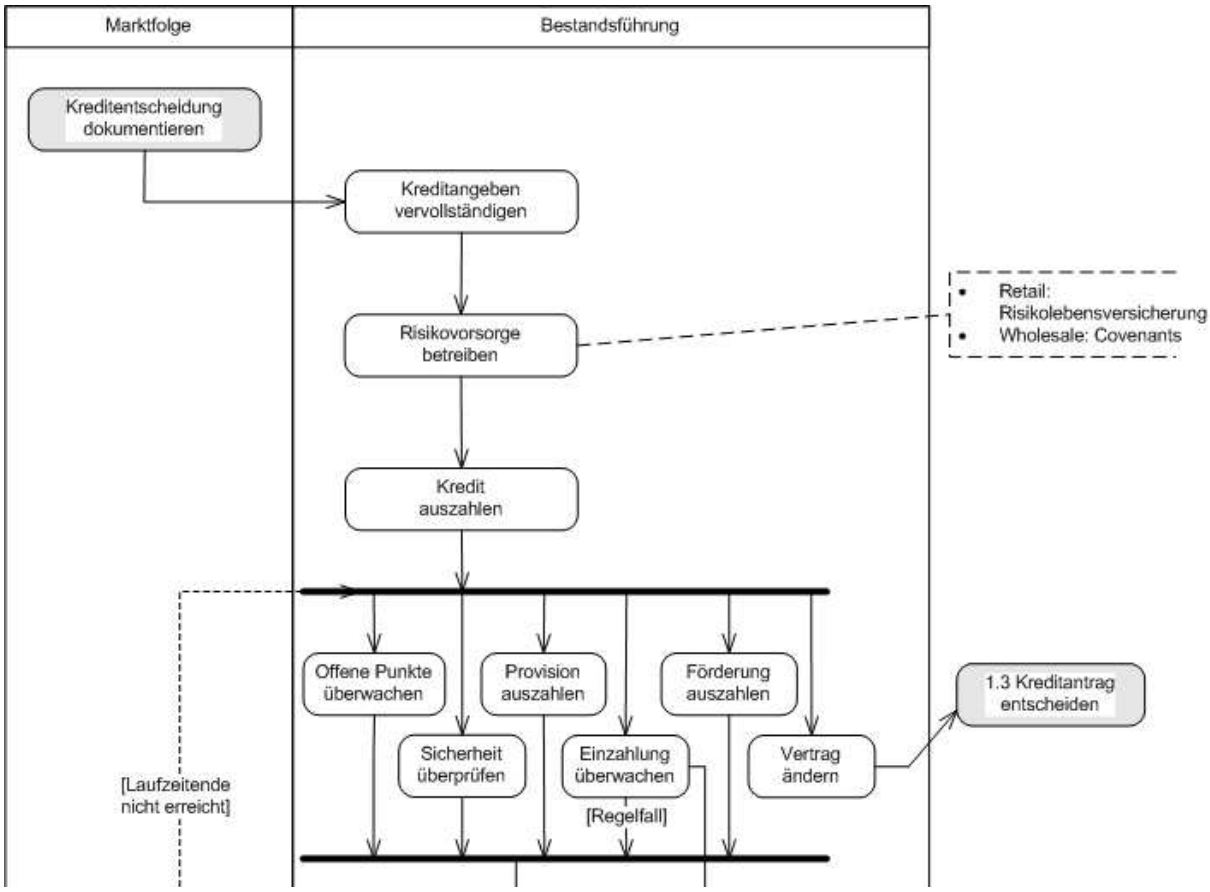


Abbildung 35: Ausschnitt einer Geschäftsprozessbeschreibung

Nr.	Name	Beschreibung
1	Kreditangaben vervollständigen	Kreditunterlagen kontrollieren auf Vollständigkeit (alle Dokumente, Legitimation, Unterschriften). Fehlende bzw. regelmäßig zu stellende Dokumente auf Wiedervorlage legen. Systemunterstützung: maschinelle Kontrollen durch Dokumentenabgleich / Plausibilitäten etc.; elektronische Prüfvermerke/Unterschrift
2	Risikovorsorge betreiben	Anlegen einer (Risiko-)Lebensversicherung, Kreditrahmenversicherung bzw. Einholen von Nebenabreden (covenants) etc.
3	Kredit auszahlen	Kreditkonto anlegen, Kredit bereitstellen. Bis zur Ausnutzung der Kreditlinie durch den Kunden: Bereitstellungszinsen einziehen, Kreditkonditionen prüfen (auf Kündigungsgründe bei Nichtabnahme). Wird der Kredit genutzt: Limitausnutzung nachführen. Schufa-Eintrag vornehmen bzw. ergänzen.

Tabelle 34: Beispiel einer textuellen Ergänzung zur Geschäftsprozessbeschreibung

Die textuellen Ergänzungen zeigen, dass es sich hierbei durchaus nicht um triviale Use Cases handelt. Der Use Case *Kreditangaben vervollständigen* mag noch als einfacher Pflege Use Case betrachtet werden können, aber auch können komplexe Folgeschritte wie Prüfungen unterschätzt werden. Beim Use Case *Kredit auszahlen* sieht es aber mindestens nach zwei weiteren Use Cases *Kreditkonto anlegen* und *Kredit bereitstellen* aus. Liegen keine weiteren Detailinformationen vor, erscheint eine UCP-Schätzung hier wenig belastbar.

3.4.3 Funktionale Beschreibung

Häufig finden sich Spezifikationen auch als Ansammlung oder Auflistung von Funktionen, die ohne Gruppierung in Anwendungsfällen vorliegen. Wir nennen diese Form kurz *Funktionale Beschreibung*. Die Vielfalt der Beschreibungsmöglichkeiten ist dabei sehr groß, damit kann eine allgemeine Richtlinie für den Umgang mit diesen Beschreibungen nur sehr grobe Hinweise enthalten.

- Die später für die Schätzung zu zählenden Schritte sind als Funktionen in der Beschreibung je nach Detaillierungsgrad bereits in der richtigen Granularität vorhanden. Bei zu feingranularer Beschreibung (siehe auch CRUD-Diagramme, Abschnitt 3.3.5) müssen Funktionen erst wieder zu Schritten zusammengefasst werden.
- In einem zweiten Schritt sind dann diese Funktionen (oder Funktionsgruppen) den Use Cases zuzuordnen (das kann man sich ruhig daneben schreiben oder in einer Matrix erfassen), bei Schwierigkeiten orientiert man sich am Hauptschritt der Verarbeitung oder an Entitäten. Auf jeden Fall gelten die Kriterien aus dem Abschnitt 3.1.3.
- Ist nicht bekannt, ob es später weitere Use Cases geben wird, die die Funktionalität nutzen, gehen wir erstmal nur von einem Use Case aus.
- Es muss für jede angeführte Funktion auch einen Use Case geben, der diese verwendet!
- Querschnittsfunktionalität kann als extra Use Case erfasst werden, um nicht ständig das Problem der Wiederverwendung zu haben (z.B. Hilfe, Login, Hauptmenu-Auswahl).

Tabelle 35 fasst die Mapping-Regeln zusammen.

Spezifikation: Funktionale Beschreibung	UCP-Sprache
Semantischer Kontext	Actor
Gruppe von Funktionen	Use Case
Gleiche Funktion für unterschiedliche Objekte	Szenario
Funktionen oder Gruppe von Funktionen	Schritt
Semantischer Kontext der Funktionen	Dialog
Semantischer Kontext der Funktionen	Schnittstelle
Semantischer Kontext der Funktionen	Bericht

Tabelle 35: Mapping-Tabelle für Funktionale Beschreibungen

Funktionsmatrizen: Ein spezieller Fall von funktional orientierten Beschreibungen sind tabellarische Aufstellungen, in denen bestimmten fachliche Abläufen oder Objekte über Matrizen den Funktionen zugeordnet sind. Man läuft leicht Gefahr, eine solche, eigentlich orthogonal zu Use Cases angelegte Beschreibung zu übernehmen, und damit durch den hohen Grad der funktionalen Zerlegung einen zu feinen "Schnitt" der Use Cases zu zählen. Dieser Hinweis gilt generell für Funktionale Beschreibungen.

Im Beispiel in Tabelle 36 sind für verschiedene Entitäten die darauf ausführbaren Funktionen aufgeführt. Je nach Ähnlichkeit der Objekte muss entschieden werden, ob jedes Kreuz ein eigener Use Case wird oder ob z.B. ähnliche Objekte gleich bearbeitet werden und dadurch nur unterschiedliche Szenarien eines Use Cases sind, d.h. Spalten zusammengefasst werden können.

Dies geht hier z.B. mit den Funktionen *exportieren* und *importieren*. Des Weiteren können evtl. spezielle Funktionen wie z.B. *Status wechseln* als Schritt eines Szenarios des Use Cases *Anzeigen (Bearbeiten)* betrachtet werden oder einzelne Funktionen sind keine eigenen Use Cases sondern immer Bestandteil eines anderen Use Cases, hier z.B. *Suchen*.

	Objekt					
Funktion	Textbau-stein	Referenz	Feature	Produkt-text	Cluster-Objekt	VIB-Template
Suchen	x	x	x	x	x	x
Anzeigen (Bearbeiten)	x	x	x	x	x	x
Ausführen						x
Anlegen	x	x	x	x	x	x
Prüfen eines Features gegen eine VIB			x			
Struktur anzeigen	x		x	x	x	x
Löschen	x	x	x	x	x	x
Übersetzung anlegen	x	x	x	(x)	(x)	x
Übersetzung editieren	x	x	x	(x)	(x)	x
Status wechseln			x	x	x	x
Exportieren	x	x	x	x	x	x
Importieren	x		x			x

Tabelle 36: Beispiel einer Funktionsmatrix

3.4.4 Fazit

Spezifikationsform	**Vorteile**	**Risiken**
Geschäftsprozess-Beschreibung	Aufzählung der Use Cases kann direkt übernommen oder einfach gefunden werden; guter Überblick der Anwendung	Unterschätzung durch zu wenig Informationen über die Szenarien/Schritte
Dialogbeschreibung	Use Cases und Szenarien sind relativ einfach zu finden; gute Abdeckung der Anwendung	Verdeckte Schritte; Anwendungslogik ist nicht erfasst, daher immer textuelle Ergänzungen beachten
Funktionale Beschreibung	Beschreibung der Schritte und Funktionen, gute Abdeckung der geforderten Funktionalität der Anwendung	Komposition zu Szenarien und Use Cases u.U. aufwändig; Gefahr, in Funktionen und damit zu detailliert zu schätzen

Tabelle 37: Eignung der weiteren Spezifikationsformen für die UCP-Methode

Auch aus nicht Anwendungsfall-basierten Spezifikationen heraus kann eine UCP-Schätzung durchgeführt werden. Dem geht allerdings immer ein mehr oder wenig aufwändiger Schritt des Mappings voraus. In diesem Schritt müssen die Use Cases gefunden werden und eine Aufteilung in Szenarien und Schritte vorgenommen werden. Mit zunehmender "Entfernung" der vorhande-

nen Spezifikation von einer Anwendungsfall-Beschreibung handelt es sich dabei um die „hohe Schule“ der UCP-Schätzung. Tabelle 37 stellt die Vorteile und Risiken der weiteren Spezifikationsformen für die UCP-Methode zusammen.

Es können hier sicherlich nicht alle Arten von Spezifikationen betrachtet und erfasst werden, für häufig vorkommende sind jedoch Hinweise für den Umgang im Rahmen einer UCP-Schätzung gegeben worden.

3.5 Zusammenfassende Hinweise für die Anwendbarkeit und Evaluierung

Der in diesem Kapitel vorgestellte Leitfaden wird bei der weiteren Entwicklung der UCP-Methode zugrunde gelegt. Er definiert Regeln anhand von Mapping-Tabellen und Praxisbeispielen, wie Anwendungsfälle aus verschiedenen Grobspezifikationen für die UCP-Methode gewonnen werden können.

Tabelle 38 fasst Hinweise und Kriterien für die Anwendbarkeit der UCP-Methode und Möglichkeiten der Evaluierung einer durchgeführten Schätzung zusammen. Zum großen Teil sind diese bereits in vorherigen Abschnitten genannt, begründet und erläutert worden.

Problem	Ursache	Auswirkung	Korrekturmöglichkeit
Schwierigkeit beim Finden der Use Cases	Beschreibung zu grob, nicht genug Informationen vorhanden	Schätzung nicht durchführbar, nicht plausibel	Vorsicht, hier hilft auch die UCP-Methode nicht!
Schwierigkeit beim Finden der Use Cases	keine Anwendungsfall-Beschreibung und Mapping von bestehender Beschreibung schwierig	Schätzung nicht oder schwierig durchführbar, nicht plausibel	Neu nachdenken ☺, zweiten Schätzer fragen
sehr viele kleine Use Cases (häufig nur 1 oder 2 Schritte)	Genereller Schnitt wahrscheinlich zu feingranular	potentiell Überschätzung	Die zugrundeliegende Anwendungsfall-Beschreibung oder das für eine andere Art der Beschreibung verwendete Mapping "passt" nicht und sollte überdacht werden.
Verhältnismäßig (zu den kleinen und mittleren) sehr viele große Use Cases	Genereller Schnitt wahrscheinlich zu grobgranular	potentiell Unterschätzung	
Sehr große Use Cases: >5 Szenarien, >7 Interaktionselemente, >16 Schritte	Schnitt wahrscheinlich zu grobgranular	potentiell Unterschätzung	Dekomposition der betroffenen Use Cases, evtl. importierte Use Cases finden
Hoher Anteil von Re-Use	Wartungs- oder Integrationsprojekt, Einsatz von Frameworks, Produkten	hohe Schätzunsicherheit	
Hoher Anteil von Re-Use	starke Kopplung der Use Cases, starke Berücksichtigung von Anwendungsfunktionen über Re-Use	Verfälschung der Schätzung	Wiederverwendete Anwendungsfunktionen/Schritte nur beim ersten Use Case zählen

Tabelle 38: Mögliche Probleme und Lösungen bei der Anwendung der UCP-Methode

4 Validierung der modellbezogenen Use Case Identifikation

Der Erfolg der UCP-Methode hängt wesentlich von der standardisierten und eindeutigen Größenbestimmung der fachlichen Anforderungen ab. Dies wurde bereits in der zentralen Problemstellung P 1 (Unterschiedlicher Schnitt der Use Cases) in Abschnitt 2.7 thematisiert. Damit kommt dem Prozess der Identifikation von Use Cases aus unterschiedlichen Spezifikationsformen eine entscheidende Bedeutung zu. Eine Lösung wurde dazu in Kapitel 3 mit der UCP-Sprache, dem Leitfaden zur Identifikation von Use Cases und dem A-Faktor angeboten.

Es ist nun zu zeigen, dass dieser Lösungsvorschlag aus Kapitel 3 zu einer guten Reproduzierbarkeit des geschätzten Aufwandes führt. Dabei sind die in Abschnitt 3.3 und 3.4 genannten unterschiedlichen Spezifikationsformen einzubeziehen. Dies ist nun Gegenstand dieses Kapitels 4.

Dazu wurden mit Hilfe einer Feldstudie an sechs Hochschulen insgesamt über 200 Aufwands-Schätzungen auf Basis von acht verschiedenen Spezifikationsformen aus der industriellen Praxis durchgeführt. Im Fokus der Untersuchung stand dabei der A-Faktor gemäß der in Kapitel 3 skizzierten UCP-Methode UCP 2.0[22]. Die Schätzungen wurden bezüglich der Reproduzierbarkeit quantitativ und qualitativ miteinander verglichen.

Man könnte nun einwenden, dass eine Feldstudie mit Studierenden nicht repräsentativ für Software-Schätzungen in der *industriellen* Praxis ist. Ebenso ist unklar, wie viel Erfahrung für die Durchführung einer UCP-Schätzung tatsächlich benötigt wird, siehe P 8 (Interpretationsspielraum von Use Cases) aus Abschnitt 2.7. Um dies zu überprüfen, wurden die Schätzungen der Studierenden mit denen von Experten verglichen (Abschnitt 4.5).

Dieses Kapitel ist wie folgt aufgebaut: Abschnitt 4.1 gibt zunächst einen kurzen Überblick in die für das weitere Verständnis notwendigen Grundlagen der experimentellen Informatik. Darauf aufbauend wird die Entwicklung und Planung der Feldstudie in Abschnitt 4.2 beschrieben. Die Auswertung der Schätzergebnisse erfolgte mit Hilfe von statistischen Methoden. Für die Grundkenntnisse der Statistik verweisen wir auf die einschlägige Fachliteratur [SH06, Sch08]. Abschnitt 4.3 beschreibt die hier verwendeten zentralen Methoden. Die Ergebnisse der Feldstudie werden dann in Abschnitt 4.4 dargestellt und anhand der zuvor eingeführten Methoden ausgewertet. Es folgt der Vergleich der Feldstudie mit Expertenschätzungen (Abschnitt 4.5) und eine Zusammenfassung aller Ergebnisse in Abschnitt 4.6.

Im Rahmen dieser Dissertation beschränken wir uns auf eine Zusammenfassung der Feldstudie und deren Ergebnisse. Eine detaillierte Darstellung der Feldstudie findet sich in [Eng08].

---

[22] UCP 2.0 und UCP 1.0 unterscheiden *nicht* hinsichtlich des A-Faktors sondern nur in den Kostenfaktoren. Damit gelten alle Aussagen aus diesem Kapitel auch für die Methode UCP 1.0. Wenn die Methode UCP 2.0 eine gute Reproduzierbarkeit zeigt, wird dies für die Methode UCP 3.0, wie sie in Kapitel 3 eingeführt wurde, erst Recht gelten, da die Verfeinerung der Dialoge in Interaktionselemente eine höhere Präzision darstellt.

4.1 Experimentelle Grundlagen

The test of all knowledge is experiment. Experiment is the sole judge of scientific "truth".
(The Feynman lectures on physics, 1964)

Der Aufbau eines Experimentes muss generell sicher stellen, dass in einer gegebenen Situation die Veränderung X der Variablen x eine Veränderung Y der Variablen y bedingt [Pre01]. Dies erfordert, dass eine beobachtete Veränderung von Y ursächlich durch X bestimmt wird. Dies ist der Fall, wenn während des Experiments sich nur X verändert. Prechelt bezeichnet in diesem Zusammenhang mit *innerer Gültigkeit* den „Grad eines kontrollierten Experiments, in dem die Änderung in den Werten der Variablen tatsächlich wie gewünscht nur auf Änderungen der unabhängigen Variablen zurückzuführen sind, d.h. wie gut letztlich alle relevanten Störvariablen kontrolliert werden".

Im Falle der Aufwandsschätzung von unterschiedlichen Spezifikationsformen ist es schwierig, sicherzustellen, dass nur die Spezifikations*form* geändert wird, ohne dass weitere Veränderungen am Schätzprozess eintreten, denn der *Schätzer* als Mensch ist immer beeinflussbar und Effekte durch Lernkurven sind nicht auszuschließen. Diese Effekte müssen durch die Methode des kontrollierten Experiments [Pre01, Bas07] möglichst klein gehalten werden.

Zunächst ist zwischen einer Feldstudie gegenüber einer Laborstudie abzuwägen: Eine Laborstudie läge vor, wenn zu jeder der *n* zu untersuchenden Spezifikationsformen die gleiche Fachlichkeit zugrunde läge, also das gleiche Vorhaben nur in unterschiedlichen Spezifikationsformen erfasst wurde. In der industriellen Praxis ist diese Situation nicht verfügbar. Daher konzentrieren wir uns im Folgenden auf eine Feldstudie, in welcher unterschiedliche Projektkontexte und Fachlichkeiten je Spezifikationsform vorliegen.

Für den Aufbau des Experiments verlangen die folgenden Störvariablen die größte Beachtung:

Auswahl (selection): Individuelle Unterschiede der Schätzer können durch eine hohe Zahl an Schätzern weitgehend eliminiert werden. Dazu ist der Erfahrungshintergrund sorgfältig zu prüfen. Es wurde dazu eine Vorlesungseinheit über Aufwandsschätzung von Software-Entwicklungsprojekten im Allgemeinen und über die UCP-Methode im Speziellen sowie die Schätzaufgabe als Übungsaufgabe entworfen und sicher gestellt, dass alle Teilnehmer der Feldstudie (Studierende) daran teilgenommen haben. Es wurden primär Studierende der (Wirtschafts-)Informatik von unterschiedlichen Hochschulen in Deutschland ausgewählt, die ein vergleichbares Wissen über die benötigten Grundlagen der Spezifikation mit UML und Anwendungsfällen haben.

Reifung (maturation): Hiermit sind Veränderungen im Verhalten der Schätzer gemeint, die über die Zeit hinweg auftreten könnten (z.B. Ermüdung, Lerneffekte, Reihenfolgeeffekte, Sequenzeffekte). Diese Effekte können im vorliegenden Kontext ausgeschlossen werden, da je Schätzer nur zwei unterschiedliche Projekte ausgewählt wurden (=>Reihenfolgeeffekt ausgeschlossen), die gesamte Schätzaufgabe in kurzer Zeit (ca. eine Stunde) durchgeführt werden konnte (=> Ermüdung ausgeschlossen) und Sequenz- und Lerneffekte kaum bis keinen Einfluss auf das Ergebnis haben, weil die Schätzer ihre Ergebnisse während der Gesamtdauer beider Schätzungen leicht anpassen können.

Sterblichkeit (mortality): Dies bedeutet das Ausscheiden von Schätzern durch Abbruch der Schätzung während der Schätzaufgabe auf eigenen Wunsch. Für die hier betrachtete Feldstudie konnte keine signifikante Sterblichkeit festgestellt werde, da die Dauer für eine Schätzung mit ca. einer Stunde sehr kurz war und die Studierenden hoch motiviert sind.

Instrumentation: Veränderungen im Verhalten über die Zeit können aber nicht nur bei den Schätzern auftreten sondern auch bei dem Experimentator oder dem Experimentaufbau selbst. Beispielsweise könnten sich im Laufe der Feldstudie bei Schätzungen durch die Studierenden an einer Hochschule Schwierigkeiten in den Übungsaufgaben abzeichnen, die dann an der nächsten Hochschule durch erweiterte Hinweise durch den Experimentator vermieden werden sollen. Daher wurden alle an der Feldstudie beteiligten Personen an den Hochschulen mit identischen Unterlagen über die Projekte und die Schätzmethodik versorgt, um gleiches Vorwissen sicher zu stellen und zu vermeiden, dass die Dokumentation der UCP-Methode sich durch zusätzliche Hilfestellungen über die Reihe der Vorlesungen und Übungen an den unterschiedlichen Hochschulen nach und nach während der Feldstudiendurchführung verändert.

Anforderungscharakteristik: Dies bezeichnet den Effekt, dass im Vorfeld der Feldstudie bereits ein bestimmtes Ergebnis erwartet wird und die Schätzer aufgrund dieser Voreinstellung beeinflusst sind und ungewollt auf dieses Ergebnis hinarbeiten. Zum Beispiel könnte die Aussage, eine bestimmte Spezifikationsform sei besser als eine andere geeignet, die Schätzer entsprechend beeinflussen. Diese Störquelle wurde dadurch eliminiert, dass alle Schätzer die exakt gleichen Instruktionen und Anleitungen in schriftlicher Form erhalten haben und die Zwischenergebnisse je Spezifikationsform bis zum Ende der Feldstudie nicht veröffentlicht wurden.

Mittels der hier beschriebenen Vorsorge wurde eine hohe innere Gültigkeit für die Feldstudie angestrebt. Eine detaillierte Diskussion der Störvariablen kann in [Eng08] nachgelesen werden.

4.2 Entwicklung und Planung der Feldstudie

In einem **ersten Schritt** ist zu bestimmen, welche Spezifikationsformen in der industriellen Praxis von betrieblichen Informationssystemen verwendet werden. Dazu wurden ca. 50 unterschiedliche Projekte bzw. Ausschreibungsunterlagen aus praktisch allen Branchen untersucht und der jeweils dominante Typ der Spezifikationsform bestimmt. Da die Unterlagen vertrauliche Informationen enthalten, wird auf eine detaillierte Darstellung der Stichprobe verzichtet. Die häufigsten Formen fasst Tabelle 39 zusammen: Drei Spezifikationsformen sind UML Diagramme, vier sind Textformen und eine beruht auf Dialog-Abbildungen (Screenshots).

Aus Kapitel 3 wurden für die Feldstudie die Spezifikationsform der CRUD-Diagramme nicht verwendet, da sie immer nur in Kombination mit weiteren Spezifikationsformen aussagekräftig ist und daher nicht in isolierter Form bewertet werden kann (Abschnitt 3.3.5). Die *Grobe textuelle Beschreibung* (4) und die *Tabellarische Beschreibung* (5) lehnen sich direkt an die Anwendungsfall-Beschreibung an und sind daher in Kapitel 3 als UML-basiert kategorisiert. Allerdings wird kein UML-Diagramm-Typ verwendet. Insbesondere finden die Anwendungsfall-Diagramme der UML in der Praxis kaum Verwendung und es werden vielmehr die beiden hier genannten textuellen Beschreibungen verwendet.

#	Spezifikationsform	Beschreibung	Beispiel siehe Abschnitt
1	Aktivitätsdiagramm	UML-Diagramm	3.3.3
2	Zustandsdiagramm	UML-Diagramm	3.3.4
3	Sequenzdiagramm	UML-Diagramm	3.3.6
4	Grobe textuelle Beschreibung	Textuelle Beschreibung	3.3.1
5	Tabellarische Beschreibung	Textuelle Beschreibung	3.3.2
6	Geschäftsprozess-Beschreibung	Textuelle Beschreibung	3.4.2
7	Funktionale Beschreibung	Textuelle Beschreibung	3.4.3
8	Dialogbeschreibung	Screenshots in Verbindung mit textueller formloser Beschreibung	3.4.1

Tabelle 39: Spezifikationsformen der Feldstudie

Für die Feldstudie sollten Spezifikationen von realen industriellen Projekten verwendet werden, um den Praxisbezug möglichst hoch zu halten. Um den zeitlichen Restriktionen der Feldstudie als studentische Übungsaufgabe und den Vertraulichkeitsanforderungen der Projektpartner (Auftraggeber) gerecht zu werden, wurden die Spezifikationen lediglich gekürzt und anonymisiert. Die Spezifikationen wurden so ausgewählt, dass möglichst eine Spezifikationsform gemäß Tabelle 39 isoliert Verwendung fand. In realen Industrieprojekten wird üblicherweise eine Mischung von unterschiedlichen Spezifikationsformen angetroffen. Für die Feldstudie musste dies vermieden werden, damit der Einfluss der einzelnen Spezifikationsform auf das Schätzergebnis verglichen werden konnte.

Ein **zweiter Schritt** war die Planung und Organisation der Feldstudie als semikontrolliertes Experiment. Dazu wurde folgender Experimentaufbau entwickelt: Die acht ausgewählten Spezifikationsdokumente mussten in eine einheitliche Form bezüglich Umfang und Kontextinformationen gebracht werden und auf die Studierenden von acht Vorlesungen an sechs Hochschulen aufgeteilt werden. Dabei stellt sich die Frage, wie viele Schätzungen *n* für jede Form mindestens benötigt werden, damit eine ausreichend große Gesamtheit für eine Varianzanalyse mit Hilfe von Methoden der Statistik vorliegt. In der Literatur sind hierzu keine einheitlichen Vorgaben zu finden, Lilja [Lil00] nennt z.B. n=30 als Scheidezahl, Westphal fordert nur n>10 [Wes71]. Engeroff konnte mit Hilfe von statistischen Methoden nachweisen, das n=20 für die hier beschriebene Aufgabenstellung ausreichend ist [Eng08, S. 67]. Dieser Wert wurde für die Feldstudie als Untergrenze festgelegt. In Summe wurden 202 Schätzungen durchgeführt, die sich gemäß Tabelle 40 auf verschiedene Hochschulen und Vorlesungen verteilen.

Die UCP-Schätzaufgabe der Studierenden beschränkte sich auf den A-Faktor gemäß UCP 2.0 und dabei auf die Ermittlung der Use Cases und deren Komplexität. Der Beitrag der Actors wurde weggelassen, da durch die gekürzte Spezifikation die Use Case übergreifenden Actors nicht angemessen in die Gesamtschätzung einzuordnen sind und bei den gewählten Spezifikationsbeispielen der Anteil des Actors im Vergleich zum Use Case vernachlässigbar klein ist.

Je nach Hochschule wurde die Schätzung in Form einer Präsenzübung oder als Hausübung durchgeführt. Vorgesehen war eine Bearbeitung jeder Schätzung in Zweiergruppen, das Ergebnis wurde in einem vorbereiteten Bogen erfasst. Dadurch lagen für die spätere Auswertung je Spezifikationsform sowohl die benannten Use Cases (und damit deren Anzahl) sowie die Komplexität jedes Use Cases bewertet durch dessen Anzahl Schritte, Szenarien und Dialoge und der Re-Use

Anteil vor. Das Gesamtergebnis je Spezifikation wird in Use Case Points (UCP) zusammengefasst, wir verwenden dafür nachfolgend der Einfachheit halber die Abkürzung *UCP* und treffen Formal die Vereinfachung, dass die Actors auf Null Punkte gesetzt sind und der T-Faktor, M-Faktor und PF jeweils Eins ist.

#	Spezifikationsform	**TU Darmstadt** Prof. Dr.-Ing. Mira Mezini	**TU Darmstadt** Prof. Dr. Andy Schürr	**TU Kaiserslautern** Prof. Dr. H. Dieter Rombach	**Universität Marburg** Prof. Dr. Gabriele Taentzer	**Universität Paderborn** Prof. Dr. Gregor Engels	**HS Darmstadt (Wi-Ing.)** Prof. Dr. Bernhard Humm	**HS Darmstadt (CNAM)** Prof. Dr. Bernhard Humm	**Universität Mainz** Dr. Andreas Winter	**Total**
1.	Aktivitätsdiagramm		19	10						**29**
2.	Zustandsdiagramm			14	11					**25**
3.	Sequenzdiagramm						9	6	6	**21**
4.	Grobe textuelle Beschreibung				8	8			8	**24**
5.	Tabellarische Beschreibung	11	18							**29**
6.	Geschäftsprozess-Beschreibung						4	3	16	**23**
7.	Funktionale Beschreibung		16			8				**24**
8.	Dialogbeschreibung	11	16							**27**
	Anzahl UCP-Schätzungen:	**22**	**69**	**24**	**19**	**16**	**13**	**9**	**30**	**202**

Tabelle 40: An der Feldstudie beteiligte Hochschulen und Anzahl der UCP-Schätzungen je Spezifikationsform

Für jede Spezifikationsform liefert die Feldstudie nun eine Reihe von Schätzwerten[23] abhängig von der Anzahl Schätzungen je Spezifikationsform (Tabelle 40). Diese verschiedenen Schätzreihen sind nun miteinander zu vergleichen.

Dabei wird die Streuung der Schätzreihe als Maß für die Güte der Reproduzierbarkeit einer Schätzung interpretiert: Je geringer die Streuung der UCP-Werte ist, desto besser ist die Reproduzierbarkeit.

Im Idealfall haben alle Studierenden je Spezifikationsform den gleichen UCP-Wert ermittelt, dann wäre die Streuung Null.

4.3 Grundlagen zu den Methoden der statistischen Auswertung

Nachfolgend werden die zentralen statistischen Methoden zur Analyse und Interpretation der Daten aus der Feldstudie kurz beschrieben. Für eine ausführlichere Darstellung verweisen wir z.B. auf [SH06, Sch08].

[23] im Folgenden auch vereinfacht als *Schätzreihe* bezeichnet

Der Aufwand eines Software-Entwicklungsprojektes kann nach Abschluss exakt in der physikalischen Größe Zeit angegeben werden, es ist der „wahre“ Aufwand, der zum Zeitpunkt der Schätzung unbekannt ist. Die Aufwandsschätzung kann diesen wahren Aufwand nie *exakt* vorhersagen. Mehrere Schätzungen durch unterschiedliche Schätzer oder durch Verwendung unterschiedlicher Methoden werden immer zumindest leicht unterschiedliche Schätzwerte ergeben, die Abweichung vom wahren Wert bezeichnen wir als *Schätzfehler*. Dieser kann grundsätzlich drei verschiedene Ursachen haben:

- Durch z.B. fehlerhafte Spezifikation entsteht ein *systematischer Fehler*, den es zu minimieren gilt bzw. dessen Ursachen es zu finden und zu beseitigen gilt. Hier helfen statistische Methoden nur sehr begrenzt. In unserem Kontext können wir aber aufgrund des Experimentaufbaus davon ausgehen, dass dieser Fehler in allen Schätzungen einer Schätzreihe gleichermaßen gemacht wird und sich daher im Vergleich innerhalb einer Schätzreihe nicht auswirkt, d.h. die Streuung davon nicht betroffen ist.
- *Methodenfehler*: Durch die Unvollkommenheit von Schätzmethoden sind diese nicht eindeutig definiert und damit nicht reproduzierbar. Genau diesen Methodenfehler wollen wir mit dem Feldversuch aufspüren bzw. dessen Größe nachweisen.
- Von ganz anderer Art sind *zufällige Fehler*. Die wesentliche Ursache liegt in der Person des Schätzers selbst mit seiner menschlichen Begrenztheit (Augen, Ohren, Gedankenfehler, etc.). Diese Begrenztheit führt zu Fehlinterpretationen der Spezifikation oder handwerklichen Fehlern in der Anwendung der Schätzmethode. Durch mehrfache unabhängige Schätzung (=*Schätzreihe*) zur Qualitätssicherung kann dieser Fehler minimiert werden.

Die Einzelergebnisse einer Schätzreihe werden aufgrund der beiden letztgenannten Fehler immer eine Streuung (allgemein: *Häufigkeitsverteilung*) um einen mittleren Wert zeigen.

Im Falle des zufälligen Fehlers handelt es sich bei der Verteilung um eine *Normalverteilung* (oder auch Gaußverteilung), die dem Fehlerverteilungsgesetzt von Gauß [BS87] folgt. Der Methodenfehler kann einer Normalverteilung folgen, muss es aber nicht. Alle drei Fehlerkategorien werden sich addieren.

Welchen Wert innerhalb des Streubereiches der Einzelwerte soll man nun als den *zuverlässigsten* oder *Bestwert* der Schätzreihe betrachten, d.h. als denjenigen, der dem wahren (zunächst unbekanntem) Wert wahrscheinlich besonders nahe kommt? Eine beweisbare Antwort auf diese Frage gibt es nicht [Wes71]; das einzig Mögliche ist, dass man ein Berechnungsprinzip postuliert, das in seinem Ergebnis zwar nicht notwendig aber einleuchtend ist. Nach Gauß gilt als Bestwert derjenige Wert, der die Summe der Quadrate der Abweichungen der Einzelwerte minimiert („Methode der kleinsten Quadrate“). Diese Bedingung erfüllt das *arithmetische Mittel* (kurz: Mittelwert) $\boldsymbol{\mu}$ als Summe aller Schätzungen x geteilt durch die Anzahl n der Einzelschätzungen (Formel 17) [BS87].

Die *Standardabweichung* $\boldsymbol{\sigma}$ (Synonym: Streuung) ist definiert durch die Quadratwurzel über die Varianz, die wiederum die Summe der quadratischen Abweichungen vom Mittelwert geteilt durch die Anzahl der Beobachtungen minus Eins ist. Der *zufällige Fehler des Mittelwertes* $\boldsymbol{\Delta\mu}$ berechnet sich dann aus σ geteilt durch die Wurzel von n (n = Anzahl der Elemente der Schätzreihe) und wird oft durch das unbestimmte Vorzeichen ± (also $\mu \pm \Delta\mu$) notiert, da der Fehler sowohl nach oben wie nach unten abweichen kann. Der Mittelwert hat also eine *Messgenauigkeit*,

die durch die Anzahl n der Schätzungen je Schätzreihe bestimmt ist: Der zufällige Fehler wird für $n \to \infty$ gegen Null laufen, bei kleinen Schätzreihen (n < 10) ist er jedoch nicht zu vernachlässigen!

Durch den Experimentaufbau mit n>20 Schätzungen je Spezifikationsform kann der zufällige Fehler eliminiert werden. Eine Streuung der Schätzreihen kann daher nur durch den Methodenfehler verursacht sein.

Der Variationskoeffizient *V* ist als Quotient aus Standardabweichung durch den Mittelwert definiert. *V* ist ein normalisiertes Maß der Streuung und drückt in Prozent aus, wie viel die Standardabweichung vom Mittelwert ausmacht. Damit eignet sich *V* zum Vergleich der Streuung von Verteilungen mit unterschiedlichen Mittelwerten - bei den UCP-Schätzungen der Feldstudie werden wir darauf zurückkommen. Alle Definitionen sind unter Formel 17 zusammengefasst.

$$\mu := \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \Delta\mu := \frac{\sigma}{\sqrt{n}} \qquad \sigma := \sqrt{\frac{\sum_{i=1}^{n}(\mu - x_i)^2}{n-1}} \qquad V := \frac{\sigma}{\mu}$$

Formel 17

N: Anzahl der Schätzwerte, μ: Mittelwert, $\Delta\mu$: zufälliger Fehler des Mittelwertes, σ: Standardabweichung oder Streuung, *V*: Variationskoeffizient

Zur *grafischen Darstellung* der statistischen Verteilung einer Schätzreihe wird der Box-Whisker-Plot (kurz: *Boxplot*) verwendet. Der Boxplot ist eine spezielle Art der Darstellung einer Häufigkeitsverteilung (Abbildung 36).

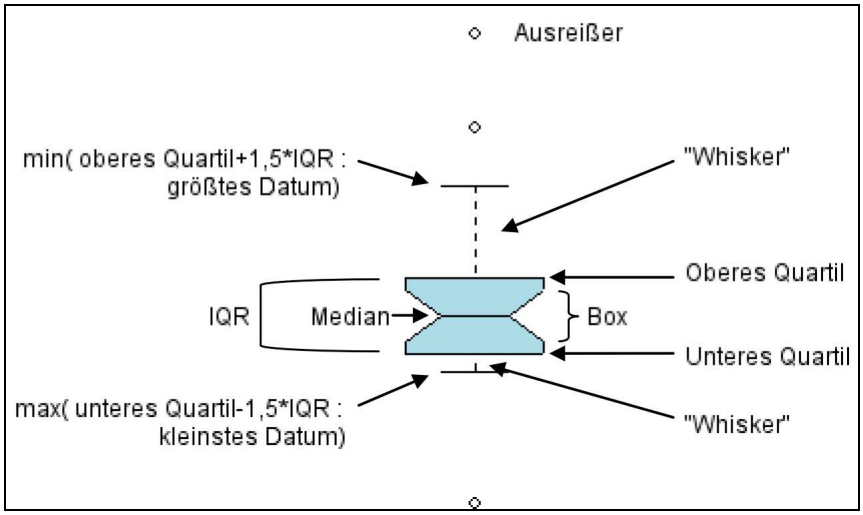


Abbildung 36: Box-Whisker-Plot (Boxplot) mit Definition der Kennwerte [Eng08]

Der Boxplot zeigt in der Mitte der „Box“ den *Median,* definiert als mittlerer Wert der Schätzreihe, wenn diese der Größe nach sortiert würde. Der Median gilt allgemein als robuster gegen Ausreißer im Vergleich zum Mittelwert. Ferner zeigt die „Box“ das obere und untere Quartil an. Als unteres *Quartil* (lat. „Viertelwerte“) bezeichnet man jenen Wert, der die Anzahl der Beobachtungen so teilt, dass unter ihm 25% der Werte fallen. Das obere Quartil ist analog für 75% definiert. Der Median ist folglich das 50%-„Quartil“.

Die Differenz zwischen dem oberen und dem unteren Quartil wird als *Quartilabstand* (engl. *InterQuartile Range = IQR*) bezeichnet. Der IQR umfasst daher 50 % der Verteilung und wird als

Streumaß verwendet [Har91]. Als *Whisker* (engl. „Fühler“ oder "Antenne") werden die vertikalen gestrichelten Linien bezeichnet. Die Länge der Whisker beträgt maximal das 1,5-fache des IQR und wird immer durch einen Wert aus den Daten bestimmt. Werte, die über dieser Grenze liegen, werden separat als Punkte (kleine Kreise) in das Diagramm eingetragen und als *Ausreißer* bezeichnet. Gibt es keine Werte außerhalb der Whisker, so wird die Länge des Whiskers durch den maximalen bzw. minimalen Wert festgelegt (Abbildung 36). Ein Ausreißer stellt sich als extrem hoher oder niedriger Wert innerhalb einer Reihe üblicher, mäßig unterschiedlicher Datenwerte dar. Er darf unter gewissen Umständen vernachlässigt werden. Eine eindeutige Entscheidung, ob dieser Wert ein Ausreißer ist, ist nur selten möglich.

Boxplots ermöglichen die Veranschaulichung von Schätzreihen und deren Streuung: Bei „schiefen“ Daten liegt der Median nicht in der Mitte der Box und die Whisker können unterschiedlich lang sein. Zum Vergleich von Verteilungen mehrerer Stichproben können mehrere Boxplots nebeneinander in einem Diagramm dargestellt werden und so leicht die Varianzhomogenität von mehreren Stichproben beurteilt werden. Des Weiteren lassen sich in Boxplots Ausreißer gut erkennen.

Boxplots erlauben leicht auf grafischem Wege festzustellen (gleiche Länge der Boxen und Whisker), ob im Rahmen der Messgenauigkeit zwischen zwei und mehr Datenreihen gleiche Varianz („Varianzhomogenität“) vorliegt. Eine zusätzliche mathematische Prüfung der Varianzhomogenität ist mit Hilfe des Levene-Tests möglich. Bekannter und weiter verbreitet ist zwar der F-Test, allerdings erfordert der Levene-Test keine Normalverteilung der Häufigkeitsverteilung und wird daher in dieser Dissertation verwendet[24]. Zur Durchführung der statistischen Berechnungen wurde das Statistikpaket *R* verwendet [SH06].

Für die Anwendung des Levene-Tests ist zunächst eine Hypothese zu formulieren, deren Gültigkeit dann mit dem Levene-Test überprüft wird. In der Statistik wird unter einer Hypothese eine anhand empirischer Daten zu prüfende Annahme verstanden. Zu jeder Hypothese existiert eine Gegenhypothese, welche in der Statistik als Alternativhypothese bezeichnet wird. Das *Hypothesenpaar* setzt sich somit aus einer *Nullhypothese* H_0 und einer zugehörigen gegensätzlichen *Alternativhypothese* H_1 zusammen. Die Nullhypothese soll stets verworfen werden, damit die Alternativhypothese als wahrscheinlich (richtig) übrig bleibt. Mit Hilfe einer Teststatistik wird aus einer bzw. häufig aus zwei gegebenen Stichproben eine Prüfgröße berechnet. Je nach dem Wert dieser Prüfgröße entscheidet der Test nun über die Annahme oder die Ablehnung der Nullhypothese.

Zusätzlich zu der Prüfgröße berechnet der Test den *p-Wert*, der angibt, mit welcher Wahrscheinlichkeit eine ebenso große oder größere Prüfgröße erwartet werden kann. Ist diese Wahrscheinlichkeit kleiner als ein zuvor festgelegtes Signifikanzniveau, wird die Nullhypothese auf diesem Signifikanzniveau abgelehnt. Üblicherweise wird das *5%-Signifikanzniveau* (kurz: 5%-Niveau) verwendet, die Nullhypothese wird also für einen p-Wert < 0,05 abgelehnt. Beim 5%-Niveau wird in 100 Fällen im Durchschnitt 5-mal irrtümlich die Nullhypothese abgelehnt.

---

[24] Der Brown-Forsythe Test verwendet den Median anstelle des Mittelwertes. Wir verwenden den Mittelwert, da er die besten Ergebnisse für symmetrische „moderate-tailed“ [BF74] Verteilungen liefert und für unsere Problemstellung am besten passt.

Für die Fragestellung der Varianzhomogenität lautet nun das Hypothesenpaar für den Levene-Test:

$H_0 = \sigma_1 = \sigma_2 = ... = \sigma_n$ $H_1 = \sigma_i \neq \sigma_j$ *für mindestens ein i, j mit* $i \neq j$	*Formel 18*

Ausgegangen vom 5%-Signifikanzniveau kann bei einem p-Wert < 0,05, die Nullhypothese, es herrscht Varianzhomogenität, verworfen und die Alternativhypothese akzeptiert werden. In diesem Fall wird von Varianzheterogenität ausgegangen. In umgekehrter Reihenfolge gilt dies aber nicht, denn es darf nicht unmittelbar von Varianzhomogenität ausgegangen werden, wenn keine Signifikanz vorliegt. Liegt der p-Wert relativ nahe am Wert 1.0, wird hier zusammen mit der Betrachtung der Boxplots davon ausgegangen, dass Varianzhomogenität vorliegt.

4.4 Auswertung der Messwerte aus der Feldstudie

Abbildung 37 zeigt eine graphische Darstellung der UCP Schätzreihen als Boxplots, sortiert nach aufsteigendem Variationskoeffizient (= Höhe der Box). Ausreißer wurden identifiziert und eliminiert. Für eine detaillierte Analyse der Ausreißer wird auf [Eng08] verwiesen.

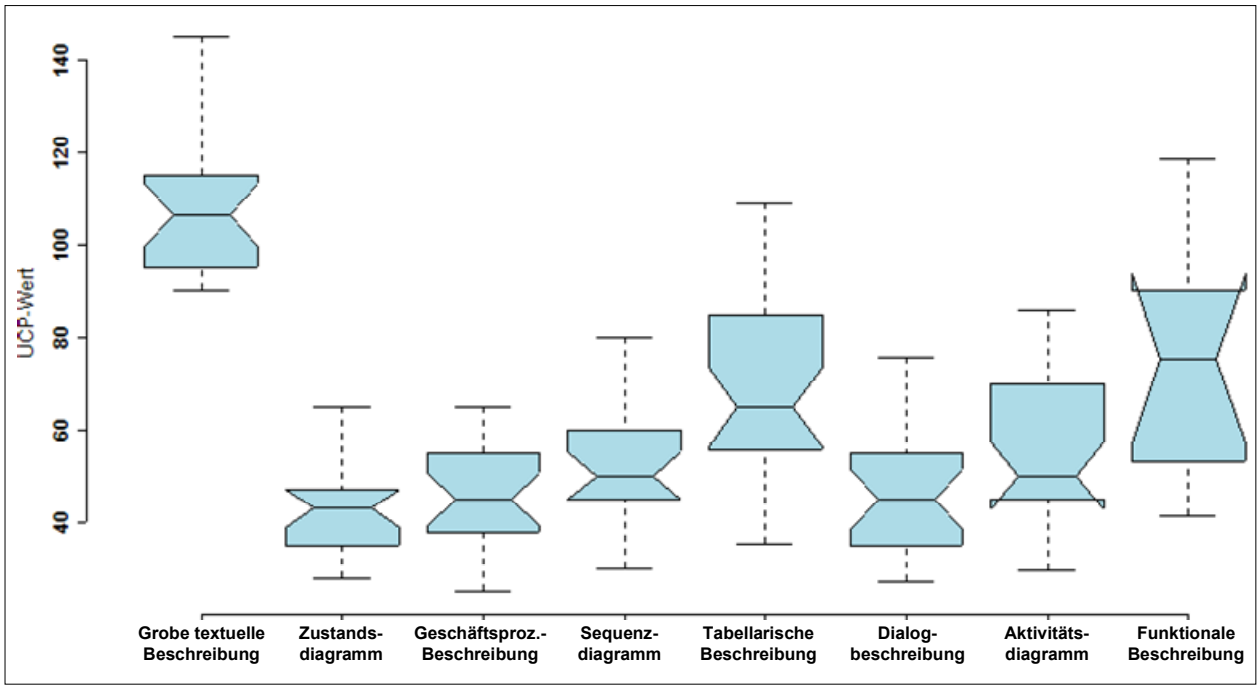


Abbildung 37: Boxplots der UCP Schätzreihen (ohne Ausreißer), Reihenfolge wie Tabelle 41

Auffällig viele Ausreißer (=14) zeigte die *Funktionale Beschreibung*, womit die Schätzreihe nur noch 10 Werte aufweist. Hier wurde von vielen Studierenden die vorgegebene Transformationsregel auf die UCP-Sprache missachtet und (fast) alle Anwendungsfunktionen als eigenständige Use Case gewertet, anstelle diese zu Anwendungsfällen zu bündeln.

Tabelle 41 fasst die Schätzreihen und die eingeführten statistischen Größen in Zahlen zusammen. Dargestellt sind sowohl die *UCP*-Werte (in Points) als auch die *Anzahl* der identifizierten Use Cases (#UC). Nachfolgend werden jetzt schrittweise die Messwerte interpretiert.

			UCP (Points)				Anzahl Use Cases	
#	**Spezifikationsform**	**n**	**Mittelwert**	**Median**	**σ**	**V_{UCP} [%]**	**Mittelwert**	**$V_{\#UC}$ [%]**
1.	**Grobe textuelle Beschreibung**	21	108±3	106,50	15	**14**	12,9±0,3	11
2.	**Zustandsdiagramm**	24	43±2	43,00	9	**22**	5,2±0,1	12
3.	**Geschäftsprozess-Beschreibung**	23	47±2	45,00	11	**24**	5,3±0,2	18
4.	**Sequenzdiagramm**	21	52±3	50,00	13	**25**	6,5±0,3	20
5.	**Tabellarische Beschreibung**	29	70±4	65,00	19	**27**	11,5±0,3	16
6.	**Dialogbeschreibung**	25	46±3	45,00	13	**28**	6,7±0,8	63
7.	**Aktivitätsdiagramm**	29	55±3	50,00	16	**29**	8,7±0,6	36
8.	**Funktionale Beschreibung**	10	75±8	75,25	24	**32**	13,0±1,5	37

Tabelle 41: Statistik zu den UCP Schätzreihen, sortiert nach aufsteigendem UCP Variationskoeffizient V. n gibt die Anzahl Daten je Reihe ohne Ausreißer an, σ die Standardabweichung.

Betrachten wir den relativen (zufälligen) Fehler $\Delta\mu/\mu$ für den UCP-Mittelwert. In Tabelle 41 ist für den Mittelwert $\mu\pm\Delta\mu$ angegeben. Daraus kann der relative Fehler leicht bestimmt werden, für die Schätzreihen Eins bis Sieben (alle mit n>20) schwankt er zwischen ca. 3% und 6% des UCP-Mittelwertes. Mögliche Unterschiede zwischen diesen Schätzreihen aufgrund des zufälligen Fehlers können also vernachlässigt werden, da n groß genug gewählt wurde. Kritisch ist Schätzreihe Nr. 8 (*Funktionale Beschreibung*) mit einem relativen Fehler des Mittelwertes von ca. 10% (75±8). Dem muss bei der Diskussion der Ergebnisse später besondere Beachtung gezollt werden. Allerdings ist hier aufgrund der hohen Zahl an Ausreißern auch die Überlagerung eines systematischen Fehlers gegeben, so dass die Belastbarkeit der Ergebnisse für diese Spezifikationsform sowieso schon sehr eingeschränkt ist.

Zwei Effekte sind bezüglich der beiden zugehörigen Variationskoeffizienten V_{UCP} und $V_{\#UC}$ zu diskutieren:

- Am Beispiel der Schätzreihe *Dialogbeschreibung* ist deutlich erkennbar, dass V_{UCP} (28%) deutlich von $V_{\#UC}$ (63%) abweichen kann. Hier zeigt sich besonders stark, dass bei der Transformation auf die UCP-Sprache Freiheitsgrade bestehen: Viele einfache Use Cases können gleichwertig alternativ durch wenigere aber komplexere Use Cases modelliert werden. Dieser Freiheitsgrad kann zu einem höheren $V_{\#UC}$ führen.
- Sind aus einer Spezifikation die Use Cases leicht ablesbar (z.B. *Grobe textuelle Beschreibung* oder *Tabellarische Beschreibung*), resultiert dies in einem niedrigen $V_{\#UC}$. In diesen Fällen kann die Streuung bei der Bewertung der Use Case Komplexität stärker ins Gewicht fallen als die Identifikation der Anzahl, und V_{UCP} ist möglicherweise größer als $V_{\#UC}$.

Diese Diskussion zeigt, dass $V_{\#UC}$ nur geringere Aussagekraft hat als V_{UCP}. Daher konzentrieren wir uns für die weitere Auswertung auf die UCP-Werte und V_{UCP}.

Es bleibt jedoch ein weiteres Problem beim Vergleich der V_{UCP} der Schätzreihen zu lösen. Für Spezifikationen, die nur sehr wenige[25] Use Cases enthalten, wird die Fehleinschätzung der Komplexität nur eines Use Cases bereits zu einer relativ hohen Abweichung vom Ideal-UCP-Wert führen. Im experimentellen Aufbau der Feldstudie ist es nicht möglich gewesen, die unterschiedlichen Spezifikationen auf die exakt *gleiche* funktionale Größe in Use Case Points zu kürzen. Bedingt durch diese unterschiedliche Größe der Spezifikationen ist nun eine nachträgliche Transformation nötig: Die UCP-Werte müsse in Relation zu ihrer mittleren Zahl an Use Cases gesetzt werden.

Für die Schätzreihen bedeutet dies, dass die Streuung der Ergebniswerte für Spezifikationen mit geringer Use Case Anzahl gestaucht und für Spezifikationen mit großer Use Case Anzahl gestreckt werden muss. Erst dadurch lassen sich die Streuungen fachlich sinnvoll miteinander vergleichen.

Die dazu notwendige Transformation erfolgt durch Multiplikation der UCP-Werte mit einem konstanten Faktor *b* für jede Schätzreihe gemäß Formel 19:

$b := \frac{\mu_{Spec}}{\mu_{all}}$	*Formel 19*

wobei μ_{Spec} die durchschnittliche Anzahl der Use Cases einer Spezifikationsform ist und μ_{all} die durchschnittliche Anzahl der Use Cases aller Spezifikationen der Feldstudie. Die Werte der Transformationskonstanten *b* liegen für die Schätzreihen (Spezifikationsformen) der Feldstudie zwischen 0,6 (*Zustandsdiagramm*) und 1,5 (*Funktionale Beschreibung*). Die transformierten UCP-Werte werden als *tUCP* bezeichnet und erlauben einen direkten Vergleich der Streuung der Schätzreihen je Spezifikationsform.

Abbildung 38 zeigt die Boxplots der transformierten tUCP Schätzreihen, sortiert nach aufsteigendem Variationskoeffizient von tUCP (V_{tUCP}). Die Reihenfolge der Spezifikationsformen weicht nun von der vor der Transformation ab (Abbildung 37 bzw. Tabelle 41). Zum Beispiel ist die *Grobe textuelle Beschreibung* nur noch an vierter Stelle gegenüber der ersten Position vor der Transformation. Grund ist, dass für diese Spezifikationsform der Mittelwert der Anzahl Use Cases aus der Schätzreihe deutlich höher ist als der für die meisten anderen Spezifikationsformen und daher die Transformationskonstante *b* (Formel 19) größer Eins ist. Durch diese Normalisierung auf vergleichbare Anzahl Use Cases erhöht sich die relative Streuung im Vergleich zu der Gesamtheit der anderen Spezifikationsformen.

Anhand der Variationskoeffizienten der tUCP-Werte können die Spezifikationsformen gemäß ihrer Streuung in vier Gruppen {*gering, mittel, hoch, sehr hoch*} mit jeweils im Rahmen der Messgenauigkeit gleichem Variationskoeffizient eingeteilt werden (Abbildung 38). Die gewählte

[25] wenige Use Cases meint hier die Größenordnung von weniger als 6 Use Cases, siehe [Eng08]

Gruppierung kann intuitiv aus der grafischen Darstellung der Boxplots geschlossen werden oder aber analytisch durch den Levene-Test bestätigt werden. Der Levene-Test prüft auf Varianzhomogenität zwischen mehreren Stichproben und setzt die Unabhängigkeit der Daten voraus, was durch den Aufbau der Feldstudie gemäß 4.1 sichergestellt ist.

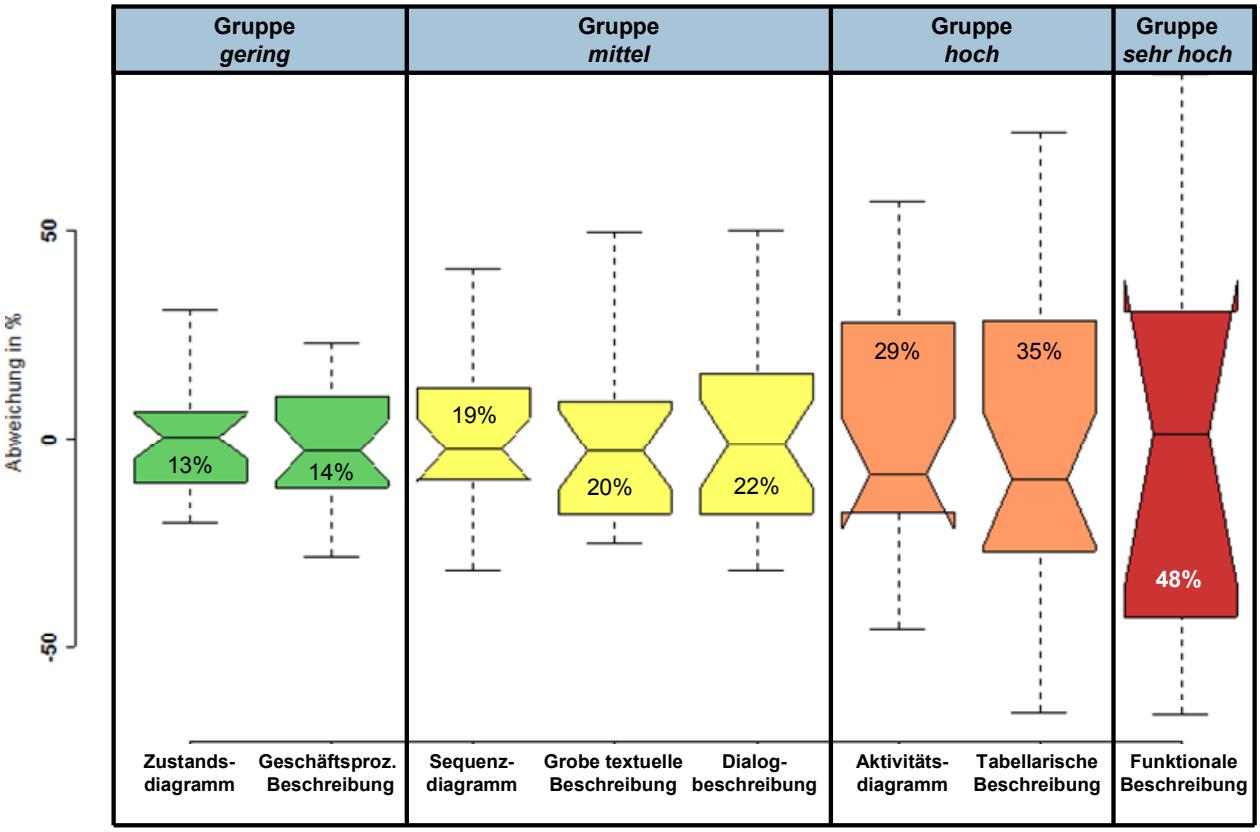


Abbildung 38: Boxplots der transformierten tUCP Schätzreihen als Abweichung vom Mittelwert in Prozent. Der Variationskoeffizient ist in den Boxen als Prozentwert angegeben.

Die Gruppierung gemäß Abbildung 38 wird durch den Levene-Test wie folgt bestätigt: Innerhalb einer Gruppe liegt keine Varianzhomogenität vor, da die ermittelten *p-Werte* mit {0,50; 0,66; 0,36} für die Gruppen {gering; mittel; hoch} jeweils größer als 0,05 (= 5%-Signifikanzniveau, siehe Abschnitt 4.3) sind und damit die Nullhypothese (es herrscht Varianzhomogenität) für jede Gruppe verworfen werden kann. Zwischen den Gruppen konnte analog keine Varianzhomogenität nachgewiesen werden (p-Wert < 5%) und für die Prüfung auf Varianzhomogenität zwischen allen Schätzreihen wurde mit einem p-Wert von $6{,}536e^{-06}$ (d.h. sehr viel kleiner als 5%) eine sehr hohe Signifikanz nachgewiesen.

Es wurde empirisch gezeigt, dass die UCP-Methode reproduzierbare Ergebnisse liefert. Die Güte der Reproduzierbarkeit wurde in der Feldstudie durch die Streuung der Schätzreihen bewertet und variiert signifikant für die unterschiedlichen Spezifikationsformen. Man könnte nun einwenden, dass diese Ergebnisse der Studierenden nicht repräsentativ für erfahrene Software-Ingenieure in der industriellen Praxis sind und möglicherweise die schlechter geeigneten Spezifikationsformen mit der Unerfahrenheit der Studierenden korrelieren. Dies wird im nächsten Abschnitt überprüft.

4.5 Vergleich der Feldstudie mit Experten-Schätzungen

Im Rahmen der Feldstudie wurde für jeden Studierenden sowohl dessen Erfahrung als Eigenbewertung sowie die aktuelle Semesterzahl erfasst. Eine Korrelation bezüglich dieser Größen als Maß für die Erfahrung der Studierenden und der Streuung des Schätzergebnisses (sowohl UCP wie #UC) konnte nicht nachgewiesen. Dies ist ein erster Hinweis, dass die UCP-Methode nicht maßgeblich von der Erfahrung des Schätzers abhängt.

In Ergänzung der Feldstudie mit Studierenden wurden nun vier Experten aus der industriellen Praxis eingewiesen, für die der Feldstudie zugrunde liegenden acht Spezifikationen jeweils eine UCP-Schätzung durchzuführen. Diese gemittelten Expertenschätzungen werden als Referenzergebnis für die verschiedenen Spezifikationen gewertet. Damit kann ein direkter Vergleich der Ergebnisse aus der Feldstudie mit Experten-Einschätzungen erreicht werden. Aufgrund der geringen Anzahl an Expertenschätzungen können weder Ausreißer identifiziert werden, noch ist eine fundierte statistische Analyse möglich, wohl ist aber eine Plausibilisierung der Feldstudien-Ergebnisse möglich. Ferner kann durch den Vergleich mit den Experten-Schätzungen auch eine Einschätzung über den Grad an notwendiger Erfahrung zur Durchführung einer UCP-Schätzung je Spezifikationsform gewonnen werden.

Tabelle 42 fasst die Schätzreihen der Experten zusammen, auf die Angabe des Fehlers der Mittelwerte wird aus Gründen der Übersichtlichkeit verzichtet, er kann aber aus den Daten leicht ermittelt werden.

	Grobe textuelle Beschreibung		Zustands-diagramm		Geschäfts prozess-Beschreibung		Sequenz diagramm		Tabellarische Beschreibung		Dialog-beschreibung		Aktivitäts-diagramm		Funktionale Beschreibung	
	#UC	UCP	#UC	UCP	#UC	UCP	#UC	UCP	#UC	UCP	#UC	UCP	#UC	UCP	#UC	UCP
Experte 1	8	80	3	35	3	25	6	35	11	90	5	55	6	30	-	-
Experte 2	17	65	5	40	-	-	7	40	11	70	5	35	6	40	10	55
Experte 3	12	95	5	45	6	40	3	25	11	90	5	45	6	42	5	25
Experte 4	12	112,5	5	40	5	40	7	40	11	81,5	5	40	7	45	5	37,5
Mittelwert	12	88	4,5	40	4,7	35	5,8	35	11	83	5	44	6,3	39	6,7	39
σ	3,7	20	1,00	4,1	1,5	8,7	1,9	7,1	0,0	9,5	0,0	8,5	0,5	6,5	2,9	15
V [%]	30	23	22	10	33	25	33	20	0	11	0	20	8	17	43	38
n	4	4	4	4	3	3	4	4	4	4	4	4	4	4	3	3

Tabelle 42: Statistik zu den Schätzreihen der Experten

Die folgenden Analysen verwenden die *Mittelwerte der Expertenschätzreihen*, um die Abweichungen der Schätzreihen der Studierenden von diesen zu berechnen. Da der „wahre" Projektaufwand unbekannt ist, wird hier der Expertenmittelwert als Bestwert ersatzweise verwendet. Es wird dabei unterstellt, dass aufgrund ihrer Erfahrung die Experten die UCP-Methode zuverlässiger anwenden können als die Studierenden.

Auch bei den Schätzungen der Experten gilt, dass die Standardabweichung wegen der unterschiedlichen Mittelwerte nicht direkt vergleichbar ist, deswegen verwenden wir wieder den Variationskoeffizienten. Der Variationskoeffizient bei den Experten variiert für die UCP-Mittelwerte zwischen 10% und 25% mit einem Ausreißer nach oben bei der *Funktionalen Beschreibung* mit 38%. Bei den Studierenden variiert der Variationskoeffizient zwischen 14% und 29% (Tabelle 41, Seite 103) und erreicht 32% für die *Funktionale Beschreibung*. Damit zeigen die Expertenschätzreihen eine im Rahmen der Messgenauigkeit gleich hohe Streuung der UCP-Mittelwerte wie die der Studierenden.

Bei der *Anzahl* identifizierter Use Cases (#UC) liegt bei den Experten eine Streuung zwischen 0% und 43% vor. Diese Streuung ist deutlich geringer als die der Studierenden, die von 11% bis 63% ausfällt.

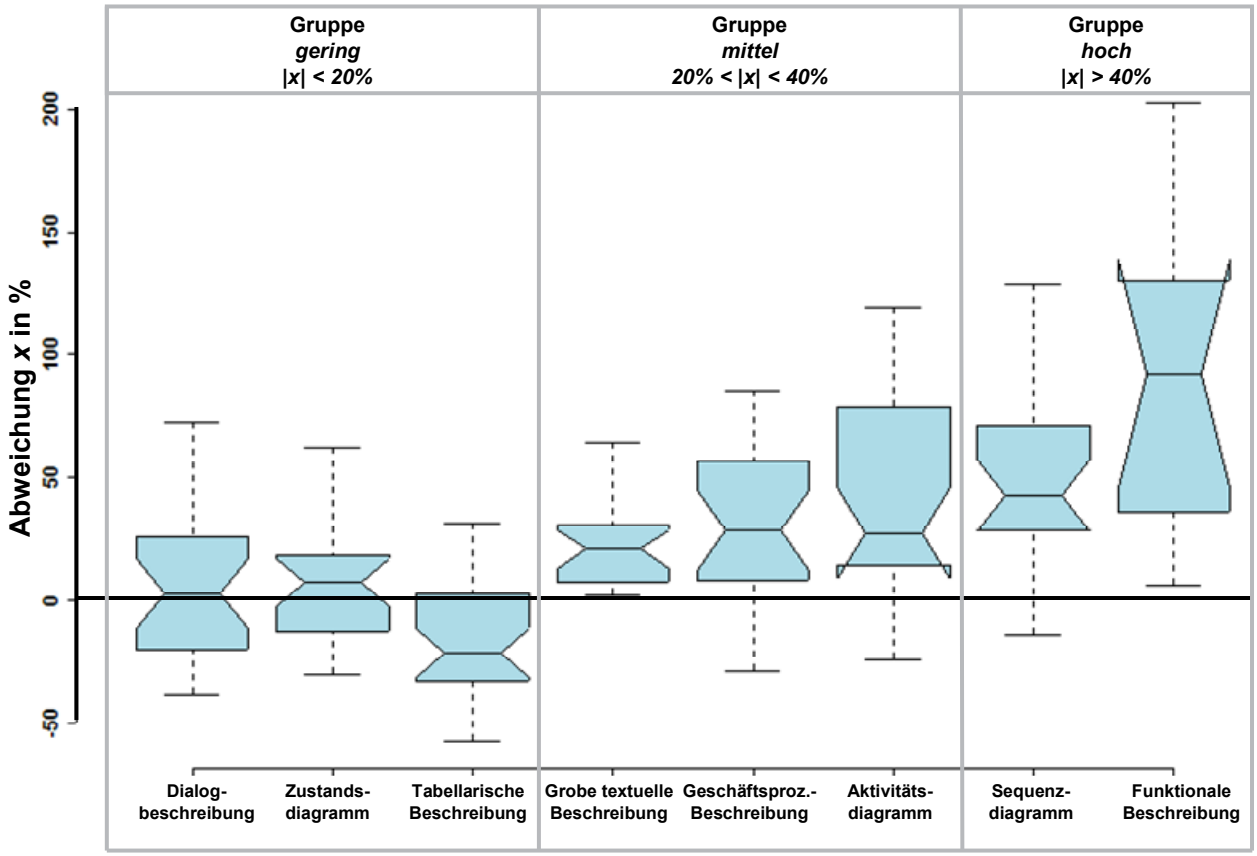


Abbildung 39: Boxplot der Abweichung der UCP-Mittelwerte der Studierenden von denen der Experten in Prozent

Der Boxplot in Abbildung 39 zeigt die Abweichung *x* je Spezifikationsform der UCP-Mittelwerte der Studierenden von denen der Experten in Prozent. Liegt ein Boxplot auf der vertikalen Achse relativ mittig auf der Nullachse, so stimmt der UCP-Mittelwert der Studierenden mit dem der Experten in etwa überein. Für die zugehörige Spezifikationsform ist die Erfahrung der Schätzperson also weniger wichtig. Liegt der Median des Boxplots sehr weit von der Nulllinie entfernt, so unterscheiden sich die beiden Schätzmittelwerte erheblich. Daraus kann der Schluss gezogen werden, dass die Erfahrung des Schätzers für diese Spezifikationsform einen größeren Einfluss auf das Schätzergebnis hat.

Betrachten wir beispielhaft die Spezifikationsform *Dialogbeschreibung*. Der Boxplot zeigt, dass ca. 50% der Studierenden einen UCP-Wert ermittelt haben, der vom Median der Experten zwischen -25% und + 30% abweicht. Der Median der Ergebnisse der Studierenden liegt sehr dicht am Expertenmittelwert. Daher ist der Einfluss von Erfahrung auf das Schätzergebnis im Falle der Dialogbeschreibung als gering einzustufen, selbst unerfahrene Schätzer können mit der UCP-Methode ein belastbares Ergebnis im Sinne der Reproduzierbarkeit durch Dritte erzielen. Damit ist für die meisten Spezifikationsformen (ausgenommen: *Sequenzdiagramm* und *Funktionale Beschreibung*) die in Anforderung *A 7* (Seite 42) enthaltene Forderung nach Anwendbarkeit auch durch Nicht-Techniker gezeigt worden.

Anders dagegen ist z.B. die Spezifikationsform *Sequenzdiagramm* sehr wohl abhängig von der Erfahrung des Schätzers, hier haben die Studierenden den UCP-Wert um mehr als 40% höher geschätzt.

In Abbildung 39 sind die Spezifikationsformen gruppiert nach dem Grad des Einflusses von Erfahrung auf die Schätzergebnisse in die Gruppen {*gering; mittel; hoch*}, wobei äquidistante Gruppen nach absoluter Höhe der Abweichung des Mittelwertes „Studierende zu Experten“ in 20%-Schritten gewählt wurden.

Aufgrund der geringeren Anzahl n=4 der Expertenschätzungen im Vergleich zu den Schätzungen der Studierenden (n>20) wird hier auf einen detaillierten Vergleich der transformierten Variationskoeffizienten V_{tUCP} verzichtet und dafür auf [Eng08] verwiesen.

Der Vergleich bestätigt die durch die Ergebnisse der Studierenden vorgenommene Gruppierung weitgehend, gilt für die Spezifikationsformen *Grobe textuelle Beschreibung*, *Aktivitätsdiagramm* und *Tabellarische Beschreibung* aber nur eingeschränkt.

4.6 Zusammenfassung der Ergebnisse der Feldstudie und Ausblick

Mit Hilfe einer Feldstudie an verschiedenen Hochschulen wurden über 200 UCP-Schätzungen von Studierenden auf Basis von acht verschiedenen praxisnahen Spezifikationsformen durchgeführt und die Ergebnisse mit denen von Experten verglichen. Die UCP-Schätzung beschränkt sich dabei auf die Identifikation von Use Cases gemäß der UCP-Sprache UCP 2.0. Grundlage waren reale Spezifikationen aus der industriellen Praxis, die für die Feldstudie gekürzt und anonymisiert wurden.

Eine Bewertung des Einflusses von Erfahrung des Schätzers auf das Ergebnis der UCP-Methode wurde durch den Vergleich der mittleren UCP-Werte von Studierenden zu Experten analysiert. Damit können die Spezifikationsformen qualitativ in drei Gruppen gemäß der benötigten Erfahrung des Schätzers aufgeteilt werden (Tabelle 43).

Die hohe Anzahl an Schätzungen erlaubt eine belastbare statistische Auswertung der Streuung der Use Case Identifikation (Kapitel 3, d.h. *Finden* und *Bewerten*) in Abhängigkeit von der Spezifikationsform. Die Varianzanalyse aus Abschnitt 4.4 zeigt auf, dass die Spezifikationsformen mit Sicherheit (im statistischen Sinne) unterschiedliche Varianzen haben. Je nach Spezifikationsform variiert die Streuung der UCP-Mittelwerte (Variationskoeffizient) zwischen 14% und 32%.

	Ergebnis der Feldstudie	
Spezifikationsform	**Gruppierung nach Variabilität**	**Bedeutung der Erfahrung des Schätzers**
Zustandsdiagramm	gering (13 – 14 %)	gering
Geschäftsprozess-Beschreibung		mittel
Sequenzdiagramm	mittel (19 – 22 %)	hoch
Grobe textuelle Beschreibung		mittel
Dialogbeschreibung		gering
Aktivitätsdiagramm	hoch (29 – 35 %)	mittel
Tabellarische Beschreibung		gering
Funktionale Beschreibung	sehr hoch (48 %)	hoch

Tabelle 43: Gegenüberstellung der Bewertung der Spezifikationsformen

Spezifikationsform	Vorteile	Risiken
Zustandsdiagramm		anderer "Schnitt" durch die Anwendung; Übersehen von Use Cases; nur im Zusammenhang mit textuellen Ergänzungen verwendbar!
Geschäftsprozess-Beschreibung	Aufzählung der Use Cases kann direkt übernommen oder einfach gefunden werden; guter Überblick der Anwendung	Unterschätzung durch zu wenig Informationen über die Szenarien/Schritte
Sequenzdiagramm	Schnitt der Use Cases bereits vorgegeben; Schritte und Szenarien einfach ersichtlich	Gefahr zu feiner Granularität; in der Regel keine vollständige Abdeckung der Anwendung; Überprüfung der textuellen Ergänzung empfohlen
Grobe (textuelle) Beschreibung	Aufzählung kann direkt übernommen werden; Szenarien und Schritte sind grob beschrieben	Unterschätzung durch zu wenig Informationen über die Szenarien/Schritte, Unterschätzung durch vergessene ganze Use Cases
Dialogbeschreibung	Use Cases und Szenarien sind relativ einfach zu finden; gute Abdeckung der Anwendung	verdeckte Schritte; Anwendungslogik ist nicht erfasst, daher immer textuelle Ergänzungen beachten
Aktivitätsdiagramm	Direkte Übernahme von Use Cases und Zählung der Szenerien und Schritte; einfach und schnell	Gefahr der Vernachlässigung der Überprüfung gegen textuelle Beschreibung, Aktivitätsdiagramme können zu fein sein.
Tabellarische Beschreibung	Aufzählung kann direkt übernommen werden; Szenarien und Schritte sind bereits beschrieben	
Funktionale Beschreibung	Beschreibung der Schritte und Funktionen; gute Abdeckung der geforderten Funktionalität der Anwendung	Komposition zu Szenarien und Use Cases u.U. aufwändig; Gefahr, in Funktionen und damit zu detailliert zu schätzen

Tabelle 44: Zusammenfassung der Vorteile und Risiken der Spezifikationsformen (aus Tabelle 31 und Tabelle 37)

Dies kann als Methodenfehler der UCP-Methode in Abhängigkeit von der Spezifikationsform interpretiert werden. Dabei ist zu beachten, dass die UCP-Methode mit sinkender Anzahl Use Cases, d.h. für sehr kleine Entwicklungsprojekte, einen zunehmenden methodischen Fehler liefert, da sich die Über- und Unterschätzungen der *Komplexität* eines jeden einzelnen Use Case durch den Schätzer nicht mehr gut herausmitteln kann. Dies bezeichnen wir als *intrinsischen Methodenfehler*, der vom Typ *zufälliger Fehler* gemäß Abschnitt 4.3 ist und gemäß Formel 17 umgekehrt proportional zur Wurzel aus der Anzahl Use Cases sein muss. Eine exakte Untergrenze kann nicht angegeben werden, weniger als 10 Use Cases wird aber zumindest als kritisch angese-

hen und bei mehr als 50 Use Cases wird dieser intrinsische Fehler mit Sicherheit vernachlässigbar sein.

Mit Hilfe der Feldstudie konnten vier Gruppen unterschiedlicher Varianz ermittelt werden (Abbildung 38). Geringe Varianz zeigt hohe Reproduzierbarkeit (als reziproke Funktion der Standardabweichung) und damit hohe Güte für die UCP-Schätzung an.

So sind die Spezifikationsformen *Zustandsdiagramm* und *Geschäftsprozess-Beschreibung* sehr gut für die UCP-Methode geeignet, wohingegen die Formen *Aktivitätsdiagramm* und *Tabellarische Beschreibung* in geringerer Reproduzierbarkeit resultieren. Die *Funktionale Beschreibung* erfordert äußerste Sorgfalt insbesondere beim Mapping auf die UCP-Sprache: Sowohl bei den Studierenden wie bei den Experten ist dies die Form mit der mit Abstand höchsten Streuung und damit geringsten Güte der Reproduzierbarkeit. Diese Ergebnisse können nun direkt mit den in Abschnitt 3.3.7 und 3.4.4 genannten Vorteilen und Risiken der Spezifikationsformen verglichen werden. Tabelle 44 fasst die detaillierten Tabellen aus Abschnitt 3.3.7 und 3.4.4 zur leichteren Nachvollziehbarkeit für die nachfolgende Diskussion nochmals zusammen.

Der Vergleich zeigt, dass die qualitative Einschätzung aus Kapitel 3 für die Spezifikationsformen *Sequenzdiagramm*, *Grobe textuelle Beschreibung* und *Dialogbeschreibung* gut mit der Varianzanalyse der Feldstudie zusammen passt: Die Variabilität ist als *mittel* eingestuft und passt zu der Bewertung der Vorteile der Spezifikationsformen in Tabelle 44. Für *Sequenzdiagramme* wurde bereits in Abschnitt 3.3.6 auf die Gefahr einer zu starken Dekomposition hingewiesen und es besteht zudem die Gefahr, dass nicht alle Use Cases erfasst werden. Dies zu erkennen erfordert viel Erfahrung und wird durch die Feldstudie bestätigt.

Für die Formen *Zustandsdiagramm*, *Grobe textuelle Beschreibung* und *Tabellarische Beschreibung* wurde aus der Feldstudie eine geringe Bedeutung der Erfahrung des Schätzers ermittelt. Der Grund dafür könnte darin liegen, dass die Use Cases sehr leicht aus der Spezifikation übernommen werden können (siehe Abschnitt 3.3.1 und 3.3.2). Diese These wird durch die sehr niedrigen $V_{\#UC}$ Werte (Tabelle 41) bestätigt: Eine geringe Streuung hinsichtlich der *Anzahl* der Use Cases belegt eine leichte Auffindbarkeit von Use Cases. Für die *Zustandsdiagramme* mag dies zwar generell nicht gelten (Abschnitt 3.3.4), das konkrete Spezifikationsbeispiel hat allerdings den Studierenden keine Schwierigkeiten gemacht. Dies kann durchaus an dem speziellen Beispiel im Feldversuch liegen und muss nicht verallgemeinerbar sein.

Als Vorteil der *Dialogbeschreibung* wurde in Abschnitt 3.4.1 genannt, dass Use Cases und Szenarien relativ einfach zu finden sind (Tabelle 44) und eine gute Abdeckung der Anwendung erzielt wird. Dies passt gut zur geringen Einstufung der erforderlichen Erfahrung aus der Feldstudie, trotzdem liegt die Variabilität nur im Mittelfeld. Grund könnte sein, dass bei dieser Spezifikationsform die Diskrepanz zwischen V_{UCP} und $V_{\#UC}$ erstaunlich hoch ist (Abschnitt 4.4) und das legt nahe, die Zählregeln in der UCP Methode (verwendet wurde UCP 2.0, die Zählregeln sind aber identisch zu UCP 1.0) zu überarbeiten und die Klassifizierung von Dialogen klarer zu fassen. Dies begründet nun auch empirisch die Überarbeitung des Konzeptes *Dialoge* in UCP 1.0 und Bewertung der Use Cases durch die präziser definierten Interaktionselemente in UCP 3.0, wie in Abschnitt 3.1 angekündigt.

Die Feldstudie lässt hinsichtlich der Spezifikationsform *Funktionale Beschreibung* die Erfordernis von sehr viel Erfahrung erkennen. Dies begründet nun auch die schon in Abschnitt 4.4 genannte ungewöhnlich hohe Zahl von 14 Ausreißern und rechtfertigt nun nachträglich, diese als Ausreißer klassifiziert zu haben. Die Variabilität ist nach Entfernen der Ausreißer immer noch höher als bei anderen Spezifikationsformen und auch in der Schätzreihe der Experten (hier wurden keine Ausreißer entfernt) zeigt die *Funktionale Beschreibung* die höchste Streuung. Damit ist diese Spezifikationsform für die UCP-Methode am ungünstigsten. In Abschnitt 3.4.4 (Tabelle 37) wurde bereits auf die große Gefahr der Dekomposition hingewiesen. Weiterhelfen können hier z.B. ergänzende textuelle Erläuterungen oder weitere Spezifikationsformen, die dann genug Information für eine angemessene Einschätzung liefern. Das in der Feldstudie verwendete Spezifikationsbeispiel *Funktionale Beschreibung* ist hiermit ein gutes Beispiel für eine besonders schwierige Spezifikationsform, die zu unterschiedlicher Dekomposition bei unterschiedlichen Schätzern verleitet: Sowohl bei den Experten wie bei den Studierenden ist $V_{\#UC}$ vergleichsweise sehr hoch. Dies beweist, dass die beschriebene Gefahr nicht gebannt werden konnte. Als Konsequenz daraus sollten unabhängige Personen UCP-Schätzungen überprüfen, die auf Basis einer *Funktionale Beschreibung* erstellt wurden. Besonders auf den „richtigen“ Schnitt der identifizierten Use Cases im Sinne von Abschnitt 3.2 ist bei der Prüfung zu achten.

Die Spezifikationsformen *Aktivitätsdiagramm* und *Tabellarische Beschreibung* zeigen in der Feldstudie eine hohe Variabilität, was eine schlechte Eignung bedeutet. Dies mag im Widerspruch zu dem beschriebenen Vorteil (Tabelle 44) stehen, dass die Use Cases direkt übernommen werden können. Zunächst fällt auf, dass in diesen Fällen die Varianz der Experteneinschätzung (=mittel bis niedrig) nicht gut zu der der Studierenden (=hoch) passt. Dies lässt den Schluss zu, dass die Experten besser mit dieser Spezifikationsform klar gekommen sind als die Studierenden. Ein Beweis dafür liefert $V_{\#UC}$ der Experten mit 8% bzw. 0% (d.h. vergleichsweise sehr niedrig) und 36% bzw. 16% bei den Studierenden. Dies steht aber im Widerspruch zu dem Ergebnis, dass die Erfahrung wenig Einfluss auf das Ergebnis zeigt. Diese Aussage wiederum ist aus der geringen Abweichung der UCP-Mittelwerte (Studierende ggü. Experten) geschlossen worden. Dies bedeutet, dass die Experten zwar mehrheitlich die gleiche Anzahl Use Cases identifiziert haben, bei der Bewertung der Komplexität der Use Cases aber erhebliche Unterschiede verzeichneten. Der beschriebene Vorteil der direkten Übernahme der Use Cases gilt also nur hinsichtlich der *Identifikation* der Use Cases, nicht jedoch hinsichtlich der Bewertung der *Komplexität*, oder anders herum gesagt: Aus der leichten Identifikation von Use Cases darf nicht automatisch auf eine gute Eignung für die UCP-Methode geschlussfolgert werden. Es ist aber auch möglich, dass zumindest für die *Tabellarische Beschreibung* das gewählte Beispiel für diese Feldstudie Schwächen aufzeigt und einfach schlecht gewählt war. Eine abschließende Beantwortung ist allerdings nicht möglich, dafür würde es umfangreichere Schätzreihen der Experten (d.h. $n > 20$) benötigen, die nicht vorliegen.

Abschließend lässt sich festhalten, dass für weitergehende Untersuchungen es notwendig wäre, den wahren (aber im Rahmen dieser Feldstudie unbekannten) Projektaufwand zu den Spezifikationsformen mit einzubeziehen. Beim Aufbau einer UCP-Schätzdatenbank für ein Unternehmen wäre dazu die jeweilige (dominante) Spezifikationsform einer UCP-Schätzung mit aufzunehmen. Diese Information liegt für die im Rahmen dieser Dissertation verfügbaren Projektschätzungen nicht vor, wird aber zukünftig bei Capgemini sd&m mit erfasst werden und sollte dann bei weitergehenden wissenschaftlichen Untersuchungen in der Zukunft genutzt werden. Diverse Studien

in der Literatur [And02, MAC05] vergleichen den eingetretenen Aufwand mit dem Schätzaufwand, berücksichtigen derzeit aber nur unzureichend oder gar nicht die den Schätzungen zugrundeliegende Spezifikationsform. Zukünftige Studien sollten aufgrund der hier erzielten Ergebnisse diesen Aspekt berücksichtigen.

Diese Feldstudie wurde auf acht unterschiedlichen Projekten, basierend auf acht unterschiedlichen Spezifikationsformen, aufgebaut. Dadurch wurde nur ein Spezifikationsdokument pro Projekt analysiert. Dies könnte eine Quelle für einen systematischen Fehler sein, auch wenn sehr viel Sorgfalt aufgebracht wurde, Störeffekte zu vermeiden. In weiterführenden zukünftigen Untersuchungen empfehlen wir daher, unterschiedliche Spezifikationsformen für den identischen fachlichen Projektkontext des Software-Entwicklungsprojektes zu verwenden. Die Spezifikation zu einem Projekt müssten dazu in unterschiedliche Spezifikationsformen übertragen werden und auf dieser Basis Schätzreihen erhoben werden. Dies hätte dann den Charakter einer Laborstudie, da die Spezifikationen „künstlich“ erstellt werden. Trotzdem wäre dieses Vorgehen zu empfehlen, um die hier präsentierten Ergebnisse der Feldstudie zu überprüfen.

Ferner wäre es hilfreich, einen größeren Satz an Expertenschätzungen zu den Spezifikationsformen dieser hier präsentierten Feldstudie zu haben, um die Expertenschätzungen ebenfalls einer Varianzanalyse unterziehen zu können.

Für die Weiterentwicklung der UCP-Methode hat diese Feldstudie jedoch bereits wertvolle Hinweise gegeben und zudem den Beweis erbracht, dass mit Hilfe des in Abschnitt 3.2 entwickelten Leitfadens die folgenden kritischen Punkte aus Abschnitt 2.7 gelöst werden:

- P 1 Unterschiedlicher Schnitt der Use Cases
- P 2 Reuse Schätzfehler
- P 4 Kaum relevante vergleichbare Praxiserfahrung verfügbar
- P 8 Interpretationsspielraum von Use Cases

Diese Feldstudie hat den neu entwickelten A-Faktor aus Kapitel 3 empirisch überprüft und bestätigt. Der T-Faktor und M-Faktor gemäß Abbildung 20 (Seite 48) wird nun in Kapitel 5 zunächst entwickelt und eine empirische Überprüfung wird in Kapitel 6 im Kontext der dann vollständig entwickelten und dargestellten Methode UCP 3.0 präsentiert werden.

5 Entwicklung eines neuen Kostenfaktor-Modells

„Was waren in deinem Projekt die drei wichtigsten Einflussgrößen von Kostenfaktoren?“ Diese Frage, gestellt an 10 Experten (Projektleiter oder Chefarchitekten) unterschiedlicher Projekte und Auftraggeber, führt zu knapp 30 höchst unterschiedlichen Antworten, denn im Kontext der Individualsoftware-Entwicklung gleichen sich Projekte nur unzureichend, die Kostenfaktoren („cost drivers“) fallen je nach Umfeld, Technologie, Team und Auftraggeber ganz unterschiedlich aus. Und hätte ich diese Frage vor 20 Jahren gestellt, wären die Antworten wiederum anders ausgefallen: So nennt COCOMO [Boe81] in seiner *Basic*-Fassung aus dem Jahr 1981 nur drei Einflussgrößen (Abschnitt 2.5, Tabelle 11). Eine deutliche Weiterentwicklung erfolgt mit der im Jahr 2000 veröffentlichten Fassung COCOMO II [Boe+00] mit fünf exponentiellen „scaling drivers“ und 17 „effort multipliers“ (Abschnitt 2.5). Ein einfaches Mapping zwischen beiden Versionen ist nicht möglich. Entweder, weil die Einflussgröße aufgrund des technischen Fortschrittes heute anders bewertet wird oder weil die Detailbeschreibung der Spezifikation geändert oder präzisiert wurde und die Bewertungen dann unterschiedlich ausfallen können.

Es stellt sich die Frage, wie die relevanten und wichtigsten Einflussgrößen auf ein Software-Entwicklungsprojekt gefunden werden und in den beiden Kostenfaktoren der UCP-Methode (T-Faktor und M-Faktor) implementiert werden können. Dadurch wird der Problempunkt P 6 (Definition von TCF und EF sind nicht mehr zeitgemäß) aus Abschnitt 2.7 gelöst.

Aufgrund der begrenzten Anzahl an Projektdaten müssen im Vorfeld der Weiterentwicklung einige einschränkende Vorüberlegungen getroffen werden (Abschnitt 5.1) und bei der Wahl der Einflussgrößen die Erfahrungen von bereits weiterentwickelten Methoden einbezogen werden (Abschnitt 5.2). Es folgt dann die Auswahl der Kostenfaktoren für UCP 3.0 anhand einer Expertenumfrage. Dazu werden die Grundlagen eingeführt (Abschnitt 5.3), die Entwicklung und Planung einer für den Kontext von UCP 3.0 geeigneten Umfrage erarbeitet (Abschnitt 5.4) und die Ergebnisse dargestellt (Abschnitt 5.5). Die Formel für die Kostenfaktoren wird unter Berücksichtigung der Aufgabenstellung und des Problempunktes P 5 (UCP Schätzmodell verwendet unzulässige Skalen-Transformationen) aus Abschnitt 2.7 dann in Abschnitt 5.6 weiterentwickelt und abschließend das finale Komplexitätsmodell vorgestellt (Abschnitt 5.7), welches dann P 7 (Komplexitätsstufen der TCF und EF sind zu vage definiert) durch eine detaillierte Definition löst.

5.1 Zu viele Freiheitsgrade in den Kostenfaktoren

Wenn nicht anders genannt, benutzen wir im Kontext der UCP-Methode die in Abschnitt 2.8 eingeführten Begriffe und Konzepte. Wird die Original-Methode von Karner gemäß Abschnitt 2.4.1 referenziert, sprechen wir kurz von der *Karner-Methode*. In diesem Abschnitt wollen wir aufgrund einiger quantitativer Vorüberlegungen ein Verfahren zur Ableitung der Kostenfaktoren für die UCP-Methode entwickeln.

Ein Ansatz wäre, mit Datamining Methoden [Lus02] aus einer großen Menge an Projektdaten die möglichen Einflussgrößen aufgrund von Korrelationen zu erkennen und empirisch durch Muster-

erkennung aus den Daten zu gewinnen. Dies setzt erstens eine sehr große Anzahl an Referenzprojekten voraus und erfordert zudem, dass für jedes dieser Projekte alle erdenklichen Einflussparameter in vergleichbarer Weise erfasst und auf einer Ordinal-Skala bewertet wurden. Diese Datenbasis ist heute nicht verfügbar und insbesondere die zweite Voraussetzung stellt eine extrem große Aufgabe dar, die in absehbarer Zeit nicht gelöst werden kann.

Eine einfache Plausibilisierung macht dies deutlich: Jede einzelne Einflussgröße der beiden Kostenfaktoren der Karner-Methode kann mit je sechs Werten [0..5] beantwortet werden. Die Kostenfaktoren haben dabei zusammen 13 + 8 = 21 Einflussgrößen. Dann würden sich zusammen 6 x 21 = 126 mögliche Freiheitsgrade ergeben. Es stellt sich die Frage, wie alle diese Einflussgrößen zueinander ins Gewicht gesetzt werden können. Ziel ist es, eine mathematische Formel zu finden, die in der Projektschätzung über alle Projekte zu einer minimalen Standardabweichung (siehe dazu auch Abschnitt 4.3) führt. Unter der Annahme, dass der Projektaufwand als stetige Funktion von den Einflussgrößen abhängt (siehe Formel 15), können durch Regressionsanalyse [DH98, UM06] numerisch die Werte für die Gewichte gefunden werden. Für eine Regressionsanalyse muss die Zahl der Datenpunkte sehr viel größer sein als die Zahl der Freiheitsgrade, damit die Lösung nicht unterbestimmt ist. Um eine statistisch belastbare numerische Lösung zu ermitteln bräuchte es folglich weit mehr als 126 Projektdaten.

Die größte öffentlich verfügbare unabhängige Projektdatenbank ist die der ISBSG[26] mit knapp 5.000 erfassten Projektschätzungen. Davon sind aber nur ca. 1.500 Neuentwicklungen und es dominiert IFPUG FPA für die Größenmessung. Schätzungen mittels der UCP-Methode sind kaum vorhanden. Eine Nacherhebung fehlender Kostenfaktoren ist praktisch nicht möglich, ebenso besteht nicht die Möglichkeit, in direkter Diskussion mit den Projektverantwortlichen wichtige ergänzende Hintergrundinformationen zu den Projekten zu erhalten.

Die Sammlung von belastbaren und hinterfragbaren Projektdaten ist eine wesentliche Grundlage für eine datenorientierte Analyse. Daher werden bei Capgemini sd&m ab 2007 alle großen Entwicklungsprojekte systematisch erhoben und ausgewertet. Für UCP 3.0 konnte auf 19 Projektschätzungen zurückgegriffen werden. Zum Vergleich: Boehm hat für COCOMO auf 83 Projekte und für COCOMO II auf 161 Projekte Zugriff. Die Kostenfaktoren aus COCOMO II sollen in die Weiterentwicklung der UCP-Methode einfließen.

Trotzdem ist zum heutigen Zeitpunkt ein ausschließlich auf Datenanalyse basierender Ansatz aufgrund zu geringer Datenbasis nicht möglich.

Es stellt sich die Frage, ob anstelle des datenbasierten Ansatzes ein expertenbasierter Ansatz trägt. Für die Weiterentwicklung der Methode UCP 1.0 zu einer Methode UCP 2.0 wurden dazu in Expertenworkshops relevante Einflussgrößen diskutiert und es konnte Einstimmigkeit erzielt werden, dass z.B. die Einflussgrößen E_1 (Familiar with Rational Unified Process) und E_5 (Object oriented experience) aus Tabelle 10 (Seite 32) der Karner-Methode nicht mehr zeitgemäß sind und weggelassen werden können. Ebenfalls wurden neue Faktoren identifiziert, wie z.B. der Einfluss des gewählten *Prozessmodells* [FJE06]. Bei der Festlegung der Gewichte der Einflussgrößen untereinander konnte jedoch weder in einem moderierten Expertenworkshop noch in einer

[26] www.isbsg.org, Non-Profit-Organisation, Stand November 2008

Umfrage unter einer großen Zahl von Experten eine Gewichtung ermittelt werden, da die Einschätzungen der Experten zu stark streuen bzw. die Experten sich gar nicht in der Lage sehen, für die Einflussgrößen verbindliche Verhältnisgrößen als Gewichte anzugeben [Lam07]. Ein ausschließlich auf Expertenanalyse basierender Ansatz ist folglich nicht möglich.

Die Kombination aus datenorientierter Analyse und expertenorientierter Analyse erscheint erfolgsversprechender zu sein. Steece beschreibt in [Ste99] einen solchen kombinierten Ansatz für die Entwicklung von COCOMO II. Die Idee dabei ist, in den Bereichen, wo die Datenbasis gut ist und die Experten unsicher oder widersprüchlich sind, a posteriori ein durch Daten bestimmtes Modell aufzubauen, und in den Bereichen, wo die Datenbasis dünn und die Expertenmeinung übereinstimmend ist, a priori ein durch Experteneinschätzung bestimmtes Modell aufzubauen.

Durch diesen Ansatz ist es möglich, für die UCP-Methode eine Einschränkung auf wenige, aus Expertensicht als relevant qualifizierte Einflussgrößen zu bekommen und die genauen Gewichte dann durch Datenanalyse abgeschlossener Projekte zu ermitteln. Aufgrund von Experteneinschätzungen können zudem bestimmte Nebenbedingungen in das Schätzmodell hinein definiert werden und damit die Zahl der Freiheitsgrade weiter reduziert werden. Dies wird in Abschnitt 5.6 weiter entwickelt werden.

Aufgrund der Vorüberlegungen leiten wir zunächst sieben Schritte als Verfahren zur Bestimmung eines Kostenfaktor-Modells ab (*Kostenfaktorbestimmungsverfahren*):

1. Literaturanalyse durchführen, um bereits bekannte Einflussgrößen aus anderen Methoden zu identifizieren
2. Liste der Einflussgrößen zusammenstellen, gruppieren und Redundanzen eliminieren
3. Expertenumfrage zur Relevanz der Einflussgrößen durchführen
4. Korrelationsanalyse durchführen, um Widersprüche und Übereinstimmung der Experten je Einflussgröße statistisch zu identifizieren => finale Liste der relevanten Einflussgrößen
5. Expertenworkshop durchführen, um für jede als relevant eingestufte Einflussgröße aus 4.) eine *Metrik* zu finden (d.h. konkrete Ausformulierung der Komplexitätsbedingung/Ausprägung je Punktwert der Einflussgröße anhand von Beispielen) => Einflussgrößen definiert
6. Projektdatenbank hinsichtlich der definierten Einflussgrößen aufbauen
7. Regressionsanalyse durchführen, um Gewichte der Einflussgrößen zu bestimmen => Kostenfaktoren definiert

Abbildung 40: Verfahren zur Bestimmung eines Kostenfaktor-Modells

Der 3. Schritt dieses Verfahrens baut dabei bewusst auf einer vorgegebenen Liste von Einflussgrößen auf. Eine Umfrage, bei der die Teilnehmer Antworten aus einer vorgegebenen Liste auswählen können, nennt man auch *gestützte* Umfrage. Zum Einen werden damit die bisherigen Erfahrungen und Erkenntnisse anderer Methoden mit eingebunden. Andererseits wird aber auch vermutet, dass eine ungestützte Umfrage zu keinen brauchbaren Ergebnissen führt, da die dann genannten Einflussgrößen nicht ausreichend vergleichbar sind. Zur Überprüfung dieser Vermu-

tung werden wir in der Umfrage trotzdem die Möglichkeit vorsehen, weitere Einflussgrößen ungestützt zu nennen.

5.2 Vergleich mit bekannten Komplexitätsfaktoren

Im ersten Schritt des Kostenfaktorbestimmungsverfahrens (Abbildung 40) werden Schätzmethoden aus der Literatur untersucht, die möglichst bekannt sind (zum Beispiel durch eine ISO-Normung) und aktiv genutzt werden. Zudem müssen die Schätzmethoden die Kostenfaktoren aufweisen, die zur Berechnung des Gesamtaufwands herangezogen werden. Wichtige Schätzmethoden wurden bereits in Abschnitt 2.2 dieser Arbeit beschrieben und - soweit erforderlich - in Abschnitt 2.6 miteinander verglichen. Eine Relevanzaussage hinsichtlich der Kostentreiber fassen wir nachfolgend zusammen:

- ISO 20968:2002. Mark II FPA wird nicht weiter untersucht, da die Methode in der nach ISO standardisierten Fassung keine Kostenfaktoren mehr berücksichtigt. Ursprünglich setzt die Methode auf die 14 Kostentreiber der Methode IFPUG FPA auf und ergänzt diese um fünf weitere Kostentreiber: *Anforderungen von anderen Anwendungsschnittstellen*, *Datenschutz- und Sicherheitsanforderungen*, *Anwenderschulungsaufwand*, *direkte Anbindung Dritter, Dokumentation* [UKS98]. Diese fünf Faktoren werden jedoch in leicht abgewandelter Form von der UCP-Methode im TCF mit erfasst und gehen daher für die weitere Analyse nicht verloren.
- ISO 19761:2003. Die Full Function Point (FFP) Methode berücksichtigt keine Kostentreiber.
- ISO 20926:2004. Die Methode IFPUG FPA berücksichtigt 14 Kostentreiber, die wir in die Untersuchung mit aufnehmen.
- ISO 24570:2006. Die NESMA-Methode ist hinsichtlich der Kostentreiber identisch zu IFPUG FPA und wird daher nicht gesondert betrachtet.
- ISO 29881:2008. Die Methode FiSMA 1.1 berücksichtigt keine Kostentreiber.

Ferner sind die Kostentreiber der Methode COCOMO II (Abschnitt 2.5) sowie der UCP-Methode nach Karner (Abschnitt 2.4) zu untersuchen. Für die UCP-Methode wurde zudem zusätzlich eine industrielle Weiterentwicklung der Credit Suisse unter dem Namen MT230 berücksichtigt. Eine detaillierte Beschreibung dieser Methode ist in [Bad08] dokumentiert. Möglicherweise existieren noch weitere UCP-Varianten in anderen Unternehmen, allerdings sind diese nicht öffentlich verfügbar und konnten im Rahmen dieser Dissertation nicht berücksichtigt werden.

Im nächsten zweiten Schritt des Kostenfaktorbestimmungsverfahrens (Abbildung 40) werden nun die Kostenfaktoren aus allen Methoden zusammengetragen und eine vergleichende Gesamtliste erstellt. Tabelle 45 stellt die Einflussgrößen aus den Kostentreibern der oben genannten Methoden gegenüber. Die Sortierung ist dabei so gewählt, dass möglichst gut Cluster gemeinsamen Vorkommens erkannt werden können, d.h. zu oberst stehen die Einflussgrößen, die von den meisten Methoden verwendet werden. Der Bezeichner der Einflussgröße wurde so gewählt, dass er allgemeinen Charakter hat. Dabei wurde bereits eine Einordnung in den T-Faktor und M-Faktor im Sinne von Abschnitt 2.8 vorgenommen, unabhängig davon, wie die Einflussgröße in der je-

weiligen Methode ursprünglich eingeordnet ist. Der T-Faktor stellt dabei nichtfunktionale Anforderungen dar, der M-Faktor erfasst Prozess- bzw. Projektmerkmale.

T-Faktor Einflussgröße	IFPUG FPA 4.1	COCOMO II	UCP (Karner)	Credit Suisse UCP MT230	Experten-Umfrage
Komplexe Geschäftsregeln	C9	CPLX	T4		1
Benutzeroberfläche	C6,C7		T3,T7	4.2.8.3	1
Schnittstellen	C1		T12	4.2.8.7	1
Verteiltes System	C2		T1	4.2.8.1	1
Verfügbarkeitsanforderungen			T10	4.2.8.2	1
Anford. Wiederverwendbarkeit	C10	RUSE	T5	4.2.8.4	?
Performance	C3,C5		T2	4.2.8.6	?
Flexibilität des Systems	C8		T9	4.1.8.5	?
Ausfallsicherheit (Reliability)		RELY			?
Installation	C11		T6		
Sicherheitsanforderungen			T11		
Anwenderschulung			T13	4.2.5	
Anforderung an Portabilität			T8		
Auslastung Zielumgebung	C4	TIME, STOR			
Anzahl Clients	C13				
Betreibbarkeit	C12				
Flexibilität der Architektur	C14				
Ähnliche Produkte		PREC			
Datenbankgröße		DATA			
Tausch Hard-/Software		PVOL			
Flexibilität der Entwicklung		FLEX			
Anzahl:	**14**	**9**	**13**	**7**	**5**

M-Faktor Einflussgröße	IFPUG FPA 4.1	COCOMO II	UCP (Karner)	Credit Suisse UCP MT 230	Experten-Umfrage
Lstg./Fähigkeit Chefdesigner		ACAP	E3		1
Reife der Prozesse		PMAT			1
Verfügbare Zeit		SCED			1
Teamplayer		TEAM			1
Kontinuität Mitarbeiter		PCON			1
Stabile Anforderungen			E8		1
Anzahl Entscheidungsträger				4.2.7	1
Integrationsabhängigkeit				4.2.8.8	1
Erfahrung Fachlichkeit		APEX	E4	4.2.6	?
Review und Architektur		RESL			?
Erfahrung Werkzeuge		LTEX			?
Verteiltes Arbeiten		SITE			?
Lstg./Fähigk. Programmierer		PCAP			?
Erfahrung Plattform/Middlew.		PLEX			
Dokumentation		DOCU			
Unterstützung d. Werkzeuge		TOOL			
Motivation			E6		
Verfügbarkeit Team			E2		
Effizienz Programmiersprache			E7		
Erfahrung mit RUP			E1		
Erfahrung Objektorientierung			E5		
Anzahl:	**0**	**13**	**8**	**3**	**9**

Legende: (Bewertung der Einflussfaktoren aus Sicht der Experten-Umfrage)
relevant | ? evtl. relevant aber Streuung hoch | fehlt | überflüssig | nicht relevant

Tabelle 45: Gegenüberstellung der jeweiligen vergleichbaren Einflussgrößen unterschiedlicher Schätzmethoden

Wird eine Einflussgröße in einer Zeile durch mehrere Methoden verwendet, kann dies im Detail durchaus in unterschiedlicher Weise erfolgen. An dieser Stelle geht es nur darum, einen weitgehend gemeinsamen Bezeichner zu finden, der den grundsätzlich bewerteten Aspekt beschreibt (z.B. *Komplexe Geschäftsregeln* als Bezeichner für *Complex Proces*sing in IFPUG FPA, *Product Complexity* in COCOMO II und *Complex Internal Processing* in der Karner-Methode). Der Kurz-Bezeichner (z.B. *C9, CPLX* oder *T4*) in den Spalten je Methode entspricht dann aber der Terminologie der jeweiligen Methode. Insgesamt werden durch die Vereinigungsmenge der Einflussgrößen je 21 unterschiedliche Aspekte für den T-Faktor und den M-Faktor genannt, zusammen also 42 Einflussgrößen.

Es wird deutlich, dass die Methoden unterschiedliche Schwerpunkte legen. IFPUG FPA z.B. setzt seinen Schwerpunkt klar auf den T-Faktor mit 14 Einflussgrößen und keinem Beitrag aus dem M-Faktor, COCOMO II berücksichtigt beide Kostenfaktoren etwa gleichgewichtig. Die UCP Karner-Methode zeigt im T-Faktor eine breite Überlappung mit IFPUG FPA, im Vergleich zu COCOMO II sind die Einflussgrößen aber sehr unterschiedlich. Die Weiterentwicklung der UCP Methode der Credit Suisse (MT230) orientiert sich im T-Faktor an einem Ausschnitt der Karner-

Methode, im M-Faktor bringt sie aber zwei neue Faktoren ein und übernimmt nur einen von acht Faktoren der Karner-Methode.

Eine Validierung der für die UCP-Methode tatsächlich relevanten Einflussgrößen erfolgt durch Auswertung einer Vielzahl von Expertenmeinungen. Dazu wird die Vereinigungsmenge aller 42 Einflussgrößen nun einer Umfrage von 25 Experten[27] unterzogen (3. Schritt des Kostenfaktorbestimmungsverfahrens, Abbildung 40). Die Grundlagen und das Vorgehen dazu werden in den nächsten Abschnitten erläutert, das Ergebnis ist aber bereits in Tabelle 45 aus Gründen der besseren Vergleichbarkeit gegenübergestellt. In der entsprechenden Spalte bedeutet der Eintrag „1“, dass diese Einflussgröße eindeutig von den Experten als sehr relevant beurteilt wurde[28], ein „?“ signalisiert zwar eine zustimmende Einschätzung der Experten, allerdings gibt es auch ablehnende Einschätzungen und insgesamt streut das Umfrage-Ergebnis zu stark für eine eindeutige Beurteilung. Diese Information wurde aus dem 4. Schritt (Korrelationsanalyse) des Kostenfaktorbestimmungsverfahrens (Abbildung 40) ermittelt.

Auffällig ist, dass die Einschätzung der als relevant eingestuften Einflussgrößen in beiden Kostenfaktoren von der Einschätzung der Credit Suisse geteilt wird, obwohl die MT230 Methode unabhängig von Capgemini sd&m entwickelt wurde und bei der Experten-Umfrage den Teilnehmern die konkrete Quelle der jeweiligen Einflussgröße nicht transparent gemacht wurde. Damit sind diese unabhängigen Bewertungen von zwei unterschiedlichen Unternehmen kongruent.

5.3 Entwicklung und Planung der Expertenumfragen zur Validierung der Einflussgrößen

Zu den empirischen Forschungsmethoden in der Softwaretechnik zählt Prechelt neben der in Abschnitt 4.1 eingeführten *Feldstudie* die *Umfrage* [Pre01]. Bei einer Umfrage beantworten eine ausgewählte Personengruppe (Zielgruppe) mehrere vom Experimentator sorgfältig formulierte Forschungsfragen. Die Beantwortung kann schriftlich (Fragebogen) oder mündlich (strukturiertes Interview) erfolgen. Die Antworten können zur quantitativen Ausrichtung der Umfrage schematisch erfolgen oder aber in der Form frei sein. „Die Auswertung einer Umfrage erlaubt vorsichtige Rückschlüsse auf die subjektive Wirklichkeit der Teilnehmer in Bezug auf die Forschungsfrage“ [Pre01].

Die zu klärende Forschungsfrage lautet:

1a) Wie wird die Relevanz (=Größe des Einflusses auf den Projektaufwand) der 42 Einflussgrößen aus Tabelle 45 im Kontext der Entwicklung betrieblicher Informationssysteme eingeschätzt?

1b) Können diese Einflussgrößen hinsichtlich ihrer Relevanz in ein Verhältnis zueinander gesetzt werden?

---

[27] Die Experten kommen aus dem Unternehmen Capgemini sd&m und verfügen über langjährige Erfahrung in Projekten der hier vorliegenden Systemklasse der betrieblichen Informationssysteme.

[28] Als Kriterium für die Relevanz wurden die Mittelwerte der Experten-Antworten der in Abschnitt 5.4 definierten Skala zugrunde gelegt. Sei *Max* der größte Wert und *Min* der kleinste Wert auf dieser Skala aller normierten Antworten., dann wurden in Tabelle 45 alle die Projekte als relevant gekennzeichnet, deren Bewertung größer als *(Max – Min) / 2* ist.

2) Gibt es weitere Einflussgrößen, die in 1) nicht berücksichtigt wurden?

Damit wir eine gute statistische Bewertung der Streuung der Experteneinschätzung erhalten, wurde zur Beantwortung der Frage 1a) ein Fragebogen anstelle eines mündlichen Interviews gewählt. In gewisser Weise gelten auch die Überlegungen aus Abschnitt 4.1 zum Aufbau eines Experimentes, insbesondere die verschiedenen Aspekte der Störvariablen. Allerdings beziehen sich die Einschätzungen der Experten auf ganz unterschiedliche Projekte und Kontexte. Die Vergleichbarkeit der Einschätzungen muss auf einer höheren Abstraktionsebene, nämlich der der Relevanz der Faktoren, erfolgen. Im Aufbau der Umfrage sind dabei die wesentlichen Aspekte des kontrollierten Experimentes aus Abschnitt 4.1 zu gewährleisten: *Auswahl, Reifung, Sterblichkeit, Instrumentation, Anforderungscharakteristik*.

Diese Aspekte sind besonders gut zu erzielen, wenn die möglichen Antworten auf eine feste Auswahl begrenzt werden. Dies ist nur für die Forschungsfragen 1a) und 1b) möglich, nicht jedoch für die offene Fragestellung 2).

Es wurde bereits dargelegt, dass Experten auf Basis einer Umfrage, welche über möglichst viele Projekte aus dem Erfahrungsschatz der Experten mitteln soll, nicht in der Lage sind, die Relevanz einer Einflussgröße auf einer absoluten Verhältnis-Skala anzuordnen (Forschungsfrage 1b), die dann die Höhe der Gewichte des Kostenfaktors ergibt. Daher haben wir uns für eine feste Vorgabe möglicher Antworten (Multiple-Choice-Verfahren) je Einflussgröße mit einer Ordinalskala entschieden, welche dann durch statistische Auswertung die Festlegung der Verhältnisskala nach Abschluss der Umfrage durch statistische Analysen erlaubt. Dieses Vorgehen wurde in einem Vortest durch eine kleine Zahl an Experten als machbar bestätigt. Folgende drei exklusiv alternative Antworten je Einflussgröße waren in der Umfrage möglich:

1. Die Einflussgröße ist vorhanden, befindet sich aber bei allen sd&m-Projekten auf einem identischen Niveau und braucht deswegen nicht erhoben zu werden. Die Metrik ist nicht relevant. Bei der Berechnung des Aufwands geht die Einflussgröße (unabhängig vom Projekt) immer mit einem festen Faktor ein. Die Größe dieses Faktors können wir mithilfe der empirischen Analyse bestimmen.
2. Die Einflussgröße ist bei sd&m-Projekten nicht relevant (oder kommt bei sd&m-Projekten sehr selten vor, ggf. Beispiel angeben)
3. Die Einflussgröße wirkt sich in unterschiedlichen sd&m-Projekten unterschiedlich stark aus. Ich beurteile die Gewichtung der Einflussgröße im Vergleich zu den anderen relevanten Einflussgrößen mit folgendem Faktor:

□ ¼-fach	□ ½-fach	□ 1-fach	□ 2-fach	□ 4-fach	□ 8-fach
(gering)	(weniger als normal)	(normal)	(mehr als normal)	(stark)	(sehr stark)

Die Forschungsfrage 2) wurde durch eine offene Frage nach weiteren Einflussgrößen abgedeckt, für die dann die 3. Antwortmöglichkeit ebenfalls vorgesehen wurde. Der Wortlaut des gesamten Fragebogens ist in Anhang A dokumentiert, weitere Details zur Umfrage finden sich in [Bad08].

Diese drei möglichen Antworten sind wie folgt motiviert:

ad 1.) Mit dieser möglichen Antwort sollen Einflussgrößen markiert werden, die sich firmenweit auf einem einheitlichen Niveau bewegen und deshalb nicht bei jedem Projekt als mögli-

che Aufwandsquelle bewertet werden müssen. Während des Vergleichens der Einflussgrößen lag die Vermutung nahe, dass die meisten Team-bezogenen Einflussgrößen in diese Rubrik fallen, da aufgrund der Firmengröße und der Firmenkultur an allen deutschen Standorten von einem einheitlichen Bild ausgegangen werden kann.

ad 2.) Zu dieser Antwort wird den Teilnehmern in der Einführung der Umfrage das *Wetter* als Beispiel gegeben: Es ist davon auszugehen, dass die Projektaufgaben unabhängig vom Wetter erledigt werden können. Das Ziel dieser möglichen Antwort ist es, Kostenfaktoren zu markieren, die zwar messbar und real sind, aber für die Projektabwicklung keine Relevanz haben, da diese zum Beispiel nicht zum Umfang des Schätzergebnisses gehören, in der Hand des Kunden liegen oder aus anderen Gründen nicht relevant sind.

ad 3.) Bei dieser möglichen Antwort sollten die Teilnehmer „die Gewichtung der relevanten Kostenfaktoren im Verhältnis zueinander" bestimmen. Bei einer Pilotumfrage mit wenigen Teilnehmern stellte sich heraus, dass die direkte Zuordnung der Gewichtung zu einem Kostenfaktor schwierig war. Deshalb musste zu dem angegebenen Zahlenwert ein ordinalskaliertes Merkmal hinzugefügt werden.

5.4 Grundlagen zur Auswertung der Expertenumfrage

Für die Auswertung der Umfrage (4. Schritt des Kostenfaktorbestimmungsverfahrens, Abbildung 40) fassen wir nachfolgend wichtige Grundlagen und Vorüberlegungen zusammen. Es gelten hinsichtlich der statistischen Behandlung zudem die bereits in Abschnitt 4.3 zusammengefassten Größen und Zusammenhänge.

Es wurden 25 Experten befragt, der Rücklauf erbrachte n=24 auswertbare Fragebögen. Damit kann der zufällige Fehler (da n>20) vernachlässigt werden, die Umfrage wird statistisch belastbar sein (Abschnitt 4.3). Zu jeder Einflussgröße erhalten wir aus der Umfrage eine *Datenreihe* mit bis zu 24 Werten[29].

Je Einflussgröße werden Gewichtungen {0, ¼, ½, 1, 2, 4, 8} genutzt. Gleichzeitig wurde jedes Merkmal mit einem der ordinalen Merkmale {gering, weniger als normal, normal, mehr als normal, stark, sehr stark} gleichgesetzt. Die Teilnehmer wurden aufgefordert, bei der Bewertung der relevanten Einflussgrößen das Verhältnis zueinander zu wahren, eine absolute Bewertung ist aber nicht möglich. Damit lassen sich die Ergebnisse *eines* Teilnehmers gut miteinander vergleichen und bewerten.

Problematisch gestaltet sich, dass die Skala nur durch die optionalen ordinalen Merkmale fixiert war. Die Bewertungen zwischen den Teilnehmern sind deshalb nur mit Vorsicht zu vergleichen: Teilnehmer A bewertet die Kostenfaktoren ausschließlich mit 2, 4 und 8. Teilnehmer B meint dasselbe, bewertet aber mit 1, 2 und 4. Deshalb muss im ersten Schritt eine Normierung vorgenommen werden. Dazu werden die Werte des Teilnehmers durch den größten Wert in seinem Fragebogen dividiert und damit auf Werte im Bereich zwischen 0 und 1 normiert. Bei Teilneh-

[29] Eine Datenreihe je Einflussgröße setzt sich nur aus der 3. Antwortmöglichkeit des Experten zusammen, andernfalls liegt kein Wert vor. Für wenig relevante Einflussgrößen kann damit auch n<20 entstehen, allerdings ist die Einflussgröße dann wahrscheinlich wenig relevant.

mer A und B im Beispiel ergeben sich damit die Werte ¼, ½ und 1. Die einzelnen Bewertungen zwischen den Teilnehmern sind nun vergleichbar.

Um intervallskalierte Merkmale auszuwerten, lässt sich der *Bestwert* über das arithmetische Mittel bilden. Die Berechnung des arithmetischen Mittelwerts verbietet sich aber bei Werten auf einer Ordinalskala, hier wird der Median für den Bestwert empfohlen [BS87]. Es werden aus den Ergebnissen der Normierung für jede Einflussgröße das arithmetische Mittel *und* der Median berechnet, da unbekannt ist, welche Skala ein Teilnehmer für die Bewertung der Einflussgröße genutzt hat. Die Ergebnisse werden nach dem arithmetischen Mittelwert absteigend sortiert. Hat sich bei den Werten des Medians auch eine absteigende Reihenfolge gebildet, kann davon ausgegangen werden, dass die so gefundene Rangfolge valide ist. Es wird sich später zeigen, dass die Unterscheidung zwischen Median und Mittelwert für die hier vorliegende Auswertung nicht notwendig ist, da die Reihenfolge für die relevanten Einflussgrößen in beiden Fällen identisch ist.

Gemäß den Vorüberlegungen aus Abschnitt 5.1 ist je Einflussgröße zu prüfen, ob die Einschätzung der Experten einheitlich oder widersprüchlich ist. Eine Bewertung kann anhand der Streuung der Datenreihen vorgenommen werden: Liegt eine Normalverteilung vor, kann davon ausgegangen werden, dass die Teilnehmer einheitlich eine ähnliche Einschätzung der Relevanz haben, welche durch den Mittelwert ausgedrückt wird. Zeigt die Datenreihe aber eine Gleichverteilung über die gesamte Skala, kann dies verschiedene Ursachen haben: Die Teilnehmer hatten keine einheitliche Meinung, die Fragestellung war unklar oder die Einschätzung der Teilnehmer ist je Einflussgröße „gewürfelt“ und damit nicht reproduzierbar – in jedem Fall sollte eine Gleichverteilung mit Vorsicht behandelt werden.

Mit dem Chi²-Verteilungstest kann geprüft werden, ob die vorliegenden Stichproben einer bestimmten Verteilung entsprechen [BS87]. Dazu wird die Nullhypothese aufgestellt, dass eine Datenreihe eine Gleichverteilung aufweist. Es wird wieder das 5%-Signifikanzniveau zu Grunde gelegt (Abschnitt 4.3), ergänzend wird auch das 10%-Signifikanzniveau untersucht, sofern das 5%-Niveau nicht erfüllt wird. In den Fällen, in denen die Nullhypothese nicht abgelehnt werden kann, muss davon ausgegangen werden, dass es sich bei der Datenreihe um eine Gleichverteilung handelt. In Tabelle 45 wurden solche Einflussgrößen bereits mit einem Fragezeichen gekennzeichnet.

5.5 Ergebnisse der Expertenumfrage

Auf Grundlage der zuvor beschriebenen Vorüberlegungen wurde eine Expertenumfrage unter 25 Experten (siehe Fußnote 27, Seite 118) durchgeführt (3. Schritt des Kostenfaktorbestimmungsverfahrens, Abbildung 40). Der Wortlaut der Umfrage ist im Anhang A dokumentiert. Bei der Auswahl der Experten wurde sichergestellt, dass ein möglichst breiter Projekt- und Erfahrungshintergrund abgedeckt wird. Als Experten wurden Projektleiter oder Chefdesigner ausgewählt, die bereits langjährige Berufserfahrung und nach eigenen Angaben mehrere Projekte geleitet haben (Mittelwert 7,6 ±0,8 Projekte; Minimum: 2, Maximum: 19), die relevant sind und aus deren Erfahrung sie den Fragebogen beantworten. Dabei wurde auf eine gute Verteilung zwischen kleinen (bis 5 Mitarbeiter), mittleren (6 – 20 Mitarbeiter) und großen (über 20 Mitarbeiter) Projekten geachtet.

Tabelle 46 fasst die Ergebnisse der Umfrage nach absteigendem Mittelwert (der Relevanz der Einschätzung) sortiert zusammen. Die Spalten bedeuten im Einzelnen:

Nr. gibt die fortlaufende Nummerierung aus der Umfrage an.

n nennt die Anzahl der auswertbaren Antworten gemäß der 3. Antwortmöglichkeit (Fußnote in Abschnitt 5.4).

Median, *Mittelwert* und *Variationskoeffizient* folgen den Definitionen aus Abschnitt 4.3 und beziehen sich auf die Relevanz-Einschätzung der Experten, normiert auf die in Abschnitt 5.4 entwickelte Skala zwischen Null und Eins.

Gleichverteilung enthält dann einen Eintrag, wenn Gleichverteilung vermutet werden muss. Angegeben ist, ob Gleichverteilung nur nach dem strengen 5%-Signifikanzniveau oder sogar auch bei dem weniger strengen 10%-Signifikanzniveau vermutet wird.

UCP-Name gibt den zukünftigen Bezeichner in der UCP 3.0 Methode an (siehe nachfolgende Diskussion der Ergebnisse). Kursiv dargestellt sind Bezeichner, die in anderer Form oder nur zur Beobachtung übernommen werden, also eingeschränkt gültig sind. Die Nummerierung der Bezeichner hat historische Gründe und ist heute ohne Bedeutung. Es wurde nicht neu durchlaufend nummeriert, um veröffentlichte Bezeichner aus einer UCP-Vorversion nicht umbenennen zu müssen.

Die Einflussgröße *Anwenderschulung* des T-Faktors aus Tabelle 45 wurde im Fragebogen nicht aufgenommen, damit umfasst die Umfrage nur 41 Einflussgrößen. Der Grund dafür liegt in der Einschätzung, dass Aufwände für umfangreichere Schulungen nicht Bestandteil einer UCP-Schätzung (UCP 3.0) sein sollten. Erfahrungsgemäß wird mit dem Auftraggeber die Anzahl Schulungstermine und der Umfang gesondert vereinbart und daraus der Aufwand direkt abgeschätzt. Dies korreliert aber nicht zwangsläufig mit der funktionalen Größe der Anwendung sondern hängt von der Anzahl und der räumlichen Verteilung der Schulungsteilnehmer, deren Vorwissen und dem Schulungskonzept ab (z.B. „Train the Trainer“ oder Schulung aller Anwender). Die Methode MT230 der Credit Suisse empfiehlt ein vergleichbares Vorgehen wohingegen die Karner-Methode den Schulungsaufwand im T-Faktor abbildet und damit proportional zum funktionalen Umfang (UUCP) definiert. Wir greifen diesen Aspekt im Kontext des T-Faktors in Abschnitt 5.5.2 und im Kontext der durch die UCP-Methode abgedeckten Projektaufwände in Abschnitt 6.1 wieder auf.

Aus dem Rücklauf der Umfrage konnten n=24 Fragebögen ausgewertet werden. Es wurden nur von einem Experten weitere Einflussgrößen über die 41 des Fragebogens hinaus angegeben. Dies ist ein Beleg für das in Abschnitt 5.1 beschriebene Vorgehen einer gestützten Umfrage auf zuvor für die Umfrage vorgegebenen Einflussgrößen.

Tabelle 46 nennt die Einflussgrößen in absteigender Reihenfolge des Mittelwertes der Relevanz. Damit hat diese Liste die Funktion eines Rankings und damit kann nun sehr einfach ausgewählt werden, wie viele oder bis zu welchem Schwellwert Einflussgrößen in die Kostenfaktoren übernommen werden. Aufgrund der in Abschnitt 5.1 erläuterten Problematik der zu vielen Freiheitsgrade und der begrenzten Zahl an Projekten zur Kalibrierung der UCP-Methode werden wir ca. 10 Einflussgrößen betrachten können. Dies passt gut zu der Zielstellung einer schlanken Methode, wie in der Anforderung *A 3* (Tabelle 14, S. 44) formuliert.

Kosten-faktor	Nr.	Einflussgröße	n	Median	Mittel-wert	Variations-koeffizient	Gleich-verteilung	UCP-Name
M-Faktor	9	Stabile Anforderungen	22	0,8	0,7	44%		M7
M-Faktor	41	Integrationsabhängigkeit	22	0,5	0,6	53%	5%	M9
M-Faktor	11	Verfügbare Zeit	22	0,5	0,6	49%		M6
M-Faktor	15	Erfahrung Fachlichkeit	21	0,5	0,6	57%	10%	M10
M-Faktor	6	Review und Architektur	20	0,5	0,6	54%	5%	M4
M-Faktor	17	Leistung/Fähigkeit Chefdesigner	19	0,5	0,6	50%		M1
T-Faktor	27	Performance	24	0,5	0,5	61%	10%	T2
T-Faktor	1	Komplexe Geschäftsregeln	21	0,5	0,5	52%		*T4*
M-Faktor	25	Anzahl Entscheidungsträger	21	0,5	0,5	52%		M8
M-Faktor	18	Leistung/Fähigkeit Programmierer	17	0,5	0,5	55%	5%	*M1b*
M-Faktor	10	Erfahrung Werkzeuge	19	0,5	0,5	66%	10%	*T15*
T-Faktor	33	Anforderungen Wiederverwendbarkeit	21	0,5	0,5	70%	10%	T5
M-Faktor	19	Kontinuität Mitarbeiter	20	0,5	0,5	55%		M3
T-Faktor	2	Schnittstellen	20	0,5	0,5	51%		T12
T-Faktor	29	Ausfallsicherheit (Reliabilität)	23	0,5	0,4	62%	10%	*T14*
T-Faktor	26	Verteiltes System	21	0,3	0,4	58%		T1
M-Faktor	20	Verteiltes Arbeiten	20	0,3	0,4	65%	10%	*M11*
M-Faktor	16	Team-Player	20	0,5	0,4	33%		M2
T-Faktor	4	Flexibilität des Systems	21	0,5	0,4	72%	10%	T9
T-Faktor	38	Verfügbarkeitsanforderungen	23	0,3	0,4	54%		T10
T-Faktor	3	Benutzeroberfläche	20	0,3	0,4	42%		T3, T7
M-Faktor	21	Reife der Prozesse	22	0,4	0,4	45%		M5
M-Faktor	24	Verfügbarkeit Team	19	0,3	0,4	71%	10%	
M-Faktor	12	Erfahrung Platform/Middleware	19	0,3	0,3	77%	10%	
M-Faktor	40	Unterstützung durch Werkzeuge	19	0,3	0,3	48%		
T-Faktor	34	Flexibilität der Architektur	18	0,3	0,3	48%		
T-Faktor	28	Flexibilität der Entwicklung	17	0,3	0,3	85%	10%	
M-Faktor	22	Motivation	14	0,3	0,3	88%	10%	
M-Faktor	13	Dokumentation	20	0,3	0,3	59%	5%	
T-Faktor	30	Datenbankgröße	17	0,3	0,3	106%	10%	
T-Faktor	32	Sicherheitsanforderungen	21	0,3	0,3	58%	5%	T11
M-Faktor	23	Erfahrung Objektorientierung	12	0,3	0,2	81%		
T-Faktor	31	Anzahl Clients	19	0,3	0,2	71%		
M-Faktor	35	Effizienz Programmiersprache	20	0,3	0,2	101%		
T-Faktor	8	Betreibbarkeit	20	0,1	0,2	73%		
T-Faktor	36	Anforderungen an Portabilität	21	0,3	0,2	86%		T8
T-Faktor	5	Installation	19	0,1	0,2	87%		T6
M-Faktor	14	Erfahrung mit RUP	15	0,1	0,1	78%		
T-Faktor	39	Tausch Hard-/Software	22	0,0	0,1	199%		
T-Faktor	7	Ähnliche Produkte	21	0,1	0,1	122%		
T-Faktor	37	Auslastung Zielumgebung	20	0,0	0,1	171%		

Tabelle 46: Ranking der Einflussgrößen nach dem Mittelwert der Relevanz

Unter den Top 10 Einflussgrößen sind 8 dem M-Faktor und nur 2 dem T-Faktor zuzurechnen. Auch die Gesamtsumme der Mittelwerte für den M-Faktor ist mit 9,1 deutlich höher als 6,4 für den T-Faktor. Dies bedeutet, dass dem M-Faktor insgesamt ein stärkeres Gewicht zukommen sollte. Andererseits nennt die Karner-Methode insgesamt mehr Einflussgrößen im *TCF* als im *EF*.

Dies ist ein weiterer Beleg für den Problempunkt P 6 (Definition von TCF und EF sind nicht mehr zeitgemäß). Bei der Weiterentwicklung der Kostenfaktoren wird dem M-Faktor daher höhere Priorität einzuräumen sein.

Wie bereits erläutert, werden wir uns für die Neuentwicklung des Kostenfaktormodells aufgrund der noch geringen Datenbasis auf nur ca. 10 Einflussgrößen beschränken. Die Auswahl der Top 10 Einflussgrößen aus Tabelle 46 hätte allerdings zur Folge, dass der T-Faktor deutlich weniger Einflussgrößen aufweist als der M-Faktor. Wir werden daher für den T-Faktor der Methode UCP 3.0 zunächst die Einflussgrößen und Gewichte aus der Karner-Methode übernehmen und nur geringfügige Modifikationen vornehmen und stattdessen den M-Faktor komplett neu aufbauen.

5.5.1 M-Faktor

Für die Auswahl der Einflussgrößen setzten wir uns nun das Ziel, 10 Einflussgrößen aus Tabelle 46 auszuwählen. Im Rahmen von Experten-Workshops werden dann die genauen Metriken je Einflussgröße entwickelt. Dabei werden Überlegungen aus den Methoden COCOMO II, IFPUG FPA, MT230 und der Karner-Methode (siehe Kapitel 2) einbezogen, bewertet, mit der Erfahrung der Experten verglichen und in einem kreativen Akt eine Metrik je Einflussgröße festgelegt (5. Schritt des Kostenfaktorbestimmungsverfahrens, Abbildung 40).

Die Bestimmung der zugehörigen Gewichte (6. und 7. Schritt des Kostenfaktorbestimmungsverfahrens) erfolgt durch empirische Datenanalyse und ist Gegenstand des Abschnittes 5.7. Es liegt aufgrund der Experteneinschätzung die Vermutung nahe, dass Einflussgrößen mit hoher Relevanz ein höheres Gewicht erhalten werden, als Einflussgrößen mit geringerer Relevanz.

Die nachfolgende Diskussion der Einflussgrößen baut auf dem absteigenden Ranking gemäß Tabelle 46 auf. Die Kurzbezeichner *M1*, *M2, M3* etc. im neuen M-Faktor sind aus Gründen der Konsistenz zu früheren Veröffentlichungen [Fro08, FE08b] nicht fortlaufend gemäß Relevanz nummeriert, sondern orientieren sich teilweise an Vorversionen der Methode UCP 2.0. Tabelle 47 nennt die detaillierte Definition (Metrik) der Einflussgröße für den neuen M-Faktor der Methode UCP 3.0. Dabei ist je Einflussgröße anhand von Beispielen und detaillierten Beschreibungen angegeben, wie viele Punkte für welche Ausprägung zu wählen sind.

Stabile Anforderungen (M7)[30]: Diese Einflussgröße erfasst den Grad der Stabilität der Anforderungen, wie sie die Spezifikation zum Zeitpunkt der Aufwandsschätzung vermuten lässt. Gründe für instabile Anforderungen können sein:

- Änderungswünsche des Auftraggebers während des Projektverlaufes (z.B. Change Requests)
- Verfeinerung der Spezifikation im Projektverlauf durch zu erwartende Korrekturen an der bisherigen Spezifikation

Change Requests (CR) werden üblicherweise gegen Aufpreis durchgeführt und müssen eigentlich in der UCP-Schätzung nicht berücksichtigt werden. Die Projektpraxis zeigt je-

[30] Als Lesehilfe sei angemerkt, dass die nachfolgend diskutierten Einflussgrößen der Reihenfolge in Tabelle 46 folgen, aber nur solche mit Eintrag *M-Faktor* in der ersten Spalte (Kostenfaktor) berücksichtigt werden.

doch, dass eine hohe CR-Rate auch insgesamt ein Zeichen für eine instabile Spezifikation ist und in einem erhöhten Projektaufwand resultiert. Wir werden diese Einflussgröße unter der Kurzbezeichnung *M7* aufnehmen und ein hohes relatives Gewicht erwarten. Diese Vermutung wird durch COCOMO II bestärkt, wo die Einflussgröße *PMAT* (Tabelle 12, Seite 38) als Scale Factor exponentiell eingeht und das größte Gewicht hat. Um diesem exponentiellen Verhalten Rechnung zu tragen wählen wir eine nach oben asymmetrische Metrik, d.h. die Ausprägung 3 = normal wird in Richtung sehr stabiler Anforderungen mit minimal einem Punkt, aber in Richtung sehr hoher Änderungsrate auf maximal sechs (statt sonst fünf) Punkte erweitert.

Integrationsabhängigkeit (M9): Die Expertenumfrage ergibt hierfür die zweithöchste mittlere Relevanz, allerdings zeigt diese Einflussgröße auch eine Gleichverteilung für das strenge 5%-Signifikanzniveau. Bei Anwendung des 10%-Signifikazniveaus liegt kein Verdacht auf Gleichverteilung vor. Eine Diskussion mit Experten und ein Vergleich mit der praxisnahen MT230 Methode der Credit Suisse lässt vermuten, dass dieser Faktor durchaus eine hohe Relevanz haben könnte, auch wenn die Expertenumfrage hinsichtlich der Streuverteilung nicht eindeutig ist (Verdacht auf Gleichverteilung). Wir übernehmen daher diese Einflussgröße (als *M9*) in den M-Faktor und bauen sie in Analogie zur Definition von MT230 auf. Sollte die empirische Datenanalyse nur ein geringes Gewicht ergeben, ist diese Entscheidung erneut zu prüfen.

Verfügbare Zeit: Aufgrund der eindeutigen Umfrageergebnisse (hohe Relevanz, kein Verdacht auf Gleichverteilung) nehmen wir diese Einflussgröße in den M-Faktor auf. Wir wählen abweichend die Bezeichnung *Termindruck (M6),* damit hohe Werte einem größeren M-Faktor entsprechen und entwickeln ansonsten die Metrik aufbauend auf *SCED* von COCOMO II (Tabelle 13, Seite 40).

Erfahrung Fachlichkeit: Die Relevanz ist sehr hoch, allerdings liegt auch bei Anwendung des nicht so strengen 10%-Signifikanzniveaus Verdacht auf Gleichverteilung vor. Die Methode MT230 legt den Schwerpunkt eher auf die Fragestellung, ob es sich um eine Folge-Release handelt als auf die grundsätzliche Erfahrung des Teams mit diesem Typ der fachlichen Anwendung (siehe *APEX* in COCOMO II, Tabelle 13, Seite 40). Eine anschließende Diskussion mit den Experten der Umfrage zeigt, dass der Fragebogen hierzu unterschiedlich verstanden wurde und dies Grund für die Streuverteilung (Verdacht auf Gleichverteilung) sein kann. Daher wird die Einflussgröße unter der neuen Bezeichnung *Reifegrad des Projektes (M10)* in den M-Faktor übernommen, um deutlich zu machen, dass es nicht um die generelle fachliche Erfahrung des Teams geht, sondern um die Frage, welcher der drei folgenden Fälle vorliegt:

- neues Projekt in neuem Umfeld (Release 1.0 bei neuen Kunden bzw. einer unbekannten Organisationseinheit eines bekannten Kunden. Typische Merkmale: neues Team, Kundenmitarbeiter nicht bekannt, Prozesse müssen etabliert/geprobt werden etc.)
- neues Projekt in bekanntem Umfeld (Release 1.0 im organisatorisch, fachlich und technisch eher vertrauten Rahmen, Entscheidungsprozesse beim Kunden erprobt. Im Gegensatz zur Folgestufe/CR treffen zumindest zwei der folgenden Kriterien zu: neues Team, neue Fachlichkeit, neue technische Basis bzw. relevante Erweiterung)

- Folgestufe/CR (Weiterentwicklung durch das weitgehend selbe Team in Zusammenarbeit mit denselben Kundenmitarbeitern. Fachlichkeit und Technik werden beherrscht und stellen kein Risiko dar).

Review und Architektur: Hier liegt hohe Relevanz mit einem Verdacht auf Gleichverteilung nur nach dem strengen 5%-Signifikanzniveau vor. Eine Diskussion mit den Experten der Umfrage lässt uns an der Relevanz der Einflussgröße festhalten, allerdings sind hier zwei unterschiedliche Aspekte im Fragebogen zusammen betrachtet worden, die diese Einflussgröße etwas überladen haben: Die *Qualität der Grobspezifikation* und die *Qualität der T-Architektur* und den damit verbundenen Risiken aus Sicht des Auftragnehmers. *T-Architektur* steht hierbei für *Technische Architektur* [Sie04] und beschreibt, wie gut das Team das technische Umfeld kennt bzw. ob die T-Architektur aus einem vorherigen Release übernommen werden kann, und wie leicht oder schwer Risiken abzuschätzen sind. Wir nehmen beide Einflussgrößen als zwei getrennte aber verwandte Aspekte in den M-Faktor als *M4a* und *M4b* auf. *M4a* baut dabei auf *APEX* von COCOMO II auf.

Leistung/Fähigkeit Chefdesigner: Die Relevanz ist hoch und es liegt kein Verdacht auf Gleichverteilung vor, daher wird diese Einflussgröße in den M-Faktor übernommen. Im Fragebogen wurden sowohl der *technische* wie der *fachliche* Chefdesigner [Sie04] adressiert. Insbesondere in großen Projekten werden diese beiden Aufgaben häufig durch unterschiedliche Personen wahrgenommen. Für den M-Faktor trennen wir daher diese Einflussgröße in *M1a* (*fachlicher Chefdesigner*) und *M1b* (*technischer Chefdesigner*) auf. COCOMO II nennt hierfür die Einflussgröße *ACAP* und bezieht dies auf die *Fähigkeit* (nicht Erfahrung) des fachlichen Chefdesigner. Wir meinen aber, dass die *Fähigkeit* losgelöst von der *Erfahrung* schwer messbar ist, wohingegen die Erfahrung in Form der Anzahl der Projekte in dieser Rolle sehr leicht messbar ist. Weiterhin gehen wir davon aus, dass nur ein sehr fähiger Chefdesigner mehrfach mit dieser Aufgabe betraut wird und nehmen daher als Maßzahl die Erfahrung ausgedrückt in Anzahl Projekte.

Anzahl Entscheidungsträger: Wir nehmen diese Einflussgröße auf, ändern allerdings den Bezeichnung in *Anzahl Ansprechpartner* (*M8*). Häufig sind auch externe Partner, weitere Dienstleister, Subunternehmer oder vom Auftraggeber weitere Ansprechpartner als fachliche oder IT-Ansprechpartner benannt, die formal keine Entscheidungsträger sind, aber im Projekt koordiniert werden müssen. Dieser Bezug erscheint uns passender und wird so auch von MT230 bestätigt. Die Metrik für *M8* ist daher in Analogie zu MT230 aufgebaut.

Leistung/Fähigkeit Programmierer: Diese Einflussgröße zeigt mittlere Relevanz bei Verdacht auf Gleichverteilung. In COCOMO wird dieser Aspekt unter *PCAP* geführt und ist analog zu *ACAP* hinsichtlich der *Fähigkeit* und nicht der *Erfahrung* der Programmierer definiert. Es gilt hier die gleiche Argumentation wie oben zu *Leistung/Fähigkeit Chefdesigner*, weswegen wir diese Einflussgröße nicht in den M-Faktor aufnehmen. Die *Erfahrung* des Teams ist unserer Einschätzung nach bereits ausreichend durch den technischen Chefdesigner stellvertretend in *M1b* abgedeckt.

Erfahrung Werkzeuge: Diese Einflussgröße adressiert die Erfahrung des Teams mit der Programmiersprache (inkl. Bibliotheken) und den Werkzeugen zur Software-Erstellung. Diese Einflussgröße zeigt mittlere Relevanz mit eindeutigem Hinweis auf Gleichverteilung, die erst ab einem Signifikanzniveau von deutlich über 10% nicht mehr bestätigt wird. Daher wollen wir diesen Aspekt *nicht* in den M-Faktor aufnehmen. Allerdings zeigten die

Diskussion mit Experten und die Auswertung von Großprojekten mit dominanter Cobol-Technologie, dass es möglicherweise eine Abhängigkeit vom gewählten Technologie-Stack gibt. Dies betrifft den T-Faktor und werden wir im nächsten Abschnitt diskutieren.

Kontinuität Mitarbeiter: Hier geht es um den Grad der Kontinuität der Teammitarbeiter im Projekt. Diese Einflussgröße wurde mit mittlerer Relevanz bewertet, ein Verdacht auf Gleichverteilung ist nicht anzunehmen, wir nehmen die Einflussgröße unter der Bezeichnung *M3* mit in den M-Faktor auf. Die detaillierte Metrik lehnt sich dabei an die Einflussgröße *PCON* von COCOMO II an.

Verteiltes Arbeiten: Diese Einflussgröße zeigt leicht unterdurchschnittliche Relevanz und eine hoch signifikante Ablehnung der Hypothese auf Gleichverteilung. Zur späteren Auswertung wird diese Einflussgröße zwar unter der Bezeichnung *M11* in zukünftigen UCP-Schätzungen erfasst, bleibt in UCP 3.0 zur Berechnung des M-Faktors unberücksichtigt (Gewicht Null).

Team-Player: Diese Einflussgröße zeigt leicht unterdurchschnittliche Relevanz und signalisiert am eindeutigsten Normalverteilung. Inhaltlich wird bewertet, wie gut die Zusammenarbeit im gesamten Projekt-Team erwartet wird. Wir definieren eine Metrik aus der Experten-Diskussion und führen die Einflussgröße unter der Bezeichnung *Zusammenarbeit* (*M2*) im M-Faktor.

Reife der Prozesse: Die Expertenumfrage ergibt für diese Einflussgröße eine leicht unterdurchschnittliche Relevanz aber signifikante Ablehnung der Hypothese auf Gleichverteilung. Die Einflussgröße wurde bereits im Rahmen der ersten eigenen Analysen der UCP-Methode entdeckt [FJE06, FE08a] und unter der Bezeichnung *Prozess-Modell* eingehend analysiert. Empirische Auswertungen deuten auf ein sehr hohes Gewicht dieser Einflussgröße bzw. auf eine sehr starke Erhöhung der Projektaufwände für hohen *Prozess-Overhead* hin. Wir entscheiden uns für die Bezeichnung *Prozess-Overhead* (*M5*), da primär bewertet wird, wie formal das Vorgehen und der Entwicklungsprozess im Projekt ist. Aufgrund der Voruntersuchungen [FE08a] wählen wir hier eine nach oben asymmetrische Metrik, d.h. die Ausprägung *3 = normal* wird in Richtung schlanker Entwicklungsprozesse mit minimal einem Punkt aber in Richtung komplexer Entwicklungsprozesse auf maximal sechs (statt sonst fünf) Punkte erweitert.

Alle weiteren Einflussgrößen gemäß Tabelle 46 zeigen nur unterdurchschnittliche Relevanz und bleiben für den M-Faktor unberücksichtigt, da wir uns zum Ziel gesetzt hatten, den M-Faktor auf die ca. 10 relevantesten Einflussgrößen zu beschränken. Mit der detaillierten Beschreibung des M-Faktors gemäß der nachfolgenden Tabelle 47 ist zudem eine möglichst exakte Definition und Normierung der Einflussgrößen gegeben, die eine gegenüber der Karner-Methode deutlich verbesserte Reproduzierbarkeit verspricht. Stichprobenartige Tests mit Experten anhand unterschiedlicher Projektkontexte bestätigen diese Vermutung.

Einflussgröße		Metrik (Beispielwerte für die Bewertung)
M1a	Leistung / Fähigkeit fachliche Chefdesigner (FCD)	Wie erfahren ist der fachliche Chefdesigner (FCD) hinsichtlich der im Projekt zu erwartenden Aufgabe? Die Kollegen sind für das Projekt zum Projektstart verfügbar und fest zugesagt. Falls keine Zusage, dann Punktabzug. *(Für Vergleichbarkeit "Umfeld" zu berücksichtigen: Branchen Know-how, spezifisches Know-how zum "Thema des Projektes", etc.)*
	1:	sehr erfahren (mind. 4 Projekte als FCD, davon mind. 2 im konkreten Umfeld)
	2:	erfahren (mind. 2 Projekte als FCD, davon mind. 1 im konkreten Umfeld)
	3:	erfahren (mind. 2 Projekte als FCD, keins im konkreten Umfeld)
	4:	geringe Erfahrung im konkreten Umfeld (mind. 1 Projekt als FCD, keins im konkreten Umfeld)
	5:	geringe Erfahrung (generell als FCD, max. 1 Projekt als FCD und keins im konkreten Umfeld)
M1b	Leistung / Fähigkeit technische Chefdesigner (TCD)	Wie erfahren ist der technische Chefdesigner (TCD) hinsichtlich der im Projekt zu erwartenden Aufgabe? Die Kollegen sind für das Projekt zum Projektstart verfügbar und fest zugesagt. Falls keine Zusage, dann Punktabzug. *(Für Vergleichbarkeit "Umfeld" zu berücksichtigen: Technologie (z.B. NET/Java/Host), Infrastruktur (Native/Web, Client-Server, etc.), Tool Know-how für einzusetzende Produkte, etc.)*
	1:	sehr erfahren im konkreten technischen Umfeld (mind. 4 Projekte als TCD, davon mind. 2 im konkreten technischen Umfeld)
	2:	sehr erfahren (mind. 3 Projekte als TCD, davon mind. 1 im konkreten technischen Umfeld)
	3:	erfahren (mind. 2 Projekte als TCD, davon mind. 1 im konkreten technischen Umfeld)
	4:	erfahren (mind. 2 Projekte als TCD, keins im konkreten technischen Umfeld)
	5:	geringe Erfahrung (generell als TCD, max. 1 Projekt als TCD und keins im konkreten technischen Umfeld)
M2	Zusammenarbeit	Wie gut erwarten wir die Zusammenarbeit im gesamten Projekt-Team (Auftragnehmer, Auftraggeber und Dritte incl. Sublieferanten des Auftragnehmers und Dienstleister des Auftraggebers)? Bewerte das Verständnis für Prozesse, die Leitungsfähigkeit der Zusammenarbeit und die gemeinsamen Ziele.
	1:	Die genannten Punkte werden vermutlich ausgesprochen gut funktionieren (das Team kennt sich, mit dem Auftraggebern (bzw. der Fachabteilung des Auftraggebern) wurde schon mindestens ein vergleichbares Projekt durchgeführt, so dass Prozesse vom Auftraggebern gelebt werden; der Auftraggeber hält sich weitestgehend an seine Prozesse; die Zusammenarbeit ist auf allen Ebenen konstruktiv)
	3:	Die genannten Punkte werden vermutlich gut funktionieren (das Team kennt sich, beim Auftraggebern werden keine Überraschungen erwartet)
	5:	Die genannten Punkte werden vermutlich schlecht funktionieren (z.B. bestehendes kritisches Auftraggeberverhältnis; Wettbewerber im engen Projektumfeld aktiv; die Ziele stehen im gegenseitigem Widerspruch; Zuständigkeiten sind weitgehend unklar; es existieren erhebliche Kulturunterschiede; es wird von einem hohen oder sehr hohen Abstimmungsaufwand ausgegangen)
M3	Kontinuität Mitarbeiter	Wie hoch ist die erwartete Fluktuation der Mitarbeiter im Projekt? Lässt die Planung Reibungsverluste durch den Wechsel von Mitarbeitern in Teilprojekten erwarten? *(Für ein neues Projekt ohne zusätzliche Indikatoren bitte "normal" annehmen)*
	1:	gering (es ist mit Wechsel/Kündigung von bis zu 10% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, Auftraggeber und Thema sind besonders attraktiv, alle Schlüsselrollen sind für die gesamte Projektdauer besetzt)
	3:	normal (es ist mit Wechsel/Kündigung von 25% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, einzelne Schlüsselrollen sind nicht über die Projektdauer besetzt)
	5:	hoch (es ist mit Wechsel/Kündigung von mehr als 50% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, das Projektumfeld führt erfahrungsgemäß zum "sauer fahren" des Teams, wichtige Kompetenzen stehen nur mit harter Zeitbeschränkung zur Verfügung)
M4a	Qualität der Grobspezifikation	Wie nachvollziehbar und detailliert ist die vorhandene fachliche Spezifikation auf der wir aufsetzen und wie gut sind fachliche Risiken bekannt?
	1:	Die Spezifikation ist ausreichend detailliert und lässt keine oder nur sehr wenige Fragen offen. (Tendenz Richtung Feinspezifikation; Grobspezifikation wurde unter Benutzung der Spezifikationsbausteine erstellt und von den relevanten Fachbereichen abgenommen.)
	3:	Die Grobspezifikation enthält eine erkennbare und plausible A-Architektur, offene Fragen sind als solche kenntlich gemacht. Die Grobspezifikation ist mit den relevanten Fachbereichen abgestimmt. Insgesamt sind die Risiken bekannt und überschaubar. Der fachliche Umfang ist klar.
	5:	Die Spezifikation enthält zahlreiche Widersprüche und offene Fragen, für die Klärung sind mehrere Workshops notwendig. Der fachliche Umfang ist noch nicht klar.
M4b	Qualität der T-Architektur	Wie gut ist die Qualität der T-Architektur und wie gut sind Risiken bekannt?
	1:	Die T-Architektur entspricht einer Standardarchitektur oder wurde in einem Vorprojekt vom Auftragnehmer bereits aufgesetzt und dokumentiert. Es sind keine Anpassungen erforderlich; eine Risikoanalyse wurde durchgeführt und ergab keine nennenswerte Risiken.
	3:	Die T-Architektur entspricht weitestgehend einer Standardarchitektur oder wurde in einem vorherigen Release bereits so aufgesetzt. Geringfügige Anpassungen sind erforderlich. Die Risiken sind bekannt.
	4:	Das technische Umfeld ist nicht gut bekannt, die T-Architektur ist zwar vorhanden, dem Team jedoch nicht ausreichend bekannt. Risiken sind schwer abzuschätzen.
	5:	Das technische Umfeld ist nicht gut bekannt, eine T-Architektur muss erst noch erstellt werden.
M5	Prozess-Overhead	Wie formal sind das Vorgehen und der Entwicklungsprozess im Projekt? *(bezieht sich auf Aufbau- und Ablauforganisation)*

	1:	schlanker Entwicklungsprozess, d.h. pragmatisches Vorgehen, wenig Dokumentation, keine formalen Reviews oder Abnahmen, niedrige Querschnittsaufwände UND max. 5 Mitarbeitern bzw. <500T€.
	2:	schlanker Entwicklungsprozess, d.h. pragmatisches Vorgehen, wenig Dokumentation, keine formalen Reviews oder Abnahmen, niedrige Querschnittsaufwände ODER max. 5 Mitarbeitern
	3:	normaler Entwicklungsprozess mit durchschnittlichen Querschnittsaufwänden (typisch für mittelgroßes Projekt (bis 2 Mio. €), aber auch für kleine Projekte mit entsprechend formalen Anforderungen des Auftraggebern)
	4:	komplexer Entwicklungsprozess, hohe Abstimmungs- und Querschnittsaufwände ODER Projekt mit mind. 20 Mitarbeitern
	5:	sehr komplexer Entwicklungsprozess, spezielles Vorgehen, hohe Abstimmungs- und Querschnittsaufwände ODER Projekt mit mind. 30 Mitarbeitern
	6:	sehr komplexer Entwicklungsprozess => Großprojekt (max. Teamstärke > 40, Umfang > 50 BJ oder 8 Mio. €), komplexer Entwicklungsprozess, d.h. sehr formales Vorgehen und Prozesse, hohe Abstimmungs- und Querschnittsaufwände, alle Querschnittsrollen besetzt
M6	Termindruck	Die optimale Projektlaufzeit wird mit 100% angesetzt. Wie viel Zeit gesteht uns der Auftraggeber zu?
	1:	100% der optimalen Projektlaufzeit (oder auch etwas mehr), bzw. der Auftraggeber hat Qualität klar vor Termin priorisiert
	3:	etwa 90% der optimalen Projektlaufzeit, oder es gibt vom Auftraggebern beeinflussbare Randbedingungen ("Politik"), die Termintreue zum primären Projektziel machen.
	5:	weniger als 80% der optimalen Projektlaufzeit, oder weniger als 90% und es gibt nicht vom Auftraggeber beeinflussbare Randbedingungen (z.B. Umsetzung rechtlicher Anforderungen), die eine Terminverschiebung grundsätzlich ausschließen ("failure is not an option").
M7	Stabile Anforderungen	Wie stabil sind die Anforderungen an das System? In welchem Umfang sind Änderungen der Spezifikation zu erwarten? *(Für ein neues Projekt ohne zusätzliche Indikatoren bitte "normal" annehmen)*
	1:	sehr stabile Anforderungen, kaum Änderungen
	3:	normale Änderungsrate, keine grundlegenden Anforderungen geändert
	5:	hohe Änderungsrate, auch grundlegender Anforderungen
	6:	sehr hohe Änderungsrate, auch grundlegender Anforderungen
M8	Anzahl Ansprechpartner	Wie viele IT- und fachliche Ansprechpartner des Auftraggebers sind involviert, welche koordiniert werden müssen? Es sind ggf. auch externe Dienstleister mitzuzählen, welche vom Auftraggeber beauftragt wurden. *(Hinweis: die Extra-Komplexität von Großprojekten wir unter anderem mit diesem Faktor gemessen. Ein Großprojekt kann hier also meist nur mit 4-5 bewertet werden!)*
	1:	wenige fachliche und technische Ansprechpartner (bis 5)
	3:	normal viele fachliche und technische Ansprechpartner (6 bis 15)
	5:	viele fachliche und technische Ansprechpartner (mehr als 15)
M9	Integrationsabhängigkeit	Wie viele Abhängigkeiten von oder zu Schnittstellen bestehen, welche aktiv koordiniert oder neu aufgesetzt werden müssen?
	1:	keine oder wenige (0 bis 4), überwiegend existieren die Schnittstellen bereits
	3:	durchschnittliche viele (5 bis 12)
	5:	sehr viele (mehr als 12), wir sind bei Tests auf diese Schnittstellen und deren Ansprechpartner angewiesen, viele der Schnittstellen sind neu
M10	Reifegrad	Welchen Reifegrad hat das Projekt? *(Hinweis: Hier sind nur die Werte 1, 3 und 5 möglich, keine Zwischenwerte!)*
	1:	Folgestufe/CR (Weiterentwicklung durch das weitgehend selbe Team in Zusammenarbeit mit denselben Auftraggebermitarbeitern. Fachlichkeit und Technik werden beherrscht und stellen kein Risiko dar)
	3:	neues Projekt in bekanntem Umfeld (Release 1.0 im organisatorisch, fachlich und technisch eher vertrauten Rahmen, Entscheidungsprozesse beim Auftraggebern erprobt. Im Gegensatz zur Folgestufe/CR treffen zumindest zwei der folgenden Kriterien zu: neues Team, neue Fachlichkeit, neue technische Basis bzw. relevante Erweiterung)
	5:	neues Projekt in neuem Umfeld (Release 1.0 bei neuen Auftraggebern bzw. einer unbekannten Organisationseinheit eines bekannten Auftraggebers. Typische Merkmale: neues Team, Auftraggebermitarbeiter nicht bekannt, Prozesse müssen etabliert/geprobt werden etc.)
M11	Verteilte Entwicklung	Anzahl der Entwicklungs-Standorte, an denen das Team arbeitet (in der Regel 1): ___
		Nearshore-Anteil in % des gesamten Teamgebirges (z. B. Polen):___
		Farshore-Anteil in % des gesamten Teamgebirges (z. B. Indien): ___

Tabelle 47: Definition der Einflussgrößen des M-Faktors

Mit dieser neuen Definition des M-Faktors sind die Problempunkte P 6 (Definition von TCF und EF sind nicht mehr zeitgemäß) und P 7 (Komplexitätsstufen der TCF und EF sind zu vage definiert) (Abschnitt 2.7) für den M-Faktor umgesetzt.

5.5.2 T-Faktor

Wie bereits in Abschnitt 5.5 begründet, werden wir den T-Faktor aufgrund der geringeren Relevanz mit nur geringfügigen Änderungen aus dem TCF der Karner-Methode übernehmen und

nicht komplett neu aufsetzen. Dabei steht im Vordergrund, die Metrik für die Einflussgrößen des T-Faktors analog zum M-Faktor möglichst exakt zu definieren. Dies erfolgt im Rahmen von Experten-Workshops je Einflussgröße in einem kreativen Akt (5. Schritt des Kostenfaktorbestimmungsverfahrens, Abbildung 40).

T4 Komplexität der Geschäftsregeln und Berechnungen: Diese Einflussgröße (Tabelle 9, Seite 31) werden wir nicht in den T-Faktor übernehmen. Diese Aspekte sind konzeptionell vollständig im A-Faktor abgebildet (Kapitel 3). Die Relevanz dieser Einflussgröße wird zwar von den Experten gemäß Tabelle 46 als sehr hoch und ohne Verdacht auf Gleichverteilung bewertet, konzeptionell ist dies aber kein Element eines Kostenfaktors, sondern ist im A-Faktor abzubilden. Damit bestätigt die Expertenumfrage nachträglich nochmals den Ansatz, den A-Faktor zu überarbeiten. In der Karner-Methode wurden Geschäftsregeln und Berechnungen nicht ausreichend berücksichtigt, da das Konzept des Umganges mit Anwendungsfunktionen nicht existierte und <<include>> Beziehungen von Anwendungsfällen unberücksichtigt blieben. Im A-Faktor ist dies nun enthalten.

T13 Trainingsintensiv: Diese Einflussgröße wird ebenfalls nicht in den T-Faktor übernommen, da hier das bereits zum Thema *Anwenderschulung* zu Beginn von Abschnitt 5.5 Gesagte gilt und daher diese Einflussgröße auch nicht in der Experten-Umfrage berücksichtigt wird.

T14 Ausfallsicherheit: Aufgrund der Expertenumfrage (hohe Relevanz aber auch Verdacht auf Gleichverteilung) nehmen wir diese Einflussgröße neu zur Beobachtung mit auf, sie wurde von Karner nicht berücksichtigt. Wenn eine ausreichend breite Datenbasis verfügbar ist, kann aufgrund der empirischen Auswertung entschieden werden, ob und mit welchem Gewicht diese Einflussgröße zu berücksichtigen ist. Im T-Faktor der Methode UCP 3.0 ist das Gewicht hierfür Null.

T2, T5, T12, T1, T9, T10, T3 und *T7* sind in dieser Reihenfolge durch die Experten hinsichtlich ihrer Relevanz bewertet worden. Dies zeigt, dass der TCF (entgegen dem EF) in der Karner-Methode hinsichtlich der meisten Einflussgrößen doch noch ausreichend aktuell ist. Bei *T2*, *T5* und *T9* ist Verdacht auf Gleichverteilung gegeben und die Relevanz dieser Einflussgrößen sollte in weiteren Experten-Diskussionen überprüft werden.

T11, T8 und *T6* wurden durch die Expertenumfrage als weniger relevant klassifiziert und könnten in späteren Versionen der UCP-Methode weggelassen werden.

T15 SEU / Programmiersprache: Diese Einflussgröße ist im TCF von Karner nicht enthalten, dessen Relevanz wurde aber bereits im vorherigen Abschnitt angesprochen. Es geht hierbei um die Berücksichtigung des Einflusses von Entwicklungsprojekten, die einen signifikanten Anteil „älterer“ Programmiersprachen einsetzten. Bei den vorliegenden Projekten ist dies im wesentlichen Cobol und Host-Technologie. Sie ist durch den Technologie-Stack Auftraggebers vorgegeben und gehört damit zu den Nicht-Funktionalen Anforderungen und ist im T-Faktor abzubilden. Eine empirische Analyse hat gezeigt, dass die Varianz der UCP-Methode deutlich verringert werden kann, wenn dieser Aspekt berücksichtigt wird. Dies wird nachfolgend an Tabelle 48 erläutert.

Tabelle 48 zeigt die Schätzdaten für 19 unterschiedliche Projekte aus unterschiedlichen Branchen. In der 3. Spalte ist der Anteil des Projektes über alle Phasen in Prozent abgeschätzt, der mit Host-Technologie umgesetzt werden wird. Kein Eintrag bedeutet, dass keine Umsetzung auf dem Host geplant ist. 6 Projekte (grau hinterlegt) weisen einen Host-Anteil von 50% - 70% auf (Spalte

Host-Anteil). Die Spalte *IST-Aufwand* nennt den tatsächlich benötigten Projektaufwand nach Projektabschluss, *PE* ist der nach der Karner-Methode berechnete Projekt-Aufwand (Formel 11, Seite 32), die relative Abweichung zum IST-Aufwand ist mit *ΔPE* benannt.

Projekt		Host-Anteil	IST-Aufwand [BT]	UUCP	TCF	EF	PE [BT]	Δ PE	TCF T_{15}	Δ2 (PE_{T15})
1	Finance 4		444	169	1,03	0,97	573	29%	1,00	24%
2	Auto 1		603	227	0,99	0,97	744	23%	1,00	19%
3	Industry 4		606	370	1,05	1,02	1.346	122%	1,00	114%
4	Logistics 2		670	231	0,98	0,99	761	14%	1,00	9%
5	Auto 9	50%	671	147	1,08	0,99	531	-21%	1,15	-12%
6	Auto 3	56%	884	177	1,04	1,03	645	-27%	1,15	-19%
7	Logistics 1		906	268	1,16	1,00	1.056	17%	1,00	12%
8	Industry 2		921	221	1,07	1,05	843	-8%	1,00	-12%
9	Finance 1		978	141	1,08	1,23	637	-35%	1,00	-37%
10	Auto 2		987	327	1,03	1,06	1.221	24%	1,00	19%
11	Auto 6	70%	1.038	187	1,01	1,09	703	-32%	1,23	-20%
12	Auto 5	60%	2.230	420	1,08	1,08	1.656	-26%	1,15	-18%
13	Auto 7	50%	2.712	717	1,03	1,03	2.583	-5%	1,15	6%
14	Industry 3		2.739	954	1,02	1,05	3.470	27%	1,00	22%
15	Auto 4		5.850	790	1,06	1,09	3.113	-47%	1,00	-49%
16	Auto 8	50%	6.385	935	1,03	1,14	3.730	-42%	1,15	-35%
17	Industry 1		6.949	1717	1,07	1,14	7.116	2%	1,00	-1%
18	Finance 2		9.199	1705	1,11	1,18	7.586	-18%	1,00	-21%
19	Logistics 3		10.380	2617	1,12	1,06	10.555	2%	1,00	-2%
Mittelwert:			**2.903**		**1,05**	**1,06**	**2.572**	**0%**	**1,05**	**0%**
Standardabweichung:								**38%**		**35%**

Tabelle 48: Vergleich Karner-Methode mit neuem TCF T_{15} (Spaltennamen siehe Text)

Für alle Projekte wurde nun zusätzlich der Host-Anteil durch eine Metrik T_{15} berücksichtigt, die wie folgt definiert ist: Gemäß Tabelle 49, Zeile *T15* werden je Projekt [1..7] Points für T_{15} definiert. In der Karner-Methode ist $TCF = 1.0$ für den Normwert 3 (Formel 8). Der neue TCF_{T15} wird berechnet zu:

$$TCF_{T15} := TCF * (1 + (T_{15} - 3) * 0{,}075) \qquad \textit{Formel 20}$$

Dies entspricht 7,5% Erhöhung oder Reduktion des *TCF* je Punkt, den T_{15} vom Normalwert *3* abweicht. Dieser Wert ist im Kontext der bekannten Gearingfaktoren (Abschnitt 2.1.3) durchaus plausibel: SLOC ist pro FP für Cobol um einen Faktor 1,3 bis 1,7 höher als für Java (Tabelle 5, Seite 13). Auch wenn SLOC nicht 1:1 in Aufwand übersetzt werden kann ist doch anzunehmen, dass ein erhöhter Aufwand für Cobol anzusetzen ist. Dies ist plausibel zu unserer Metrik: 100% Host-Anteil entspricht $T_{15} = 7$ und damit TCF = 1,3 gemäß Formel 15.

Einflussgröße		Metrik (Beispielwerte für die Bewertung)
T1	Verteiltes System	Wie stark verteilt ist die Architektur des Systems? 1: 2-tier 3: 3-tier 5: Hochverteilte Systemarchitektur
T2	Performanz- und Lastanforderungen	Wie hoch sind Performanz- und Lastanforderungen an das System? 1: keinerlei Anforderungen 3: übliche Performance-Lastanforderungen 5: hohe Performance-Lastanforderungen (z.B. Lastverteilung)
T3	Effizienz der Benutzungsschnittstelle	Wie effizient muss die Benutzungsschnittstelle zu bedienen sein? 1: keine Anforderungen 3: normale Benutzungsschnittstelle, z.B. Web-GUI, einfaches Swing-GUI 5: hoch integrierte, effiziente Benutzungsschnittstelle, z.B. Akkord-Anforderungen, Makros
T4	*Komplexität Geschäftsregeln und Berechnung*	*Wie komplex sind die Geschäftsregeln / die Berechnungen im System?* *1: Nur einfachste Regeln, keine Berechnungen* *3: Normale Komplexität* *5: Sehr komplexe Regeln / Berechnungen*
T5	Wiederverwendbarkeit	Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes? 1: keine Anforderungen, z.B. Software, die nur einmal läuft 3: normale Anforderungen 5: hohe Anforderungen, z.B. Framework
T6	Installationsfreundlichkeit	Wie einfach muss die Software zu installieren sein? 1: Keine Installationsanforderungen 3: Normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen) 5: Hohe Installationsanforderungen (Eingeständige Installation durch eine hohe Zahl von Endkunden)
T7	Benutzerfreundlichkeit	Wie hoch sind die Anforderungen an die Benutzerfreundlichkeit des Systems? 1: keine Anforderungen (keine Benutzer) 3: normale Anforderungen (GUI, Hilfesystem) 5: hohe Anforderungen (z.B. GUI-Varianten für Benutzergruppen, Internationalisierung, Wizards, Fehlertoleranz)
T8	Portabilität	Wie hoch sind die Anforderungen an die Portabilität des Systems? 1: keine Anforderungen z.B. Software, die nur einmal läuft 3: normale Anforderungen (eine Zielplattform) 5: hohe Anforderungen (z.B. Cross-Plattform, Windows + Unix)
T9	Variablilität	Wie variabel (änderungsfreundlich und anpassbar) muss das System sein? 1: keine Anforderungen, z.B. Software, die nur einmal läuft 3: normale Anforderungen, z.B. Konfigurierbarkeit 5: hohe Anforderungen, z.B. Anpassbarkeit über Templates
T10	Verfügbarkeit	Wie hoch sind die Verfügbarkeitsanforderungen an das System? 1: keinerlei Anforderungen 3: übliche Verfügbarkeitsanforderungen 5: Hohe Verfügbarkeitsanforderungen (24/7-Betrieb, 99,x% Verfügbarkeit)
T11	Sicherheitsanforderungen	Wie hoch sind die Sicherheitsanforderungen an das System? 1: keinerlei Anforderungen 3: übliche Sicherheitsanforderungen (Authentifizierung und Authorisierung) 5: Hohe Sicherheitsanforderungen (Zertifikate, verschlüsselte Kommunikation, Kryptographie)
T12	Systemnutzung durch Dritte	Bietet das System direkte Zugänge für Dritte (andere als den Kunden) an? 1: keine Anforderungen 3: Systemnutzung durch Endkunden (B2C-Kunden des Kunden) 5: z.B. externe Serviceschnittstellen, B2B-Systeme, Handelsplattformen
T13	*Trainingsintensiv*	*Ist die Software so komplex, dass besondere Anwenderschulungen erforderlich sind?* *1: keine Schulung erforderlich, intuitiv nutzbar* *3: übliche Anwenderschulung oder Selbststudium* *5: Training erforderlich*
T14	Ausfallsicherheit	Wie hoch ist die Anforderung an Ausfallsicherheit der Software? 1: kein oder nur geringer Schaden, ausfallbedingte Verluste können leicht wiederhergestellt werden 3: normaler Schaden, ausfallbedingte Verluste können wieder hergestellt werden 5: hoher finanzieller Schaden, ausfallbedingte Verluste können nicht wieder hergestellt werden
T15	SEU, Programmiersprache (Gearingfaktor)	Wie hoch ist der Anteil an "älteren" Programmiersprachen (in der Regel Cobol): 1: - entfällt hier - 2: - entfällt hier - 3: kein Anteil, die gesamte Entwicklung erfolgt in modernen Programmiersprachen in entsprechenden Entwicklungsumgebungen 4: geringer Anteil an Host-Technologie, die Entwicklung erfolgt zu weniger als 25 % auf dem Host 5: gemischter Anteil Host und moderner Programmiersprachen, die Entwicklung erfolgt zu etwa 50 % auf dem Host 6: die Entwicklung erfolgt zu mehr als 75 % auf dem Host , geringer Anteil moderner Programmiersprachen, 7: die Entwicklung erfolgt zu 100 % auf dem Host, kein Anteil moderner Programmiersprachen,

Tabelle 49: Definition der Einflussgrößen des T-Faktors (kursiv = entfällt für UCP 3.0)

Mit dieser Einflussgröße T_{15} wurden für die 19 Projekte jeweils der *TCF* korrigiert und der neue UCP-Aufwand (PE_{T15}) analog zur Formel 10 und Formel 11 (Seite 32) ermittelt. Dafür wurde der

Produktivitätsfaktor *PF* von 27,3 auf 26,3 Bh/Point abgesenkt, damit der Mittelwert über alle Projekte wieder Null ergibt und damit vergleichbar ist[31]. Die relative Abweichung des neuen PE_{T15}-Aufwandes zum *IST-Aufwand* ist in Tabelle 48 in der Spalte *Δ2* aufgeführt. Der Vergleich der Standardabweichungen der Schätzreihen mit altem original *TCF* und um Einflussgröße T_{15} korrigiertem *TCF* zeigt eine deutliche Verbesserung von 38% auf 35%. Dies rechtfertigt die Übernahme der Einflussgröße T_{15} in den neuen T-Faktor.

Tabelle 49 stellt alle Einflussgrößen für den neuen T-Faktor zusammen, die kursiv dargestellten Zeilen sind nur zur Information angegeben, wurden aus dem TCF übernommen und entfallen im T-Faktor der Methode UCP 3.0. Die Gewichte (bis auf T_{15}) für die Einflussgrößen des T-Faktors werden von dem *TCF* der Karner-Methode aus Tabelle 9 (Seite 31) übernommen.

Mit dieser präzisierten und überarbeiteten Definition des T-Faktors sind die Problempunkt P 6 (Definition von TCF und EF sind nicht mehr zeitgemäß) und P 7 (Komplexitätsstufen der TCF und EF sind zu vage definiert) (Abschnitt 2.7) für den T-Faktor umgesetzt.

5.6 Bestimmung der Kostenfaktor-Formel

In diesem Abschnitt werden nun die Formeln für die beiden Kostenfaktoren überarbeitet. Darauf aufbauend schließt sich dann die Bestimmung der Gewichte für die Einflussgrößen des M-Faktors in Abschnitt 5.7 gemäß des 6. und 7. Schrittes des Kostenfaktorbestimmungsverfahrens (Abbildung 40, Seite 115) an.

Die Entwicklung eines neuen Kostenfaktormodells ist notwendig, um den Problempunkt P 5 (UCP Schätzmodell verwendet unzulässige Skalen-Transformationen) gemäß Abschnitt 2.7 auszuräumen.

Neben dieser formalen Kritik sprechen aber auch praktische Gründe für ein anderes Skalenmodell als die Summation über die Einflussgrößen gemäß Formel 21 (hier am Beispiel des TCF). Eine Produktdarstellung (Multiplikation statt Summation) würde jede Einflussgröße als Korrektur*faktor* ausweisen und die Gewichte könnten dann als Prozentwert interpretiert werden, um den der A-Faktor dann durch die Ausprägungen dieser Einflussgröße auf- bzw. abgewertet wird. Diesen Vorteil haben wir im vorherigen Abschnitt bei der Erweiterung des TCF um die Einflussgröße T_{15} schon benutzt.

Zunächst wird aufgrund von Experteneinschätzungen festgelegt, wie die verschiedenen 6 Ausprägungen ([0..5] Punkte) je Einflussgröße untereinander gewichtet werden sollen. Hier haben wir uns für das möglichst einfachste Modell mit einer linearen äquiproportionalen Skala entschieden, d.h. 2 Punkte gehen doppelt so stark wie 1 Punkt ein und 4 Punkte doppelt so stark wie 2 Punkte. Dadurch reduziert sich die Zahl der Freiheitsgrade von 6 auf nur noch einen Freiheits-

[31] Bei Kalkulation mit dem unveränderten Produktivitätsfaktor PF würde sich ein Mittelwert von 4% und eine Standardabweichung von 36% ergeben, das Ergebnis hängt also nicht von PF ab, wie man sich auch leicht aus der Theorie überlegen kann. Die Bedingung, dass der Mittelwert über eine Schätzreihe Null sein soll, wird im folgenden Abschnitt noch weiter formalisiert werden, ist für die Überlegungen hier aber nicht bedeutend.

grad pro Einflussgröße, ausgedrückt durch sein Gewicht W_i und wir bezeichnen dies als **Nebenbedingung N1**.

Für den T-Faktor setzen wir auf den TCF auf und können aufbauend auf Formel 8 aus Abschnitt 2.3 definieren, wobei die Gewichte W_i später noch zu definieren sind:

$$T\text{-}Faktor := c_1 + c_2 \cdot \sum_{i=1}^{13} (T_i \cdot W_i)$$ mit den Konstanten: c_1 := 0,58 und c_2:= 0,01	*Formel 21*

Analog setzen wir auf dem EF auf und können aufbauend auf Formel 9 definieren:

$$M\text{-}Faktor := c_1 + c_2 \cdot \sum_{i=1}^{9} (M_i \cdot W_i)$$ mit den Konstanten: c_1 := 1,405 und c_2:= -0,03	*Formel 22*

Die Gewichte W_i der Karner-Methode sind im Abschnitt 2.4.1 in Tabelle 9 und Tabelle 10 aufgeführt. Für jede Einflussgröße werden [0..5] Punkte vergeben. Hierfür steht in der Formel 21 die Variable T_i und in Formel 22 die Variable M_i.

Es fällt auf, dass im EF einige Gewichte negativ sind, was nur an der Formulierung der Einflussgrößen liegt. Eine negierte Formulierung würde diese Unsymmetrie vermeiden, z.B. E_2: statt: „viele Teilzeitmitarbeiter“ wäre „wenig Personalkontinuität“ gleichbedeutend, oder E_7: statt „stabile Anforderungen“ wäre „keine Anforderungsänderung“ gleichbedeutend und ein positives Gewicht, d.h. eine Umkehr des Skalenverlaufes möglich. Diese Unsymmetrie wollen wir im neuen Kostenfaktor-Modell vermeiden und für die noch festzulegenden W_i nur noch Gewichte $W_i \geq 0$ zulassen (**Nebenbedingung N2**) und stattdessen durch die oben dargestellte Wahl der Formulierungen einen gleichen Verlauf der Ordinalskala erreichen.

Es lassen sich weitere Nebenbedingungen formulieren, welche die oben genannten Freiheitsgrade weiter einschränken. Die Ordinalskala für die Werte T_i und M_i sind so entworfen, dass drei Punkte einen normalen Wert bedeuten: Werden also für den Kostenfaktor alle Einflussgrößen mit drei Punkten bewertet, so ist der T-Faktor gleich Eins, analoges gilt für den M-Faktor (**Nebenbedingung N3**):

$$\text{T-Faktor} = 1 \quad \Leftrightarrow \quad \forall i \in [1..13] : T_i = 3$$ $$\text{M-Faktor} = 1 \quad \Leftrightarrow \quad \forall i \in [1..9] : M_i = 3$$	*Formel 23*

Über diese Nebenbedingung lässt sich die Konstante c_1 aus dem T-Faktor bzw. M-Faktor bestimmen zu:

T-Faktor: $c_1 = 1 - 3 \cdot c_2 \cdot \sum_{i=1}^{13} W_i \quad = 0{,}58$	*Formel 24*
M-Faktor: $c_1 = 1 + 3 \cdot c_2 \cdot \sum_{i=1}^{9} W_i \quad = 1{,}405$	*Formel 25*

Bei der numerischen Ermittlung der UCP-Formel aus einer Projektdatenbank fordern wir ferner, dass die Summe der Abweichungen über alle Projekte Null sein soll (**Nebenbedingung N4**). Dies wird mathematisch durch Multiplikation mit einer geeigneten Konstante erreicht und entspricht genau der Funktion des Produktivitätsfaktors *PF*, die einheitenlose Größe *Points* in die physikalische Einheit *Zeit* bzw. Bearbeiterstunden [Bh] zu transformieren.

Der bisher definierte T-Faktor kann durch Taylor-Entwicklung [BS87] leicht in eine Produktdarstellung überführt werden. Es gilt folgender Zusammenhang:

$\prod_{i=1}^{13}(1 + c \cdot W_i \cdot T_i) = 1 + c \cdot \sum_{i=1}^{13}(W_i \cdot T_i) + c^2 \cdot \sum_{i,j=1;i\neq j}^{13}(W_i \cdot T_i \cdot W_j \cdot T_j) + \ldots$ $\approx 1 + c \cdot \sum_{i=1}^{13}(W_i \cdot T_i)$ mit Konstante $c << 1$	*Formel 26*

Die quadratischen *c*-Terme und Terme höherer Ordnung sind mit *c*<<1 vernachlässigbar klein und werden weggelassen (Taylor-Entwicklung).

Mit Hilfe von Formel 26 lässt sich der T-Faktor gemäß Formel 24 in eine Produktdarstellung überführen:

T-Faktor := $c_1 + c_2 \cdot \sum_{i=1}^{13}(W_i \cdot T_i) = 1 + c_2 \cdot \sum_{i=1}^{13}(W_i \cdot T_i) - 3 \cdot c_2 \cdot \sum_{i=1}^{13}(W_i \cdot 3) = 1 + c_2 \cdot \sum_{i=1}^{13}(W_i \cdot (T_i - 3))$ $\approx \prod_{i=1}^{13}(1 + c_2 \cdot (T_i - 3) \cdot W_i)$ mit Konstanten: $c_2 := 0{,}01 << 1$ $c_1 := 1 - 3 \cdot c_2 \cdot \sum_{i=1}^{13} W_i \quad = 0{,}58$ $W_i \geq 0$	*Formel 27*

Und analog für den M-Faktor:

$\text{M-Faktor} := c_1 + c_2 \cdot \sum_{i=1}^{9}(W_i \cdot M_i) = 1 + c_2 \cdot \sum_{i=1}^{9}(W_i \cdot M_i) + c_2 \cdot \sum_{i=1}^{9}(W_i \cdot 3) = 1 + c_2 \cdot \sum_{i=1}^{9}\left(W_i \cdot (M_i - 3)\right)$ $\approx \prod_{i=1}^{9}\left(1 + c_2 \cdot (M_i - 3) \cdot W_i\right)$ mit Konstanten: $c_2 := -0{,}1 << 1$ $c_1 := 1 + 3 \cdot c_2 \cdot \sum_{i=1}^{9} W_i \quad = 1{,}405$ $W_i \geq 0$	*Formel 28*

Dies hat folgende Vorteile:

- Jeder einzelne M-Faktor (analog für T-Faktor) geht hinsichtlich seines Gewichtes relativ gesehen immer gleich stark in den M-Faktor ein, eine gegenseitige Abhängigkeit des relativen Beitrages, wie bei der Summenformel, wird vermieden. Dies wäre insbesondere dann problematisch, wenn die einzelnen Einflussgrößen einen Beitrag in der Größenordnung von Eins beisteuern.
- Der Problempunkt P 6 mit der Forderung nach algebraischer Konformität wird erfüllt.
- Die Formel ist intuitiver durch den Schätzer nachvollziehbarer, da jede Einflussgröße des Kostenfaktors nun einen multiplikativen Beitrag leistet. Damit sind ggf. zukünftig auch Einflussgrößen >> 1 besser darstellbar.
- Jedes einzelne Gewicht muss positiv sein. Durch diese einfache Nebenbedingung wird nun leicht vermieden, dass bei der numerischen Ermittlung der Gewichte durch Regressionsanalyse ein negativer Wert resultiert, was bei der Summenformel des M-Faktors nicht einfach formulierbar gewesen wäre.
- Der Term *(T_i-3)* bzw. *(M_i-3)* sorgt für die Reskalierung des neutralen Wertes Drei auf das Gewicht Null und ist in der Formel nun leicht sichtbar und nicht mehr in den Gewichten und Konstanten „versteckt".

Ferner ist bei Karner durch den Wertebereich beginnend bei Null die Punkte-Skala nicht symmetrisch gewählt. Wenn 3 der Normal-Fall ist, also die Mitte der Skala, dann gibt es drei Punktewerte unterhalb der Skala = {0,1, 2}, aber nur zwei Punktewerte oberhalb = {4,5}. Dafür gibt es keinen plausiblen Grund. Wir werden daher zukünftig in der Regel eine Punkte-Skala aus [1..5] vorsehen mit 3 für den Normalfall und Werte größer 3 für die Ausprägungen, die den Aufwand vergrößern (Faktor > 1) und Werte kleiner 3, also {1,2}, die den Faktor verkleinern. Damit ist die Skala grundsätzlich „nach oben geöffnet", d.h. werden besonders starke Einflussgrößen identifiziert, welche den Faktor >> 1 zur Folge haben sollten, ist dies durch entsprechend höhere Punkte abbildbar. Die Skala soll äquiproportional gewählt sein, d.h. bei gleichen W_i ist die Einflussgröße

bei einem Punkt halb so groß wie bei drei Punkten, bei drei Punkten ist die Einflussgröße halb so groß wie bei fünf Punkten usw., wie bereits oben unter Nebenbedingung N1 gefordert. Damit wird zukünftig die Konstante c_1 auch für den M-Faktor ein positives Vorzeichen erhalten, d.h. $c_1 > 0$ (**Nebenbedingung N5**). Die Formulierungen der Gewichte werden entsprechend zu definieren sein.

Da wir die Konstante c_2 in Formel 27 und Formel 28 eliminieren konnten, wird die Formel nur noch von einer Konstanten c geprägt. Wir benennen c_1 nun in c_T und c_M für den T- bzw. M-Faktor um. Die Konstante spiegelt jeweils das Gesamtgewicht des jeweiligen Faktors in Bezug auf den Gesamtaufwand wider. Die Konstante kann durch Expertenabschätzung in Abhängigkeit der jeweiligen Anzahl der Einzelfaktoren je Kostenfaktor gewählt werden und wird für das neue Kostenfaktormodell nicht numerisch ermittelt, sondern beim Entwurf der neuen Formel sinnvoll vorbelegt (**Nebenbedingung N6**).

COCOMO II hat exponentielle Kostenfaktoren (Formel 13) eingeführt. Es stellt sich also grundsätzlich die Frage, ob das hier bisher vorgestellte lineare UCP-Modell nicht auch exponentielle Anteile enthalten sollte. Dazu ist zunächst festzuhalten, dass bereits bei der Definition, wann wie viele Punkte je Einzelwert je Einflussgröße vergeben werden, eine nicht lineare Skala erreicht werden kann, hierzu aber das Modell von Karner keine Aussagen macht. Eine exponentielle Berücksichtigung bei der Bewertung der Use Cases und Actors ziehen wir für die Methode UCP 3.0 nicht in Erwägung, da es hierfür keine fachlichen Anhaltspunkte gibt.

In den Vorgaben des EF und TCF findet sich keine detaillierte und normierbare Vorgabe, wann wie viele Punkte vorzusehen sind. Ein wesentlicher Verbesserungsschritt ist also in Abschnitt 5.5 bereits bei der Definition der Einflussgrößen erreicht worden, in dem durch konkrete Beispiele und „Best Practices“ eine einheitliche und standardisierte Punkte-Vorgabe je Kostenfaktor gegeben wurde, so dass unabhängige Schätzer für das gleiche oder vergleichbare Projekte auch gleiche Werte vergeben.

Ein Beispiel aus dem EF mag dies verdeutlichen: E_2 – „Part time Workers“. Ist ein hoher Wert anzusetzen, wenn 25% des Teams nur in Teilzeit arbeiten oder sind erst 50% Teilzeit Grund für einen hohen Wert? UCP nach Karner macht hier keine Vorgaben, auch wird nicht definiert, ab welchem Teilzeit-Grad negative Auswirkungen auf den Projektverlauf zu erwarten sind: Wenn jemand zu 80% (z.B. 4 Tage pro Woche) im Projekt arbeitet, ist dies nach Erfahrung des Autors keine wesentliche Einschränkung, eine 40% Verfügbarkeit aber sehr wohl. COCOMO II z.B. hat für den *PCON* Cost Driver (= Personell Continuity) feste Prozentwerte vorgegeben (Table 2.28 in [Boe+00]). Daher werden wir in dem zu entwickelten Kostenfaktor-Modell aufgrund von Experteneinschätzungen eine Punkte-Matrix je Kostenfaktor vorgeben, die genauen Definitionen sind nicht numerisch aus Datenbanken abzuleiten sondern werden durch Expertenwissen vorbelegt. Dadurch können nichtlineare Effekte berücksichtigt werden, wie dies für die Einflussgrößen M5 und M7 in Tabelle 47 und für T15 in Tabelle 49 bereits vorgesehen wurde.

5.7 Neues Komplexitätsmodell

In diesem Abschnitt werden nun die Gewichte für die Einflussgrößen des in den vorherigen Abschnitten entwickelten M-Faktors bestimmt, dies entspricht dem 6. und 7. Schritte des Kostenfak-

torbestimmungsverfahrens (Abbildung 40, Seite 115). Dazu wurden insgesamt 19 geeignet große Projekte ausgewertet und insbesondere die neuen Einflussgrößen für den T-Faktor und den M-Faktor erhoben. Darauf aufbauend werden dann durch numerische Analyse die Gewichte bestimmt.

Zeigen die Projektdaten eine gute Korrelation mit den Einflussgrößen, dann lassen sich die Gewichte sicher bestimmen. Dies ist dann eine nachträgliche empirische Validierung der ausgewählten Einflussgrößen. In [FE08a] wird z.B. eine solche Korrelationsanalyse für die Einflussgröße *Prozessmodell* (entspricht *Prozess-Overhead* (M5) in UCP 3.0) beschrieben. Wir verzichten hier auf eine detaillierte Beschreibung für jede Einflussgröße und halten fest, dass eine ausreichende Korrelation für alle ausgewählten Einflussgrößen festgestellt werden konnte.

Um die Zahl der Freiheitsgrade bei der Bestimmung des Kostenfaktor-Modells klein zu halten, wurden im vorherigen Abschnitt bereits Nebenbedingungen formuliert, die nachfolgend in Tabelle 50 zusammengefasst sind:

N1:	Lineare äquiproportionale Skala je Einflussgröße
N2:	Die Gewichte je Einflussgröße sind positiv, d.h. $W_i \geq 0$
N3:	Kostenfaktor = 1,0 genau dann, wenn alle Einflussgrößen drei Punkte erhalten
N4:	Die Summe aller Abweichungen über alle Projekte soll Null sein
N5:	Konstante $c_1 > 0$
N6:	c_T und c_M werden per Definition sinnvoll vorbelegt

Tabelle 50: Nebenbedingungen zur numerischen Ermittlung der Gewichte je Kostenfaktor

Aufgrund von N1 müssen wir nur *ein* Gewicht je Einflussgröße bestimmen. Die Gewichtung der verschiedenen Ausprägungen je Einflussgröße, wie sie in Tabelle 47 und Tabelle 49 bereits definiert wurden, kann 1:1 in einen Punktewert übernommen werden. Formel 27 und Formel 28 erfüllen Nebenbedingung N3.

Wir definieren in der Methode UCP 3.0 für beide Kostenfaktoren einheitlich die Konstante $c_T = c_M := 0{,}01 > 0$. Damit werden Nebenbedingungen N5 und N6 erfüllt: Numerische Simulationen zeigen, dass dafür Gewichte überwiegend im einstelligen Bereich ermittelt werden können und somit praktikabel sind. Die Konstante erlaubt zudem die Interpretation der Höhe der Gewichte in Prozent. Die Gewichte erhalten damit eine anschauliche Bedeutung, da eine Abweichung vom Normwert drei z.B. um einen Punkt in einer prozentualen Abweichung vom Kostenfaktor in der Höhe des Gewichtes resultiert.

Ein Beispiel verdeutlich das: Betrachten wir die bereits ausführlich in Abschnitt 5.5.2 diskutierte Einflussgröße T_{15} mit einem Gewicht von 7,5. Alle Einflussgrößen seien mit drei bewertet (d.h. T-Faktor = 1,0) und T_{15} wird nun von drei auf vier erhöht. Dann bestimmt sich der T-Faktor gemäß Formel 27 zu 1,075. Die Erhöhung um einen Punkt bewirkt einen um 7,5% erhöhten T-Faktor und damit ein um die gleiche Größe erhöhter geschätzter Projektaufwand *PE*.

Für den M-Faktor wäre gemäß Karner, wie die Nebenbedingungen zu Formel 28 zeigen, die Konstante c = -0,1. Durch die Festlegung von c_M auf ebenfalls +0,01 wird sich die „Richtung" der Skala je Einflussgröße umdrehen und die Gewichte etwa verzehnfachen. Die umgedrehte Skalenfolge wurde bereits bei der Definition der Ausprägungen je Einflussgröße (Tabelle 47) berücksichtigt.

Die Gewichte sind nun so zu wählen, dass die Abweichung σ (Standardabweichung) zwischen dem tatsächlichen Projektaufwand nach Projektabschluss (IST-Aufwand) und dem mit UCP 3.0 geschätzten Projektaufwand *PE* minimal wird. Diese Minimierungsaufgabe ist unter den noch offenen Nebenbedingungen N2 und N4 numerisch durchzuführen. Wir verwenden dafür die Funktion *Zielwertsuche* mit Nebenbedingungen von Excel [Bad08].

Tabelle 51 zeigt einen Auszug aus der aufgebauten Datenbank der 19 Projekte, die zur numerischen Bestimmung der Gewichte des M-Faktors herangezogen werden. Die Spalte *IST-Auf.* nennt den tatsächlich benötigten Projektaufwand in Bearbeitertagen (BT) nach Projektabschluss, *PE* ist der nach der UCP-Methode berechnete Projekt-Aufwand. Die relative Abweichung zum IST-Aufwand ist mit Δ_{PE} benannt. Die Einflussgrößen des M-Faktors wurden teilweise schon für Vorversionen der UCP 3.0 Methode erhoben und die Werte je Einflussgröße mussten aufgrund der geänderten Skala transformiert werden. Daher kommen teilweise gebrochene Werte zustande.

Projekt		IST-Auf.	Faktoren			Einflussgrößen des M-Faktors													PE	Δ_{PE}
		[BT]	A-	T-	M-	M1a	M1b	M2	M3	M4a	M4b	M5	M6	M7	M8	M9	M10	M11	[BT]	[%]
1	Finance 4	444	31	1,00	0,95	1	1	2	1	2	2	4	3	3	3	3	3	4,3	507	14%
2	Auto 1	603	38	0,99	1,07	3	3	3	3	3	3	3	3	3	2	4,3	5	3	580	-4%
3	Industry 4	606	64	1,03	0,68	3,7	3,7	2	3	3,7	3,7	2	1	3	1	1	3	4,3	728	20%
4	Logistics 2	670	38	0,98	1,00	2	2	3,7	3,7	3,7	3,7	3	1	3,5	2,5	2	5	3	629	-6%
5	Auto 9	671	23	1,17	1,54	3,7	3,7	4,3	3,7	3	3	4	3	3	5	4,3	5	3	749	12%
6	Auto 3	884	24	1,17	1,25	4,3	4,3	4,3	4,3	4,3	4,3	3	3	3	3	3	3	3	750	-15%
7	Logistics 1	906	46	1,10	1,26	3	3	3	1	3	3	5	1	3	2	1	3	3	887	-2%
8	Industry 2	921	43	1,09	1,01	3	3	3	1	3	3	3	3	3	3	5	1	3	694	-25%
9	Finance 1	978	25	1,03	2,05	3	3	3	1	5	5	5	5	6	1	3	1	3	1.013	4%
10	Auto 2	987	57	1,03	1,30	4,3	4,3	4,33	2	3	3	3,5	4,3	3,5	2,5	2	3	3	1.066	8%
11	Auto 6	1.038	32	1,24	1,48	1	1	2	3	5	5	5	1	5	3	1	3	3	935	-10%
12	Auto 5	2.230	76	1,19	1,59	4,3	4,3	3	3	2	2	5	1	3	3,7	3,7	1	3,7	1.843	-17%
13	Auto 7	2.712	102	1,18	1,80	3	3	3	3	3	3	5	3	4	4,3	3,7	5	3	3.129	15%
14	Industry 3	2.739	176	1,02	1,16	3	3	4,3	4,3	5	5	3	5	3	3	3	1	3	2.860	4%
15	Auto 4	5.850	161	1,01	2,08	2	2	4,3	2	3,7	3,7	6	5	4	3	5	1	3	5.233	-11%
16	Auto 8	6.385	147	1,18	1,69	4,3	4,3	3	3	3	3	5	3	4	4,3	1	1	3	4.128	-35%
17	Industry 1	6.949	243	1,03	2,06	3,7	3,7	2	1	4,3	4,3	5	3	5	4,3	5	3	3	7.360	6%
18	Finance 2	9.199	339	1,07	2,41	3,7	3,7	4,3	3	3	3	6	2	5	5	3	1	3	10.603	15%
19	Logistics 3	10.380	504	1,10	1,83	3,7	3,7	3	3	3	3	6	4,3	3	3,7	3	1	4,3	12.430	20%
Mittelwert:		**2.903**	114	1,09	1,48	3,1	3,1	3,2	2,6	3,5	3,5	4,3	2,9	3,7	3,1	3,0	2,6	3,2	**2.954**	**0%**
σ:																				**16%**

Tabelle 51: Projektdatenbank (Auszug) für UCP 3.0

Der Produktivitätsfaktor wurde zu 18,9 Bh/Point bestimmt, womit der Mittelwert des IST-Aufwandes und der Mittelwert von PE annähernd gleich sind (Mittelwert $\Delta_{PE} \approx 0\%$).

Die Minimierung der Standardabweichung der Mittelwerte (ca. 16%) führt zu den zwölf Gewichten gemäß Tabelle 52. *M11* wurde nicht berechnet, sondern das Gewicht zuvor aufgrund der Diskussion in Abschnitt 5.5.1 auf Null gesetzt, womit dieser Einflussgröße nicht in den M-Faktor eingeht. Die Werte für die Spalten *Minimum* und *Maximum* repräsentieren jeweils den kleinsten bzw. größten Faktor, den diese Einflussgröße unter Berücksichtigung der Punktewerte aus Tabelle 47 hervorrufen kann.

Einflussgröße		Gewicht W_i	i	Minimum	Maximum
M1a	Leistung/Fähigkeit fachliche Chefdesigner (FCD)	3,6	1	0,93	1,07
M1b	Leistung/Fähigkeit technische Chefdesigner (TCD)	3,6	2	0,93	1,07
M2	Zusammenarbeit	4,0	3	0,92	1,08
M3	Kontinuität Mitarbeiter	2,7	4	0,95	1,05
M4a	Qualität der Grobspezifikation	2,0	5	0,96	1,04
M4b	Qualität der T-Architektur	1,0	6	0,98	1,02
M5	Prozess-Overhead	25,0	7	0,50	1,75
M6	Termindruck	1,0	8	0,98	1,02
M7	Stabile Anforderungen	13,0	9	0,74	1,39
M8	Anzahl Ansprechpartner	1,0	10	0,98	1,02
M9	Integrationsabhängigkeit	4,3	11	0,91	1,09
M10	Reifegrad	1,0	12	0,98	1,02
M11	*Verteilte Entwicklung*	*0*		*1,00*	*1,00*

Tabelle 52: Gewichte W_i des M-Faktors

Einflussgröße		Gewicht W_i	i	Minimum	Maximum
T1	Verteiltes System	2,0	1	0,96	1,04
T2	Performanz- und Lastanforderungen	1,0	2	0,98	1,02
T3	Effizienz der Benutzungsschnittstelle	1,0	3	0,98	1,02
T4	*Komplexität der Geschäftsregeln und Berechnung*	*0,0*		*1,00*	*1,00*
T5	Wiederverwendbarkeit	1,0	4	0,98	1,02
T6	Installationsfreundlichkeit	0,5	5	0,99	1,01
T7	Benutzerfreundlichkeit	0,5	6	0,99	1,01
T8	Portabilität	2,0	7	0,96	1,04
T9	Variablilität	1,0	8	0,98	1,02
T10	Verfügbarkeit	1,0	9	0,98	1,02
T11	Sicherheitsanforderungen	1,0	10	0,98	1,02
T12	Systemnutzung durch Dritte	1,0	11	0,98	1,02
T13	*Trainingsintensiv*	*0,0*		*1,00*	*1,00*
T14	*Ausfallsicherheit*	*0,0*		*1,00*	*1,00*
T15	SEU, Programmiersprache	7,5	12	1,00	1,38

Tabelle 53: Gewichte W_i des T-Faktors

Der Vollständigkeit halber und zum Vergleich geben wir analog noch die zwölf Gewichte für den T-Faktor an, die weitgehend aus dem TCF übernommen wurden und für das neue Kostenfaktormodell (Formel 27) angepasst wurden. T4, T13 und T14 werden, wie in Abschnitt 5.5.2 diskutiert, nicht berücksichtigt und haben daher ein Gewicht von Null.

Damit wurden die Kostenfaktoren vollständig entlang des Vorgehens des Kostenfaktorbestimmungsverfahrens (Abbildung 40) entwickelt. Im nächsten Kapitel wird die gesamte Methode UCP 3.0 zusammengefasst und diskutiert werden. Wir stellen dabei auch die Verbesserung der Methode UCP 3.0 gegenüber der Karner-Methode dar.

6 Synthese

Der Aufgabenstellung aus Abschnitt 2.7 folgend wurde in den vorangehenden Kapiteln schrittweise eine Schätzmethoden für betriebliche Informationssysteme entwickelt, welche insbesondere zum Zeitpunkt der Angebotserstellung gut anwendbar sein soll. Ausgehend von der Terminologie aus Abschnitt 2.8 wurden dazu im Verlauf der Dissertation mehrere Versionen einer überarbeiteten UCP-Methode entwickelt, die hier nicht im Detail beschrieben werden.

Die finale Schätzmethode UCP 3.0 wird nun in diesem Kapitel zusammengefasst. Dazu wird die Schätzmethode in Abschnitt 6.1 nochmals zusammenfassend beschrieben und der Anwendungsbereich der Methode (Umfang) abgeleitet. Zur effizienten Durchführung einer Schätzung mittels UCP 3.0 wurde ein Schätzwerkzeug entwickelt, welches in Abschnitt 6.2 vorgestellt wird. Die Anwendung in der Praxis wird durch ein Anwendungsbeispiel in Abschnitt 6.3 präsentiert. Es schließt sich in Abschnitt 6.4 die Validierung der Methode durch einen empirischen Vergleich des Schätzergebnisses zwischen der Karner-Methode und UCP 3.0 anhand von den tatsächlich benötigten Entwicklungsaufwänden abgeschlossener Software-Projekte an. Dafür wurde das Schätzwerkzeug Abschnitt 6.2 wie in Abschnitt 6.3 dargestellt verwendet.

Eine Bewertung und Diskussion der Methode UCP 3.0 erfolgt später in Kapitel 7. Dabei werden auch die zu Beginn dieser Arbeit in Abschnitt 2.7 genannten Problempunkte der UCP-Methode aufgegriffen.

6.1 Darstellung der Schätzmethode UCP 3.0

In diesem Kapitel werden nun die zuvor entwickelten Methoden für die Funktionalen und Nichtfunktionalen Anforderungen zur Methode UCP 3.0 zusammengefügt:

- Für die Funktionalen Anforderungen wird zur Identifikation von Use Cases das in Kapitel 3 dargestellte modellbasiertes Vorgehen verwendet, welches unterschiedliche Spezifikationsformen auf eine neu entwickelte UCP-Sprache transformiert. Abbildung 14 (Seite 27) stellte dies bereits schematisch dar. In einer zweiten Transformation erfolgt die Abbildung der UCP-Sprache auf den A-Faktor. Der Leitfaden zur Use Case Identifikation und die Mapping Regeln aus Kapitel 3 liefern hierfür die Grundlagen.
- Für die Nichtfunktionalen Anforderungen wird das in Kapitel 5 entwickelte Kostenfaktor-Modell des T-Faktors und M-Faktors verwendet (Abschnitt 5.7)

Daraus wird für die Methode UCP 3.0 der Schätzaufwand *PE* in Bearbeiterstunden (Bh) mit Hilfe von Formel 29 ermittelt. Werden nachfolgend Aufwände in Bearbeitertagen (BT) dargestellt, erfolgt die Umrechnung gemäß der Zuordnung 1 BT = 8 Bh.

$PE := 5 \cdot A\text{-}Faktor \cdot T\text{-}Faktor \cdot M\text{-}Faktor \cdot PF$	*Formel 29*
mit:	

A-Faktor $:= \left(\sum_{i=1}^{\#A} A_i\right) + \left(\sum_{i=1}^{\#UC} U_i\right)$ mit *#A* die Anzahl aller Actors A_i und *#UC* die Anzahl aller Use Cases U_i der UCP-Sprache	*Formel 30*
T-Faktor $:= \prod_{i=1}^{12}\left(1 + c_T \cdot (T_i - 3) \cdot W_i\right)$ *mit Konstanten:* $c_T := 0{,}01$, $W_i \geq 0$ *entsprechend Tabelle 53* *T-Faktor = 1* $\Leftrightarrow \quad \forall i \in [1..12] : T_i = 3$	*Formel 31*
M-Faktor $:= \prod_{i=1}^{12}\left(1 + c_M \cdot (M_i - 3) \cdot W_i\right)$ *Mit Konstanten:* $c_M := 0{,}01$, $W_i \geq 0$ *entsprechend Tabelle 52* *M-Faktor = 1* $\Leftrightarrow \quad \forall i \in [1..12] : M_i = 3$	*Formel 32*
PF := *18,9 (Produktivitätsfaktor, je Unternehmen zu kalibrieren)*	*Formel 33*

In Formel 29 wird der Multiplikator 5 explizit ausgewiesen und nicht in die Konstante *PF* mit integriert. Der Grund hierfür ist, dass der A-Faktor aufgrund seiner neuen Definition etwa um einen Faktor fünf kleiner ist als der Wert für die ungewichteten Use Case Points in der Karner-Methode. Um einen Vergleich mit anderen Varianten der UCP-Methode in der Literatur sowohl für *PF* als auch für UCP-Werte zu ermöglichen, wird der Faktor explizit in der Formel ausgewiesen. Damit entspricht der Faktor (5 · A-Faktor) ungefähr dem UUCP-Wert der Karner-Methode (Formel 7, Seite 31). Aufgrund des stärkeren Gewichtes für die Actors und neu entwickelten Transformationsregeln von der Spezifikation über die UCP-Sprache zum A-Faktor sind die Werte nur in ihrer allgemeinen Größenordnung vergleichbar, eine einfache Umrechnung ist nicht möglich.

Der T-Faktor kann nun Werte im Bereich zwischen 0,58 und 1,28 annehmen, was einer Spreizung von 221% entspricht (=Minimum/Maximum). Für den M-Faktor ist die Spreizung mit 397% fast doppelt so hoch und basiert auf einem Wertebereich von 0,43 bis 1,705.

Es bestätigt sich also auch nach finaler Festlegung der Gewichte, dass der M-Faktor weit mehr korrigierenden Einfluss auf den A-Faktor, und damit auf das Gesamtergebnis nehmen kann, als der T-Faktor. Der Produktivitätsfaktor *PF* ist mit 18,9 Bearbeiterstunden (Bh) pro Point geringer als in der Karner-Methode (27,3 Bh/UCP). Dies liegt an der geänderten Punkteskala des A-Faktors und den neuen Formeln der Kostenfaktoren.

Der Produktivitätsfaktor kann für unterschiedliche Organisationen unterschiedlich sein und ist bei Anwendung der Methode in einem Unternehmenskontext gemäß Abschnitt 5.7 zu kalibrieren, d.h. so einzustellen, dass N4 (Tabelle 50) erfüllt wird.

Im Rahmen der Definition der Methode UCP 3.0 wurde genau festgelegt, welche Projektaufwände durch *PE* erfasst sind. Dies ist für die Anwendung und Nachvollziehbarkeit der Methode wichtig und löst damit die Problemstellung P 9 (Umfang der durch UCP abgedeckten Projektaufwände) aus Abschnitt 2.7.

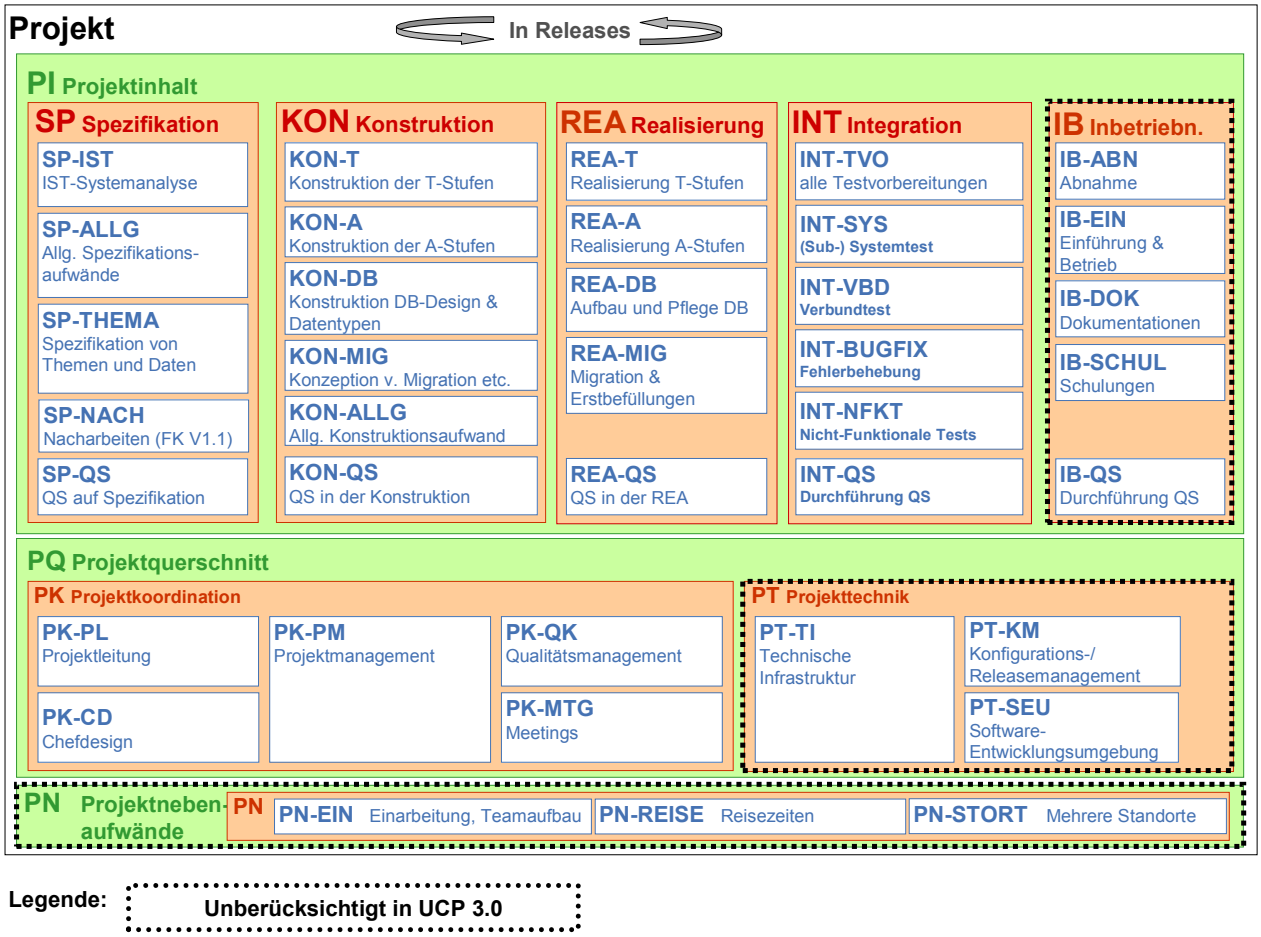


Abbildung 41: Umfang der durch UCP 3.0 geschätzten Projektaufgaben (Kontenrahmen)

Dazu wurde für alle Projekte ein einheitliches *Aufwandsmodell* zugrunde gelegt, welches die Aufwandsblöcke für Software-Entwicklungsprojekte betrieblicher Informationssysteme auf drei Detaillierungs-Ebenen definiert. In Abbildung 41 sind die drei Ebenen abgebildet, die beiden obersten Ebenen sind:

1. Ebene (grün) = Projektinhalt (PI), Projektquerschnitt (PQ), Projektnebenaufwände (PN)
2. Ebene (rot) = Spezifikation (SP), Konstruktion (KON), Realisierung (REA), Integration (INT), Inbetriebnahme (IB), Projektkoordination (PK), Projekttechnik (PT) und Projektnebenaufwände (PN, identisch zur 1. Ebene)

UCP 3.0 umfasst aus dem Projektinhalt die Spezifikation (SP), Konstruktion (KON), Realisierung (REA) und Integration (INT) sowie aus dem Projektquerschnitt die Projektkoordination

(PK). Im Rational Unified Process (RUP, siehe Abschnitt 2.3.1) entspricht dies in etwa den Disciplines in den Phasen *Inception*, *Elaboration* und *Construction*. *Transition* bleibt unberücksichtigt.

UCP 3.0 umfasst nicht die in Abbildung 41 gepunktet hinterlegten Bereiche: Inbetriebnahme (Abnahme, Einführung und Betrieb, Dokumentation über die Spezifikation und Konstruktion hinaus, Schulung) und Aufbau einer allgemeinen Projekttechnik (Technische Infrastruktur, Konfigurations- und Releasemanagement, Software-Entwicklungsumgebung) und Projektnebenaufwände (Einarbeitung, Teamaufbau, Reisezeiten, Mehraufwand für standortübergreifendes Arbeiten oder Offshore Entwicklung). Ebenso sind keine Aufwände für Gewährleistung, Vorstudien oder die Grobspezifikation gemäß Abbildung 10, Seite 21 berücksichtigt.

Damit liegt eine klar nachvollziehbare und einheitliche Klassifikation der Projektaufwände vor. Auf dieser Basis ist ein systematischer Vergleich von Projektaufwänden unterschiedlicher Teams und Projekte in unterschiedlichen Branchen möglich und wird in Abschnitt 6.4 wieder aufgegriffen.

6.2 Unterstützung der Methode durch ein Schätzwerkzeug

Zur Durchführung der in diesem Kapitel vorgestellten Methode UCP 3.0 wurde ein Werkzeug auf Basis von Microsoft Excel entwickelt. Es setzt auf den in der UCP-Sprache identifizierten Actors und Use Cases auf und unterstützt in sieben Schritten (Abbildung 42) die Erfassung und Zählung für die UCP-Schätzung Für jeden Schritt ist ein eigenes Arbeitsblatt (Reiter) in Excel zuständig:

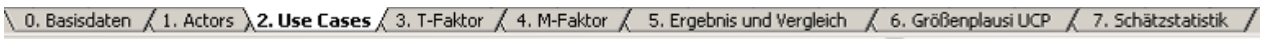


Abbildung 42: Schätzwerkzeug - Die sieben Schritte der UCP 3.0 Schätzung

Eine Farblogik hilft bei der Orientierung: Gelbe Felder enthalten Beschriftungen, weiße oder hellblaue Felder sind Eingabefelder für den Schätzer, graue Felder sind Auswahlfelder und blaue Felder zeigen das Ergebnis einer Berechnung an, wobei die Beschriftung dazu ebenfalls blau hinterlegt ist.

In einem Schritt Null werden Basisdaten (Stammdaten) zum Projekt und Informationen zum Schätzer, zur Dauer der Schätzung etc. erfasst. Diese Daten sind für die eigentliche Durchführung einer Schätzung nicht zwingend notwendig und werden hier nicht weiter erläutert.

	A	B	C	D	E	F	G
1	**Actors Bewertung**						
2	**Actors Beschreibung**				**Bewertung**		**Kommentar**
3	**Gruppe**	**Nr.**	**Name**	**Typ:**	**Komplexität**	**Points**	
4					**Summe:**	**9**	
5	Nutzer	1	Beispiel: Endnutzer	Benutzer-Interface	komplex	3	
6	Nutzer	2	Beispiel: Administrator	Benutzer-Interface	komplex	3	
7	Randsystem	3	Beispiel: über eine zustandsorientierte SSt. angebundenes Nachbarsystem	Nachbarsystem	mittel	2	keine Versionierung von Schnittstellen, keine fachlichen Transaktionen
8	Randsystem	4	Beispiel: über eine zustandslose SSt. angebundenes Nachbarsystem	Nachbarsystem	einfach	1	service-orientierte Anbindung, keine Versionierung, keine Transaktionen
9		5					
10		6					

0. Basisdaten | 1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 43: Arbeitsblatt Actors

Im ersten Schritt sind die Actors in Form einer Liste zu erfassen und hinsichtlich ihrer Komplexität zu bewerten, Abbildung 43 zeigt beispielhaft die Erfassung von vier Actors (Endnutzer, Administrator und zwei Schnittstellen), in Spalte A können Actors nach fachlichen Gesichtspunkten durch einen frei definierbaren Namen gruppiert werden, Spalte B sorgt für eine fortlaufende Nummerierung zwecks besserer Referenzierbarkeit, in Spalte C kann der Name des Actors erfasst werden. Die Spalte D bietet eine Auswahlbox mit den Möglichkeiten *Benutzer-Interface* (dann ist gemäß Abschnitt 3.1.2 der Actor als *komplex* zu werten und es werden automatisch 3 Points in der Spalte *Points* angesetzt) oder *Nachbarsystem* (dann kann der Actor wahlweise als *einfach*, *mittel* oder *komplex* eingestuft werden und es werden automatisch 1, 2 oder 3 Points zugeordnet). Die Summe aller Points aller Actors wird automatisch in Feld F4 berechnet und in das Arbeitsblatt *Ergebnis* übertragen.

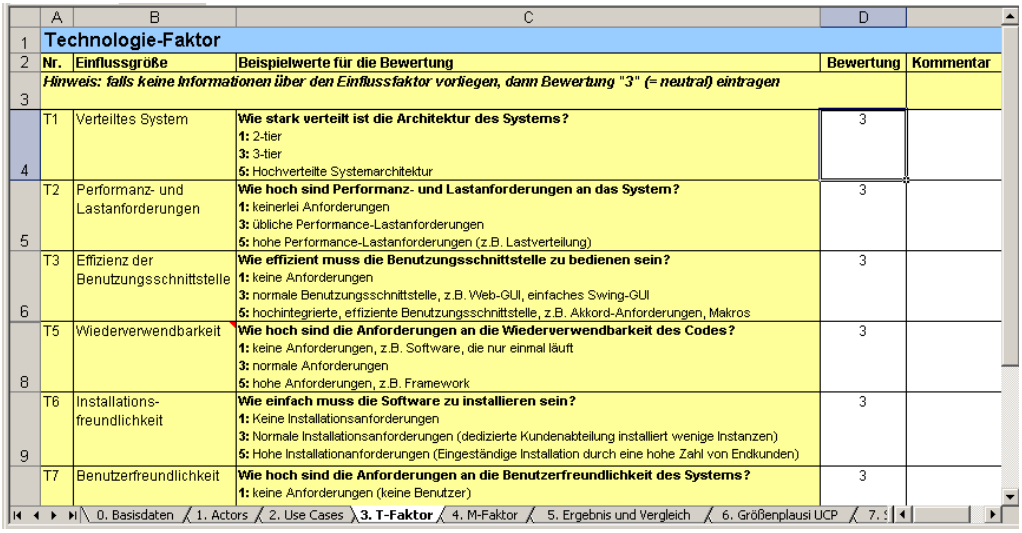
Im zweiten Schritt sind im Arbeitsblatt *Use Cases* diese zu erfassen (Abbildung 44). Analog zu den Actors ist eine Gruppierung, Nummerierung und Vergabe eines Namens möglich. Die Komplexität kann dann entweder durch Zählen der Anzahl Szenarien (Spalte D), Schritte (Spalte E) und Interaktionselemente (Spalten F-N) oder in der Spalte S durch direkte „intuitive“ Vergabe eines *Points*-Wertes erfolgen. Re-Use kann in Spalte U in Prozent eingetragen werden. Die Spalte R *Effektiv* zeigt das gemäß Abschnitt 3.1.2 automatisch berechnete Ergebnis der Bewertung in Points an, Feld T6 berechnet die Summe über alle Use Cases und überträgt diese Summe ebenfalls in das Arbeitsblatt *Ergebnis*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	**Use Case Bewertung**																					
2	**Use-Case-Beschreibung**			**Kenngrößen**											**Bewertung**							**Kommentar**
3						**Anzahl Interaktionselemente (IAEe)**										**in Points**							
4							**Dialoge**			**Schnittstellen**			**Berichte**		*Berechnete Points je Zählmaß*							
5	**Gruppe**	**Nr.**	**Name**	**Szenarien**	**Schritte**	einfach	mittel	komplex	einfach	mittel	komplex	einfach	mittel	komplex	*Szenarien*	*Schritte*	*IAEe*	**Berechnet**	**Intuitiv**	**Effektiv**	**Re-Use in %**	
6			**Summen:**	**22**	**71**	**1**	**4**	**3**		**2**		**2**	**2**					**21**	**7**	**24**		
7	Dialog	1	Beispiel: einfacher Use Case	1	3		1								1,0	1,0	0,8	1		**1,0**		
8	Dialog	2	Beispiel: mittlerer Use Case	2	5		2								2,0	1,7	1,5	2		**2,0**		
9	Dialog	3	Beispiel: komplexer Use Case	2	9	1		1		2					2,0	3,0	3,4	3		**3,0**		
10	Dialog	4	Beispiel: großer Use Case mit Wiederverwendung	4	14							2	2		4,0	4,7	2,3	4		**4,0**		Wiederverwendung aus Use Case XY (über alle Phasen)
11	Batch	5	Beispiel: Use Case korrigiert durch intuitiven Wert	1	8		1								1,0	2,7	0,8	1	2	**2,0**		Schritte sind eher umfangreich, daher ist das Vertrauen in die Bewertung durch Schritte am größten und der Use Case wurde intuitiv auf 2 UCP nach oben korrigiert.
12	Batch	6	Beispiel: Warnung	5	9			2							5,0	3,0	3,0	3		**3,0**		Die Szenarien beschreiben den Use Case nicht so gut wie die anderen Kenngrößen. Der Vorschlag von 3 UCP ist daher in Ordnung.
13	Batch	7	Beispiel: Warnung und Korrektur	1	15										1,0	5,0		3	5	**5,0**		Hochkomplexe Analyse, die immer nach demselben Schema abläuft. Hier sind nur die Schritte repäsentativ.
14	Batch	8	Beispiel: Warnung ohne Korrektur	6	8										6,0	2,7		4		**4,0**		Durch die Bewertung der Szenarien wird dieser Use Case eventuell überbewertet. Eine manuelle Kontrolle ergab aber, dass die Szenarien die Komplexität erhöhen, damit ist der berechnete Wert in diesem Fall
15		9																				
16		10																				
17		11																				

0. Basisdaten | 1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 44: Arbeitsblatt Use Cases

Rot hinterlegt sind berechnete Felder als Warnhinweis, sofern die absolute Abweichung zwischen dem Wert aus der Spalte O, P oder Q und dem insgesamt durch den Median berechneten Points-Wert des Use Cases in Spalte R um Zwei oder mehr abweicht. Die Warnung soll den Schätzer darauf hinweisen, dass der Points-Wert in Spalte R möglicherweise durch eine der Einzelkenngrößen stark verfälscht wird. Dies kann beispielsweise der Fall sein, wenn ein Use Case sehr viele, aber sehr „einfache“ Szenarien enthält.

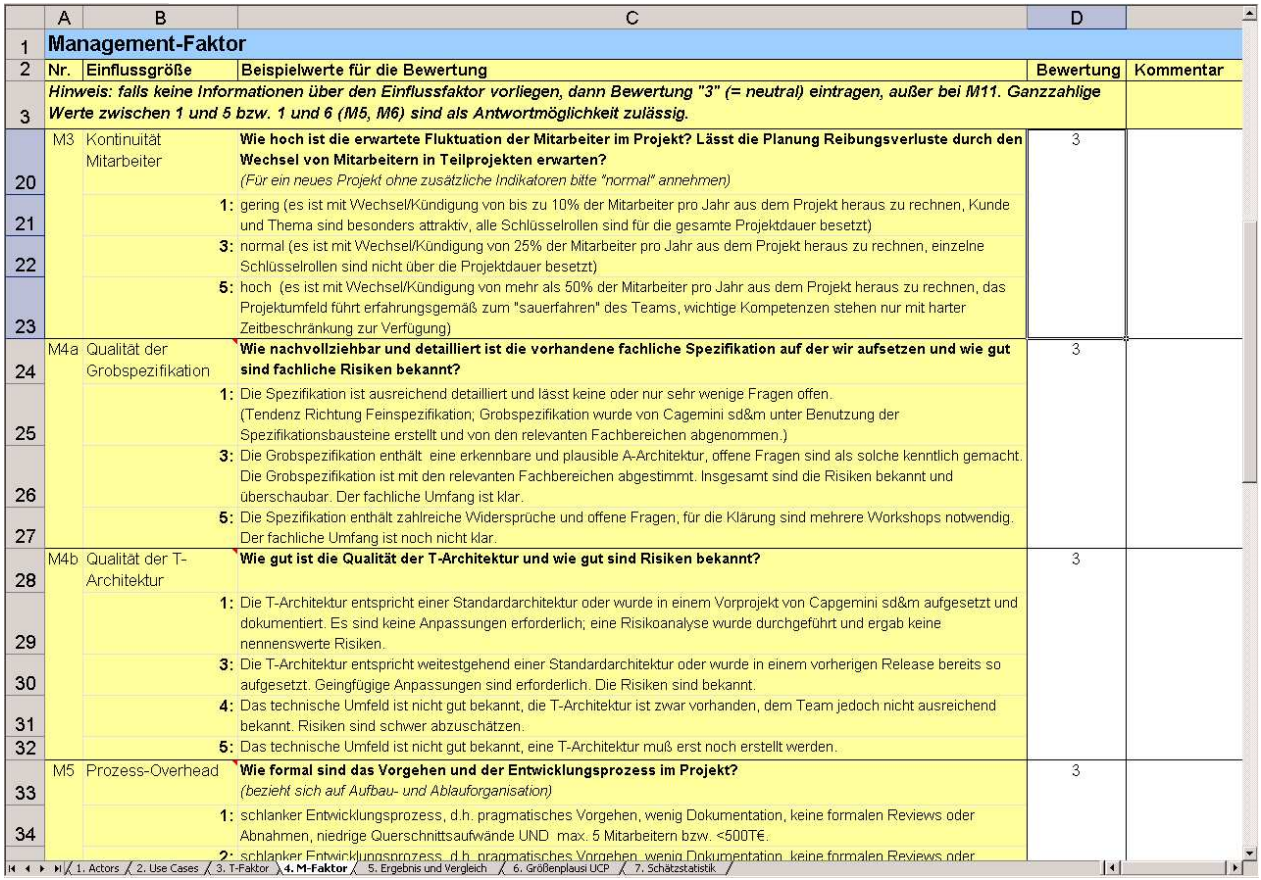


	A	B	C	D	
1	**Technologie-Faktor**				
2	**Nr.**	**Einflussgröße**	**Beispielwerte für die Bewertung**	**Bewertung**	**Kommentar**
3	***Hinweis: falls keine Informationen über den Einflussfaktor vorliegen, dann Bewertung "3" (= neutral) eintragen***				
4	T1	Verteiltes System	**Wie stark verteilt ist die Architektur des Systems?** **1:** 2-tier **3:** 3-tier **5:** Hochverteilte Systemarchitektur	3	
5	T2	Performanz- und Lastanforderungen	**Wie hoch sind Performanz- und Lastanforderungen an das System?** **1:** keinerlei Anforderungen **3:** übliche Performance-Lastanforderungen **5:** hohe Performance-Lastanforderungen (z.B. Lastverteilung)	3	
6	T3	Effizienz der Benutzungsschnittstelle	**Wie effizient muss die Benutzungsschnittstelle zu bedienen sein?** **1:** keine Anforderungen **3:** normale Benutzungsschnittstelle, z.B. Web-GUI, einfaches Swing-GUI **5:** hochintegrierte, effiziente Benutzungsschnittstelle, z.B. Akkord-Anforderungen, Makros	3	
8	T5	Wiederverwendbarkeit	**Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes?** **1:** keine Anforderungen, z.B. Software, die nur einmal läuft **3:** normale Anforderungen **5:** hohe Anforderungen, z.B. Framework	3	
9	T6	Installations-freundlichkeit	**Wie einfach muss die Software zu installieren sein?** **1:** Keine Installationsanforderungen **3:** Normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen) **5:** Hohe Installationanforderungen (Eingeständige Installation durch eine hohe Zahl von Endkunden)	3	
	T7	Benutzerfreundlichkeit	**Wie hoch sind die Anforderungen an die Benutzerfreundlichkeit des Systems?** **1:** keine Anforderungen (keine Benutzer)	3	

0. Basisdaten | 1. Actors | 2. Use Cases | **3. T-Faktor** | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. …

Abbildung 45: Arbeitsblatt T-Faktor (Auszug)

Im dritten Schritt ist der T-Faktor zu bestimmen. Abbildung 45 zeigt einen Ausschnitt des entsprechenden Arbeitsblattes. Je Einflussgröße ist eine der angegebenen Ausprägungen aus Spalte C in Spalte D einzutragen. Das berechnete Ergebnis wird im Arbeitsblatt *Ergebnis* angezeigt. Aus psychologischen Gründen erfolgt die Anzeige nicht im gleichen Dialog, da bei der Bewertung der Einflussgrößen ein intuitiv hoch oder niedrig wahr genommener T-Faktor den Schätzer nicht beeinflussen soll.



	A	B	C	D	
1	**Management-Faktor**				
2	**Nr.**	**Einflussgröße**	**Beispielwerte für die Bewertung**	**Bewertung**	**Kommentar**
3	***Hinweis: falls keine Informationen über den Einflussfaktor vorliegen, dann Bewertung "3" (= neutral) eintragen, außer bei M11. Ganzzahlige Werte zwischen 1 und 5 bzw. 1 und 6 (M5, M6) sind als Antwortmöglichkeit zulässig.***				
20	M3	Kontinuität Mitarbeiter	**Wie hoch ist die erwartete Fluktuation der Mitarbeiter im Projekt? Lässt die Planung Reibungsverluste durch den Wechsel von Mitarbeitern in Teilprojekten erwarten?** *(Für ein neues Projekt ohne zusätzliche Indikatoren bitte "normal" annehmen)*	3	
21			**1:** gering (es ist mit Wechsel/Kündigung von bis zu 10% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, Kunde und Thema sind besonders attraktiv, alle Schlüsselrollen sind für die gesamte Projektdauer besetzt)		
22			**3:** normal (es ist mit Wechsel/Kündigung von 25% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, einzelne Schlüsselrollen sind nicht über die Projektdauer besetzt)		
23			**5:** hoch (es ist mit Wechsel/Kündigung von mehr als 50% der Mitarbeiter pro Jahr aus dem Projekt heraus zu rechnen, das Projektumfeld führt erfahrungsgemäß zum "sauerfahren" des Teams, wichtige Kompetenzen stehen nur mit harter Zeitbeschränkung zur Verfügung)		
24	M4a	Qualität der Grobspezifikation	**Wie nachvollziehbar und detailliert ist die vorhandene fachliche Spezifikation auf der wir aufsetzen und wie gut sind fachliche Risiken bekannt?**	3	
25			**1:** Die Spezifikation ist ausreichend detailliert und lässt keine oder nur sehr wenige Fragen offen. (Tendenz Richtung Feinspezifikation; Grobspezifikation wurde von Cagemini sd&m unter Benutzung der Spezifikationsbausteine erstellt und von den relevanten Fachbereichen abgenommen.)		
26			**3:** Die Grobspezifikation enthält eine erkennbare und plausible A-Architektur, offene Fragen sind als solche kenntlich gemacht. Die Grobspezifikation ist mit den relevanten Fachbereichen abgestimmt. Insgesamt sind die Risiken bekannt und überschaubar. Der fachliche Umfang ist klar.		
27			**5:** Die Spezifikation enthält zahlreiche Widersprüche und offene Fragen, für die Klärung sind mehrere Workshops notwendig. Der fachliche Umfang ist noch nicht klar.		
28	M4b	Qualität der T-Architektur	**Wie gut ist die Qualität der T-Architektur und wie gut sind Risiken bekannt?**	3	
29			**1:** Die T-Architektur entspricht einer Standardarchitektur oder wurde in einem Vorprojekt von Capgemini sd&m aufgesetzt und dokumentiert. Es sind keine Anpassungen erforderlich; eine Risikoanalyse wurde durchgeführt und ergab keine nennenswerte Risiken.		
30			**3:** Die T-Architektur entspricht weitestgehend einer Standardarchitektur oder wurde in einem vorherigen Release bereits so aufgesetzt. Geingfügige Anpassungen sind erforderlich. Die Risiken sind bekannt.		
31			**4:** Das technische Umfeld ist nicht gut bekannt, die T-Architektur ist zwar vorhanden, dem Team jedoch nicht ausreichend bekannt. Risiken sind schwer abzuschätzen.		
32			**5:** Das technische Umfeld ist nicht gut bekannt, eine T-Architektur muß erst noch erstellt werden.		
33	M5	Prozess-Overhead	**Wie formal sind das Vorgehen und der Entwicklungsprozess im Projekt?** *(bezieht sich auf Aufbau- und Ablauforganisation)*	3	
34			**1:** schlanker Entwicklungsprozess, d.h. pragmatisches Vorgehen, wenig Dokumentation, keine formalen Reviews oder Abnahmen, niedrige Querschnittsaufwände UND max. 5 Mitarbeitern bzw. <500T€.		
			2: schlanker Entwicklungsprozess, d.h. pragmatisches Vorgehen, wenig Dokumentation, keine formalen Reviews oder		

1. Actors | 2. Use Cases | 3. T-Faktor | **4. M-Faktor** | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 46: Arbeitsblatt M-Faktor (Auszug)

Im vierten Schritt ist der M-Faktor zu bestimmen, Abbildung 46 zeigt einen Ausschnitt des entsprechenden Arbeitsblattes. Je Einflussgröße ist eine der angegebenen Ausprägungen aus Spalte C in Spalte D einzutragen. Das berechnete Ergebnis wird im Arbeitsblatt *Ergebnis* angezeigt.

Im fünften Schritt wird das Ergebnis der UCP-Schätzung zusammengefasst. Abbildung 47 zeigt das entsprechende Arbeitsblatt. Im oberen Teil sind die Ergebnisse aus den Schritten Eins bis Vier zusammengefasst und in Feld B12 der Aufwand *PE* in Bearbeitertagen berechnet. Im unteren Bereich (Zeilen 23 bis 33) kann ein direkter Vergleich mit einer Bottom-Up Schätzung gemäß Kontenrahmen (Abschnitt 6.1 und Abbildung 41) erfolgen. Es werden nur jene Konten auf der zweiten Ebene aggregiert berücksichtigt, die durch die UCP-Methode abgedeckt werden. In Spalte D besteht die Möglichkeit, Korrekturwerte einzutragen, falls in der Bottom-Up Schätzung (Expertenschätzung) Aufwände berücksichtigt wurden, die nicht durch die Use Case bezogene Schätzung abgebildet werden könnten, z.B. falls die Spezifikation schon in Teilen vorliegt und daher in der Bottom-Up Schätzung zu knapp geschätzt wurde.

	A	B	C	D	E	F
1	**Ergebnis der UCP-Schätzung**					
2	**Größe**	**Value**				
3	Points aus Actors	9				
4	Points aus Use Cases	24				
5	A-Faktor	33	*= Points aus Actors + Points aus Use Cases*			
6	T-Faktor	1,00				
7	M-Faktor	1,00				
8	**Bereinigte Use Case Points (UCP)**	**33**				
9						
10	Produktivitätsfaktor (PF)	18,90				
11						
12	**Aufwand in BT**	**390**	*= 5 x UCP x PF / 8*			
15						
16						
17	*Hinweis: UCP schätzt den Aufwand ab Feinspezifikation bis BzA. Nicht enthalten sind Aufwand für Grobspezifikation, PN, PT, BERAT und IB).* *Daher ist vor dem Vergleich ggf. eine manuelle Korrektur des geschätzten Aufwands notwendig.*					
18	**Manuelle Korrektur der Expertenschätzung/Nachkalkulation und Vergleich mit UCP**					
20	Der hier vorliegende Vergleichsaufwand entspricht		dem Ergebnis der Expertenschätzung			
22	**Aufwand nach Kategorie**	**korrigiert**	**Schätzung**	**Korrektur**	*<die Korrektur kann positiv oder negativ sein: positiv bedeutet, Aufwand in Schätzung aber nicht in UCP enthalten, negativ bedeutet, Aufwand nicht in Schätzung enthalten>*	
23	Spezifikation (SP)	50 BT	50 BT	0 BT	*<Begründung für manuelle Korrektur (z.B. 20 BT für Grobkonzept)>*	
24	Konstruktion (KON)	45 BT	45 BT		*<Begründung für manuelle Korrektur>*	
25	Realisierung (REA)	170 BT	170 BT		*<Begründung für manuelle Korrektur>*	
26	Integration (INT)	50 BT	50 BT		*<Begründung für manuelle Korrektur>*	
27	Inbetriebnahme (IB)	0 BT	15 BT	15 BT	*wird durch UCP nicht abgedeckt*	
28	Projektkoordination (PK)	80 BT	80 BT		*<Begründung für manuelle Korrektur>*	
29	Projekttechnik (PT)	0 BT	44 BT	44 BT	*wird durch UCP nicht abgedeckt*	
30	Projektnebenaufwände (PN)	0 BT	33 BT	33 BT	*wird durch UCP nicht abgedeckt*	
32	**Gesamt Schätzung (korrigiert)**	395 BT				
33	**Abweichung UCP ggü. Schätzung**	-1%				

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | **5. Ergebnis und Vergleich** | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 47: Arbeitsblatt Ergebnis und Vergleich

Feld B33 zeigt die Abweichung der UCP-Schätzung von der Expertenschätzung an und zeigt farblich in Ampel-Logik die absolute Abweichung für den Quality Gate Prozess (Abschnitt 2.6) an (grün: 0-20% Abweichung, gelb: 20-40%, rot > 40%).

Im sechsten Arbeitsblatt kann eine einfache Plausibilisierung des Schätzergebnisses durch einen Vergleich mit Mittelwerten aus der Projektdatenbank vorgenommen werde. Eingangsgrößen sind

lediglich die Anzahl Actors und Use Cases sowie die Einschätzung der Größenklasse {*klein, mittel, groß*}. Daraus wird anhand von Erfahrungswerten ein Vorschlag für die mittlere Größe der Use Cases und Actors gemäß Tabelle 23 (Seite 67) sowie für den T-Faktor und M-Faktor (Tabelle 51) gemacht und direkt der Projektaufwand ermittelt.

Im siebten Arbeitsblatt (Abbildung 48) werden verschiedene Auswertungen zur UCP-Schätzung dargestellt. Insbesondere die graphische Darstellung der Häufigkeitsverteilungen je Use Case Komplexitätsklasse soll dabei helfen, die Schätzung hinsichtlich ihrer Qualität bewerten zu können. Eine angemessene Verteilung der Use Cases auf die verschiedenen Komplexitätsklassen kann ein Hinweis auf eine gute und sinnvolle Granularität der Use Cases sein. Liegt der größte Teil aller Use Cases in *einer* Komplexitätsklasse, so ist dies ein Indiz für einen falschen Schnitt der Use Cases (Abschnitt 3.2).

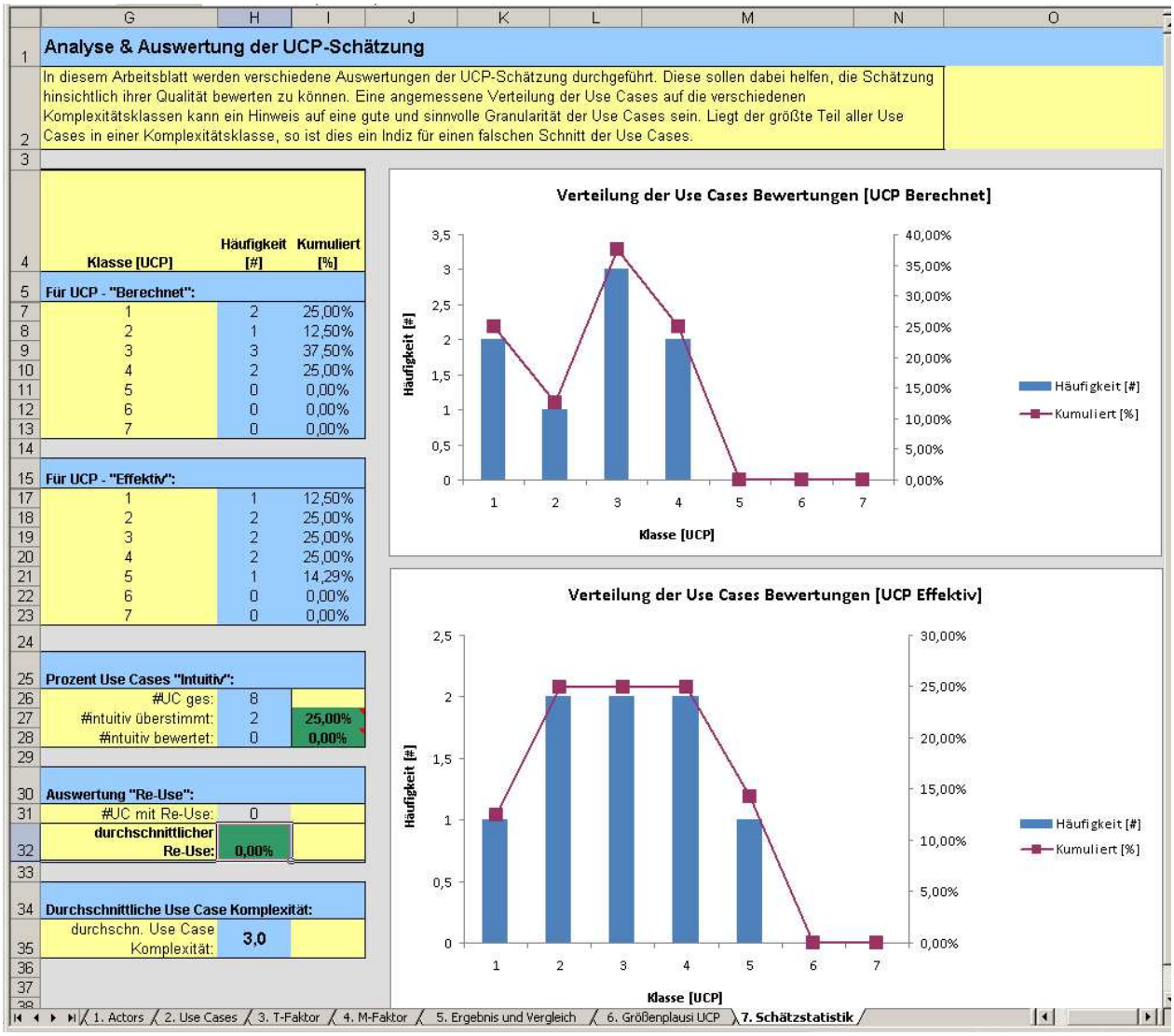


Abbildung 48: Arbeitsblatt Schätzstatistik (Auszug)

Insgesamt ist das Schätzwerkzeug einfach zu bedienen und Excel als Plattform hat sich als völlig ausreichend erwiesen.

6.3 Beispielhafte Anwendung des Schätzwerkzeuges

Anhand eines Beispiels aus der industriellen Praxis soll die Anwendung des in Abschnitt 6.2 beschriebenen Werkzeuges für die Methode UCP 3.0 gemäß Abschnitt 6.1 vorgestellt werden. Wir verwenden dafür Auszüge aus einer realen Spezifikation. Kundenname und Kontext sind anonymisiert bzw. verfremdet worden, um Vertraulichkeit zu gewährleisten.

Zum Kontext des Beispiels: Die Hochschule XY finanziert sich als gemeinnützige Einrichtung wesentlich über Spenden von ehemaligen Studierenden sowie Großspendern aus der Industrie. Zur Verwaltung ihrer Spenden und Fördermittel benötigt sie eine neue *Förderverwaltung*.

Aus Gründen der besseren Lesbarkeit werden wir nachfolgend die dem Beispiel *Förderverwaltung* zugrunde liegende Spezifikation schrittweise vorstellen und dabei jeweils bereits die Anwendungsfälle ableiten, auf Use Cases transformieren, die Zählung der Szenarien, Schritte und Interaktionselemente durchführen und die Points je Use Case bestimmen. Erst danach bestimmen wir die Actors, da dann die gesamte Spezifikation dem Leser bekannt ist. Eine Übersicht aller identifizierten Use Cases gibt bereits Abbildung 51 auf Seite 151.

Für die Aufwandsschätzung werden die Anwendungsfälle des Geschäftsprozesses *Verwaltung eines Förderers* betrachtet, siehe Abbildung 49. Das Diagramm zeigt alle Anwendungsfälle dieses Geschäftsprozesses und deren Verknüpfung. Im System werden Förderer (natürliche Personen, Organisationen) gesucht, angelegt, bearbeitet und es werden Bescheinigungen ausgestellt. *Förderer anlegen* besteht (mindestens) aus den Schritten *Persönliche Daten anlegen*, *Adresse anlegen*, *Beitragsvereinbarung anlegen* und *Bankverbindung anlegen*. Dies ist die Geschäftsprozess-Sicht, in einer Anwendungsfall-Modellierung werden diese Schritte zu eigenen Anwendungsfällen.

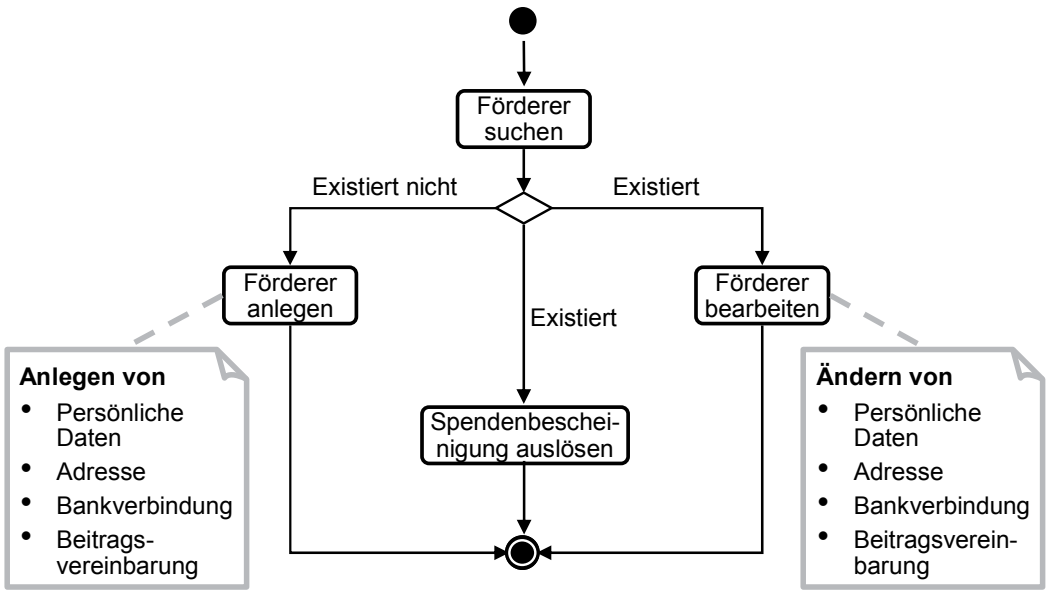


Abbildung 49: Geschäftsprozess „Verwaltung eines Förderers“

Die Anwendungsfälle *Förderer anlegen* und *Förderer bearbeiten* sind auf den ersten Blick sehr ähnlich.

Förderer suchen ist fachlich gesehen kein eigener Anwendungsfall, sondern eine Voraussetzung für die anderen genannten Anwendungsfälle. Es ist aber einfacher, ihn gesondert zu zählen und wir bilden ihn daher auf einen eigenen Use Case *Förderer Suchen* ab.

Die Spezifikation liefert hierzu noch eine Grobe textuelle Beschreibung (Abbildung 50). Wir lesen daraus gemäß Abschnitt 3.3.1 zwei Szenarien ab (eindeutige und nichteindeutige Suchergebnisse), fünf Schritte (*Suchkriterien anzeigen*, *Suchkriterien eingeben*, *Suche durchführen*, *Vorschlagsliste anzeigen*, *Förderer anzeigen*) und drei Dialoge (*Suchkriterien* {einfach}, *Vorschlagsliste* {mittel}, *Detailanzeige* {mittel}). Dies resultiert gemäß Abschnitt 3.1.2 in einer mittleren Komplexität, also 2 Points für den Use Case *Förderer Suchen*.

Beschreibung

Die zukünftige Anwendung muss die Suche nach folgenden Kriterien ermöglichen:

- Suche nach Name
- Suche nach Adresse und Adressteilen
- Suche nach aktuellen, vordatierten und historischen Adressen und Adressteilen
- Suche nach Bankverbindung
- Suche nach Kontaktinformation(E-Mail, Telefonnummer)
- Suche mit Wildcard (*)

Bei uneindeutigen Suchergebnissen soll eine Vorschlagliste vorgesehen werden. Die Anzahl von Ergebnissätzen soll durch einen Benutzer konfigurierbar bzw. änderbar sein. Alle Kriterien müssen sowohl als einzelne Suchparameter als auch in Kombination benutzt werden können.

Abbildung 50: Grobe textuelle Beschreibung „Förderer Suchen“

Abbildung 51 zeigt, wie dieser Use Case im Schätzwerkzeug in der Zeile 7 (*Nr. 1)* erfasst wird. Die weiteren Use Cases sind bereits ebenfalls schon erfasst und werden nun erläutert.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	**Use Case Bewertung**																				
2	**Use-Case-Beschreibung**			**Kenngrößen**											**Bewertung**						
3						**Anzahl Interaktionselemente (IAE)**										**in Points**						
4						**Dialoge**			**Schnitt-stellen**			**Berichte**			*Berechnete Points je Zählmaß*						
5	**Gruppe**	**Nr.**	**Name**	**Szenarien**	**Schritte**	einfach	mittel	komplex	einfach	mittel	komplex	einfach	mittel	komplex	*Szenarien*	*Schritte*	*IAE*	**Berechnet**	**Intuitiv**	**Effektiv**	**Re-Use in %**
6			**Summen:**	**13**	**40**	**2**	**8**						**1**					**14**	**3**	**14**	
7		1	Förderer suchen	2	5	1	2								2	2	2	2		**2**	
8		2	Förderer anlegen	4	5	1									4	2	1	2	1	**1**	
9		3	Persönliche Daten anlegen	2	5		1								2	2	1	2		**2**	
10		4	Adresse anlegen	3	10		3								3	3	2	3		**3**	
11		5	Bankverbindung anlegen	1	3		1								1	1	1	1		**1**	
12		6	Beitragsvereinbarung anlegen	1	4		1						1		1	1	2	1	2	**2**	
13		7	Förderer bearbeiten																		
14		8	Spendenbescheinigung auslösen		8											3		3		**3**	

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 51: Erfassung der Use Cases im Schätzwerkzeug

Der Anwendungsfall *Förderer Anlegen* bietet keine eigene Funktionalität, sondern zerfällt in vier Unteranwendungsfälle, die hier nur aufgerufen werden (*Persönliche Daten anlegen*, *Adresse anlegen*, *Bankverbindung anlegen* und *Beitragsvereinbarung anlegen*, siehe Abbildung 49).

Das Anlegen der persönlichen Daten ist die Voraussetzung für die anderen Anwendungsfälle. Wir leiten aus der Spezifikation einen Auswahl-Dialog {einfach} ab (Auswahl der vier Unteranwendungsfälle – die eigentlichen Pflegedialoge werden in den aufgerufenen Anwendungsfällen gezählt) und vier Szenarien (die vier alternativen Anwendungsfälle). Insgesamt ergeben sich damit fünf Schritte (jeder Aufruf der vier Anwendungsfälle zählt als ein Schritt sowie der Aufruf eines Dialoges zählt ebenfalls als ein Schritt), siehe Zeile 8 (*Nr. 2*) in Abbildung 51. Daraus würde ein Use Case mittlerer Komplexität folgen, was allerdings aufgrund der einfachen Struktur nicht gerechtfertigt erscheint. Die hohe Zahl an sehr einfachen Szenarien führt zu einer Überbewertung des Use Cases. Dies wird durch die rote „Ampelfarbe“ im Feld O8 angezeigt. Wir überstimmen daher die exakt gezählte *mittlere* Bewertung durch eine *einfache* intuitive Bewertung (= 1 Point) und geben in Feld S8 eine Eins ein, was zu einem Point für den Use Case in Feld T8 führt.

Der Anwendungsfall *Persönliche Daten anlegen* wird in der Spezifikation durch Abbildung 52 und den folgenden textuellen Erläuterungen beschrieben: „Es werden die persönlichen Daten, wie z.B. Name, Vorname, Geschlecht, Geburtsdatum aufgenommen. Anschließend werden die Daten im Datenbestand der Förderer auf Dubletten geprüft. Der Benutzer kann dann in einer Liste von bereits im System vorhandenen Förderern auswählen und bearbeiten oder einen neuen Förderer anlegen.“

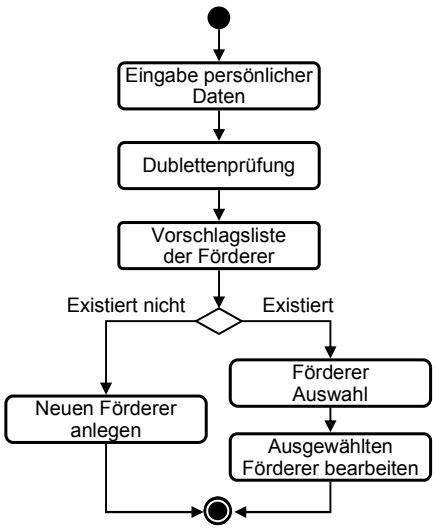


Abbildung 52: Anwendungsfall „Persönliche Daten anlegen“

Wir leiten aus der Spezifikation zwei Szenarien ab (Förderer existiert/existiert nicht). Die Verarbeitung wird in beiden Fällen mit der in der Spezifikation nicht explizit genannten Anzeige des Förderers abschließen, dies ergibt sich aus dem Kontext. Diese Anzeige ist aber schon im Use Case *Förderer suchen* gezählt worden und wird daher hier nicht noch mal gezählt (alternativ könnte ein Re-Use Faktor vergeben werden, siehe Abschnitt 3.2.6). Ebenfalls nicht gezählt wird der Schritt *Förderer Auswahl,* da er dem gleichnamigen Schritt im Use Case *Förderer suchen* entspricht. Der Dialog zur Anzeige der Vorschlagsliste der Förderer entspricht ebenfalls dem bei

Förderer suchen verwendeten Dialog *Vorschlagsliste anzeigen* und wird nicht nochmals gezählt. Insgesamt identifizieren wir für den Use Case *Persönliche Daten anlegen* fünf neue Schritte und einen Dialog (Abbildung 53). Im Schätzwerkzeug (Abbildung 51) wird dieses Ergebnis in Zeile 9 (*Nr. 3*) eingetragen.

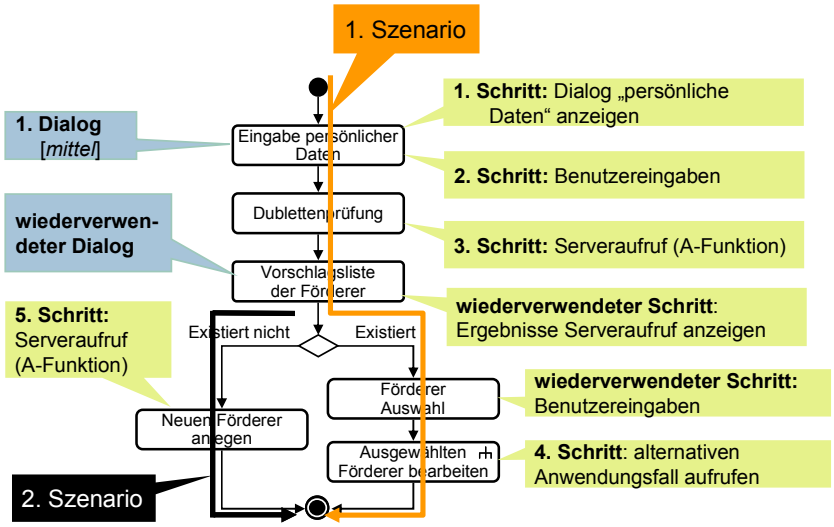
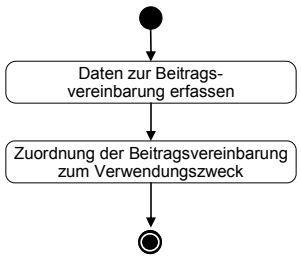


Abbildung 53: Zählung für Use Case „Persönliche Daten anlegen“

Der Anwendungsfall *Adresse anlegen* wurde bereits in Abbildung 27 (Seite 78) als Beispiel verwendet. Wir hatten 3 Szenarien, 10 Schritte und 3 Dialoge mittlerer Komplexität ermittelt (Abbildung 51, Zeile 10, *Nr. 4*).

Den Anwendungsfall *Bankverbindung anlegen* hatten wir ebenfalls als Beispiel für einen elementaren Anwendungsfall in Abbildung 22 (Seite 57) vorgestellt und ein Szenario, drei Schritte und einen Dialog mittlerer Komplexität identifiziert (Abbildung 51, Zeile 11, *Nr. 5*).



Beschreibung

- Als Vorbedingung gilt, dass vor dem Anlegen einer Beitragsvereinbarung die Bankleitzahl auf Gültigkeit geprüft wurde
- Im ersten Schritt werden die Daten für eine Beitragsvereinbarung aufgenommen, z.B. der Förderbetrag und die Periodizität
- Im zweiten Schritt wird die Beitragsvereinbarung einem bestimmten Zweck zugeordnet
- Für einen Förderer muss es möglich sein, beliebig viele Beitragsvereinbarungen anzulegen
- Außerdem wird durch das Anlegen einer Beitragsvereinbarung automatisch ein Dankschreiben erzeugt und verschickt

Abbildung 54: Anwendungsfall „Beitragsvereinbarung anlegen“

Für den Anwendungsfall *Beitragsvereinbarung anlegen* liegt die Spezifikation gemäß Abbildung 54 vor. Wir identifizieren ein Szenario. Aus der textuellen Beschreibung können vier Schritte identifiziert werden, die sich alleine aus dem Aktivitätsdiagramm nicht ergeben: *Dialog zur Datenerfassung*, *Eingabe*, *Verarbeitung*, *Dankschreiben erzeugen*. Wir identifizieren ferner zwei

Interaktionselemente: Ein Dialog zur Datenerfassung {mittel} und ein Bericht (Dankschreiben) {mittel}. Dies würde zu einem Use Case einfacher Komplexität führen (Abbildung 51, Feld R12), aufgrund der zwei Interaktionselemente korrigieren wir die Bewertung jedoch auf mittel (= 2 Points).

Der Geschäftsprozess *Förderer bearbeiten* besteht laut Spezifikation „aus den vier alternativen Schritten *Persönliche Daten bearbeiten, Adresse bearbeiten / sperren, Beitragsvereinbarung bearbeiten, Bankverbindung bearbeiten*. Die Änderungen erfolgen analog zum Anlegen. Aber auch beim Ändern von Fördererdaten müssen die gleichen Prüfungen auf Korrektheit und Vollständigkeit durchgeführt werden, wie beim Anlegen der Fördererdaten." Üblicherweise werden bei Datenpflegen die Anwendungsfälle zum *Anlegen*, *Ändern* und *Löschen* nicht gesondert gezählt, wenn sie sich nicht unterscheiden (andere Prüfungen, andere Dialoge). In diesem Fall ist aus dem Text die Analogie bereits ersichtlich, es sind auch keine zusätzlichen Dialoge vorgesehen. Daher wird hieraus kein weiterer Use Case abgeleitet. Im Schätzwerkzeug in Abbildung 51 wird zwar in Zeile 13 (*Nr. 7)* der Use Case *Förderer bearbeiten* zur Information notiert, aber keine Points zugeordnet.

Für den Anwendungsfall *Spendenbescheinigung auslösen* liefert die Spezifikation folgende Beschreibung: „Damit eine Spendenbescheinigung verschickt werden kann, muss der Hochschule XY mindestens eine dem Förderer zugeordnete Spende vorliegen. Der Zeitpunkt der Versendung wird gesteuert über den Spendenbescheinigungsstatus [...][32]. Der Förderer kann aber [...] auf eine Spendenbescheinigung verzichten. Hat der Förderer mehr als eine Spende geleistet, werden alle bisherigen Spenden gemeinsam bestätigt. Darüber hinaus muss jede einzelne Spende für eine Bescheinigung gesperrt werden können. Es ist notwendig, verschiedene Spendenbescheinigungsformulare verwenden zu können. Alle Daten zur Bescheinigung müssen so archiviert werden, dass die Bescheinigungen für jeden Förderer nachvollzogen werden können."

Dieser serverseitige Anwendungsfall wird durch einen komplexen Batch *Spendenbatch* realisiert werden, der Prozess kann aber auch individuell für einen Spender ausgelöst werden. Es sind im Original weitere Alternativszenarien beschrieben, die hier nicht betrachtet werden und eine Zerlegung des Use Cases erforderlich machen würden. Über den eigentlichen Versand ist hier nichts ausgesagt, wir gehen für unser Beispiel von einem getrennten Prozess aus. Folgende acht Schritte lassen sich aus obiger Beschreibung ableiten: 1. Fördererdaten einlesen; 2. Spendendaten für Förderer einlesen; 3. Bescheinigungsstatus für Spenden prüfen (ist schon verschickt, ist gesperrt, Förderer verzichtet); 4. je nach Status diese Daten akkumulieren; 5. Daten über Bescheinigungsformular einlesen; 6. Formular aus akkumulierten Daten erzeugen und ablegen; 7. Bescheinigungsstatus für Spenden aktualisieren; 8. Spendenbescheinigung archivieren. Hieraus ergibt sich ein Use Case hoher Komplexität (3 Points), siehe Abbildung 51, Zeile 14, *Nr. 8.*

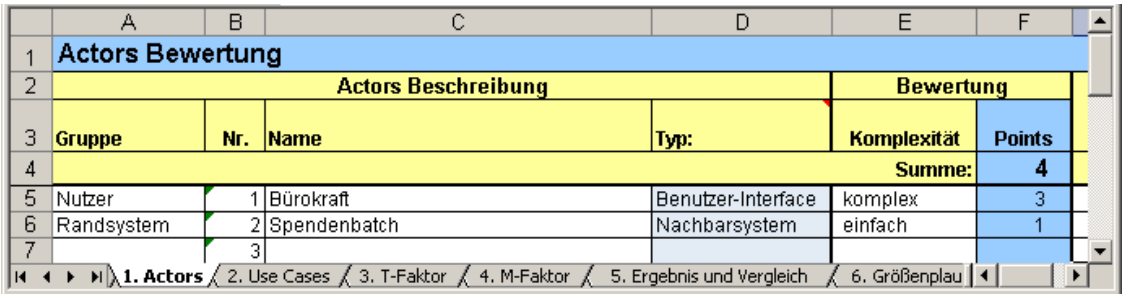
Insgesamt haben wir aus der Spezifikation acht Use Cases mit zusammen 14 Points ermittelt (Abbildung 51, Feld T6).

Aus der bis hierher vorgestellten Spezifikation der *Förderverwaltung* lassen sich insgesamt zwei Actors ableiten:

---

[32] Verschiedene Auslöser des Prozesses werden beschrieben, insbesondere Jahresabschluss oder Spendeneingang.

- 1. Benutzer des Systems (Bürokraft), wobei keine weiteren Rollen unterschieden werden. Dabei handelt es sich um ein Benutzer-Interface, wofür per Definition (Abschnitt 3.1.2, Seite 58) 3 Points (komplex) vergeben werden.
- 2. Randsystem („Spendenbatch“), wobei hier genau genommen abhängig vom Spendenstatus auch die *Zeit* (Jahresende) der Auslöser (siehe Abschnitt 3.1.1, Zeitereignis) des Use Cases und damit der Actor ist. Es handelt sich um einen einfachen Actor (1 Point).



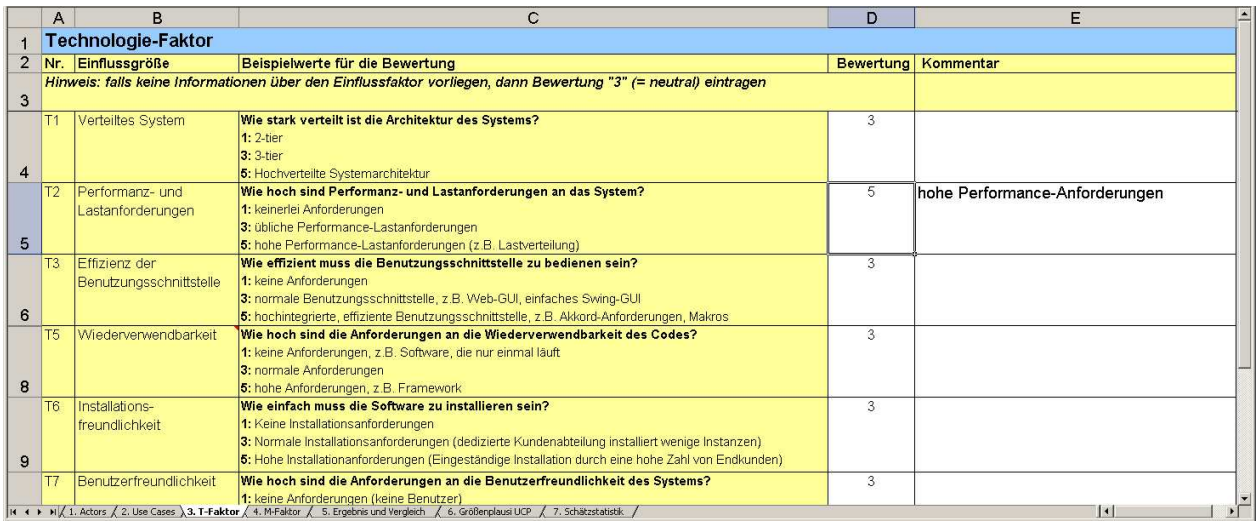
	A	B	C	D	E	F
1	**Actors Bewertung**					
2		**Actors Beschreibung**			**Bewertung**	
3	**Gruppe**	**Nr.**	**Name**	**Typ:**	**Komplexität**	**Points**
4					**Summe:**	**4**
5	Nutzer	1	Bürokraft	Benutzer-Interface	komplex	3
6	Randsystem	2	Spendenbatch	Nachbarsystem	einfach	1
7		3				

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplau

Abbildung 55: Erfassung der Actors im Schätzwerkzeug

Damit ergeben sich insgesamt 4 Points für die Actors (Abbildung 55, Feld F4).

Die Spezifikation enthält noch die Hinweise auf hohe Performance-Anforderungen (Nicht-Funktionale Anforderungen). Daher wird für den T-Faktor die Einflussgröße T2 von Standard-Wert 3 auf 5 (hohe Performance-Anforderungen) geändert. Alle anderen Einflussgrößen verbleiben in der Default-Einstellung (Wert=3). Abbildung 56 zeigt den relevanten Auszug aus dem entsprechenden Dialog im Schätzwerkzeug. Dies führt zu einem Wert von 1,02 für den T-Faktor (Abbildung 58, Feld B6).



	A	B	C	D	E
1	**Technologie-Faktor**				
2	Nr.	Einflussgröße	Beispielwerte für die Bewertung	Bewertung	Kommentar
3	*Hinweis: falls keine Informationen über den Einflussfaktor vorliegen, dann Bewertung "3" (= neutral) eintragen*				
4	T1	Verteiltes System	**Wie stark verteilt ist die Architektur des Systems?** **1:** 2-tier **3:** 3-tier **5:** Hochverteilte Systemarchitektur	3	
5	T2	Performanz- und Lastanforderungen	**Wie hoch sind Performanz- und Lastanforderungen an das System?** **1:** keinerlei Anforderungen **3:** übliche Performance-Lastanforderungen **5:** hohe Performance-Lastanforderungen (z.B. Lastverteilung)	5	hohe Performance-Anforderungen
6	T3	Effizienz der Benutzungsschnittstelle	**Wie effizient muss die Benutzungsschnittstelle zu bedienen sein?** **1:** keine Anforderungen **3:** normale Benutzungsschnittstelle, z.B. Web-GUI, einfaches Swing-GUI **5:** hochintegrierte, effiziente Benutzungsschnittstelle, z.B. Akkord-Anforderungen, Makros	3	
8	T5	Wiederverwendbarkeit	**Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes?** **1:** keine Anforderungen, z.B. Software, die nur einmal läuft **3:** normale Anforderungen **5:** hohe Anforderungen, z.B. Framework	3	
9	T6	Installations-freundlichkeit	**Wie einfach muss die Software zu installieren sein?** **1:** Keine Installationsanforderungen **3:** Normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen) **5:** Hohe Installationanforderungen (Eingeständige Installation durch eine hohe Zahl von Endkunden)	3	
	T7	Benutzerfreundlichkeit	**Wie hoch sind die Anforderungen an die Benutzerfreundlichkeit des Systems?** **1:** keine Anforderungen (keine Benutzer)	3	

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 56: Erfassung des T-Faktors

Der Auftraggeber ist dem Dienstleister durch vorhergehende Projekte gut vertraut. Der Projektumfang ist sehr gut überschaubar, daher plant der Auftraggeber, die Rollen des fachlichen Chefdesigners (FCD) und die des technischen Chefdesigners (TCD) mit der gleichen Person zu beset-

zen, die sich im Kundenkontext als auch in der fachlichen und technischen Fragestellung gut auskennt. Für den M-Faktor M1a und M1b wird der Wert 1 angesetzt, die anderen Einflussgrößen sind mit der normalen Ausprägung 3 gewertet worden, der M-Faktor beträgt daher 0,86 (Abbildung 57 und Abbildung 58, Feld B7).

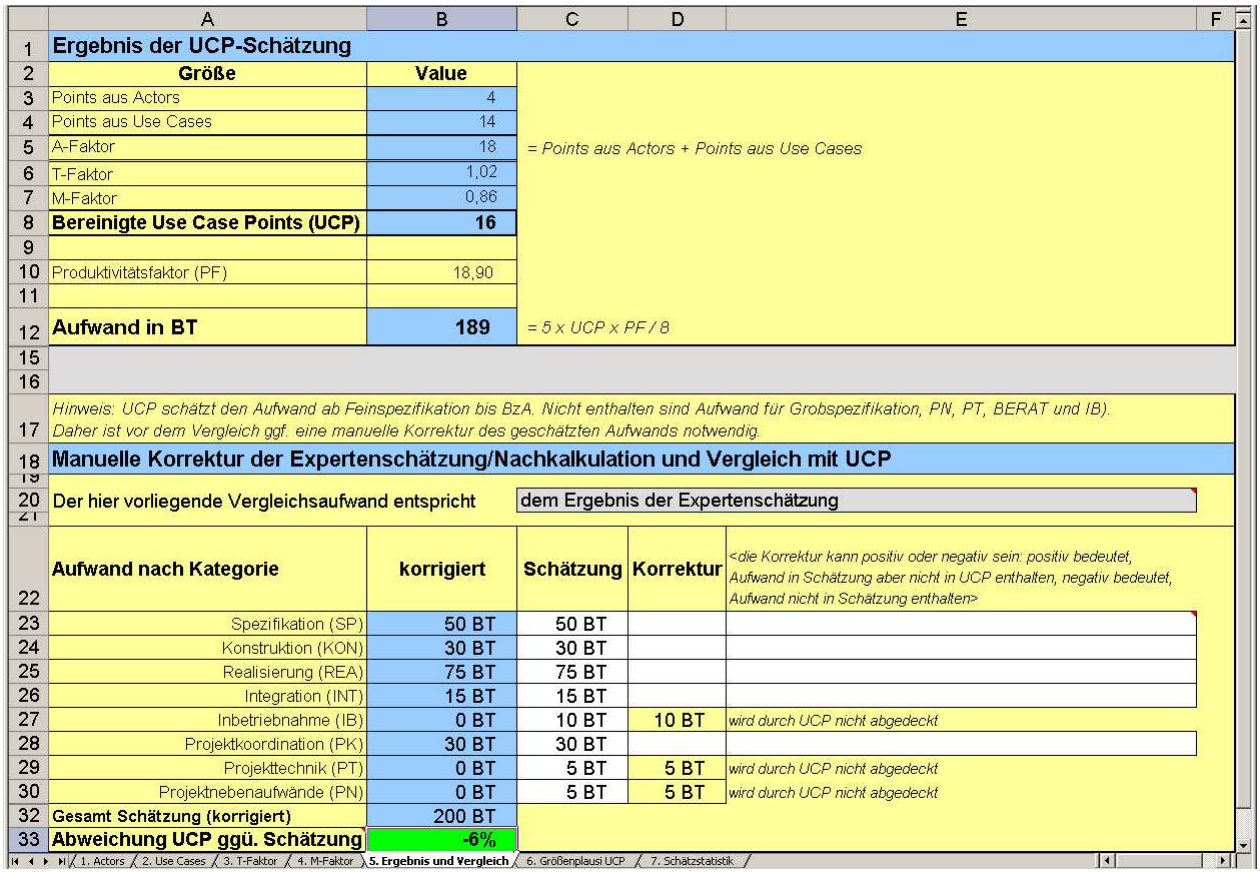
	A	B	C	D	E
1	**Management-Faktor**				
2	**Nr.**	**Einflussgröße**	**Beispielwerte für die Bewertung**	**Bewertung**	**Kommentar**
3	***Hinweis: falls keine Informationen über den Einflussfaktor vorliegen, dann Bewertung "3" (= neutral) eintragen, außer bei M11. Ganzzahlige Werte zwischen 1 und 5 bzw. 1 und 6 (M5, M6) sind als Antwortmöglichkeit zulässig.***				
4	M1a	Leistung/Fähigkeit fachliche Chefdesigner (FCD)	**Wie erfahren ist der fachliche Chefdesigner (FCD) hinsichtlich der im Projekt zu erwartenden Aufgabe?** Die Kollegen sind für das Projekt zum Projektstart verfügbar und fest zugesagt. Falls keine Zusage, dann Punktabzug. *(Für Vergleichbarkeit "Umfeld" zu berücksichtigen: Branchen-Know-How, spezifisches Know-how zum "Thema des Projektes", etc.)*	1	in diesem Fall ist der FCD sehr erfahren, hat bei gleichem Kunden im konkreten Umfeld schon diverse Projekte durchgeführt
5			**1:** sehr erfahren (mind. 4 Projekte als FCD, davon mind. 1 im konkreten Umfeld)		
6			**2:** erfahren (mind. 2 Projekte als FCD, davon mind. 1 im konkreten Umfeld)		
7			**3:** erfahren (mind. 2 Projekte als FCD, keins im konkreten Umfeld)		
8			**4:** geringe Erfahrung im konkreten Umfeld (mind. 1 Projekt als FCD, keins im konkreten Umfeld)		
9			**5:** geringe Erfahrung (generell als FCD, max. 1 Projekt als FCD und keins im konkreten Umfeld)		
10	M1b	Leistung/Fähigkeit technische Chefdesigner (TCD)	**Wie erfahren ist der technische Chefdesigner (TCD) hinsichtlich der im Projekt zu erwartenden Aufgabe?** Die Kollegen sind für das Projekt zum Projektstart verfügbar und fest zugesagt. Falls keine Zusage, dann Punktabzug. *(Für Vergleichbarkeit "Umfeld" zu berücksichtigen: Technologie (z.B. .NET/Java/Host), Infrastruktur (Native/Web, Client-Server, etc.), Tool-Know-How für einzusetzende Produkte, etc.)*	1	in diesem Fall ist der TCD sehr erfahren, hat bei gleichem Kunden im konkreten Umfeld schon diverse Projekte durchgeführt
11			**1:** sehr erfahren im konkreten technischen Umfeld (mind. 4 Projekte als TCD, davon mind. 2 im konkreten technischen Umfeld)		
12			**2:** sehr erfahren (mind. 3 Projekte als TCD, davon mind. 1 im konkreten technischen Umfeld)		
13			**3:** erfahren (mind 2 Projekte als TCD, davon mind. 1 im konkreten technischen Umfeld)		
14			**4:** erfahren (mind 2 Projekte als TCD, keins im konkreten technischen Umfeld)		
15			**5:** geringe Erfahrung (generell als TCD, max. 1 Projekt als TCD und keins im konkreten technischen Umfeld)		
16	M2	Zusammenarbeit	**Wie gut erwarten wir die Zusammenarbeit im gesamten Projekt-Team (Capgemini sd&m, Kunde und Dritte incl. Sublieferanten von Capgemini sd&m und Dienstleister des Kunden)?** Bewerte das Verständnis für Prozesse, die Leitungsfähigkeit der Zusammenarbeit und die gemeinsamen Ziele.	3	
			1: Die genannten Punkte werden vermutlich ausgesprochen gut funktionieren (das Capgemini sd&m-Team kennt sich, mit dem Kunden (bzw. der Fachabteilung des Kunden) wurde schon mindestens ein vergleichbares Projekt durchgeführt, so dass Prozesse vom Kunden gelebt werden, der Kunde hält sich weitestgehend an seine Prozesse;		

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | 5. Ergebnis und Vergleich | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 57: Erfassung des M-Faktors

Abbildung 58 zeigt die Ergebnisdarstellung im Schätzwerkzeug. Die Points aus Actors und Use Cases sind in den Zeilen 3 und 4 zusammengefasst. Der Wert für den T-Faktor und M-Faktor wurde aus den entsprechenden Reitern berechnet und in den Zeilen 6 und 7 zusammengefasst. In Summe ergeben sich 16 bereinigte Use Case Points. Für den Produktivitätsfaktor sind 18,9 Stunden pro Point gemäß Formel 33 (Seite 143) anzusetzen, daraus errechnet sich der Projektaufwand zu 189 Bearbeitertagen (1 BT = 8 Stunden).

Eine Expertenschätzung hat für die *Förderverwaltung* einen vergleichbaren Aufwand von 200 Bearbeitertagen (BT) ergeben, dies ist in Abbildung 58 in Feld B32 vermerkt. Dazu wurde die Expertenschätzung in den Zeilen 23 bis 30 zum Vergleich auf die Konten der Ebene 1 gemäß Abbildung 41 aufgeteilt. Die Aufwände in Summe von 20 BT für Inbetriebnahme (IB, Zeile 27), Projekttechnik (PT, Zeile 29) und Projektnebenaufwände (PN, Zeile 30) sind durch die UCP-Schätzung nicht abgedeckt und werden in dem Vergleichswert der Expertenschätzung in Feld B32 nicht berücksichtigt. Damit ergibt sich eine Abweichung von 6% (Feld B33), um die die UCP-Schätzung niedriger als die Expertenschätzung ausgefallen ist. Beide Schätzungen liegen also dicht beieinander.



	A	B	C	D	E	F
1	**Ergebnis der UCP-Schätzung**					
2	**Größe**	**Value**				
3	Points aus Actors	4				
4	Points aus Use Cases	14				
5	A-Faktor	18	*= Points aus Actors + Points aus Use Cases*			
6	T-Faktor	1,02				
7	M-Faktor	0,86				
8	**Bereinigte Use Case Points (UCP)**	**16**				
9						
10	Produktivitätsfaktor (PF)	18,90				
11						
12	**Aufwand in BT**	**189**	*= 5 x UCP x PF / 8*			
15						
16						
17	*Hinweis: UCP schätzt den Aufwand ab Feinspezifikation bis BzA. Nicht enthalten sind Aufwand für Grobspezifikation, PN, PT, BERAT und IB). Daher ist vor dem Vergleich ggf. eine manuelle Korrektur des geschätzten Aufwands notwendig.*					
18	**Manuelle Korrektur der Expertenschätzung/Nachkalkulation und Vergleich mit UCP**					
20	Der hier vorliegende Vergleichsaufwand entspricht		dem Ergebnis der Expertenschätzung			
22	**Aufwand nach Kategorie**	**korrigiert**	**Schätzung**	**Korrektur**	*<die Korrektur kann positiv oder negativ sein: positiv bedeutet, Aufwand in Schätzung aber nicht in UCP enthalten, negativ bedeutet, Aufwand nicht in Schätzung enthalten>*	
23	Spezifikation (SP)	50 BT	50 BT			
24	Konstruktion (KON)	30 BT	30 BT			
25	Realisierung (REA)	75 BT	75 BT			
26	Integration (INT)	15 BT	15 BT			
27	Inbetriebnahme (IB)	0 BT	10 BT	10 BT	*wird durch UCP nicht abgedeckt*	
28	Projektkoordination (PK)	30 BT	30 BT			
29	Projekttechnik (PT)	0 BT	5 BT	5 BT	*wird durch UCP nicht abgedeckt*	
30	Projektnebenaufwände (PN)	0 BT	5 BT	5 BT	*wird durch UCP nicht abgedeckt*	
32	**Gesamt Schätzung (korrigiert)**	200 BT				
33	**Abweichung UCP ggü. Schätzung**	**-6%**				

1. Actors | 2. Use Cases | 3. T-Faktor | 4. M-Faktor | **5. Ergebnis und Vergleich** | 6. Größenplausi UCP | 7. Schätzstatistik

Abbildung 58: Ergebnisdarstellung im Schätzwerkzeug

6.4 Validierung in der Praxis

Zur gesamthaften Validierung wurde die Methode UCP 3.0 in einem Praxistest anhand von 19 industriellen Software-Entwicklungsprojekten im Kontext betrieblicher Informationssysteme mit in Summe über 275 Bearbeiterjahren an Aufwand überprüft. Die Schätzungen wurden dazu zu einem frühen Zeitpunkt auf Basis einer Grobspezifikation durchgeführt und mit den tatsächlichen Projektaufwänden nach Projektabschluss verglichen.

Für die in dem Praxistest ausgewerteten 19 Projekte wurden während der Projektdurchführung die Aufwände aller Teammitarbeiter täglich in einer Granularität von 15 Minuten in einem Zeiterfassungssystem auf sogenannten *Konten* festgehalten. Für eine Vergleichbarkeit aller Schätzungen ist es entscheidend, dass im Team ein einheitliches Verständnis besteht, welche Aufwände auf welchem Konto verbucht werden. Ein Beispiel: Gehört das tägliche Lesen von Emails zum Projektaufwand oder wird es gesondert erfasst? Wie werden Teambesprechungen verbucht? Hier gibt es kein „richtig“ oder falsch“, es bedarf nur der eindeutigen Festlegung und der Abgrenzung, was für die Methode UCP 3.0 enthalten sein soll. Daher ist zunächst für alle Projekte, deren Aufwände verglichen wurden, ein einheitlicher Kontenrahmen mit Kontierungsregeln gemäß Abschnitt 6.1 definiert worden.

Für die 19 Projekte wurde eine Schätzung gemäß UCP 3.0 mit dem in Abschnitt 6.2 vorgestellten Schätzwerkzeug durchgeführt. Da in Abschnitt 6.3 die Anwendung des Werkzeuges bereits bei-

spielhaft gezeigt wurde, beschränken wir uns hier auf die Darstellung der Ergebnisse und einen Vergleich mit der Karner-Methode. Letztere wurde bereits in Abschnitt 2.4.1 dargestellt. Tabelle 54 zeigt diesen Vergleich zwischen der Aufwandsschätzung gemäß der Karner-Methode und der Methode UCP 3.0 für 19 unterschiedliche Software-Entwicklungsprojekte (sortiert nach aufsteigender Größe).

			UCP nach Karner					UCP 3.0				
Projekt		IST-Aufwand [BT]	UUCP	TCF	EF	PE [BT]	Δ_{Karner}	A-Faktor	T-Faktor	M-Faktor	PE [BT]	$\Delta_{3.0}$
1	Finance 4	444	169	1,03	0,97	573	29%	31	1,00	0,95	507	14%
2	Auto 1	603	227	0,99	0,97	744	23%	38	0,99	1,07	580	-4%
3	Industry 4	606	370	1,05	1,02	1.346	122%	64	1,03	0,68	728	20%
4	Logistics 2	670	231	0,98	0,99	761	14%	38	0,98	1,00	629	-6%
5	Auto 9	671	147	1,08	0,99	531	-21%	23	1,17	1,54	749	12%
6	Auto 3	884	177	1,04	1,03	645	-27%	24	1,17	1,25	750	-15%
7	Logistics 1	906	268	1,16	1,00	1.056	17%	46	1,10	1,26	887	-2%
8	Industry 2	921	221	1,07	1,05	843	-8%	43	1,09	1,01	694	-25%
9	Finance 1	978	141	1,08	1,23	637	-35%	25	1,03	2,05	1.013	4%
10	Auto 2	987	327	1,03	1,06	1.221	24%	57	1,03	1,30	1.066	8%
11	Auto 6	1.038	187	1,01	1,09	703	-32%	32	1,24	1,48	935	-10%
12	Auto 5	2.230	420	1,08	1,08	1.656	-26%	76	1,19	1,59	1.843	-17%
13	Auto 7	2.712	717	1,03	1,03	2.583	-5%	102	1,18	1,80	3.129	15%
14	Industry 3	2.739	954	1,02	1,05	3.470	27%	176	1,02	1,16	2.860	4%
15	Auto 4	5.850	790	1,06	1,09	3.113	-47%	161	1,01	2,08	5.233	-11%
16	Auto 8	6.385	935	1,03	1,14	3.730	-42%	147	1,18	1,69	4.128	-35%
17	Industry 1	6.949	1.717	1,07	1,14	7.116	2%	243	1,03	2,06	7.360	6%
18	Finance 2	9.199	1.705	1,11	1,18	7.586	-18%	339	1,07	2,41	10.603	15%
19	Logistics 3	10.380	2.617	1,12	1,06	10.555	2%	504	1,10	1,83	12.430	20%
Mittelwert:		**2.903**		**1,05**	**1,06**	**2.572**	**0%**	**114**	**1,09**	**1,48**		**0%**
σ:							**38%**					**16%**

Tabelle 54: Vergleich der Schätzergebnisse UCP 3.0 zur Karner-Methode

Die Spalte *IST-Aufwand* nennt den tatsächlich benötigten Projektaufwand nach Projektabschluss in Bearbeitertagen (BT). Die Größen *UUCP*, *TCF* und *EF* der Karner-Methode sind in Abschnitt 2.4.1 definiert worden. Der daraus berechnete Projektaufwand *PE* ist ebenfalls in Bearbeitertagen angegeben, die Abweichung gegenüber dem IST-Aufwand in Prozent wurde mit Δ_{Karner} bezeichnet. Für die Methode UCP 3.0 sind gemäß dem vorherigen Abschnitt die Größen A-Faktor, T-Faktor und M-Faktor angegeben sowie der daraus berechnete Projektaufwand *PE*. Die Abweichung zum IST-Aufwand in Prozent wurde mit $\Delta_{3.0}$ bezeichnet. Zur Vergleichbarkeit der Methoden UCP 3.0 und UCP nach Karner wurde der Produktivitätsfaktor jeweils so kalibriert, dass die Mittelwerte über die Schätzreihen Null ergeben (siehe Nebenbedingung N4 in Tabelle 50, Seite 138).

Insgesamt wurde die Standardabweichung σ der mittleren Abweichung von 38% (Karner-Methode) auf 16% mehr als halbiert. Besonders auffällig ist, dass sehr große Ausreißer vermieden werden. Zum Beispiel wurden die Projektaufwände für Projekt 3 mit der Karner-Methode um

122% überschätzt und mit UCP 3.0 nur noch um 20%. Die größte Unterschätzung beträgt bei der Karner-Methode noch -47%, bei UCP 3.0 nur noch -35%.

Damit wird die Methode UCP 3.0 in der Praxis gesamthaft bestätigt, die Schätzergebnisse sind signifikant besser als die der Karner-Methode. Eine Bewertung der Methode im Kontext der industriellen Praxis sowie im Vergleich zur Karner-Methode erfolgt nun im nächsten Kapitel in Abschnitt 7.2.

7 Zusammenfassung, Bewertung und Ausblick

In diesem Kapitel wird das wissenschaftliche Ergebnis dieser Arbeit zunächst zusammengefasst (Abschnitt 7.1). Dann wird die Methode UCP 3.0 abschließend bewertet (Abschnitt 7.2) und es werden insbesondere die Vorteile gegenüber der ursprünglichen Karner-Methode dargestellt. Dabei werden die *Problempunkte* aus der Aufgabenstellung dieser Dissertation aus Abschnitt 2.7 wieder aufgegriffen. In Abschnitt 7.3 folgen dann einige Gedanken zu möglichen weiterführenden Forschungsschwerpunkten, die sich aus dieser Arbeit ableiten lassen.

7.1 Zusammenfassung

Eine fortschreitende Industrialisierung im Software-Engineering führt in der IT-Industrie zu einem Bedarf nach besseren Schätzmethoden für betriebliche Informationssysteme schon zum Zeitpunkt der Angebotserstellung (Kapitel 1). Daraus erwachsen Anforderungen an Methoden zur Aufwandsschätzung, welche keine der heute bekannten Schätzmethoden vollkommen erfüllen kann (Abschnitt 2.6). In der Praxis ist häufig eine Spezifikationsform mit Anwendungsfällen anzutreffen. Es wäre wünschenswert, aus diesen Anwendungsfällen die zu schätzenden funktionalen Größen unmittelbar ableiten zu können (Abschnitt 2.7). Eine Analyse bekannter Schätzmethoden in Kapitel 2 zeigt, dass die Use Case Points (UCP) Methode diese Anforderung gut erfüllt, jedoch noch einige konzeptionelle Schwächen gegenüber weit verbreiteten Verfahren wie der Function Point Methode oder COCOMO II aufweist.

In dieser Dissertation wurde die UCP-Methode grundlegend überarbeitet und wesentliche konzeptionelle Schwächen beseitigt. Dabei wurden zahlreiche Anleihen und Erkenntnisse aus der experimentellen Physik und der Statistik in das Gebiet des Software-Engineering übertragen.

Zunächst wurde eine neue konzeptionelle Sicht auf die UCP Methode entwickelt (Abschnitt 2.8). Für die Identifikation und Abbildung der Funktionalen Anforderungen einer Spezifikation auf Use Cases wurde in Kapitel 3 ein modellbasiertes Vorgehen definiert, welches unterschiedliche Spezifikationsformen auf eine neu entwickelte UCP-Sprache transformiert. Einzelne Beschreibungen in der UCP-Sprache können dann auf ein Größenmaß (Points) abgebildet werden. Dieses neue UCP-Modell wurde in Kapitel 4 durch eine Feldstudie mit über 200 Einzelschätzungen experimentell validiert.

Für die Nichtfunktionalen Anforderungen wurde in Kapitel 5 aufbauend auf COCOMO II und weiteren Verfahren aus der Literatur und der industriellen Praxis ein neues Kostenfaktor-Modell entwickelt und durch eine Expertenumfrage mit 25 Teilnehmern validiert.

Die Gesamtlösung *UCP 3.0* wurde in einem Praxistest anhand von 19 industriellen Software-Entwicklungsprojekten mit in Summe über 275 Bearbeiterjahren an Aufwand überprüft. Die Schätzungen wurden dazu zu einem frühen Zeitpunkt auf Basis einer Grobspezifikation durchgeführt und mit den tatsächlichen Projektaufwänden nach Projektabschluss verglichen. UCP 3.0 schafft zusammen mit einem Anwendungsleitfaden ein normiertes Verfahren für den Schätzprozess und zeigt in der industriellen Praxis ein hohes Maß an Reproduzierbarkeit mit deutlich ver-

besserter Schätzgenauigkeit. Ein Schätzwerkzeug sowie eine Beispielanwendung ergänzen die im Rahmen dieser Dissertation vorgestellte Lösung.

Die Methode UCP 3.0 hat sich im industriellen Einsatz des Softwarehauses Capgemini sd&m bereits bewährt und wird dort im Quality Gate Prozess seit Dezember 2008 eingesetzt. Es ist ein moderner Lösungsansatz für die Use Case bezogene Aufwandsschätzung von Individualsoftware-Entwicklungsprojekten für betriebliche Informationssysteme. Durch die dieser Dissertation zugrunde liegenden Arbeiten konnte zudem ein breites Praxis-Fundament für die UCP-Methode entwickelt werden. Der zeitliche Aufwand zur Durchführung einer Schätzung wird als gering eingestuft [FJE06].

7.2 Bewertung der Methode UCP 3.0

In Abschnitt 6.4 wurde bereits die Güte der Methode UCP 3.0 anhand der Abweichung zwischen der UCP-Schätzung und den tatsächlichen Projektaufwänden nach Projektabschluss für eine Reihe von 19 Projekten genannt. Dabei wurde eine Standardabweichung als Streumaß der mittleren Abweichung von 16% ermittelt. Die Karner-Methode zeigte eine deutlich schlechtere Güte der Schätzergebnisse mit einer Standardabweichung von 38%. Wir setzten dieses Ergebnis nun in den Kontext der industriellen Anwendung.

Grundlage für die UCP-Schätzung in der Industrie ist eine Grobspezifikation gemäß Abschnitt 2.3.1. In diesem Abschnitt hatten wir für die Schätzunsicherheit eine Streuung zum Zeitpunkt der Grobspezifikation von 40% und weniger angegeben, wie sie aus Expertenschätzungen erzielt wird. Dies steht in Übereinklang mit der bereits in Kapitel 1 genannten mittleren Unterschätzung von großen Projektvorhaben im neuen Umfeld um 40%.

Durch die Methode UCP 3.0 konnte die Schätzgüte also so signifikant verbessert werden, dass die Streuung unterhalb der allgemein erwarteten Streuung der Expertenschätzung von 40% liegt. Damit ist die Methode sehr gut zur unabhängigen Validierung von Expertenschätzungen geeignet, insbesondere auch im Kontext der Quality Gates zum Angebotszeitpunkt (Abschnitt 2.6 und darin insbesondere Abbildung 18).

7.2.1 Vorteile gegenüber der Karner-Methode

In Abschnitt 2.7 wurden insgesamt neun Problempunkte der UCP-Methode nach Karner aufgeführt. Mit der neuen Methode UCP 3.0 wurden diese Probleme nun gelöst. Durch die Überprüfung der Gesamt-Methode in der Praxis (Abschnitt 6.4) wurden die Lösungsansätze für alle Problempunkte ganzheitlich validiert. Nachfolgend fassen wir die Lösungsansätze je Problempunkt nochmals zusammen:

P 1 **Unterschiedlicher Schnitt der Use Cases:** Ein wesentlicher Kritikpunkt an der UCP-Methode war bisher, dass Use Cases in unterschiedlicher Granularität beschrieben werden können und dies unmittelbar Einfluss auf das Schätzergebnis hat. Damit drohte die Gefahr, dass Systeme gleicher fachlicher Größe je nach Spezifikation in unterschiedlichen Use Case Points Werten resultierten.

In Kapitel 3 wurde dazu eine detaillierte Lösung entwickelt. Zunächst wurde eine formale UCP-Sprache entworfen und die Transformation von unterschiedlichen Spezifikationsformen auf diese Sprache beschrieben. In einem zweiten Transformationsschritt wurde genau festgelegt, wie die Komplexität der Use Cases und Actors in Points zu bewerten ist. Diese formale Definition wurde durch einen Leitfaden für die Use Case Identifikation ergänzt und in Kapitel 4 wurde diese modellbezogene Use Case Identifikation validiert. Damit wurde ein praxiserprobter Standard geschaffen, der die Unschärfe der bisherigen UCP-Methode hinsichtlich des Größenverständnisses für Use Cases auflöst.

P 2 **Re-Use Schätzfehler:** Für die UCP-Methode fehlte ein Konzept, wie die Wiederverwendung (Re-Use) von Anwendungsfällen zu berücksichtigen ist. Im Rahmen des Lösungsansatzes zu P 1 wurde dafür ein Konzept entwickelt (Abschnitt 3.2.6) und in Kapitel 4 validiert. Dieses Konzept wurde in das Schätzwerkzeug mit aufgenommen (Abschnitt 6.2).

P 3 **Actor-Gewichte zu gering:** Die Anwendung der Karner-Methode für große Software-Projekte zeigte, dass der Einfluss der Akteure in der UCP-Schätzung vernachlässigbar klein ist. Grund ist die deutlich geringere Gewichtung der Akteure gegenüber den Anwendungsfällen. Es konnte im Rahmen dieser Dissertation gezeigt werden, dass durch ein größeres Gewicht für die Actors eine bessere Schätzgenauigkeit erreicht werden kann. In UCP 3.0 gehen die Actors in etwa fünfmal stärker ein als in der Karner-Methode.

P 4 **Kaum relevante vergleichbare Praxiserfahrung verfügbar:** In der Literatur sind bisher nur wenige Projektdaten zu UCP-Schätzungen veröffentlicht worden. Meistens handelt es sich dabei um kleinere Projekte mit bis zu zwei Bearbeiterjahren. Im Rahmen dieser Dissertation wurde eine Projektdatenbank für die UCP-Schätzung aufgebaut und in Teilen veröffentlicht. Sie enthält die Schätz- und Ist-Daten von 19 abgeschlossenen Projekten in der Größe zwischen 444 und 10.380 Bearbeitertagen Aufwand (insgesamt in Summe über 275 Bearbeiterjahre).

P 5 **UCP Schätzmodell verwendet unzulässige Skalen-Transformationen:** In der Karner-Methode werden unterschiedliche Typen von mathematischen Skalen verwendet und beliebig von einer Skala in eine andere transformiert. Diese konzeptionelle Diskussion ist Mitte der 90er Jahre bereits in die Function Points Methode eingebracht worden und stand für die UCP-Methode noch aus. Insbesondere wurde aus algebraischen Gründen eine Produktformel (Verhältnis-Skalentyp) für die Komplexitätsfaktoren gefordert. Dieser Punkt wurde in UCP 3.0 umgesetzt (Formel 31 und Formel 32).

P 6 **Definition von TCF und EF sind nicht mehr zeitgemäß:** Die Faktoren des EF und TCF der Karner-Methode wurden in der Literatur als nicht mehr zeitgemäß kritisiert. Im Rahmen dieser Dissertation konnte diese Kritik durch eine Expertenumfrage bestätigt werden (Abschnitt 5.5) und ein Ranking der aus heutiger Sicht relevanten Einflussgrößen entwickelt werden. Dabei wurden neuere Erkenntnisse, insbesondere aus COCOMO II, berücksichtigt. Es wurde ein Kostenfaktorbestimmungsverfahren entwickelt, welches der grundsätzlich sehr limitierten Anzahl an verfügbaren Projektdaten im Verhältnis zur Anzahl der prinzipiell möglichen Freiheitsgrade Rechnung trägt. Damit wurde ein neues Kostenfaktor-Modell entwickelt (Kapitel 5).

P 7 **Komplexitätsstufen der TCF und EF sind zu vage definiert:** Für die Kostenfaktoren TCF und EF der Karner-Methode fehlt eine ausreichend präzise und zeitgemäße Definiti-

on, bei welcher Ausprägung je Einflussgröße wie viel Punkte zu vergeben sind. Damit wird das Ergebnis subjektiv und führt in der Konsequenz zu einem Mangel an Reproduzierbarkeit der UCP-Schätzungen. Analog zu den detaillierten Evaluationskriterien im Counting Practise Manual (CPM) der Function Point Methode oder den detaillierten Vorgaben in COCOMO II wurde eine präzise und detaillierte Beschreibung der Ausprägungen jeder Einflussgröße der beiden Kostenfaktoren für UCP 3.0 ausgearbeitet und mit Experten abgestimmt. Für die 19 untersuchten Projekte in der aufgebauten Projektdatenbank gab es keine signifikant unterschiedliche Interpretation der jeweiligen Einflussgrößendefinition.

P 8 **Interpretationsspielraum von Use Cases:** Es war bisher unklar, wie viel Erfahrung für die Durchführung einer UCP-Schätzung tatsächlich benötigt wird und ob die Bewertung von Akteuren und Anwendungsfällen eineindeutig erfolgen wird. Mit Hilfe einer Feldstudie wurde die Reproduzierbarkeit der Methode UCP 3.0 hinsichtlich der Identifikation und Bewertung von Akteuren und Anwendungsfällen eindeutig nachgewiesen, wenn auch die Güte je nach Spezifikationsform variiert. Es konnte statistisch nachgewiesen werden, dass für bestimmte Spezifikationsformen die Erfahrung des Schätzers keinen signifikanten Einfluss hat (Abschnitt 4.5).

P 9 **Umfang der durch UCP abgedeckten Projektaufwände:** Die Karner-Methode nennt nur vage, welche Aufwandsteile genau aus einem Entwicklungsprojekt im Schätzergebnis enthalten sind. Es fehlt ein klarer Aufgabenrahmen der mit der Schätzung abgedeckten Projektaufwände. Für die Methode UCP 3.0 wurde dazu ein Kontenrahmen (Abbildung 41, Seite 144) vorgestellt und detailliert der Umfang der Methode abgegrenzt (Abschnitt 6.1).

7.2.2 Probleme und Grenzen der Methode UCP 3.0

Die Methode UCP 3.0 enthält gegenüber der Karner-Methode keine nennenswerten Nachteile. In diesem Abschnitt brauchen wir daher nur auf die Probleme und Grenzen der UCP-Methode im Allgemeinen einzugehen.

Die Methode UCP 3.0 ist gut geeignet, wenn der Projektinhalt gut durch Anwendungsfälle beschrieben werden kann. Dies ist z.B. der Fall für den Kontext der Individualentwicklung von betrieblichen Informationssystemen (z.B. Stammdaten-Verwaltungssysteme, Auskunftssysteme, etc.) oder allgemein der Neuentwicklung fachlicher Funktionalität. Die UCP-Methode schätzt dabei über alle Phasen des Projektes (Abschnitt 6.1), eine Schätzung z.B. nur einer einzelnen Phase *Spezifikation* ist ohne weiteres nicht möglich. Aufgrund von Erfahrungswerten der Aufwände der Projektphasen zueinander kann jedoch eine Umrechnung auf Projektphasen aus dem Gesamtaufwand erreicht werden.

Im Umkehrschluss ist die Methode UCP 3.0 nicht geeignet für Projekte, deren Umfang kaum oder gar nicht vom fachlichen Spezifikationsumfang abhängt, wie z.B. technische Systemmigrationen (Umstellung auf eine andere Plattform, Datenbank oder allgemein andere Technologie), hardwarenahe Projekte, Studien oder Beratungsprojekte. Ebenso ist sie ohne weiteres nicht anwendbar für Produktanpassungen, Integrationsprojekte oder Wartungsprojekte, da hier der Projektaufwand nicht primär mit dem fachlichen Umfang korreliert, sondern vor allem durch den

Anpassungsaufwand und damit von der Änderungsfreundlichkeit des bereits bestehenden Software-Systems bestimmt wird.

Ferner ist eine gewisse Mindestgröße für das zu Schätzende Projekte notwendig, damit eine statistische Mittelung greift. In Tabelle 23 (Seite 67) hatten wir bereits als untere Grenze für kleine Projekte einen Umfang von 5.000 h = 625 BT (Bearbeitertage) angegeben. Dies entspricht etwa 35 Use Cases[33].

Grundsätzlich ist zu bedenken, dass das Ergebnis eine Schätzmethode nur so gut sein kann wie die zugrunde liegende (Grob-)Spezifikation. Ist die Grobspezifikation lückenhaft, inkonsistent oder widersprüchlich, kann die strukturierte Abbildung auf Use Cases zwar helfen, diese Schwächen zu identifizieren. Überwinden kann UCP 3.0 solche Schwächen aber nicht. In der Praxis wird man in solchen Fällen Prämissen formulieren und als Erläuterung zu der UCP-Schätzung festhalten. Ein Indikator für solche Schwächen der Spezifikation liegt vor, wenn sich Schwierigkeiten beim Identifizieren der Use Cases einstellen. Dann ist möglicherweise die Beschreibung der Spezifikation zu grob oder es fehlen einfach wichtige Informationen.

Zeigt die UCP-Schätzung einen hohen Anteil an Re-Use, kann dies mehrere Gründe haben: Entweder, es handelt sich um ein Wartungs- oder Integrationsprojekt, oder es werden verstärkt Produkte oder Frameworks eingesetzt. In diesen Fällen wird eine hohe Schätzunsicherheit zu erwarten sein, da die UCP-Methode für solche Projekttypen (noch) nicht geeignet ist. Andererseits könnte der hohe Re-Use Anteil auch durch eine starke Kopplung der Use Cases begründet sein oder sehr viele Anwendungsfunktionen werden wiederverwendet, d.h. durch Re-Use berücksichtigt.

Ein hoher Re-Use Anteil in der Größenordnung von 30 % und mehr könnte die Schätzung verfälschen, da die Methode Re-Use nicht exakt genug kalibrieren kann: Die Prozentwerte im Schätzwerkzeug verführen zu einer eher intuitiven Bewertung und basieren nicht auf einer exakten Metrik des *Zählens* wie bei den Schritten, Szenarien und Interaktionselementen. Es empfiehlt sich, wiederverwendete Schritte bzw. Anwendungsfunktionen nur beim ersten Use Case zu zählen. Ist dies nicht möglich und ein hoher Re-Use Anteil nicht zu vermeiden, ist dies ein Indikator für eine weniger gut belastbare UCP-Schätzung.

7.3 Ausblick

Mit dieser Dissertation wurde eine Top-Down Schätzmethode entwickelt, die auf Use Cases basiert und den Anforderungen der Software-Entwicklung betrieblicher Informationssysteme genügt. Damit ist ein wichtiger Beitrag für eine zunehmend industrialisierte Software-Industrie geschaffen worden. Die ursprüngliche UCP-Methode von Karner kann durch die hier präsentierte Methode UCP 3.0 ersetzt werden. UCP 3.0 wird seit Dezember 2008 bei Capgemini sd&m erfolgreich in der Praxis eingesetzt. In ein bis zwei Jahren sollte dann eine deutlich erweiterte Datenbasis mit über 50 Projekten für weitere Untersuchungen dieser Methode vorliegen.

---

[33] Bei dieser Umrechnung lösen wir Formel 29 nach *A-Faktor* auf, setzen die Anzahl Actors auf Null, den T-Faktor = M-Faktor = 1 und gehen von im Mittel 1,5 Points pro Use Case aus (Tabelle 23).

Obwohl die Methode bereits eine hohe Reife zeigt und die Standardabweichung ausreichend gering für einen breiten Einsatz ist, gibt es weitere Verbesserungspunkte, für deren Behebung allerdings eine deutlich größere Anzahl an Projekten notwendig wäre. An oberster Stelle steht dabei die Überarbeitung des T-Faktors, der, wie in Abschnitt 5.7 dargestellt, in weiten Teilen veraltet ist. Hinzu kommt eine genauere Untersuchung der neu eingeführten Einflussgröße T_{15} zur Berücksichtigung von Host-Projekten (Gearingfaktor).

Eine weitere Untersuchung wird für das Konzept des Re-Use empfohlen, insbesondere für den Kontext der Wartung und Weiterentwicklung großer Software-Systeme. Es ist prinzipiell vorstellbar, auch für diesen Projekttyp die UCP-Methode zu adaptieren. Gleiches gilt für den Bereich der Integration von Standard-Software und großen ERP-Einführungsprojekten[34]. Es wird vermutet, dass der Aufwand zur Konfiguration und Einführung solcher Systeme ebenfalls mit der fachlichen Spezifikation der Änderungen korreliert. Hierzu wäre zunächst eine geeignete Projektdatenbank aufzubauen. Die Methoden und Verfahren zur Ableitung des A-Faktors und der Kostenfaktoren sind mit dieser Arbeit bereits gegeben.

Auf Basis einer deutlich größeren Anzahl an Projektschätzungen sollte es ferner möglich sein, eine Aussage zur mittleren Prognose-Genauigkeit einer Einzelschätzung in Abhängigkeit der bewerteten Kostenfaktoren auszusprechen. Es wäre durchaus wünschenswert, als Ergebnis der UCP-Schätzung neben dem Projektaufwand *PE* auch dessen zu erwartende Standardabweichung *ΔPE* (Prognosegüte, also $PE \pm \Delta PE$) mit angeben zu können, wobei die Standardabweichung eine Funktion der Eingangsgrößen der Schätzung ist. So ist z.B. davon auszugehen, dass *ΔPE* empfindlich davon abhängt, ob z.B. für die Einflussgröße M4a (Qualität der Grobspezifikation) die Ausprägung 1 (Spezifikation ist ausreichend detailliert) oder 5 (Spezifikation enthält zahlreiche Widersprüche) gewählt wurde. Der Einfluss auf die Prognosegüte (Streuung) wird möglicherweise für jede Einflussgröße unterschiedlich sein. Für eine solche Erweiterung der UCP-Methode bedarf es jedoch ca. 20 Projekte je Ausprägung jeder Einflussgröße, also ca. 1.000 Projekte alleine für den M-Faktor.

In Abschnitt 4.6 wurde bereits ein Ausblick auf weitere Forschungsfragen im Kontext der Feldstudie zur Reproduzierbarkeit der UCP-Methode gegeben. Dazu zählt die vorgeschlagene Laborstudie zur Untersuchung unterschiedlicher Spezifikationsformen für den identischen fachlichen Projektkontext sowie die Varianzanalyse von Expertenschätzungen auf Basis eines größeren Satzes an Expertenschätzungen.

In dieser Dissertation wurden mit der Feldstudie und der Expertenumfrage nach Kenntnis des Autors erstmals umfangreiche experimentelle Untersuchungen von Aufwandsschätzungen betrieblicher Informationssysteme in der Informatik durchgeführt. Dabei wurden zahlreiche Anleihen und Erkenntnisse aus der experimentellen Physik und der Statistik in das Gebiet des Software-Engineerings übertragen. Sicher stellt dies nur den Grundstein für weitere Untersuchungen mit den hier dargestellten Verfahren dar und weitere Analysen werden folgen, welche die hier vorgestellten Ergebnisse überprüfen und erweitern werden.

---

[34] ERP = Enterprise Resource Planning Systems, z.B. R/3 von SAP

Literaturverzeichnis

[ADS+01] Anda, B.; Dreiem, D.; Sjøberg, D.I.K.; Jørgensen, M.: Estimating software development effort based on use cases - Experiences from industry. In Gogolla, M.; Kobryn, C. (Eds.): *UML 2001 - The Unified Modeling Language*, 4th Int'l Conference, Springer-Verlag LNCS 2185, 2001, pp. 487-502.

[Alb79] Albrecht, A.J.: Measuring Application Development Productivity. In *Proc. IBM Applications Development Symposium*. GUIDE Int. and Share Inc., IBM Corp., Monterey, CA, 1979.

[AN05] Arlow, J.; Neustadt, I.: UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2. ed., Addison-Wesley Professional, 2005.

[And02] Anda, B.: Comparing effort estimates based on use cases with expert estimates. In *Proceedings of Empirical Assessment in Software Engineering (EASE 2002)*, UK, 2002.

[AR94] Abran, A.; Robillard, P.N.: Function Points: A Study of their Measurement Processes and Scale Transformations. In *Journal of Systems and Software*, 25 (1994), pp. 171-184.

[Bad08] Badent, P.: Projektaufwand in Abhängigkeit von Organisation und Vorgehen, Masterarbeit, Hochschule München, Februar 2008.

[Bal00] Balzert, H.: Lehrbuch der Software-Technik, Band 1, Software-Entwicklung, Spektrum Akademischer Verlag, 2. Auflage, 2000.

[Bas07] Basili, V.: The Role of Controlled Experiments in Software Engineering Research, Basili et al. in: *Empirical Software Engineering Issues*, LNCS 4336, pp. 33–37, Springer-Verlag, 2007.

[BD08] Bundschuh, M.; Dekkers, C.: The IT Measurement Compendium, Springer Verlag, 2008.

[BEHS07] Blau, H.; Eicker, S.; Hofmann, A.; Spies, T.: Reifegradüberwachung von Software, ICB-Research Report No. 20, Universität Duisburg-Essen, 2007.

[BF74] Brown, M.B.; Forsythe, A.B.: Robust tests for the equality of variances. In: *Journal of the American Statistical Association* 69, 1974, pp. 364-367.

[BF04] Bundschuh, M.; Fabry, A.: Aufwandsschätzung von IT-Projekten, 2. Aufl., mitp-Verlag, Bonn 2004.

[Boe81] Boehm, B.W.: Software Engineering Economic, Prentice Hall, 1981.

[Boe86] Boehm, B.W.: Wirtschaftliche Software- Produktion, Forkel-Verlag, 1986.

[Boe+00] Boehm, B.W. et al.: Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[Boe07] Boehm, B. W.: Software Engineering, IEEE Computer Society, Los Altimos, 2007.

[Boo94] Booch, G.: Object-oriented analysis and design with applications, 2nd ed., Benjamin/Cummings, Redwood City, 1994.

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modeling Language User Guide, Addison-Wesley Object Technology Series, 1999.

[BS87] Bronstein, I.N.; Semendjajew, K.A.: Taschenbuch der Mathematik, 23. Aufl., Verlag Harri Deutsch, 1987.

[Bug+07] Buglione, L. et al.: Project Sizing and Estimating: A Case Stud using PSU, IFPUG and COSMIC in Software Process and Product Measurement, Dumke, R. et al. Eds., In *Proceedings of International Conferences*, Springer Verlag, 2008.

[BZ07] Basili, V.R.; Zelkowitz, M. V.: Empirical Studies to Build a Science of Computer Science, Comm. of the ACM, 50(2007)11, pp. 33-43.

[Cle05] Clem, R.: Project Estimation with Use Case Points, 2005. http://www.codeproject.com/gen/design/usecasep.asp, 15.03.2007.

[Coc95] Cockburn, A.: Structuring Use Cases with Goals, 1995. http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm, 15.01.2007.

[Coc98] Cockburn, A.: Use Case fundamentals, 1998. http://alistair.cockburn.us/index.php/Use_case_fundamentals, 29.02.2008.

[Coc03] Cockburn, A.: Writing Effective Use Cases. In *The Agile Software Development Series*, 8. Auflage, Addison-Wesley, 2003.

[Coe08] Coe, J.: Improving Consistency of Use Case Points Estimates, University of Alabama, Huntsville, In *CrossTalk, The Journal of Defense Software Engineering*, 2008. http://www.stsc.hill.af.mil/crosstalk/2008/03/0803Coe.html, 10.09.2008.

[Coh05] Cohn, M.: Estimating with Use Case Points, Methods & Tools, Vol. 13, No. 3, 2005. http://www.methodsandtools.com/archive/archive.php?id=25

[DAS05] DASMA (2005): Metrik-Glossar, http://www.dasma.org/contray/html/informationen/metrik-glossar/e, 15.03.2007

[Den91] Denert, E.: Software-Engineering, Springer-Verlag, 1991.

[DH98] Draper, N.R.; Smith, H.: Applied Regression Analysis, New York, Wiley, 1998.

[DS06] Duschinger, H.; Sauer, S.: Leitfaden der sd&m Spezifikationsmethodik für betriebliche Informationssysteme, Capgemini sd&m, internes Dokument, 2006.

[Dum01] Dumke, R.: Softwarequalitätsmanagement, Otto-von-Guericke-Universität Magdeburg, 2001.

[Dum03] Dumke, R.: Software Engineering, 4. Auflage, Vieweg Verlag, 2003.

[Dut05] Dutoit, M.: Aufwandsschätzung und Größenmessung für SW-Entwicklung – MT230, Credit Suisse, internes Dokument, 2005.

[ED07] Ebert, C.; Dumke, R.: Software Measurement – Establish, Extract, Evaluate, Execute. Springer Verlag, 2007.

[Eng08] Engeroff, T.: Analyse der Use-Case-Points-Methode hinsichtlich der zugrunde liegenden Spezifikationsformate. Diplomarbeit, Fachbereich Elektrotechnik und Informationstechnik, Technische Universität Darmstadt, April 2008.

[FAD02] Fetcke, Th.; Abran, A.; Dumke, R.: Eine verallgemeinerte Repräsentation für ausgewählte Functional Size Measurement Methoden In *Current Trends in Software Measurement*, Dumke, R. and Rombach, D. Eds., Deutscher Universitäts Verlag, 2002, pp. 50-75.

[FE08a] Frohnhoff, S., Engels, G.: Revised Use Case Point Method - Effort Estimation in Development Projects for Business Applications. In *Proceedings of CONQUEST 2008, 11th International Conference on Quality Engineering in Software Technology*, Potsdam, Germany, dpunkt.Verlag, 2008.

[FE08b] Frohnhoff, S., Engeroff, T.: Field Study: Influence of Different Specification Formats on the Use Case points Method. In *Software Process and Product Measurement*, Dumke, R. et al. Eds., Proceedings of International Conferences, Springer Verlag, 2008.

[FJ06] Frohnhoff, S; Jung, V.: Abschlussbericht zum 10-Projekte-Programm, internes Dokument der Capgemini sd&m AG, April 2006.

[FJE06] Frohnhoff, S.; Jung, V.; Engels, G.: Use Case Points in der industriellen Praxis. In *Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress*; Abran, A. et al. Eds. Shaker Verlag, 2006, pp. 511-526.

[FKD07] 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.

[Fra07] Fraunhofer IESE: Virtuelles Software Engineering Kompetenzzentrum. http://www.software-kompetenz.de/?4784, 15.03.2007

[Fro08] Frohnhoff, S.: Große Softwareprojekte – Aufwandsschätzung mit ´Use Case Points´. In *Informatik-Spektrum*, Volume 31, Number 6, Bode, A. et al. Eds., Springer Verlag, 2008, pp. 566-575.

[GH96] Garmus, D.; Herron, D.: Measuring the Software Process – A Practical Guide to Functional Measurement, Prentice Hall PTR, 1996.

[HAL+08] Habra, N.; Abran, A; Lopez, M.; Sellami, A.: A framework for the design and verification of software measurement methods. In *Journal of Systems and Software,* Vol. 81, May 2008.

[Har91] Hartung, J.: Statistik, Lehr- und Handbuch der angewandten Statistik. R. 8.Auflage, Oldenbourg Verlag, 1991.

[Hel08] Heltewig, S.: Improving the Use Case Point Method, Diplomarbeit, Fachbereich Informatik, Technische Universität Kaiserslautern, Dezember 2008.

[HGS+05] Habela, P.; Głowacki, E; Serafiński, T; Subieta, K.: Adapting Use Case Model for COSMIC-FFP Based Measurement. In *Proceedings of the 15th International Workshop on Software Measurement*, Montreal, Canada 2005. Shaker Verlag 2005.

[HRZ06] Hericko, M.; Rozman, I.; Zivkovic, A.: A formal representation of functional size measurement methods. In *The Journal of Systems and Software*, 79(2006), pp. 1341-1358.

[Hum04] Humm, B.: Technische Open Source Komponenten implementieren die Referenzarchitektur Quasar. In: *ISOS 2004 - Informationssysteme mit Open Source*, *Proceedings GI-Workshop,* Eirund, H.; Jasper, H.; Zukunft, O., pp. 77-87. Gesellschaft für Informatik, 2004.

[ISO-C03] ISO/IEC 19761:2003, Software Engineering – COSMIC-FFP: A Functional Size Measurement Method, International Organization for Standardization, 2003.

[ISO-F08] ISO/IEC 29881:2008, Software Engineering – FiSMA functional size measurement method version 1.1, International Organization for Standardization, 2008.

[ISO-I03] ISO/IEC 20926:2003, Software Engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual, International Organization for Standardization, 2003.

[ISO-M02] ISO/IEC 20968:2002, Software Engineering – MK II Function Point Analysis – Counting Practices Manual, International Organization for Standardization, 2002.

[ISO-N05] ISO/IEC 24570:2005, Software Engineering – NESMA functional size measurement Method version 2.1 – Definitions and counting guidelines for the application of Function Point Analysis, International Organization for Standardization, 2005.

[IFP03] International Organization for Standardization: Software Engineering – IFPUG 4.1 Unadjusted functional size measurement method –Counting practices manual; http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=35582; 2003.

[JBR99] Jacobson, I.; Booch, G.; Rumbaugh, J.: The unified software development process, Addison-Wesley, 1999.

[JCJO93] Jacobson, I.; Christerson, M.; Jonsson, P.; Overgaard, G.: Object-Oriented Software Engineering. A Use Case Driven Approach, 4. Auflage, Addison-Wesley, 1993.

[Jer03] Jerala, S.: Aufwandsschätzung, iZB Soft, internes Dokument, 2003.

[JFB03] Junior, O.; Farias, P.; Belchior, A.: Fuzzy Modeling for Function Point Analysis. In *Software Quality Journal*, 1(2003), pp. 149-166.

[Kar93] Karner, K: Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. No. LiTHIDA-Ex-9344:21. December 1993.

[Kar93a] Karner, Gustav: Resource Estimation for Objectory Projects, Objective Systems SF AB, September 1993.

[Koi04] Koirala, S.: How to Prepare Quotation Using Use Case Points, 2004. http://www.codeproject.com/gen/design/usecasepoints.asp , 15.01.2007.

[Kru03] Kruchten, P.: The Rational Unified Process: An Introduction, 3rd ed., Addison-Wesley, 2003.

[Lam07] Lamprecht, B.: Umfrage für UCP Kostenfaktoren, internes Dokument, Capgemini sd&m AG, 2007.

[LD01] 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, 2001, pp. 228-267.

[Lil00] Lilja, D.J.: Measuring Computer Performance: A Practitioner's Guide, Cambridge University Press, 2000.

[Lus02] Lusti, M.: Data Warehousing and Data Mining: Eine Einführung in entscheidungsunterstützende Systeme, 2. Auflage, Springer Verlag, 2002.

[MAC05] Mohaghedi, P.; Anda, B.; Conradi, R.: Effort Estimation of Use Cases for Incremental Large-Scale Software Development, ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA. http://www.idi.ntnu.no/grupper/su/publ/parastoo/icse05-ucpeffort-25may05.pdf.

[McC06] McConell, S.: Software Estimation, Microsoft Publication, 2006.

[OA06] Ouwerkerk, J.; Abran, A.; An Evaluation of the Design of Use Case Points (UCP). In: *Mensura 2006 – International Conference on Software Process and Product Measurement*, Cadiz, Spain 2006.

[PAC07] Capgemini: PAC's market segmentation; evolution of the market for custom software development in German-speaking Europe. Interne Marktanalyse, 2007.

[Park92] Park, R. E.: Software Size Measurement: A Framework for Counting Source Statements, Technical Report CMU/SEI-92-TR-020, Carnegie Mellon University, 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html

[PB05] Poensgen, B.; Bock, B.: Function-Point-Analyse, dpunkt.Verlag, 2005.

[PM92] Putnam, L.H.; Myers, W.: Measures for Excellence: Reliable Software on Time, within Budget, Yourdon Press, 1992.

[PMI04] Project Management Institute, A Guide to the Project management Body of Knowledge, 3rd edn., (2004), ANSI/PMI 99-001-2004, ISBN 1-930699-45-X.

[Pre01] Prechelt, L.: Kontrollierte Experimente in der Softwaretechnik: Potenzial und Methode, Springer, Berlin, 2001.

[Pro02] Probasco, L.: Dear Dr. Use Case: What About Function Points and Use Cases? Rational Edge Magazine, August 2002.
http://www-128.ibm.com/developerworks/rational/library/2870.html

[QSM08] Homepage der Firma QSM, 20.12.2008.
http://www.qsm.com/FPGearing.html#MoreInfo

[RQZ07] Rupp, C.; Queins, S. ; Zengler, B. : UML 2 glasklar, 3rd ed., Hanser Verlag, 2007.

[Sch08] Schlittgen, R.: Einführung in die Statistik, 11. überarb. Auflage, Oldenbourg Verlag, 2008.

[SH06] Sachs, L.; Hedderich , J.: Angewandte Statistik – Methodensammlung mit R. Springer Verlag, 12. Auflage, 2006.

[Sie03] Siedersleben, J.: Softwaretechnik – Praxiswissen für Software-Ingenieure, 2. überarbeitete und aktualisierte Auflage, Hanser Verlag, 2003.

[Sie04] Siedersleben, J.: Moderne Software-Architektur, dpunkt.verlag, 2004.

[Smi99] 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

[Sne05] Sneed, H.: Software-Projektkalkulation. Hanser Verlag, 2005.

[Ste99] Steece, B. M.: From Multiple Regression to Bayesian Analysis for COCOMO II. In: *International Society of Parametric Analysts*, 1999.

[SW98] Schneider, G.; Winters, J.P.: Applying Use Cases - A Practical Guide. Addison-Wesley Longman, 1998.

[Tau05] Taubner, D: Software-Industrialisierung. In *Informatik-Spektrum*, Volume 28, Number 4, Bode, A. et al. Eds., Springer Verlag, 2005, pp. 292-296.

[UKS98] UKSMA: Mark II Function Point Analysis - Counting Practices Manual. United Kingdom Software Metrics Association, 1998.

[UM06] Urban, D.; Mayerl, J.: Regressionsanalyse: Theorie, Technik und Anwendung, 2. überarb. Auflage, VS Verlag, Wiesbaden, 2006.

[Wes71] Westphal, W.H.: Physikalisches Praktikum, 13. Aufl., Vieweg Verlag, 1971.

Anhang A1 – Expertenumfrage

Laufende Nr.	Name des Kostenfaktors[35]

Hier findet sich die konkrete Beschreibung des Kostenfaktors. Der Kostenfaktor wurde aus den genannten Schätzmethoden entnommen und bislang nicht auf sd&m angepasst. Bitte verwendet keine Zeit die Kostenfaktoren zu überarbeiten, sondern akzeptiert diese so wie vorgegeben.

Metrik:
Die Metrik wurde aus der Schätzmethode entnommen, aus dem der Kostenfaktor stammt. Im weiteren Verlauf unserer Arbeiten ist geplant die Metriken anzupassen. Die Metrik ist wie folgt aufgebaut:
(a) verringert den Aufwand, gemäß dem Beispiel um - x %
(b) neutral, wirkt sich nicht auf den Aufwand aus
(c) erhöht den Aufwand, gemäß dem Beispiel + x %

Die Antworten sind immer nach dem folgenden Muster aufgebaut und sollen auf der letzten Seite unter Antworten eingetragen werden.

□ Der Kostenfaktor ist bei sd&m-Projekten nicht relevant (oder kommt bei sd&m-Projekten sehr selten vor, ggf. Beispiel angeben).
Beispiel: Das Wetter. Dies ist anzunehmen, da wir alle ein Dach über dem Kopf haben. Diesen Kostenfaktor müsste z. B. ein Bauunternehmen durchaus beachten.

□ Der Kostenfaktor ist vorhanden, befindet sich aber bei allen sd&m-Projekten auf einem identischen Niveau und braucht deswegen nicht erhoben zu werden. Die Metrik ist nicht relevant. Bei der Berechnung des Aufwands geht der Kostentreiber (unabhängig vom Projekt) immer mit einem festen Faktor ein. Die Größe dieses Faktors können wir mit Hilfe der Statistik bestimmen.
Beispiel: Die technische Ausstattung der sd&m-Mitarbeiter (Notebook, Monitor, Telefon), da sich unsere Ausstattung auf einem vergleichbaren Stand befindet.

□ Der Kostenfaktor wirkt sich in unterschiedlichen sd&m-Projekten unterschiedlich stark aus. Ich beurteile die Gewichtung des Kostenfaktors im Vergleich zu den anderen relevanten Kostenfaktoren mit folgendem Faktor:
Wenn ihr das denkt, versucht bitte das Verhältnis zu bestimmen, die Umschreibung mit Worten dient hierbei als Hilfe:

□ ¼-fach	□ ½-fach	□ 1-fach	□ 2-fach	□ 4-fach	□ noch höher
gering	weniger als normal	normal	mehr als normal	stark	sehr stark

Bemerkung:
Bitte hier eine Bemerkung ergänzen, falls die Beschreibung des Kostenfaktors nicht verständlich oder passend ist oder falls die Metrik nicht stimmig ist. Hier kann auch eine Bemerkung zur Gewichtung angegeben werden. Oder z. B.: Der Kostenfaktor ist nur in ganz wenigen sd&m-Projekten relevant, geht hier dann aber sehr stark ein.

[35] In der Umfrage wurde nicht zwischen *Kostenfaktor* und *Einflussgröße* unterschieden, sondern jede *Einflussgröße* als eigener *Kostenfaktor* bezeichnet. Um den Wortlaut der Frage nicht nachträglich zu verändern wird an dieser Stelle der konzeptionell unsaubere Begriff *Kostenfaktor* beibehalten, eigentlich geht es um *Einflussgrößen.*

Persönliche Daten des Befragten

Dein Name: _____

Ungefähre Anzahl der relevanten Projekte, die Du schon geleitet hast und aus denen Erfahrung in Deine Antworten einfließen:

kleine Projekte: _____ (Gesamtgröße bis 5 Mitarbeiter)

mittelgroße Projekte: _____ (Gesamtgröße 6 bis 20 Mitarbeiter)

große Projekte: _____ (Gesamtgröße größer als 20 Mitarbeiter)

Wie lange bist Du schon bei sd&m (Anzahl Jahre): _____

01 Komplexe Geschäftsregeln
Wie komplex sind die Geschäftsregeln und Verarbeitungsprozeduren des Systems? Hierzu zählen Kontrollfunktionen, rechenbetonte Operationen, hardwareabhängige Funktionen und datenabhängige Funktionen.
Metrik: (a) nur einfachste Regeln, keine Berechnungen (b) normale Komplexität (c) sehr komplexe Regeln / Berechnungen
Deine Antwort.
Bemerkung:

02 Schnittstellen
Benötigt das System Schnittstellen über welche Daten- und Kontrollinformationen ausgetauscht werden?
Metrik: (a) keine oder sehr geringe Anforderungen (b) Schnittstellen werden benötigt im normalen Umfang (c) komplexe Schnittstellen müssen mit verschiedenen Techniken/Middleware realisiert werden
Deine Antwort.
Bemerkung:

03 Benutzeroberfläche
Wie hoch sind die Anforderungen an die Benutzeroberfläche des Systems?
Metrik: (a) keine besonderen Anforderungen (b) normale Anforderungen (c) hohe Anforderungen (z. B. GUI-Varianten für Benutzergruppen, Internationalisierung, Fehlertoleranz, Makros, Wizards)
Deine Antwort.
Bemerkung:

04 Flexibilität des Systems
Wie flexibel muss das System sein? Hierzu zählen Regeln, Bedingungen und Abläufe welche durch den Benutzer bzw. Administrator geändert werden.
Metrik

(a) keine Anforderungen, z. B. Software, die nur einmal läuft (b) normale Anforderungen, z. B. Konfigurierbarkeit (c) hohe Anforderungen, z. B. Anpassbarkeit über Templates / Skins, Plugin-Schnittstelle
Deine Antwort.
Bemerkung:

05 Installation
Wie einfach muss die Software zu installieren sein?
Metrik: (a) für die Installation ist eine entsprechend ausgebildete Person notwendig (b) normale Installationsanforderungen (dedizierte Kundenabteilung installiert wenige Instanzen) (c) hohe Installationsanforderungen (Eingeständige Installation durch eine hohe Zahl von Endkunden)
Deine Antwort.
Bemerkung:

06 Review und Architektur
Wie gut ist die Grobspezifikation welche wir vom Kunden erhalten haben? Wurde eine Risikoanalyse bereits durchgeführt? Müssen umfangreiche T-Komponenten erstellt werden? (Der Kostentreiber ist leider so aufgebaut.)
Metrik (a) die Grobspezifikation lässt keine oder nur sehr wenige Fragen offen; eine Risikoanalyse wurde durchgeführt und ergab keine nennenswerte Risiken, die T-Architektur existiert (b) die Grobspezifikation enthält offene Fragen welche mit dem Kunden zu klären sind; eine Risikoanalyse wurde durchgeführt und es sind Risiken bekannt, die T-Architektur kann weitestgehend mit Frameworks erstellt werden (c) die Grobspezifikation enthält teilweise Widersprüche und offene Fragen, für die Klärung dieser Punkte sind mehrere Workshops mit dem Kunden notwendig; eine Risikoanalyse muss durchgeführt werden; es sind umfangreiche Arbeiten notwendig um eine T-Architektur zu erstellen
Deine Antwort.
Bemerkung:

07 Ähnliche Produkte
Wie ähnlich ist die Fachlichkeit des Projekts zu anderen Produkten oder Systemen, welche schon auf dem Markt existieren?
Metrik: (a) ähnliche Systeme gibt es schon auf dem Markt und sind uns teilweise bekannt, die Fachlichkeit von unserem System weicht nicht wesentlich ab (b) eine geringe Ähnlichkeit zu auf dem Markt verfügbaren Systemen lässt sich ableiten, innovative Ansätze werden nicht benötigt oder sind uns zumindest teilweise bekannt (c) ähnliche System gibt es nicht auf dem Markt, wir müssen erst innovative Ansätze entwickeln, ggf. befinden sich relevante Nachbarsysteme oder Hardware aktuell in der Entwicklung
Deine Antwort.
Bemerkung:

08 Betreibbarkeit
Wie leicht muss das Anwendungssystem zu betreiben und zu bedienen sein (von technischer Seite)?
Metrik: (a) der Betreiber kennt sich technisch sehr gut aus und kann deshalb viele Probleme selbst lösen, er benötigt für einfache Fehler keine konkreten Handlungsanweisungen

(b) normale Anforderungen (c) von seitens des Betreibers existieren hohe Anforderungen an die Betreibbarkeit des Systems, für alle möglichen Fehler werden klare Handlungsanweisungen benötigt
Deine Antwort.
Bemerkung:

09 Stabile Anforderungen
Wie stabil sind die Anforderungen an das System?
Metrik: (a) sehr stabile Anforderungen, kein Änderungsmanagement erforderlich (b) normale Änderungsrate, übliches Änderungsmanagement (c) sehr hohe Änderungsrate auch grundlegender Anforderungen
Deine Antwort.
Bemerkung:

10 Erfahrung Werkzeuge
Wie viel Erfahrung hat das Team mit der Programmiersprache (inkl. Bibliotheken) und den Werkzeugen für Analyse und Darstellung von Anforderungen und Entwürfen, Konfigurationsmanagement, Dokumentationsgenerierung, Bibliotheksverwaltung, Programmstil und Formatierung?
Metrik: (a) die meisten Teammitglieder haben bereits ein Projekt mit den selben oder ähnlichen Werkzeugen durchgeführt (b) die Hälfte der Teammitglieder kennt wie Werkzeuge (c) sehr wenige Teammitglieder kennen die Werkzeuge
Deine Antwort.
Bemerkung:

11 Verfügbare Zeit
Steht ausreichend Zeit für die Projektabwicklung zu Verfügung?
Metrik: (a) es steht mehr Zeit zur Verfügung, als wir vom Kunde fordern würden (b) es steht nur 100 % der benötigten Zeit für die Projektabwicklung zur Verfügung (c) da nur 85% der geschätzten Zeit zur Verfügung stehen, muss das Team steil aufgebaut werden
Deine Antwort.
Bemerkung:

12 Erfahrung Plattform/Middleware
Wie viel Erfahrung hat das Team über die zu nutzende Plattform und deren Middleware?
Metrik: (a) die meisten Teammitglieder haben bereits ein Projekt auf dieser oder einer sehr ähnlichen Plattform/Middleware durchgeführt (b) die Hälfte der Teammitglieder kennen die Plattform/Middleware (c) sehr wenige Teammitglieder kennen die Plattform/Middleware
Deine Antwort.
Bemerkung:

13 Dokumentation
Wie aufwändig ist die Dokumentation (auch innerhalb der einzelnen Projektphasen) für uns bzw. als Lieferleistung für den Kunden (soweit im Aufwandsmodell rot eingerahmt)?
Metrik:

(a) der Aufwand für Dokumentation kann gering gehalten werden, z. B. müssen grundlegende Funktionen (Drop-Down-Menu, Sortierung) nicht erklärt werden oder Nutzungskonzepte können entfallen (b) durchschnittlicher Aufwand für die Dokumentation (c) erhöhter Aufwand für die Dokumentation, auch der Kunde fordert mehr Dokumentation als üblich, für jede Besprechung ist eine Niederschrift notwendig welche auch von allen Seiten kommentiert wird
Deine Antwort.
Bemerkung:

14 Projektabwicklung
Wie vertraut ist das Team mit dem sd&m-Vorgehensmodell? (Im Original bezog sich die Frage auf RUP.)
Metrik: (a) fast alle (b) etwa die Hälfte (c) fast niemand
Deine Antwort.
Bemerkung:

15 Erfahrung Fachlichkeit
Wie viel Erfahrung hat das Team (Analysten und Programmierer) mit der Fachlichkeit der zu entwickelnden Software?
Metrik: (a) Anwendung und Umfeld sind dem Team vertraut (typisch bei hohen Release-Nummern und Minor Releases) (b) Anwendung und Umfeld sind dem Team zum Teil bekannt bzw. einem Teil des Teams bekannt (c) komplettes Neuland
Deine Antwort.
Bemerkung:

16 Teamplayer
Wie gut funktioniert das Team? Bewerten Sie die Zusammenarbeit, Kompromissbereitschaft, Erfahrung und gemeinsame Visionen und Ziele aller beteiligten Stakeholder (Auftraggeber, Anwender, Entwickler, Projektmanagement, usw.).
Metrik: (a) ausgesprochen gut, da das Team sich schon länger kennt (b) gut (c) das Team wurde neu gebildet
Deine Antwort.
Bemerkung:

17 Leistung Chefdesigner
Wie viel Leistungsvermögen besitzen die technischen und fachlichen Chefdesigner (nicht Erfahrung)? Hierzu zählen z. B. Analyse- und Designfähigkeit, Effizienz, Gründlichkeit und Kommunikationsfähigkeit.
Metrik: (a) sehr gut (b) gut (c) durchschnittlich
Deine Antwort.
Bemerkung:

18 Leistung Programmierer
Wie viel Leistungsvermögen besitzen die Programmierer (nicht Erfahrung)? Bei der Bewertung sollte nicht jeder Programmierer separat betrachtet werden, sondern die Leistung im Team.
Metrik: (a) sehr gut (b) gut (c) durchschnittlich
Deine Antwort.
Bemerkung:

19 Kontinuität Mitarbeiter
Wie hoch ist die Kontinuität der Mitarbeiter im Projekt?
Metrik: (a) gering (12% / Jahr) (b) normal (24% / Jahr) (c) hoch (48% / Jahr)
Deine Antwort.
Bemerkung:

20 Verteiltes Arbeiten
Wird standortübergreifend gearbeitet und welche Möglichkeit zur Kommunikation stehen zur Verfügung?
Metrik: (a) gleiche Niederlassung (b) national verteilt (c) international verteilt, Kommunikation über Email und Telefon
Deine Antwort.
Bemerkung:

21 Prozessreife
Wie gut und wie effektiv sind die Prozesse zwischen sd&m und Kunde? (Im Original bezog sich die Frage auf den CMM-Level.)
Metrik: (a) sehr effektiv, wenig Aufwand (b) durchschnittlich (c) weniger effektiv, hoher Aufwand (in der Regel bei großen Kunden der Fall)
Deine Antwort.
Bemerkung:

22 Motivation
Wie motiviert ist das Team?
Metrik: (a) ausgezeichnet motiviert (b) motiviert (c) unmotiviert
Deine Antwort.
Bemerkung:

23 Objektorientierung
Wie vertraut ist das Team mit objektorientierter Programmierung?

Metrik: (a) fast alle (b) etwa die Hälfte (c) fast niemand
Deine Antwort.
Bemerkung:

24 Verfügbarkeit Team
Wie hoch ist die Verfügbarkeit der Teammitglieder?
Metrik: (a) alle Teammitglieder > 90% verfügbar (b) einige Teammitglieder < 70% verfügbar (c) signifikanter Anteil des Teams ist wenig verfügbar
Deine Antwort.
Bemerkung:

25 Anzahl Entscheidungsträger
Wie viele IT- und fachliche Ansprechpartner des Kunden sind in Entscheidungen involviert?
Metrik: (a) im Bezug auf die Projektgröße gibt es wenige fachliche und technische Ansprechpartner, Entscheidungen auf Seiten des Kunde werden schnell gefällt (b) im Bezug auf die Projektgröße gibt es durchschnittliche viele fachliche und technische Ansprechpartner (c) im Bezug auf die Projektgröße gibt es überaus viele fachliche und technische Ansprechpartner, Entscheidungen auf Seiten des Kunde benötigen lange
Deine Antwort.
Bemerkung

26 Verteiltes System
Wie stark verteilt ist die Architektur des Systems?
Metrik: (a) einfache Systemarchitektur (b) verteilte Systemarchitektur auf mehren Rechnern (z. B. Applikationsserver, Webserver und Lastverteiler) (c) verteilte Systemarchitektur auf verschiedenen Plattformen
Deine Antwort.
Bemerkung:

27 Performance
Wie hoch sind Performanz- und Lastanforderungen an das System?
Metrik: (a) keinerlei Anforderungen, niedrige Transaktionsrate (b) übliche Performance-Lastanforderungen, nur ein Teil der bei (c) beschriebenen Beispiele trifft zu (c) hohe Performance-Lastanforderungen, hohe Transaktionsrate, z. B. Lastverteilung, Number-Crunching, Stored Procedures müssen erstellt werden
Deine Antwort.
Bemerkung:

28 Flexibilität der Entwicklung
Wie flexibel können wir die Software realisieren bezüglich Anforderungen und Schnittstellen?

Metrik: (a) keine besonderen Anforderungen, z. B. wir können selbst Framework bestimmen (b) entsprechende Vorgaben existieren, Änderungen sind nicht zu erwarten (c) entsprechende Vorgaben wie z. B. Schnittstellendefinitionen sind erst sind in Arbeit, Framework oder Bibliothek sind vorgegeben
Deine Antwort.
Bemerkung:

29 Ausfallsicherheit (Reliabilität)
Wie zuverlässig und ausfallsicher muss die Software sein?
Metrik: (a) gering oder normal (b) hoch (c) hoch, da Risiko für menschliches Leben bestehen könnte
Deine Antwort.
Bemerkung:

30 Datenbankgröße
Wie groß sind die Datenbankobjekte (in Byte) im Verhältnis zu der Größe des Gesamtsystems?
Metrik: (a) eher kleiner Datenbestand (Datenbank [Bytes] / SLOC < 20) (b) mittelgroßer Datenbestand (Datenbank [Bytes] / SLOC zwischen 20 und 500) (c) großer Datenbestand (Datenbank [Bytes] / SLOC > 500)
Deine Antwort.
Bemerkung:

31 Anzahl Clients
Von wie vielen Anwendern in wie vielen (Teil-) Organisationen wird das Anwendungssystem genutzt?
Metrik: (a) relativ wenige (b) durchschnittliche viele (c) extrem viele
Deine Antwort.
Bemerkung:

32 Sicherheitsanforderungen
Wie hoch sind die Sicherheitsanforderungen an das System?
Metrik: (a) keinerlei Anforderungen (b) übliche Sicherheitsanforderungen (Authentifizierung und Autorisierung) (c) hohe Sicherheitsanforderungen (Zertifikate, verschlüsselte Kommunikation, Kryptographie)
Deine Antwort.
Bemerkung:

33 Anforderungen Wiederverwendbarkeit
Wie hoch sind die Anforderungen an die Wiederverwendbarkeit des Codes im selben System oder als Framework in anderen Projekten oder als Lieferleistung an den Kunden?
Metrik: (a) keine Anforderungen, z. B. Software, die nur einmal läuft

(b) normale Anforderungen (c) hohe Anforderungen, z. B. Framework
Deine Antwort.
Bemerkung:

34 Flexibilität der Architektur
Wie flexibel soll die Architektur im Hinblick auf Erweiterungen gestaltet werden?
Metrik: (a) keine Anforderung (b) normal (c) flexibel
Deine Antwort.
Bemerkung:

35 Komplexität Programmiersprache
Wie Effizient ist die Programmiersprache mit welcher das System zu erstellen ist?
Metrik: (a) einfach (z. B. Java mit Standard-Frameworks) (b) normal (z. B. Java mit speziellen Frameworks) (c) komplex, z.B. C++ oder Hochsprachen wenn Anteile von anderen Programmiersprachen enthalten sind
Deine Antwort.
Bemerkung:

36 Anforderungen Portabilität
Wie hoch sind die Anforderungen an die Portabilität des Systems?
Metrik: (a) keine Anforderungen z. B. Software, die nur einmal läuft (b) normale Anforderungen (eine Zielplattform, üblicher Grad an Abstraktion) (c) hohe Anforderungen (z. B. Cross-Platform, Windows + Unix)
Deine Antwort.
Bemerkung:

37 Auslastung Zielumgebung
Wie stark ist die Zielumgebung während dem Betrieb ausgelastet (CPU, Storage-System)?
Metrik: (a) gering, < 50% (b) mittel, 50 bis 90 % (c) hoch, > 90%
Deine Antwort.
Bemerkung:

38 Verfügbarkeitsanforderungen
Wie hoch sind die Verfügbarkeitsanforderungen an das System?
Metrik: (a) keinerlei Anforderungen (b) übliche Verfügbarkeitsanforderungen (c) hohe Verfügbarkeitsanforderungen (24/7-Betrieb, 99,x% Verfügbarkeit, Skalierbarkeit)
Deine Antwort.
Bemerkung:

39 Tausch Hard-/Software
Wie oft wird die Hard- und Software (auch Datenbank oder Middleware) gewechselt?
Metrik: (a) alle 12 Monate oder seltener, kleine Updates monatlich (b) alle 6 Monate, kleine Updates 2-wöchentlich (c) jeden Monat, kleine Updates jede Woche
Deine Antwort.
Bemerkung:

40 Unterstützung durch Werkzeuge
Wie gut ist die Unterstützung der Entwickler durch Werkzeuge und wie gut sind diese in die Prozesse eingebunden?
Metrik: (a) sehr hoch, Tools sind in Prozesse eingebunden (b) üblich (c) gering, z. B. Entwicklung auf Host-System mit halbmanueller Übertragung
Deine Antwort.
Bemerkung:

41 Integrationsabhängigkeit
Wie viele Abhängigkeiten von oder zu Schnittstellen bestehen, welche aktiv gemanagt werden müssen?
Metrik: (a) keine oder wenige (b) durchschnittliche viele (c) sehr viele, wir sind bei Tests auf diese Schnittstellen und deren Ansprechpartner angewiesen
Deine Antwort.
Bemerkung:

An dieser Stelle gibt es die Möglichkeit, weitere Kostenfaktoren aus eigener Erfahrung zu ergänzen.

	Name des Kostenfaktors
Beschreibung des Kostenfaktors	
Metrik: (a) (b) (c)	
Antwort: □ Der Kostenfaktor ist bei sd&m-Projekten nicht relevant (oder kommt bei sd&m-Projekten sehr selten vor, ggf. Beispiel angeben). □ Der Kostenfaktor ist vorhanden, befindet sich aber bei allen sd&m-Projekten auf einem identischen Niveau und braucht deswegen nicht erhoben zu werden. Die Metrik ist nicht relevant. Bei der Berechnung des Aufwands geht der Kostentreiber (unabhängig vom Projekt) immer mit einem festen Faktor ein. Die Größe dieses Faktors können wir mit Hilfe der Statistik bestimmen. □ Der Kostenfaktor wirkt sich in unterschiedlichen sd&m-Projekten unterschiedlich stark aus. Ich beurteile die Gewichtung des Kostenfaktors im Vergleich zu den anderen relevanten Kostenfaktoren mit folgendem Faktor: □ ¼-fach□ ½-fach □ 1-fach □ 2-fach □ 4-fach □ noch höher	
Bemerkung:	