BAMBERGER BEITRÄGE

ZUR WIRTSCHAFTSINFORMATIK UND ANGEWANDTEN INFORMATIK

ISSN 0937-3349

Nr. 104

Die COCOMO-Modelle im Licht der agilen Softwareentwicklung

Daniel Hallmann

October 2018

URN: urn:nbn:de:bvb:473-opus4-532118
DOI: https://doi.org/10.20378/irbo-53211

FAKULTÄT WIRTSCHAFTSINFORMATIK UND ANGEWANDTE INFORMATIK

OTTO-FRIEDRICH-UNIVERSITÄT BAMBERG

Die COCOMO-Modelle im Licht der agilen Softwareentwicklung

Daniel Hallmann

daniel.hallmann@uni-bamberg.de

Abstrakt: Aufwandsschätzungen sind wichtig, um ökonomische und strategische Entscheidungen in der Softwareentwicklung treffen zu können. Verschiedene Veröffentlichungen propagieren das Constructive Cost Model (COCOMO) als ein algorithmisches Kostenmodell, basierend auf Formeln mit objektiven Variablen für Schätzungen in der klassischen Softwareentwicklung (KS). Arbeiten aus der agilen Softwareentwicklung (AS) verweisen auf den Einsatz von erfahrungsbasierten Schätzmethoden und von subjektiven Variablen. Aufgrund der schwachen Operationalisierung im agilen Kontext sind Aussagen über konkrete Ursache- und Wirkungszusammenhänge schwer zu treffen. Hinzu kommt der einseitige Fokus der klassischen und agilen Untersuchungen auf den eigene Forschungsbereich, der nach sich zieht, dass eine Verwendung von Variablen aus COCOMO in der AS unklar ist. Wenn hierzu Details bekannt wären, könnten operationalisierte Variablen aus COCOMO auch in der AS eingesetzt werden. Dadurch wird es möglich, in einer wissenschaftlichen Untersuchung eine Konzeptionierung von konkreten kausalen Abhängigkeiten vorzunehmen – diese Erkenntnisse würden wiederum eine Optimierung des Entwicklungsprozesses erlauben. Zur Identifikation von Variablen wird dazu eine qualitative und deskriptive Arbeit mit einer Literaturrecherche und einer Auswertung der Quellen durchgeführt. Erste Ergebnisse zwischen beiden Welten zeigen dabei sowohl Unterschiede als auch Gemeinsamkeiten. Eine Vielzahl von Variablen aus COCOMO kann in der AS verwendet werden. Inwieweit dies möglich ist, ist von den objektiven und subjektiven Anteilen der Variablen abhängig. Vertreter mit erfahrungsbasiertem Hintergrund wie Analyst Capability (ACAP) und Programmer Capability (PCAP) lassen sich aufgrund von Übereinstimmungen mit personenbezogenen Merkmalen gut in die AS übertragen. Parallel dazu sind Variablen aus dem Prozess- und Werkzeugumfeld weniger gut transferierbar, da konkret die AS einen Fokus auf solche Projektmerkmale ablehnt. Eine Weiterverwendung von Variablen ist damit grundsätzlich unter der Berücksichtigung von gegebenen Rahmenbedingungen möglich.

Keywords: COCOMO, Agile Softwareentwicklung, Größen- und Korrektheitsmetriken, Kostentreiber, Faktoren und Exponenten

Inhaltsverzeichnis

1 Einführung

Zu den wichtigsten Aufgaben innerhalb der Planungsphase von Softwareprojekten gehören Aufwandsschätzungen. Durch die Ermittlung von Umfang und Entwicklungszeit können entsprechende Ressourcen für die Umsetzung eingeplant werden, was wiederum eine Kostenprognose und eine Orientierung für ökonomische und strategische Entscheidungen erlaubt. Je nachdem, wie umfangreich und teuer die Entwicklung ausfällt, können etwa Inhalte ausgeplant und verzögert werden, wenn diese zum gegenwärtigen Zeitpunkt eine niedrigere Priorität aufweisen.

Für die KS ist COCOMO ein etabliertes Schätzverfahren aus dem Bereich der algorithmischen Kostenmodelle, die auch manchmal als parametrische Modelle bezeichnet werden [vgl. Hummel 2011, S. 13]. Für die Berechnung der Aufwände werden mathematische Formeln eingesetzt, die eine Beziehung zwischen Eingabeparametern und Ergebnisgrößen herstellen. Ähnliche Ansätze stellen SLIM [vgl. Putnam 1978], RCA PRICE-S [vgl. PRICE Systems, LCC 2017], SEER-SEM [vgl. Galorath, Inc. 2017] und COBRA [vgl. Briand, El Emam und Bomarius 1998]. Eine Übersicht von Modellen zeigt die Arbeit von Boehm [vgl. 1981, S. 511]. In einer zeitlichen Reihenfolge angeordnet existieren für COCOMO die Versionen COCOMO 81 [vgl. Boehm 1981], Ada COCOMO [vgl. Boehm und Royce 1987], COCOMO 2.0 [vgl. Boehm, Clark u. a. 1995] und COCOMO 3.0 [vgl. Clark 2016]. Letztere befindet sich aktuell in Entwicklung, weshalb bisher lediglich Arbeitsartikel veröffentlicht sind. Ein wesentliches Merkmal des Ansatzes sind Formeln, mit deren Hilfe die Aufwände respektive die Dauer der Projekte ermittelt werden. Der Aufwand für ein Softwareprojekt wird dabei in Man-Months (MM) mit folgender Formel berechnet:

$$MM = a \prod_{1}^{18} cd_i (Size)^b \tag{1}$$

Neben den MM als Darstellung des Aufwands werden in der Formel noch $Size$ als Parameter für die Größe, a als Faktor für die Produktivität, b als Exponent für die Skalierungseffekte und $\prod_1^{18} cd_i$ als erweiterte Einstellmöglichkeit von Projektrahmenparametern über die Kostentreiber (KT) verwendet. Auf Basis der MM kann nun weiterführend die zeitliche Dauer einer Entwicklung kalkuliert werden. Das Ergebnis wird als Development Schedule ($TDEV$) bezeichnet und in Monaten angegeben:

$$TDEV = c(MM)^d \tag{2}$$

Auch hier gibt es in ähnlicher Weise wie bei der MM-Formel einen Faktor c für die Produktivität und einen Exponent d für das Skalierungsverhalten mit gleichem Hintergrund und Funktionsweise. Lediglich die konkreten Werte variieren zwischen beiden Formeln. Parallel zu diesen Gleichungen wird außerdem die Korrektheitsmetrik ‚Percentage of Relative Error' (PRE) innerhalb des Modells eingesetzt, mit dessen Hilfe die Prognosequalität geprüft wird. Hierbei werden die geschätzten und realen Werte nach Abschluss des Projektes verglichen und bei Abweichungen wird das Modellverhalten angepasst. Der Prozess in COCOMO nennt sich Kalibrierung und trägt zur kontinuierlichen Verbesserung der Vorhersagen bei.

In der AS gibt es ebenfalls Techniken für Schätzungen aus dem Bereich der mathematischen Modelle. Beispiele sind das ‚Web Model' (WebMo) [vgl. Reifer 2000], das ‚Resistance Factor Model' (RFM) [vgl. Popli und Chauhan 2014a; Popli und Chauhan 2014b], das ‚Principal Component Analysis Model' (PCAM) [vgl. Garg und Gupta 2015] und das ‚Radial Basis Function Model' (RBFM) [vgl. Moser, Pedrycz und Succi 2007]. Parallel zu diesen Modellen existieren außerdem subjektive Ansätze wie ‚Planning Poker', ‚Analogy' und ‚Expert Judgement' [vgl. Keaveney und Conboy 2006; Nguyen-Cong und Tran-Cao 2013; Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a.

2014]. Auch in empirischen und literaturbasierten Übersichtsstudien – wie die von Usman, Mendes und Börstler [vgl. 2015]; Usman, Mendes, Weidt u. a. [2014] – werden vorkommende Größenmetriken wie Story Points (SP), beeinflussenden KT wie Erfahrungslevel der Teammitglieder und prüfende Korrektheitsmetriken wie ‚Magnitude of Relative Error' (MRE) bestimmt. Nicht zu vergessen sind auch theoretische Studien – etwa von Reifer [vgl. 2000] mit Fokus auf der Konzeption von Modellen, die auch Faktoren und Exponenten beinhalten.

In der AS werden vorwiegend die benannten subjektiven Schätzmethoden für informelle Anforderungen wie User Stories eingesetzt, die auf den Erfahrungen der Personen im Team beruhen. Die Metriken für die Softwaregröße – beispielsweise SP – basieren auf abstrakten Größen, die nicht konkret definiert sind. Je nach Team kann dabei das gemeinsame Bild zu Faktoren wie Aufwand, Zeit, Komplexität und Risiko schwanken. Weiterhin werden Einflussgrößen auf die Aufwände nur benannt und nicht im Detail spezifiziert. Eine Operationalisierung gibt es an dieser Stelle nicht, im Anschluss bleiben die konkreten inhaltlichen Merkmale deswegen unklar. Dadurch sind auch Rückschlüsse auf Auswirkungen im Projekt nur sehr ungenau oder gar unmöglich. In der Praxis ist diese offene Formulierung durchaus praktikabel, wenn nicht sogar förderlich, da schneller agiert werden kann. Für eine genaue wissenschaftliche Untersuchung reicht diese grobe Nennung allerdings nicht aus. Hier werden exakte inhaltliche Definitionen benötigt, die eine Aussage zu Korrelationen zwischen Ursache und Wirkung erlauben.

Wie eingangs beschrieben, veröffentlichte Boehm Untersuchungen zu COCOMO, die den Aufbau mit Formeln, Größen- und Korrektheitsmetriken, Faktoren, Exponenten und KT darstellen. In ähnlicher Weise tragen Studien in der AS ebenfalls den aktuellen Stand der agilen Schätzmodelle zusammen. Zusätzlich existieren vergleichende Studien zu COCOMO [vgl. Menzies u. a. 2017] für eine Validierung von Vor- und Nachteilen zwischen unterschiedlichen Schätzmethoden. Hinzu kommen die Übersichtsstudien in der AS, die die Variablen der Schätzmethoden vergleichend beschreiben. Jedoch fokussieren sich die Autoren bei der Zusammenstellung der Inhalte auf eine Bewertung der Methoden und Variablen innerhalb des klassischen oder agilen Vorgehens. Nicht bearbeitet werden dagegen Aspekte zwischen den unterschiedlichen Entwicklungsparadigmen, womit die Frage weiterhin offenbleibt, inwieweit Variablen von COCOMO auch innerhalb der AS zum Einsatz kommen können.

Sobald jedoch etablierte formale Variablen und deren Operationalisierung aus COCOMO für die AS identifiziert sind, lassen sich diese für wissenschaftliche Arbeiten im agilen Kontext weiterverwenden. Mit den geprüften Variablen ist infolgedessen eine Beschreibung von konkreten kausalen Beziehungen möglich und erlaubt zusätzlich eine Messung und eine statistische Auswertung dieser unter Einhaltung von wissenschaftlichen Qualitätsmerkmalen wie Validität, Reliabilität und Objektivität. Mit den gewonnenen Erkenntnissen kann weiterführend außerdem der agile Prozess gezielt mit Handlungsoptionen optimiert werden, die langfristig die Möglichkeit besserer Ergebnisse bieten. Aufgrund der unklaren Variablenmerkmale, der fehlenden Studien und der Möglichkeit zu wissenschaftlichen Untersuchungen im agilen Kontext soll eine vergleichende Arbeit von COCOMO mit AS durchgeführt werden. Das Schließen der Forschungslücke erfordert dabei die Beantwortung folgender Fragen:

Frage 1 (F1): *Welche messbaren Variablen gibt es in COCOMO und gleichzeitig in der AS?*

Frage 2 (F2): *Welche konkreten Variablen existieren für die Qualität von Anforderungen, die Teamerfahrungen, das gemeinsame Verständnis und die Auswirkungen im Projekt?*

Frage 3 (F3): *Wie wurden generell die Variablen operationalisiert?*

Zur Klärung der Fragen wird im Detail eine qualitative und deskriptive Arbeit durchgeführt. Dabei werden die benötigten Daten mit Hilfe einer Literaturrecherche auf Basis der Schlüsselwörter ‚COCOMO', ‚Cost Model', ‚Cost', ‚Effort', ‚Agile', ‚Agile Development', ‚Agile Software Development' und ‚Agile Softwareentwicklung' über die einschlägigen Datenbanken DBLP, ACM, IEEExplorer und SpringerLink erhoben. Für die gezielten Anfragen der Datenbanken ist darüber hinaus die Verknüpfung der Schlüsselwörter mit dem logischen Operator AND im jeweiligen Format notwendig. Um die Qualität zu steigern, werden die zu untersuchenden Medien zusätzlich auf Zeitschriften- und Konferenzartikel sowie Bücher mit Peer Review beschränkt. Nach einer ersten qualitativen Sichtung und Vorauswahl der Ergebnisse folgten im Anschluss tiefgreifende Textanalysen und Vergleiche mit dem Ziel der Identifikation, der Musterkennung und der Strukturierung von Variablen.

Erste Ergebnisse belegen grundsätzlich die Verwendung von Variablen aus den Formeln von COCOMO in der AS, die in vielen Fällen identisch sind oder über ein Pendant bereits dort verwendet werden. Jedoch muss betont werden, dass diese Verwendung eher die Ausnahme als die Regel darstellt, was sich durch die Verwendung unterschiedlicher methodischer Ansätze ergibt. COCOMO verfolgt einen mathematischen, formalen Ansatz, der sich von dem eher erfahrungsbasierten, informellen Vorgehen der AS unterscheidet. Damit lassen sich Variablen aus COCOMO mit erfahrungsbasiertem und subjektivem Hintergrund (z. B. ACAP, PCAP) eindeutiger weiterverwenden als objektive, prozess- und werkzeugbezogene (z. B. Platform Constraints (PLAT), Process Capability & Usage (PCUS)) Variablen. Eine Entscheidung für oder gegen eine Übernahme zeigt sich auch in der Operationalisierung der Variablen. So forciert COCOMO mit einem formalen Ansatz eine exakte Operationalisierung, die so im personenbezogenen agilen Bereich nicht praktiziert wird, da hier nur durch eine Nennung ein Hinweis entsteht. Grundsätzlich können die Ausdifferenzierungen jedoch übernommen werden, solange eine Prüfung der Variablencharakteristik mit den Merkmalen der AS stattfindet und sich Übereinstimmungen herausstellen. Prinzipiell können damit vollumfänglich Variablen (inkl. Operationalisierung) von COCOMO in die AS übernommen werden, wenn Überschneidungen von gleichen Merkmalen zwischen den unterschiedlichen Welten vorliegen.

Es folgt nun eine detaillierte Reflektion von Bestandteilen aus den Formeln 1 und 2 aus COCOMO im Vergleich zur AS, zur Darstellung von möglichen überführbaren Merkmalen in die neueren Methoden. Dazu wird im Kapitel 2 die Bestimmbarkeit der Softwaregröße mithilfe von Größenmetriken betrachtet. Danach folgt in Kapitel 3 die Auseinandersetzung mit den Faktoren und Exponenten. Die KT als Möglichkeit zum Einbringen von Projektrahmenparametern in die Berechnungen wird im Anschluss in Kapitel 4 analysiert. Zur Qualitätssicherung und Optimierung von Prognosen sind die Korrektheitsmetriken von Vorteil. Eine Gegenüberstellung wird in Kapitel 5 vorgestellt. Im letzten Kapitel 6 werden abschließend die Ergebnisse zusammengefasst und ein Fazit herausgestellt.

2 Metriken für die Berechnung der Softwaregröße

Die Größe ist die primäre Einheit, mit der Aufwand in COCOMO bestimmt wird (siehe Formel 1). Bei COCOMO 81 und Ada COCOMO werden dafür vorzugsweise Source Lines of Code (SLOC) und Delivered Source Instructions (DSI) auf Ebene der Quellcodezeilen verwendet. Diese beiden Einheiten operieren damit auf

der untersten Ebene der Programmerstellungen. Dabei handelt es sich um objektive Messkriterien, die zählbar einen Größenwert ergeben. DSI ist eine von Boehm entwickelte Metrik, die neben der Zeilenberechnung zusätzlich einen abgeschlossenen Auslieferungsaspekt für den Kunden enthält. Diese spezielle Verwendungsform von SLOC konnte so in der AS nicht ermittelt werden. SLOC werden aufgrund ihrer Einfachheit und den damit erzielten guten Ergebnissen allerdings auch bei neueren Entwicklungsmethoden eingesetzt [vgl. Britto, Usman und Mendes 2014; Nguyen-Cong und Tran-Cao 2013; Usman, Mendes und Börstler 2015].

Aus dem objektiven Bereich sind außerdem die Chidamber- und Kemerer-Metriken (CK) [vgl. Chidamber und Kemerer 1994] ergänzend aufzuführen, mit denen sich anhand der Klassen und strukturellen Möglichkeiten von OOP (z. B. Kapselung, Vererbung) die Größe der Software messen lässt. OOP ist heute ein Standardvorgehen und wird sowohl in der KS als auch in der AS verwendet. An dieser Stelle zeigt sich eine konkrete Überschneidung bei COCOMO 2.0 und den agilen Entwicklungsmethoden. Für den Aufbau der historischen Daten wird bei COCOMO der Quellcode von abgeschlossenen Projekten automatisch auf Ebene der SLOC untersucht [vgl. Boehm, Clark u. a. 1995, S. 71]. Dabei werden für die Bestimmung der Größe die Metriken von CK verwendet. In der AS verweisen parallel dazu sowohl die Arbeiten von Alshayeb und Li [vgl. 2003] als auch die von Abrahamsson u. a. [vgl. 2007] auf die CK und stellen deren positive Vorhersagekraft in agilen Projekten heraus.

Weitere Metriken wie Function Points (FP) und Object Points (OP) werden zusätzlich bei COCOMO 2.0 und 3.0 eingesetzt. Hierbei handelt es sich ebenfalls um berechenbare Einheiten, die jedoch ausgehend vom Quellcode auf einer Strukturebene höher ansetzen. Im Detail werden hier abgegrenzte und für den Nutzer sinnvolle Funktionseinheiten mit den Ein- und Ausgaben sowie den enthaltenen Verarbeitungsschritten gezählt. Diese Art der Größenbetrachtung entlang von Funktionen und Objekten ist auch in der AS ein praktikables Vorgehen. Hierbei finden sich sowohl für die FP [vgl. Britto, Usman und Mendes 2014; Garg und Gupta 2015; Lenarduzzi u. a. 2015; Nguyen-Cong und Tran-Cao 2013; Santana u. a. 2011; Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015] als auch für die OP [vgl. Usman, Mendes und Börstler 2015] Ansätze in der Literatur. Das Vorgehen bei den Metriken ist hier das gleiche und wird unverändert in der agilen Welt übernommen.

Trotz positiver Anwendung in der AS sind traditionelle Ansätze nur bedingt auf die neuen agilen Methoden übertragbar [vgl. Reifer 2000]. Die Ursache dafür liegt im Entwicklungsvorgehen zu der Zeit von COCOMO. Damals wurden Systeme vorwiegend nach Anforderungen von Grund auf neu gebaut und die Abschätzung der Größe basierte hauptsächlich auf SLOC und FP. Entgegen diesem Vorgehen werden heute Anwendungen bspw. im Web unter der Verwendung von Vorlagen und Webobjekten (z. B. HTML, Applets, Komponenten) erstellt. Webprojekte basieren auf Designvorlagen, wodurch SLOC die Größe nicht mehr passend abbilden. Außerdem haben sich die Abläufe in den Programmen verändert, da diese heutzutage mehr Aufgaben haben, als nur Eingaben zu Ausgaben transformieren, weswegen FP die Größe ebenfalls unzureichend beschreibt. Bestehende Modelle herzunehmen ist demnach aufgrund der starken Abhängigkeit von der Softwaregröße schwierig. Stattdessen werden also angepasste Größenmetriken benötigt. Deshalb haben sich andere Ansätze wie Web Objects (WO) [vgl. Reifer 2000], Application Points (AP) und Multimedia Points (MP) [vgl. Cowderoy 1999] herausgebildet. Mit der Verbreitung der Metriken in der AS und einer Unterstützung von agilen Methoden in COCOMO 3.0 haben diese auch Einzug in die klassischen Ansätze gehalten. Konkret sind hier die AP als eine Möglichkeit für die Angabe der Größe geplant.

Bei der weiteren Reflektion von Metriken in COCOMO wird man jedoch feststellen, dass wie eben schon angesprochen, die traditionellen und speziell die objektiven Ansätze in der AS nur noch eine untergeordnete Rolle spielen, da dort primär subjektive Metriken auf Basis von Expertenschätzungen eingesetzt werden. Ein erster Ansatz sind dabei die COSMIC Function Points (CFP) [vgl. Desharnais, Buglione und Kocatürk 2011] als Weiterentwicklung der FP. Die Größe wird hier auf Basis der Anforderungen durch geschulte Personen geschätzt. Zentral ist die Bewertung von abgegrenzten Funktionseinheiten aus Nutzersicht, die damit bereits stark angelehnt ist an eine Unterstützung der User Stories. Diese Trends werden jetzt entgegengesetzt nun auch in die Entwicklung von COCOMO 3.0 aufgenommen, womit auch der Support agiler Methoden angedacht ist. Eine wichtige Komponente spielt dabei die Unterstützung der eben genannten CFP.

Eine weitere Metrik im agilen Bereich sind die Use Case Points (UCP) [vgl. Nguyen-Cong und Tran-Cao 2013; Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a. 2014]. Hier werden sogenannte Use Cases geschätzt, die im Gegensatz zu den User Stories Softwarefunktionen in detaillierteren Schritten beschreiben. Für alle Abfolgen werden Größenschätzungen gegeben, die in Summe den Umfang der Funktion darstellen. Auch hier handelt es sich wiederholt um einen subjektiven Ansatz, der nach einer vollständigen Entwicklung und Veröffentlichung von COCOMO 3.0 dort ebenfalls Anwendung finden kann.

Die am meisten unterstützte Metrik in den neuen Entwicklungsmethoden [vgl. Britto, Usman und Mendes 2014; Nguyen-Cong und Tran-Cao 2013; Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a. 2014] sind jedoch die SP [vgl. Cohn 2004]. Diese beschreiben ein teamspezifisches Zeit-, Komplexitäts- und Risikomaß für beispielsweise die Bewertung von User Stories. Die konkreten Faktoren und Ausprägungen können hier zwischen Teams und Projekten variieren. Nicht untypisch sind dabei Abweichungen für Schätzungen von ähnlichen Funktionalitäten über Projektgrenzen hinweg. Auch diese sollen in COCOMO 3.0 integriert werden und als Größenmaß zur Verfügung stehen.

3 Faktoren und Exponenten für die Produktivitäts- und die Skalierungseffekte

Die Formeln von COCOMO zur Berechnung der MM (1) und des $TDEV$ (2) besitzen jeweils einen Faktor a für MM und einen Faktor c für $TDEV$, der den Einfluss auf die Produktivität bestimmt. Hierbei werden erstens der entsprechende Aufwand und zweitens die zeitliche Entwicklungsdauer beeinflusst. Die Werte wurden auf Basis der historischen Daten und der anschließenden Modellkalibrierung ermittelt. Diese unterscheiden sich je nach COCOMO-Version und differieren je nach verwendetem Modus (Organic Mode, Semidetached Mode, Embedded Mode), wobei sich der Wert dabei zwischen $2,4$ und $3,6$ bewegt.

Weiter besitzen die Formeln mit b einen Exponenten für MM und mit d einen für $TDEV$, der den relativen Einfluss auf die Wirtschaftlichkeit der Software in Abhängigkeit von der Softwaregröße beschreibt. Verändert sich somit die Größe der Software, so variiert auch die Wirtschaftlichkeit des Softwareproduktes. Auch hier schwankt der Wert ähnlich wie bei den Faktoren über die Modi in COCOMO 81 beziehungsweise über verschiedene Skalierungsfaktoren (Stable Requirements, Precedentedness, Conformity, Development Flexibility, Architecture/Risk Resolution, Team Cohesion, Process Maturity) auf Basis einer 6-Punkte-Likert-Skala [vgl. Likert 1932] in Ada COCOMO und COCOMO 2.0 im Bereich zwischen $0,32$ und $1,26$. COCOMO 3.0 für Schätzungen in der AS befindet sich aktuell in Entwicklung. Ohne finale Veröffentlichung der dort verwendeten Formeln kann jedoch mit Hilfe der aktuellen Arbeitsartikel von einer ähnlichen Struktur ausgegangen

werden, was eine Weiterverwendung beider Komponenten in der AS nahelegt. Hierdurch ergibt sich eine direkte Möglichkeit, diese auch in anderen agilen Ansätzen weiter fortzuführen.

Für einen weiteren Vergleich der Faktoren und Komponenten mit der AS kommen zwei Ansätze aus der agilen Literatur in Frage. Als erstes handelt es sich dabei um das Framework von Benediktsson und Dalcher [vgl. 2004], das die Formeln von COCOMO als Ausgang für den eigenen Ansatz nutzt; als zweites WebMo als direkter Nachfolger von COCOMO 2.0 für die Webentwicklung. Beim Framework Benediktssons und Dalchers werden die Faktoren und Exponenten aufgeführt und als Anregung genutzt. Sie werden in der weiteren Modellkonstruktion jedoch nicht berücksichtigt, wodurch der Ansatz hier nicht weiter thematisiert werden soll.

Aus diesem Grund liegt an dieser Stelle der Fokus auf einer Betrachtung der Formeln aus WebMo [vgl. Reifer 2000, S. 61]. Für den ersten Teil sind das die Faktoren A und B und für den zweiten Teil die Exponenten $P1$ und $P2$. Dabei kann gleich vorab festgehalten werden, dass diese Komponenten für die gleiche Funktionsweise vorgesehen sind, wodurch hier grundsätzlich eine Übereinstimmung zu COCOMO 2.0 besteht. Somit ist über die Faktoren auch eine Einstellmöglichkeit für Projektrahmenparameter mit Einfluss auf die Produktivität gegeben. Hinzu kommen die Exponenten für die Parametrisierung der Skalierungseffekte. Lediglich aufgrund der Eigenheiten der Webentwicklung kommt es bei konkreten Werten zu unterschiedlichen Ausprägungen. Ähnlich den drei Modi bei COCOMO wird auch bei WebMo eine Unterteilung in vier Bereiche vorgenommen. Je nach Einsatzgebiet der zukünftigen Software (Webshops, Handel- und Verkaufssysteme, B2B-Anwendungen, Informationswerkzeuge) gestalten sich dabei die Werte für die Faktoren sowie die Exponenten unterschiedlich. Was auffällt ist die abweichende Basis der konkreten Einteilung. Diese entspricht bei COCOMO den Eigenschaften der Projektumgebung (z. B. Teamgröße, Erfahrungen, Komplexität) und bei WebMo den Eigenschaften des Industriezweiges (z. B. Einzelhandel, Geldgeschäfte und Börse, Informationsdienstleistungen). Zu jedem Bereich werden Werte definiert, die bei den Faktoren zwischen $1,5$ bis $2,7$ und bei den Exponenten zwischen $1,0$ und $1,05$ liegen.

4 Der Einfluss von Kostentreibern auf die Schätzungen

In COCOMO werden darüber hinaus KT für die Feinabstimmung der Aufwandsberechnung eingesetzt. Damit hat ein Nutzer die Möglichkeit Rahmenparameter im Projekt anzugeben, wodurch eine präzisere Schätzung möglich wird. Der Aufbau erfolgt auf Basis eines Prozesses [vgl. Boehm und Valerdi 2008; Boehm und Wolverton 1980] innerhalb der Gesamtkonzeption von COCOMO. Dabei werden zunächst die KT auf Basis einer Analyse bestehender Modelle definiert, sodass sich eine Grundmenge ergibt. In den nachfolgenden Phasen erfolgt weiter die Operationalisierung und Feinabstimmung der Stufen und der Werte auf Basis von Workshops, Expertendiskussionen, empirischen Datenerhebungen in Softwareprojekten und statistischen Verfahren mit einer Datenauswertung.

Die Einteilung der Stufen wird ähnlich wie bei den Exponenten in Ada COCOMO und COCOMO 2.0 über eine Likert-Skala abgebildet. Die Gestaltung über alle KT ist grundsätzlich gleich aufgebaut und schwankt zwischen vier bis sieben Stufen, was im Umkehrschluss jeweils einer 4- bis 7-Punkte-Einteilung bei Likert entspricht. Eine Person kann damit eine Bewertung von Very Low bis Extra Extra High durchführen und sich an Informationen aus Zahlen, Gleichungen, Prozentangaben, Perzentilen, Zeitangaben und Text orientieren. Im Hintergrund ist jeder Bewertungsstufe eine Gleitkommazahl zugeordnet, die den Einfluss auf den Basisaufwand reflektiert. Hier reicht der Definitionsbereich von $0,61$ bis $1,74$.

Bei einer konkreten Ausführung können die KT einzeln bewertet werden und der ausgewählte Wert fließt in die Gesamtberechnung ein. Dargestellt wird die Berechnung in der Formel 1 mit der Produktbildung $\prod_1^{18} cd_i$. Dabei steht cd_i für jeweils einen spezifischen KT.

Da für diese Bewertung mehrere Informationen notwendig sind, ist die Verwendung erst in späteren Projektphasen und mit detaillierteren Untermodellen vorgesehen. Eingeführt wurden die KT bereits im Intermediate Model von COCOMO 81. Durch diese erfolgreiche Einführung wurde das Konzept ebenfalls in Ada COCOMO, dort auch im Intermediate Model und später in COCOMO 2.0 im Late Estimation Model weitergeführt. Durch veränderte Rahmenparameter in der Softwareentwicklung haben sich die KT im Laufe der Zeit evolutionär weiterentwickelt. Bestehende wurden aktualisiert, neue entsprechend hinzugefügt oder in Folge eines nicht mehr zeitgemäßen Einsatzes auch entfernt.

Bei WebMo werden die KT ähnlich wie bei COCOMO operationalisiert. In erster Linie liegt das an der zeitlichen Nähe zu COCOMO 2.0 und damit dem Trend für parametrische Modelle. Die Einteilung bleibt grundsätzlich so erhalten; lediglich die beiden obersten Stufen Extra High und Extra Extra High werden nicht verwendet, wodurch sich die gesamte Skala auf Very Low bis Very High als 5-Punkte-Likert-Skala beschränkt. Durch das unterschiedliche Einsatzgebiet im Web schwanken außerdem die Gleitkommazahlen aufgrund der Kalibrierung mit neuen Rahmenparametern. Der kleinste Wert entspricht $0,58$, der größte Wert $1,67$, wodurch der Definitionsbereich damit gegenüber COCOMO etwas nach unten verschoben ist. Bei der weiteren Betrachtung der KT im agilen Umfeld wird der Einfluss häufig auf Basis von nicht oder gering quantifizierten Größen bestimmt. Hier entscheiden grobe Anhaltspunkte, was bei einer Bewertung zu beachten ist; anders als bei den festgelegten Quantifizierungsstufen und -werten bei COCOMO. Werden sie bei COCOMO exakt spezifiziert, so bleiben sie in der AS weitestgehend offen. Eine Einschätzung der unterschiedlichen Ansätze legt jedoch nahe, dass der quantitative operationalisierte Ansatz von früher heute weitestgehend entfällt, da jetzt der Mensch stärker in den Fokus rückt und daher eine subjektive Bewertung näherliegt [vgl. Britto, Mendes und Börstler 2015].

Grundsätzlich werden die KT aus COCOMO in vier Bereiche eingeteilt. Hierzu gehören der Produkt-, der Computer-, der Personal- und der Projektbereich. Jeder dieser Bereiche enthält zwischen fünf und sieben KT und variiert in Abhängigkeit von der entsprechenden COCOMO-Version. Der hier dargestellte Ansatz stammt ursprünglich aus COCOMO 81 und damit aus der KS. Im Zuge der Reflektion mit der AS soll hier der grundsätzliche Ansatz validiert und die mögliche Weiterverwendung von KT in die AS geprüft werden. Zur besseren Vergleichsbildung erfolgt eine Eingruppierung der neueren Ansätze in die Bereiche aus COCOMO.

Produktbereich

Im ersten Produktbereich lassen sich KT für die Verlässlichkeit und die Komplexität der Software sowie die Wiederverwendbarkeit von Komponenten identifizieren, die sich in ähnlicher Form auch in der agilen Welt widerspiegeln. Hierzu gehören für die Verlässlichkeit ‚Required Software Reliability' (RELY) und ‚Impact of Software Failure' (FAIL), die nicht funktionale Anforderung für die Bewertung eines Risikos von Fehlerauswirkungen in der Anwendung der Software abbilden. Ein hoher Wert wird hier vergeben, wenn bei Fehlern eine Gefahr für Leib und Leben entsteht. Zum zweiten ist das ‚Product Complexity' (CPLX), mit dem die Komplexität der Software spezifiziert wird. Speziell sind hier Eigenschaften der Anwendung wie die Art der Kontrollflussoperationen, mathematischen Berechnungen und Datenbankzugriffe abgebildet. Im direkten Vergleich mit der AS lassen sich Anknüpfungspunkte für eine Weiterverwendung finden. Ein ers-

ter Ansatzpunkt ist der KT aus WebMo mit dem Namen ‚Product Reliability and Complexity' (RCPX), der sich aus beiden zusammensetzt und für die Produktzuverlässigkeit und -komplexität der Software steht. Auch hier wird bei der Zuverlässigkeit und der Auswahl einer hohen Stufe ein hoher Einfluss mit den genannten Risikofaktoren bestätigt. Die Bedeutung von Fehlern und Nacharbeiten wird unter anderem in der Arbeit von Britto, Usman und Mendes [vgl. 2014] für das agile Entwicklungsvorgehen bekräftigt.

Der Ansatz für die Bewertung von Komplexität ist bei WebMo wieder ähnlich und setzt sich aus verschiedenen Komponenten zusammen. Im Rahmen von COCOMO werden Eigenschaften einer Technologie – wie beispielsweise einer konkreten Datenbank – definiert. Bei WebMo dagegen wurde dieses Vorgehen ersetzt und dafür das Verhalten von unterschiedlichen Ansätzen in diesem Fall von verschiedenen Datenbanken aufgenommen. Ein weiterer Unterschied lässt sich mit einem anderem Fokus beschreiben. In COCOMO wird die einzelne Anwendung auf der Ebene der Operationen (z. B. Kontrollfluss, mathematisch) bewertet. Bei WebMo vergrößert sich diese Sichtweise in Richtung einer Betrachtung von Komponenten (z. B. Client, Server, Webservices), was sich durch die verteilte Natur der Webanwendungen ergibt. Der Einfluss von Komplexität auf die Aufwände taucht nicht nur bei WebMo auf, sondern wird auch von Garg und Gupta [vgl. 2015]; Usman, Mendes, Weidt u. a. [2014] bestätigt.

Ein weiterer KT aus COCOMO lässt sich mit ‚Required Reusability' (RUSE) in vergleichbaren agilen Ansätzen [vgl. Britto, Usman und Mendes 2014; Mendes u. a. 2003] identifizieren. Es handelt sich dabei um die geforderte Wiederverwendbarkeit von Komponenten der Software. Laut den Beschreibungen von COCOMO ist ein erhöhter Aufwand notwendig, um Teile so zu konzipieren, dass diese unverändert oder angepasst in neuen Versionen wiederverwendet werden können. Ein Schlagwort ist hier auch Modularität oder das Baukastensystem, das kleine Änderungen an dedizierten Einheiten erlaubt und dadurch die Anpassungsfähigkeit erhöht. In der AS – konkret in der Webentwicklung – hat sich dieser Trend weiterentwickelt, was sich durch den Einsatz modularer Architekturen (z. B. SOA) sowie von Frameworks und Templates zeigt. Auch aufgrund von hohem Zeitdruck und Grundprinzipien (z. B. Reaktionsfähigkeit) der AS ist es heute Standard, flexible Strukturen und vorgefertigte Komponenten in der Programmierung zu verwenden. Außerdem darf man davon ausgehen, dass sich der Mehraufwand für die Erstellung von wiederverwendbaren Komponenten reduziert oder sogar positiv verändert hat. Denn angesichts der umfangreichen Erfahrungen mit dem etablierten komponentenorientierten Vorgehen haben sich Routinen eingespielt, die eine schnelle und effektive Softwareentwicklung ermöglichen.

Computerbereich

In diesem Bereich sind die zwei Aspekte Laufzeitumgebung und Leistungsfähigkeit von Interesse für Abweichungen in den Entwicklungskosten. Erste KT unter COCOMO sind ‚Platform Constraints' (PLAT) und ‚Platform Volatility' (PVOL), die unter anderem die Eigenheiten der Betriebskomponenten (z. B. OS, DBMS) und der Releases für die Betrachtung der Aufwände abbilden. Konkret betrifft das Themen, die den normalen Betrieb der Anwendungen beeinflussen können. Vergleichbare KT auf der Seite der AS für PLAT sind das ‚Operating System' (OPSY) aus der Arbeit von Garg und Gupta [vgl. 2015] sowie die ‚Platform Difficulty' (PDIF) aus WebMo, ebenfalls für eine Einschätzung der Plattformeigenschaften. Hierbei geht es um Unterschiede im Betriebssystem (z. B. Windows, Linux, MacOS) und deren mögliche Auswirkungen auf die Entwicklung. Die Optimierung für bestimmte Zwecke ist dabei ebenso entscheidend wie Eigenheiten und Einschränkungen. Abweichungen, wie beispielsweise eine Webentwicklung

mit PHP als Programmiersprache auf Windows, kann damit zu einem erhöhten Entwicklungsaufwand führen. Dabei fällt die unterschiedliche Sichtweise zwischen beiden Entwicklungswelten auf. Wird bei COCOMO von einer Systemumgebung ausgegangen und die Nutzung der Ressourcen ausgehend vom Programm thematisiert, so werden in den neueren Entwicklungsmethoden mehr die Eigenschaften von verschiedenen Systemen mit deren Vor-/Nachteilen herausgestellt, die sich unabhängig vom Programm auf dessen Verhalten auswirken.

Hinzu kommt die Variabilität der Komponenten über unterschiedliche Versionen. Sobald hier eine Komponente der Laufzeitumgebung aktualisiert wird, kann dies einen Einfluss auf die Schätzungen der Anforderungen und die Entwicklung nehmen. Die Eigenheiten von PVOL lassen sich für die AS mit PDIF sowie den Arbeiten von Britto, Mendes und Börstler [vgl. 2015] und Tanveer, Guzmán und Engel [vgl. 2016] bestätigen. Bei der Bewertung geht es vorwiegend um die Beurteilung der Server und der Netzwerkverbindungen für etwa Webanwendungen. In diese Bereiche fallen beispielsweise die Aktualisierungszyklen der Systeme und die Verbindungsgeschwindigkeiten zwischen Servern, Datenbanken und eventuell zusätzlichen Schnittstellen. Neben einer ähnlichen Zielstellung gibt es auch Unterschiede. Bei COCOMO wird eine zeitliche Beurteilung von großen und kleinen Änderungen an der Laufzeitumgebung vorgenommen. In der AS geschieht das im Gegensatz dazu über die Variabilität der Updates.

Im Bereich der nicht funktionalen Anforderungen kann auch der KT ‚Execution Time Constraint' (TIME) als Verbindung für einen KT im Bereich der AS ausgewählt werden. Das Pendant hier ist der KT ‚Performance Requirements' (PREQ) [vgl. Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a. 2014]. Bei TIME geht es um die benötige Ausführungszeit von einzelnen Komponenten der Anwendung, die je nach zu erledigenden Aufgaben unterschiedlich ausfallen kann. In Summe wird dadurch auch die Gesamtzeit von einer Nutzeraktion bestimmt, sobald sich diese aus einzelnen Aufgaben zusammensetzt. Gleiche Voraussetzungen werden hier auch durch den neuen KT PREQ abgedeckt, da in der AS die Leistungsfähigkeit ebenfalls einen wichtigen Stellenwert einnimmt. Besonders in der Webentwicklung sind Reaktions- und Antwortzeiten ein entscheidendes Qualitätskriterium, besonders was Absprungraten während des Surfverhaltens von Nutzern anbetrifft.

Personenbereich

Der Personenbereich lässt sich in zwei Unterkategorien aufteilen: zum einen in KT für ein Team und zum zweiten KT für Personen. Bei ersterer spielen Teamaspekte wie die Erfahrungsverteilung und der Zusammenhalt eine Rolle. Bei der zweiten dreht sich alles um den Grad personenbezogener Erfahrungen, bezüglich Technik, Projektdomains, anderer Projekte, Schätzungen und agiles Wissen.

Die ersten beiden KT ‚Analyst Capability' (ACAP) sowie ‚Programmer Capability' (PCAP) in COCOMO beschreiben eine Kosteneinfluss, der von einer unterschiedlichen prozentualen Verteilung von Teamkenntnissen im Bereich Kommunikationsfähigkeiten und technischen sowie kundenspezifischen Wissen ausgeht. Die Einschätzung erfolgt dabei in Perzentilen und thematisiert eine Teambewertung für Erfahrungen. Je mehr Kompetenzen demzufolge prozentual im Team vorhanden sind, desto höher ist der Wert. Beispiele sind hier Kommunikationsfähigkeiten sowie ausreichendes technisches und kundenspezifisches Wissen. Auf agiler Seite lautet das Gegenstück ‚Personnel Capability' (PERS) aus WebMo, das nahe dem Original aus COCOMO die gleiche Verteilung von Fähigkeiten im Team symbolisiert. Identisch zwischen beiden Ansätzen sind Perzentilen als Einheit für die Capability. Unterstützend wirkt hier die ‚Perfect Team Composition' (PTCP) [vgl. Popli und Chauhan 2014b], die die Wichtigkeit einer guten Verteilung von

Kenntnissen im Team in den Fokus setzt. Dies entspricht auch dem Wunsch nach interdisziplinären Teams im agilen Umfeld. Hierbei geht es explizit um die Zusammenstellung von benötigten Fähigkeiten für die erfolgreiche Umsetzung des Projektes. Konkret befürworten etwa die neuen Methoden wie Scrum und XP interdisziplinäre Teams, die mit ihren Mitgliedern alle Bereiche eines Softwareprojekts abdecken können. Hierzu gehören Management, Anforderungsentwicklung, Programmierung, Tests, Administration und Betrieb. Sollten nicht alle Fähigkeiten im Team vorhanden sein, kann es zu Mehraufwand kommen, sobald zusätzlich Personen von außerhalb angefragt und eingebunden werden müssen.

Ein weiterer KT in COCOMO trägt den Namen ‚Stakeholder Team Cohesion' (TEAM), der ebenfalls in der AS verwendet wird. Im Detail wird hier der Zusammenhalt der Auftraggeber innerhalb eines Teams umrissen. Wichtig ist hier z. B. die Vision, die nicht vorhanden oder nicht auf einer gemeinsamen Basis ruhen kann. Parallel dazu stellt die gemeinsame Verfolgung einer Zielstellung einen weiteren entscheidenden Faktor dar. Im negativen Fall präferieren dabei alle ein unterschiedliches Ergebnis. Entgegen diesem Verhalten ist jedoch auch der positive Fall möglich, dass alle zusammen in eine Richtung ziehen – was hier zu bevorzugen ist. Eine konkrete Übereinstimmung liefert hier wieder das WebMo mit einem KT, der unter gleichem Namen auch eine identische Beschreibung vorgibt. Hinzu kommt ein weiterer agiler KT: ‚Lack of Team Cohesion' (LOTC) [vgl. Britto, Mendes und Börstler 2015] bei Entwicklerteams, der konkret einen geringen Teamzusammenhalt beschreibt. Hier ist zum einen der Fall zu berücksichtigen, wenn Projektmitglieder noch wenig bis gar nicht zusammengearbeitet haben. Durch die fehlenden Erfahrungen über die Fähigkeiten im Team, können so untereinander falsche Annahmen zur Arbeitsweise und -geschwindigkeit entstehen, was sich wiederum negativ auf die Schätzungen auswirkt.

Erschwerend für den Teamzusammenhalt sind auch Wechsel im Team, wodurch sich das soziale Gefüge innerhalb der Gruppe ändert. Speziell bei COCOMO wurde hier ein KT mit dem Namen ‚Personal Continuity' (PCON) definiert, der die Volatilität der Teammitglieder umschreibt. Hierbei geht es konkret um eine prozentuale Einschätzung dahingehend, wie viele Personen im Laufe eines Kalenderjahres stabil im Team bleiben. Wichtig ist hier auch der Aspekt, ob und wie viele Kenntnisse im Team weiterhin zur Verfügung stehen. In der AS beeinflussen außerdem Bewegungen im Team die Zusammenarbeit. Für eine Bewertung dieser Einflussgröße wird bei WebMo der KT PERS verwendet. COCOMO definiert hier Prozentangaben, bezogen auf die Teilnehmeranzahl; WebMo dagegen unterscheidet weiterführend die Höhe von Auswirkungen der Veränderungen. Der Aspekt von Bewegungen im Team wird außerdem durch den KT ‚Expected Team Change' (EXTC) [vgl. Popli und Chauhan 2014b] aus der AS bekräftigt. So können durch etwaige Kompetenzverluste auf Basis von Rotationen im Team negative Effekte auf Schätzungen entstehen. Hinzu kommt das Risiko für erhöhte Kosten, das mit Blick auf die Wirtschaftlichkeit vermieden werden sollte. Wichtig zu betonen ist hier noch die differenzierte Sichtweise zwischen COCOMO und AS. Bei COCOMO ergibt sich der Wert für den KT aus der Anzahl von Mitgliedern, die auf ein Jahr gesehen im Team verbleiben. Die Stabilität des Teams ist hier das Kernmerkmal. In der AS wird dagegen der Wert über die Anzahl von Mitgliedern bestimmt, die im selbigen Jahreszeitraum das Team verlassen. Die Volatilität des Teams ist hier das bestimmende Merkmal.

Im Personalbereich lassen sich noch weitere KT aus COCOMO ausmachen, die für eine Verwendung im agilen Umfeld in Frage kommen. Konkret sind diese ‚Application Experience' (AEXP), ‚Platform Experience' (PEXP) und ‚Language and Tool Experience' (LTEX). Entgegen den gerade beschriebenen KT für eine Teambewertung, steht bei diesen die Einschätzung der Eigenschaften von einzelnen Personen im Vordergrund. Wie aus dem Namen ableitbar, werden an diesem Punkt

die technischen Erfahrungen mit der Software, der Laufzeitumgebung, der Programmiersprachen sowie den Werkzeugen kategorisiert. Die Verbindung zwischen AS und COCOMO wird beispielsweise über WebMo mit dem KT ‚Personal Experience' (PREX) in unveränderter Form durch gleiche Bewertungskriterien realisiert. Der KT ‚Individual Personal Capability' (IPCP) [vgl. Čelar, Turić und Vicković 2014] bekräftigt an dieser Stelle die Notwendigkeit von ausreichenden technischen und methodischen Erfahrungen im Team, die demnach für die Aufgabenbewältigung notwendig sind. Konkrete Aspekte sind Soft Skills wie Konzentrations-, Kommunikations-, Entschluss-, Entscheidungs- und Leistungsfähigkeit. Hinzu kommt der Grad einer Lernbereitschaft für neue Methoden und Techniken, die für das Projekt hilfreich sind. Auch die Befähigung zum autonomen Arbeiten wird unter diesen Punkten bewertet. Hilfreich sind die Kenntnisse für Abstimmungen und das Treffen von Entscheidungen etwa bei Architekturfragen und dem Lösen von schwierigen Programmieraufgaben. Zusätzlich wird der Stellenwert für ausreichende Erfahrungen in den Studien [vgl. Azevedo Santos u. a. 2013; Britto, Mendes und Börstler 2015; Lang, Conboy und Keaveney 2013; Tanveer, Guzmán und Engel 2016; Usman und Britto 2016] bestätigt. Damit sind Fähigkeiten und Kenntnisse auch in den agilen Projekten Einflussgrößen für die Aufwände. Zu beachten ist hier, dass der Einfluss von personenbezogenen KT für variierende Erfahrungen an dieser Stelle sowohl für lokale als auch für verteilte Teams gilt.

Essenziell von Vorteil sind auch Erfahrungen mit der Domain des Kunden für ein Verständnis der Prozesse und Zielstellungen. Wenn diese Punkte beachtet werden, können zugeschnittene und passende Anwendungen implementiert werden. Thematisiert und eingruppiert werden Kundenerfahrungen in COCOMO unter den eben beschriebenen KT-Gesichtspunkten aus COCOMO [vgl. Boehm 1981, S. 429]. Bei einer konkreten Betrachtung handelt es sich hier jedoch um rein technische und für die Umsetzung wichtige KT ohne einen konkreten Bezug zum Kundenbereich. Dieses Thema wird eher mit dem KT ACAP aus dem Bereich der Capability abgedeckt, da hier konkret eine Auseinandersetzung mit den Anforderungen stattfindet und somit ein Verstehen der Kundendomain unerlässlich ist. Obwohl kein konkreter KT vorliegt, soll hier die Zuordnung wie von COCOMO vorgesehen übernommen werden. Einen Anknüpfungspunkt in der AS gibt es über den KT ‚Domain Knowledge' (DNKW) [vgl. Britto, Mendes und Börstler 2015; Schweighofer u. a. 2016; Usman, Mendes, Weidt u. a. 2014], der hier die Kenntnisse aus dem geschäftlichen Umfeld des Kunden beschreibt. Neben dem Hintergrundwissen ist es außerdem wichtig, dass die Erfahrungen mit dem Kundenumfeld akkurat [vgl. Schweighofer u. a. 2016] und konsistent [vgl. Azevedo Santos u. a. 2013] sind, da dann über Sprache und Diskussionen weniger falscher Annahmen und Entscheidungen getroffen werden. Neben der Notwendigkeit für ausreichender Erfahrungen der Entwickler, thematisiert die AS zusätzlich auch den Umfang und die Güte von Kenntnissen des Kunden. Ein entsprechender KT ist die ‚Client-Specific Knowledge' (CSKW) [vgl. Britto, Usman und Mendes 2014], nach dem die Kunden alle notwendigen Erfahrungen zum einen aus der eigenen geschäftlichen Domain ziehen und zum anderen ergänzende Kenntnisse zum gewählten Softwareentwicklungsprozess besitzen sollen. Nur so können fachliche Anfragen korrekt beantwortet und eine richtige Umsetzung gewährleistet werden. Zusätzlich sind methodische Kenntnisse von Vorteil, um mit dem Bewusstwerden der eigenen Rolle (z. B. Product Owner aus Scrum) eine zielgerichtete Mitarbeit zu ermöglichen.

Wichtig bei der Betrachtung sind neben den Erfahrungen zum aktuellen Projekt auch diejenigen, die über bereits abgeschlossene Projekte erworben wurden. In COCOMO unterstützen dabei die KT PEXP und LTEX diese historische Sicht, die die Erfahrungen mit der damals verwendeten Plattform sowie der Programmiersprache und den Werkzeugen beschreiben. Diese können losgelöst vom aktuellen

Projekt als übergreifendes Wissen betrachtet werden. In Vergleich dazu ist AEXP für die Abbildung der Kenntnisse mit der konkreten Anwendung vorgesehen und eng mit den dedizierten Inhalten verknüpft und daher nur für eine lokale Verwendung vorgesehen. Reflektiert mit dem Bereich der AS wird hierzu in der Literatur der agile KT ‚Team's Prior Experience' (TPEX) [vgl. Keaveney und Conboy 2006; Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a. 2014] aufgestellt. Konkret geht es dabei um projektübergreifende Kenntnisse (z. B. Risiken, Aufwände, Komplexität) als Hilfestellungen für Schätzungen in aktuellen Projekten.

Außerdem sind Erfahrungen für Schätzmethoden in der AS wichtig, da diese vorwiegend auf Expertenmeinungen beruhen. Der für diesen Themenbereich definierte agile KT ‚Programmer Estimation Capability' (PSCP) [vgl. Keaveney und Conboy 2006; Lang, Conboy und Keaveney 2013] und der KT ‚Developers Estimation Experience' (DVEE) [vgl. Tanveer, Guzmán und Engel 2016] umfassen dabei gezielt die Kenntnisse zu Methoden aus dem Bereich der Softwareanforderungen und der eigentlichen Anforderungen (z. B. Inhalt, Komplexität, Abhängigkeiten, Risiko, Entwicklungszeit). Hier sind demzufolge für eine gute Schätzung Kenntnisse zum methodischen Ablauf, der Zielstellung und der richtigen Anwendung durch alle Teilnehmer wichtig. Denn nur wenn sich alle bei diesen Aspekten einig sind, kann ohne offene Konflikte bezüglich der Methode oder verdeckten Fehlausführungen ein schnelles und einheitliches Ergebnis erreicht werden. Außerdem – wie hier explizit erwähnt – ist es entscheidend, dass neben dem Wissen zum Vorgehen auch inhaltliche Erfahrungen über Schätzungen vorliegen, um keine unnötigen Ungenauigkeiten durch eine unsachgemäße Anwendung einzufügen. Generell können Schätzerfahrungen als KT dem Bereich individuelle Erfahrung in COCOMO zugeordnet werden, wobei jedoch hier explizit keine Thematisierung stattfindet. Dies wird auch deutlich angesichts der unterschiedlichen methodischen Ausgangssituationen. Wird in der AS explizit das Wissen der Personen bei den Expertenschätzungen benötigt, so sind Kenntnisse zu vergangenen Projekten bei COCOMO nicht notwendig, da hier die historischen Projektdaten in Form einer Datenbank vorliegen.

Bezüglich dem Themenfeld der Erfahrungen soll hier außerdem noch auf den KT ‚XP Practices' (XPPR) [vgl. Usman, Mendes, Weidt u. a. 2014] aufmerksam gemacht werden, der den Grad an individuellen Erfahrungen mit dem agilen Vorgehen widerspiegelt. Aufgrund der personenbezogenen Einordnungen ist hier ebenfalls eine Zuordnung in den Erfahrungsbereich von COCOMO möglich. Im Detail umfasst der KT die Erfahrungen mit den agilen Projektmethoden und in diesem Fall konkret XP. Für einen reibungslosen Projektverlauf ist demnach das Wissen zu Aspekten (z. B. Rollen, Meetings, Artefakte) der Projektmethode wichtig. Denn nur, wenn diese bekannt sind und – wenn möglich – automatisiert im Hintergrund ablaufen, kann die Konzentration auf die eigentliche Fertigstellung der Anforderungen gelenkt werden.

Projektbereich

Der letzte Bereich beinhaltet die größte Anzahl an KT und ist in Unterbereiche für die verteilte Zusammenarbeit, die Prozessmodelle, den Zeitplan, die Anforderungen, die Sicherheit und die Werkzeuge in Projekten eingeteilt. Auch hier gibt es wiederholt Überschneidungen zwischen beiden Welten.

In COCOMO wird dem Aspekt der verteilten Zusammenarbeit im Team über die KT ‚Multisite Collocation' (SITE: Collocation) und ‚Multisite Communication' (SITE: Communication) eine Gewichtung beigemessen. Die Ausprägungen des Verteilungsgrads reichen von einem international arbeitenden Team bis hin zu einer ausschließlich lokal entwickelnden Gruppe. Ein weiterer Aspekt wird mit

der Charakteristik der Kommunikationswerkzeuge herausgestellt. Die Bandbreite reicht dabei von der einfachen Telefonnutzung bis hin zu einer umfassenden Multimediaplattform. Mit dem KT ‚Facilities' (FCIL) aus dem WebMo ergibt sich hier eine gute Übereinstimmung zwischen COCOMO und der AS. Bei der Entwicklung hat sich Reifer stark an dem Stand von Boehm orientiert und die Einteilung der Bewertungsstufen sowie die enthaltenen Werte (z. B. Verteilungsgrad des Projektteams) übernommen. Weiterhin ergänzen die KT ‚Process Model of the Sites' (PRMS) [vgl. Usman und Britto 2016], ‚Language Differences' (LNGD) [vgl. Britto, Mendes und Börstler 2015; Britto, Usman und Mendes 2014; Usman und Britto 2016] und ‚Travel' (TRVL) [vgl. Britto, Usman und Mendes 2014] den Aspekt der verteilten Arbeit in agilen Softwareprojekten, der sich erschwerend auf die Zusammenarbeit als Team auswirkt [vgl. Usman, Mendes, Weidt u. a. 2014]. So haben das Kommunikationsmodell, die Anzahl der Standorte, die geographische Distanz, die Unterschiede in den Zeitzonen, die kulturellen Differenzen, die Abweichungen in den Arbeitszeiten und die Reisezeiten einen Einfluss auf die entstehenden Entwicklungskosten.

Weiterhin ist eine gemeinsame Basis bezüglich der methodischen Prozesse innerhalb eines Softwareprojekts für eine zielgerichtete Zusammenarbeit von zentraler Bedeutung. Hinzu kommt die Ausführungsqualität der einzelnen Schritte, damit keine unnötigen Verzögerungen und Fehler entstehen. Ein weiterer Punkt ist die Effizienz des Prozesses in Abstimmung auf die Wertschöpfungskette. Dabei bewegt sich der Korridor bei der Bewertung von konfusen bis hin zu effektiven Prozessen, wobei erstere keine und letztere verstärkt die Zielerreichung unterstützen. Die Abbildung dieser Aspekte werden in COCOMO unter dem KT ‚Process Capability & Usage' (PCUS) zusammengefasst. Einen gleichen Qualitätsanspruch stellt auch die AS, obwohl die Prozesse hier weniger formell sind und auch die Flexibilität innerhalb der Strukturen größer ist. Der hier verwendete KT lautet ‚Capability Maturity Model' (CMMI) [vgl. Garg und Gupta 2015] und stimmt im Namen und Inhalt mit dem bekannten Modell des Software Engineering Institutes überein.

Weiterführend ist im Projektbereich noch ein KT aus COCOMO zu nennen, der in der AS einen vergleichbaren Partner besitzt. Speziell geht es um ‚Required Development Schedule' (SCED) als Einflussgröße für die Manipulation des Projektplans. Der Definitionsbereich reicht hier ausgehend von einem normalen Zeitplan mit keinem zusätzlichen Einfluss bis hin zu einer zeitlichen Kompression sowie Expansion mit jeweils negativem Einfluss auf die Aufwände. An dieser Stelle führt der Versuch, den Zeitplan zu kürzen oder unnötig in die Länge zu ziehen, unweigerlich zu einer Erhöhung der Kosten. Ein konkretes agiles Gegenstück lautet ebenfalls SCED und stammt aus dem WebMo, der exakt die gleiche Zielstellung mit einem identischen Aufbau der Bewertungsstufen abbildet. Eine Kompression sowie eine Ausdehnung des zeitlichen Ablaufs hat hier die gleiche Auswirkung.

Anforderungen verkörpern einen zentralen Aspekt sowohl in klassischen als auch agilen Softwareprojekten. Hierdurch ergibt sich eine umfangreiche Anzahl an KT auf beiden Seiten mit Einflussnahme auf den Projektverlauf. Im Detail spielen dabei die Autoren, die Qualität und die Änderungen an den Dokumenten eine entscheidende Rolle. In der KS schreiben vorzugsweise Analysten die Anforderungen, die AS präferiert dagegen eine direkte Erstellung seitens des Kunden. Der zugehörige agile KT lautet ‚Features Defined by the Customer' (FEDC) [vgl. Azevedo Santos u. a. 2013] und zielt auf die direkte Einbindung des Kunden in den Prozess ab. Sobald der Kontakt hier abreißt, kann es zum Verlust der Synchronität zwischen Anforderungs- und Entwicklungsseite kommen, was wiederum durch Nacharbeiten zu mehr Aufwand führen kann. Außerdem wird die Verantwortung des Kunden durch die Übernahme von Aufgaben gestärkt, was auch zu einer stärkeren Identifikation mit dem Produkt führt.

Die agile Welt forciert zusätzlich auch qualitative Eigenschaften der Anforderungen über die KT ‚Requirements Legibility' (REQL) [vgl. Britto, Usman und Mendes 2014; Usman, Mendes und Börstler 2015], ‚Lack of Clarity in Requirements' (LCYR) [vgl. Popli und Chauhan 2014b; Tanveer, Guzmán und Engel 2016] und ‚User Stories Description Quality' (USDQ) [vgl. Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015]. Dabei geht es um fehlende Anforderungen und die richtige Formulierung dieser. Sollten damit Beschreibungen fehlen, unvollständig, zweideutig oder nicht richtig verfasst sein, kann dies zu Verzögerungen durch Fragen – oder gravierender – zu einer falschen Implementierung führen, was unbedingt vermieden werden sollte.

Als Einflussgröße auf die Aufwände steht außerdem die Aufgabengröße in besonderem Fokus. Dieser Punkt unterscheidet den Entwicklungsaufwand in Abhängigkeit der Inhalte. Da dieser Aspekt sehr fundamental ist, ist es nicht verwunderlich, dass er eine identische Rolle sowohl in COCOMO als auch in der AS spielt. Bedeutsam ist hier die differenzierte Betrachtung zwischen beiden Welten aufgrund unterschiedlicher Sichtweisen. So handelt es sich bei den parametrisierten Modellen um das Basiskriterium, was den Aufwand bestimmt; hier beispielsweise angegeben über die Anzahl von Quellcodezeilen (siehe Kap. 2). In Bezug dazu steht die Größe der agilen Methoden – hier beschrieben mit dem KT ‚Task Size' (TASI) [vgl. Schweighofer u. a. 2016; Usman, Mendes und Börstler 2015; Usman, Mendes, Weidt u. a. 2014] – nicht allein da, sondern reiht sich mit weiteren Merkmalen (z. B. Komplexität, Risiko) in ein kombiniertes Maß ein; hier etwa abgebildet über die SP als teamspezifische Einheit für den Umfang einer Anforderung.

Weiterhin wird der Aspekt Komplexität bezüglich der Abhängigkeiten der Anforderungen konkret mit dem agilen KT ‚Requirements Complexity' (REQX) [vgl. Tanveer, Guzmán und Engel 2016] umschrieben. Im Detail geht es dabei um die Bewertung der Systemkomponenten (z. B. Anzahl, Art), die bei der Umsetzung involviert sind. Denn je nachdem, wie verknüpft die Anforderung ist, werden die Aufwände für eine Integration in das bestehende System höher sein. Diese Verbindungen werden in der AS sowie in COCOMO als Thema behandelt, da hier lediglich die Ebene der Betrachtung (Operationen vs. Komponenten, siehe Unterkap. Produktbereich) unterschiedlich ist. Konkret wird deutlich, dass sich komplexe Arbeitsschritte auf die Genauigkeit der Schätzungen auswirken. Zusätzlich erweitert der KT ‚Dependencies' (DEPD) [vgl. Tanveer, Guzmán und Engel 2016] die Betrachtung von Anforderungen auf der Komplexitätsebene. Auf einer Ebene höher werden dabei die Abhängigkeiten zwischen den einzelnen Vorgaben thematisiert. Sollten dabei Verbindungen bestehen, müssen diese bei den Schätzungen und folglich bei der Umsetzung mit beachtet werden. Dadurch erhöht sich die Schwierigkeit für die Entwickler, eine passende Schätzung zu treffen; besonders wenn nicht klar ist, ob nur die Funktionalität oder auch die abhängigen Funktionalitäten mit geschätzt werden sollen. Weiterführend bedarf es bei der Entwicklung dann eines vermehrten Abstimmungsaufwands, wenn parallel an diesen Inhalten gearbeitet wird. An dieser Stelle ist auch ein Eingriff in das System durch die unterschiedliche Anzahl von Verbindungen entscheidend. Sobald hier mehrere Einheiten involviert sind, kann das Risiko für unentdeckte Problemstellungen steigen. Dabei spielt die Relevanz der verknüpften Einheiten eine besondere Rolle. Droht etwa durch eine fehlerhafte Anbindung ein Ausfall der Anwendung, sollte diese Komponente besonders beachtet werden. Aus dem agilen Umfeld stammt hierzu der KT ‚Impact on Existing System' (IMES) [vgl. Tanveer, Guzmán und Engel 2016], der konkret auf die komponentenorientierten Systeme (z. B. Client, Server, Datenbank) abzielt.

In COCOMO beschreiben weiter die KT ‚Breakage' (BKGE) oder auch ‚Requirements Volatility' (RVOL) die Variabilität der Anforderungen im Projekt, die sich vom Anfang bis zum Abschluss des Projektes verändern können. Einen ähnlichen

KT gibt es auch im agilen Umfeld mit ‚Volatility of Requirements' (VREQ) [vgl. Britto, Mendes und Börstler 2015; Keaveney und Conboy 2006; Popli und Chauhan 2014b; Usman und Britto 2016; Usman, Mendes und Börstler 2015], der die gleiche Aufgabe als Einflussgröße auf die Schätzungen übernimmt. Der Unterschied besteht lediglich in der Grundannahme für die Änderungen. Bei COCOMO wird die Stabilität der Anforderung als Zielstellung vorausgesetzt – im Gegensatz hierzu sind variable Vorgaben im agilen Ansatz der Normalfall und eine Stabilität eher die Ausnahme. In dieses Thema passt auch der KT ‚Changing Requirements from Customers' (CHRC) und ‚New Requirements' (NREQ) [vgl. Schweighofer u. a. 2016] für eine Bewegung in den Anforderungen. Demnach wirkt sich ein spontaner Richtungswechsel im Entwicklungszeitfenster unproduktiv aus, da entsprechend mehr Zeit benötigt wird, um auf die Neuerungen reagieren zu können. Änderungen sind bis zur Planung möglich, danach sind konstante Vorgaben wichtig. Parallel dazu wirken sich auch unkontrolliert einfließende neue Anforderungen negativ für das Projekt aus. Denn dadurch entstehen Abhängigkeiten zu bereits bestehenden Funktionalitäten, die gelöst werden müssen. Hinzu kommt das Risiko für die Verschiebung der Auslieferung für den Kunden, was wiederum den Zeitpunkt bis zur Integration von wertvollem Feedback verzögert. Der zugehörige KT wird als ‚Scope Creep' (SCCP) [vgl. Britto, Mendes und Börstler 2015; Usman und Britto 2016] bezeichnet.

Neben den wechselnden Anforderungen beeinflussen parallel auch fehlende Vorgaben die Schätzungen. Der KT ‚Missing Functional and Non-Functional Requirements' (MIFN) [vgl. Usman und Britto 2016; Usman, Mendes und Börstler 2015] umschreibt dabei diesen Aspekt, der mit den wechselnden Anforderungen wesentlich dafür verantwortlich ist, dass es zu Unterschätzungen von Aufwänden in Projekten kommt. Für die schnellen Wechsel und die Antwort auf Unvollständigkeit sind zusätzlich auch die Reaktionszeiten im Team ausschlaggebend, die durch den KT ‚Time Response' (TRSP) [vgl. Azevedo Santos u. a. 2013] abgedeckt werden. Wichtig ist dabei die schnelle Einplanung der Änderungen mit Unterstützung der iterativen Vorgehensweise der AS.

Ergänzend soll weiter der KT ‚Security Application' (SECU) aus COCOMO genannt werden. Die Sicherheit spielt hier besonders bei der Einstufung der zu entwickelnden Software eine entscheidende Rolle. So existieren unbedenkliche Anwendungen in einem nicht bis wenig kritischen Umfeld, aber auch klassifizierte Anwendungen (Secret oder Top Secret) in kritischen Umgebungen. Besonders wird dieser Aspekt in Ada COCOMO hervorgehoben, da mit Ada beispielsweise für Behörden und das Militär entwickelt wird. Wichtig sind hier vor allem die Verlässlichkeit und Robustheit der Software gegenüber Fehlern oder äußeren Einflüssen wie Angriffen und der Schutz von Daten. Vergleichbare agile KT sind ‚Security' (SECY) in [vgl. Usman, Mendes, Weidt u. a. 2014] und ‚Risk Taking' (RISK) in [vgl. Garg und Gupta 2015]. Hier sind ebenfalls Sicherheits- und Risikopunkte mit Auswirkungen auf die Kosten relevant. Je nachdem, wie umfangreich die Maßnahmen sind, kann der Zeitaufwand erheblich schwanken. So ist davon auszugehen, dass etwa bei einer Webseite für Internetbanking die Vorkehrungen gegen Ausfälle und Fehler stärker ausgebaut sind als dieses bei einem persönlichen Blog der Fall ist.

Zu eingesetzten Werkzeugen in Projekten ergeben sich weiter KT aus COCOMO, die sich in ähnlicher Form in der AS widerspiegeln. Die Eigenheiten der Entwicklungsumgebung und der Programmiersprache bestimmen dabei den Einfluss auf die Kosten. Ein Aspekt ist ‚Use of Software Tools' (TOOL), bei dem es vorrangig um die Einschätzung der eingesetzten Werkzeuge im Projekt geht. Dies können einfache Editoren und Debugger sein, bis hin zu Programmen für eine vollständige Produktionsprozessunterstützung von der Planung, die über die Programmerstellung bis hin zu den Tests und dem Betrieb reichen. Aus dem agilen Feld erzeugt

WebMo hier wiederholt einen guten Anknüpfungspunkt und definiert aus dem vorgestellten KT ein eigenes Pendant mit dem Namen ‚Facilities' (FCIL), das exakt den gleichen Inhalt wie in COCOMO abbildet. Dieses Auftreten beinhaltet auch die Übernahme von identischen Werten für die Berechnungsformeln. Einen weiteren Anknüpfungspunkt seitens der Werkzeuge bietet der KT ‚Development Platform' (DEVP) [vgl. Garg und Gupta 2015] aus der AS. Hier spielen besonders die Tools zur Unterstützung der Programmierung eine entscheidende Rolle. Ähnlich wie bei COCOMO werden auch hier einfache Konsolenprogramme bis hin zu graphischen Entwicklungsumgebungen unterschieden. Die KT auf beiden Seiten erfüllen dabei die gleiche Zielstellung, jedoch sollte hier der Einfluss auf die Kosten kritisch betrachtet werden. Per se darf nicht von einem negativen Einfluss beim Einsatz von einfachen Werkzeuge ausgegangen werden. Geübte Entwickler auf der Konsole können durchaus schnell und effektiv arbeiten. Gerade Tastenkombinationen und Shortcuts führen häufig schneller zum Ziel als Mausbewegungen in einer IDE.

Diesen Fokus auf die Werkzeuge ergänzen weiter die agilen KT ‚Programming Language' (PRGL) [vgl. Garg und Gupta 2015; Keaveney und Conboy 2006] und ‚Type of Implementation' (TYOI) [vgl. Tanveer, Guzmán und Engel 2016] mit Aspekten der verwendeten Programmiersprache und der Art des zugrundeliegenden Entwicklungskonzeptes. So unterscheiden sich etwa die Sprachen beim Einsatzgebiet, der Syntax oder dem Befehlsumfang. Je nach Einsatz bspw. im Webumfeld (z. B. PHP, JS) oder für Business-Anwendungen (z. B. Java, C++) schwankt dabei der Aufwand (z. B. Zeit, Ressourcen) für die jeweilige Implementierung einer Funktionalität. Hinzu kommt der Einfluss von unterschiedlichen Prinzipien für das Schreiben von Quellcode. Dabei ist das Vorgehen bei OOP anders als etwa bei einer funktionalen oder prozeduralen Programmierung. Durch das Vorgehen entstehen unterschiedliche Aufgaben mit verschiedenem zeitlichen Umfang, wodurch wiederum der Gesamtaufwand beeinflusst wird. ‚Clean Code Development' (CCOD) soll hier als weiterer technischer KT aus der AS benannt werden. Dieser fokussiert besonders die Wichtigkeit, einfachen und sauberen Programmcode zu schreiben [vgl. Azevedo Santos u. a. 2013]. Wichtig ist dieser Aspekt auch bei der Teamarbeit, insbesondere wenn im Pair Programming ein gemeinsames Verständnis des Quellcodes notwendig ist. Hinzu kommt die zeitverzögerte Arbeit an gleichen Codestellen – auch hier wird mit einem sauberen Stand das Verstehen vereinfacht. Dieses zeigt sich besonders bei der Fehlerbehebung nach Abschluss der Entwicklung im Betriebsmodus, sobald andere Personen als die Entwickler Änderungen durchführen müssen.

Außerdem wird der KT ‚Defects in Third Party Tools' (DTPT) [vgl. Popli und Chauhan 2014b] immer wichtiger für aktuelle Entwicklungen. Wenn Fehler in externen Bibliotheken oder Schnittstellen auftreten, kann dies die Lauffähigkeit der eigenen Anwendung beeinflussen, so dass die Verwaltung von Updates in diesen Komponenten essenziell ist. Dies verursacht allerdings einen erhöhten Zeit- und Kostenaufwand. ‚Technology Newness' (TNEW) [vgl. Britto, Usman und Mendes 2014; Lang, Conboy und Keaveney 2013; Popli und Chauhan 2014b] und ‚Introduction of New Technology' (INTY) [vgl. Keaveney und Conboy 2006] werden ebenfalls als Einflussgrößen in der AS genannt. Hierbei ist es von Vorteil, offen für neue Technologien zu sein. Je nach Aufgabe und Einsatzzweck kann eine aktuelle Programmiersprache, Entwicklungsumgebung, Hardware oder Architektur den Entwicklungsprozess beschleunigen und im besten Fall Wettbewerbsvorteile generieren. Die Verwendung einer neuen Programmiersprache und Technologie erzeugt jedoch auf der anderen Seite einen Einfluss auf die Kosten. Für eine Verwendung müssen diese erst gelernt werden, damit eine effektive Nutzung möglich ist. Dabei ist es von Vorteil, wenn das Softwareprojekt eine lernfördernde Umgebung bietet [vgl. Jørgensen 2005].

Generell haben laut der Studie von Schweighofer u. a. [vgl. 2016] im agilen Umfeld die personenbezogenen KT eine stärkere Wirkung auf entstehende Kosten als die projektbezogenen KT. Dies bedeutet, dass für Schätzungen die Fertigkeiten und Erfahrungen der Experten im Team essenziell sind. Hinzu kommt der Stellenwert der Zusammenarbeit im Team, die wiederum gute Teamprozesse im fachlichen sowie zwischenmenschlichen Bereich voraussetzt. Außerdem zeigt sich – obwohl es Ansätze für KT in der AS gibt – dass wenig Einigkeit darüber besteht, welche KT sich wirklich als Einflussgrößen auf die anfallenden Kosten eignen [vgl. Usman, Mendes, Weidt u. a. 2014].

5 Korrektheitsmetriken zur Qualitätssicherung

Sowohl in COCOMO als auch in der AS ist ein immer größer werdendes Anliegen, die Prozesse und die Ergebnisse kontinuierlich zu verbessern. In COCOMO wird dieses über die Kalibrierung der Modelle auf Basis von Daten aus abgeschlossener Projekte durchgeführt. Die Fehleranalyse basiert dabei auf Projektebene und wird jeweils am Ende eines Projektes innerhalb eines durchschnittlichen Zeitfensters von zehn bis achtzehn Monaten durchgeführt [vgl. Reifer 2000, S. 58]. In der AS findet diese dagegen über eine Reflektion auf Basis der vergangenen Iteration(en) statt. Die Fehleranalyse ist damit kürzer angelegt und erfolgt bereits während des laufenden Projekts. Das Zeitfenster bewegt sich dabei je nach Iterationslänge zwischen einer und vier Wochen.

In beiden Umgebungen gestaltet sich die Möglichkeit zur Datenerhebung für eine anschließende Fehlerberechnung ähnlich. Die Größe eines jeweiligen Projekts wird im Vorfeld bei COCOMO auf Basis der historischen Daten und bei AS mithilfe von Erfahrungswerten der Teammitglieder geschätzt. Hierzu werden unter anderem vorhandene Rahmenparameter mit in die Berechnungen und Überlegungen einbezogen. Ist das Projekt – beziehungsweise die Iteration – beendet, liegen danach zusätzlich die tatsächlichen Werte vor. Sowohl die Prognosewerte als auch die realen Werte sind einfach zu erhalten und liegen im Normalfall für eine Qualitätssicherung vor. Danach ist es unproblematisch mit statistischen Methoden die Genauigkeit der Schätzungen zu messen.

In COCOMO wird dafür – wie eben beschrieben – die sogenannte ‚Percentage of Relative Error' (PRE) [vgl. Boehm 1981, S. 495] als relativer Fehler aus den Prognose- und Ergebniswerten mit der Formel 3 berechnet. Das Ergebnis wird in Prozent angegeben und zeigt die Fehlerhöhe an. Je größer damit der Wert ist, desto stärker ist die Abweichung der Schätzung von den Realwerten. Die Fehleranalyse kann dabei für die Aufwände in MM (wie dargestellt) und für den Zeitplan in $TDEV$ ausgewertet werden. Nicht zuletzt durch die einfache Anwendung hat sich die Formel als etablierte Methode entwickelt und bildet heute die Basis für nachfolgende Ansätze.

$$PRE = \frac{(MM)_{EST} - (MM)_{ACT}}{(MM)_{ACT}} \tag{3}$$

Eine direkte Weiterführung mit einer kleinen Anpassung beschreibt die ‚Magnitude of Relative Error' (MRE) [vgl. Kemerer 1987, S. 420], deren Aufbau (siehe Formel 4) identisch mit der vorherigen Berechnung ist. Hinzu kommen hier lediglich die Betragsstriche im Zähler der Division und die dadurch gewährleistete positive Ergebnismenge. Die Methode wird generell in der Softwareentwicklung eingesetzt [vgl. Hummel 2011, S. 30], was in der KS durch die direkte Nähe zu COCOMO und

das ähnliche Erscheinungsdatum und in der AS aufgrund der zeitlosen Möglichkeit zur Fehleranalyse [vgl. Mendes u. a. 2003] begründet ist.

$$MRE = \frac{|actual - estimated|}{actual} \tag{4}$$

Nach der Messung und auf Basis der Abweichungen erfolgt nun die Anpassung der für die Schätzungen zugrundeliegenden Daten, die angesichts der unterschiedlichen Voraussetzungen von COCOMO und AS verschieden durchgeführt wird. Bei COCOMO läuft diese über eine Anpassung der Modellkonfiguration ab, beispielsweise über die Kalibrierung der Faktoren und Exponenten (siehe Kap. 3) und der Multiplikatoren für die KT (siehe Kap. 4). Verändert werden hier Werte sowohl für die Berechnung von MM als auch $TDEV$. Bei der AS wird in Bezug auf die Anpassung des Referenzmediums für die Schätzungen anders vorgegangen. Entgegen den historischen Daten für die parametrischen Modelle, basieren die Daten hier auf den individuellen Erfahrungen der Teammitglieder. Für die Optimierung im Vorgehen ist demzufolge die Einbindung der Individuen von entscheidender Bedeutung. So wird in gemeinsamen Gesprächen in der Form von Retrospektiven in strukturierter Form eine Auseinandersetzung ermöglicht, um neue Erkenntnisse zu produzieren, die eine Optimierung des Arbeitsprozesses und eine Steigerung der Leistungsfähigkeit ermöglichen.

6 Zusammenfassung und Fazit

Inhalt dieser Studie ist die Reflektion von COCOMO im Licht der AS. Konkret geht es um die verwendeten Variablen in COCOMO und die Klärung einer Verwendung dieser auch in der AS. Im Kern ist zu prüfen, welche der Größen sich in welcher Form auch in den neuen Methoden einsetzen lassen. Hierzu ist es notwendig, die einzelnen Bestandteile von COCOMO separat mit der AS zu reflektieren. Mit Hilfe dieser qualitativen und deskriptiven Analyse entsteht so eine sinnbildliche Brücke zwischen beiden Welten. Für eine Zusammenfassung der Ergebnisse werden dazu nachfolgend die eingangs erwähnten Forschungsfragen noch einmal betrachtet und deren Beantwortung vorgestellt.

F1: Grundsätzlich lassen sich Variablen aus COCOMO wie Größen- und Korrektheitsmetriken, Faktoren, Exponenten und KT auch in der AS verwenden. Sollten diese nicht explizit bereits dort zum Einsatz kommen, so gibt es entsprechende Pendants, die den Inhalt in ähnlicher Art und Weise abbilden. Explizit gelten für beide Seiten sowohl objektive (z. B. SLOC, FP) als auch subjektive Größenmetriken (z. B. SP, UCP) (siehe Anhang Tab. 1). Auch die Faktoren und Exponenten (siehe Anhang Tab. 2) sind mit einer entsprechenden Anpassung für agile Rahmenbedingungen nutzbar. Bei den KT können Variablengruppen (siehe Anhang Tab. 3) aus dem Produkt- (z. B. Verlässlichkeit, Komplexität), dem Computer- (z. B. Laufzeitumgebung, Leistungsfähigkeit), dem Personen- (z. B. Team, Person) und dem Projektbereich (z. B. Verteilte Zusammenarbeit, Prozessmodelle) identifiziert werden. Ergänzend ergibt sich auch bei den Korrektheitsmetriken (siehe Anhang Tab. 4) mit PRE für COCOMO und MRE für die AS eine passende Übereinstimmung.

F2: Für die Variable ‚Qualität von Anforderungen' ergeben sich KT aus dem Projektbereich im agilen Umfeld. Kandidaten sind REQL, LCYR, USDQ, TASI, REQX, DEPD und IMES, nach denen sich eine gute oder schlechte Qualität entlang von linguistischen und textuellen Formatkriterien sowie semantischen Inhaltsaspekten bemessen lässt. Konkret geht es dabei um die Größe, die Komplexität und die Abhängigkeiten der Dokumente in Korrelation zum Grad eines Verstehens und dem Umfang von Systemveränderungen. Teamerfahrungen als

weitere Variable sind eine Größe für den Personenbereich, die sowohl in COCOMO als auch in der AS gleichermaßen verwendet wird. Vertreter aus COCOMO sind ACAP und PCAP, parallel dazu stammen PERS sowie PTCP aus der AS. Neben den KT reflektieren auch die Exponenten über die Modi in COCOMO erfahrungsspezifische Unterschiede in Projekten. Generell ist hier der Grad und die Verteilung von Kenntnissen im Team wichtig, im besten Fall mit einer gesunden Mischung aus Juniors und Seniors im Projekt. Ein gemeinsames Verständnis von Anforderungen lässt sich als Variable ähnlich wie die Qualität der Dokumente in der AS identifizieren. Als Kandidat kommen hierzu die SP der Größenmetriken in Frage, die als abstraktes Maß aus einer Kombination von Zeit, Komplexität und Risiko die Größe der Anforderung abbilden. Die Höhe wird dabei über einen iterativen Einigungsprozess im Team bis zum Erlangen eines gemeinsamen Bildes ermittelt. Ausgehend von Unterschieden in der Qualität und dem gemeinsamen Verständnis von Anforderungen ergeben sich mögliche Auswirkungen auf den Entwicklungsprozess. Diese letzte Variable kann mit Messgrößen innerhalb der Korrektheitsmetriken konkret mit PRE aus COCOMO und MRE aus AS beschrieben werden. Mithilfe von Formeln ist hier eine Identifizierung von Abweichung in den Schätzungen in Reflektion von den tatsächlichen Werten möglich. Der tatsächliche Aufwand kann sich von den Prognosen etwa durch zusätzliche Aufgaben unterscheiden, die durch fehlende Informationen in den Anforderungen (Qualitätsmerkmal) oder ein unterschiedliches gemeinsames Bild (Verstehensmerkmal) entstanden sind.

F3: Bei den Größenmetriken werden objektive und subjektive Messkriterien unterschieden. Der erste Ansatz, der verstärkt in COCOMO eingesetzt wird, basiert dabei auf zählbaren, berechenbaren und analysierbaren Eigenschaften des Programmcodes. Vertreter sind SLOC, CK, FP, OP und AP. Subjektive Messkriterien mit größerem Anteil in der AS setzen dagegen auf ein qualitatives Vorgehen, wodurch die Abstufungen der individuellen Erfahrungen eine große Rolle spielen. Metriken hierzu sind CFP, UCP und SP. Die Operationalisierung der Faktoren und Exponenten gestaltet sich im Vorgehen zwischen COCOMO und der AS in etwa vergleichbar. Grundsätzlich werden Stufen über eine Likert-Skala definiert, die je nach konkreten Rahmenbedingungen ausgewählt werden können. Die Anzahl liegt zwischen einer und sechs Stufen, in Reflektion zu den unterschiedlichen Modi bei COCOMO und den Definitionen für Projektumgebungen bei WebMo in der AS. Jeder Stufe ist eine Fließkommazahl aus dem Definitionsbereich $0,32$ bis $3,0$ zugeordnet, wodurch ein unterschiedlicher Einfluss in den Berechnungen abgebildet werden kann. Die KT sowohl in COCOMO als auch im WebMo für die AS wurden über eine Likert-Skala operationalisiert. Die Auswahlmöglichkeiten schwanken zwischen vier bis sieben Stufen mit textuellen Wahlmöglichkeiten von Very Low bis Extra Extra High. Die einzelnen Stufen enthalten operationalisierte Werte in Form von Zahlen, Gleichungen, Prozenten und Perzentilen, Zeiten und Text. Für die Integration in die Berechnungsformeln ist jeder Stufe – ähnlich wie bei den Faktoren und Exponenten – eine Fließkommazahl zugeordnet. Je nach kalibriertem Einfluss liegen die Werte zwischen $0,58$ bis $1,74$. Generell wird jedoch in der AS keine beziehungsweise nur eine geringe Quantifizierung der KT vorgenommen und die Variablen werden dazu lediglich als Einflussgröße auf die Aufwände benannt. Die Ansätze sind damit wenig bis gar nicht formalisiert und setzen verstärkt auf eine erfahrungsbasierte und subjektive Herangehensweise, mit Einbindung der Bewertungspersonen. Bei den Korrektheitsmetriken sowohl in COCOMO als auch in der AS existieren keine konkreten Operationalisierungen. Vielmehr sind unterschiedliche Formeln implementiert, die die Güte der Prognose bei den Schätzungen berechnen. Verwendet werden hierzu Eingangswerte, wie etwa die Zahlenwerte aus den Vorhersagen und die tatsächlichen Werte nach Fertigstellung der Entwicklung.

Nach der Beantwortung der Fragen zeigen sich Übereinstimmungen und Unterschiede im generellen Ansatz und im Detail zwischen COCOMO und der AS. Grundsätzlich wurde das algorithmische Vorgehen in der Übergangsphase zwischen der KS und der AS in den neueren Methoden weiterverwendet. Ein prominentes Beispiel hierzu ist WebMo. Konkret fand dabei ein Transfer von inhaltlichen Merkmalen wie Variablen und Werte identisch oder mit Anpassungen statt. Aufgrund einer zeitlichen Weiterentwicklung der Schätzmethoden – von einer Entwicklung mit formalen Prozessmodellen, die Gewicht auf eine günstige Umsetzung setzen, hin zu Expertenschätzungen mit Fokus auf schnellen Ergebnisse für den Kunden – ergeben sich auch Differenzen. Aufgrund der unterschiedlichen Ausrichtung variiert dabei die Charakteristik der Variablen abhängig von der Tendenz zu klassisch vs. agil. Je nach Umgebung werden verstärkt objektive vs. subjektive Schätzmethoden, quellcodebasierte vs. erfahrungsbasierte Größenmetriken und prozess- und werkzeugbezogene vs. personenbezogene KT verwendet. Mit der zeitlichen Distanz zwischen COCOMO und der AS und dem unterschiedlichen Methodenansatz ist damit die Auswahl an Variablen begrenzt. Je besser hier die jeweiligen Eigenschaften der Variablen aus COCOMO (z. B. KT aus Personenbereich), mit den Eigenschaften der AS (z. B. Einbindung der Personen) übereinstimmen, desto besser lassen sich diese übertragen. Ratsam ist jedoch immer eine Prüfung auf eine valide Verwendung innerhalb der aktuellen agilen Rahmenparameter.

Danksagung: Als erstes möchte ich Herrn Prof. Dr. Lüttgen vom Lehrstuhl Softwaretechnik und Programmiersprachen an der Universität Bamberg für seine Unterstützung bei der Erstellung dieser Arbeit danken. Mit seiner intensiven Prüfung der Inhalte und seinen hilfreichen Anmerkungen zu Verbesserungen hat er maßgeblich bei der Entwicklung mitgewirkt.

Außerdem möchte ich meiner Familie und meinen Freunden danken. Insbesondere meiner Frau Daniela für Ihren unermüdlichen Einsatz in der Familie und für unsere lieben Kinder. Dieser Rückhalt und diese Freiräume ermöglichten erst die Erstellung dieser Arbeit. Meinem Vater für die Korrekturhinweise und allen Anderen mit denen ich Ideen und Gedanken austauschen durfte.

Tabellen

Tabelle 1: Metriken zur Softwaregröße

COCOMO	AS
Source Lines of Code (SLOC)	Source Lines of Code (SLOC)
Chidamber- und Kemerer-Metriken (CK)	Chidamber- und Kemerer-Metriken (CK)
Function Points (FP)	Function Points (FP)
Object Points (OP)	Object Points (OP)
Application Points (AP)	Application Points (AP), Web Objects (WO), Multimedia Points (MP)
COSMIC Function Points (CFP)	COSMIC Function Points (CFP)
Use Case Points (UCP)	Use Case Points (UCP)
Story Points (SP)	Story Points (SP)

Tabelle 2: Faktoren und Exponenten

COCOMO	AS
Produktivitätsfaktor in COCOMO (a), Projektgrößenskalierungsexponent in COCOMO (b)	Produktivitätsfaktor in WebMo (A), Projektgrößenskalierungsexponent in WebMo (B)
Produktivitätsfaktor in COCOMO (c), Projektgrößenskalierungsexponent in COCOMO (d)	Produktivitätsfaktor in WebMo (P1), Projektgrößenskalierungsexponent in WebMo (P2)

Tabelle 3: KT

COCOMO	AS
Produktbereich	
Verlässlichkeit	
Required Software Reliability (RELY), Impact of Software Failure (FAIL)	Product Reliability and Complexity (RCPX)
Komplexität	
Product Complexity (CPLX)	Product Reliability and Complexity (RCPX)
Wiederverwendbarkeit	
Required Reusability (RUSE)	Required Reusability (RUSE)

Weiter auf der nächsten Seite

Tabelle 3: KT

COCOMO	AS
Computerbereich	
Laufzeitumgebung	
Platform Constraints (PLAT), Platform Volatility (PVOL)	Operating System (OPSY), Platform Difficulty (PDIF)
Leistungsfähigkeit	
Execution Time Constraint (TIME)	Performance Requirements (PREQ)
Personenbereich	
Team	
Analyst Capability (ACAP), Programmer Capability (PCAP)	Personnel Capability (PERS), Perfect Team Composition (PTCP)
Stakeholder Team Cohesion (TEAM), Personal Continuity (PCON)	Stakeholder Team Cohesion (TEAM), Lack of Team Cohesion (LOTC), Expected Team Change (EXTC)
Person	
Application Experience (AEXP), Platform Experience (PEXP), Language and Tool Experience (LTEX)	Personal Experience (PREX), Individual Personal Capability (IPCP), Domain Knowledge (DNKW), Client-Specific Knowledge (CSKW), Team's Prior Experience (TPEX), Programmer Estimation Capability (PSCP), Developers Estimation Experience (DVEE), XP Practices (XPPR)
Projektbereich	
Verteilte Zusammenarbeit	
Multisite Collocation (SITE: Collocation), Multisite Communication (SITE: Communication)	Facilities (FCIL), Process Model of the Sites (PRMS), Language Differences (LNGD), Travel (TRVL)
Prozessmodelle	
Process Capability & Usage (PCUS)	Capability Maturity Model (CMMI)
Zeitplan	
Required Development Schedule (SCED)	Required Development Schedule (SCED)
Anforderungen	

Weiter auf der nächsten Seite

Tabelle 3: KT

COCOMO	AS
Breakage (BKGE), Requirements Volatility (RVOL)	Features Defined by the Customer (FEDC), Requirements Legibility (REQL), Lack of Clarity in Requirements (LCYR), User Stories Description Quality (USDQ), Task Size (TASI), Requirements Complexity (REQX), Dependencies (DEPD), Impact on Existing System (IMES) Volatility of Requirements (VREQ), Changing Requirements from Customers (CHRC), New Requirements (NREQ), Scope Creep (SCCP), Missing Functional and Non-Functional Requirements (MIFN), Time Response (TRSP)
Sicherheit	
Security Application (SECU)	Security (SECY), Risk Taking (RISK)
Werkzeuge	
Use of Software Tools (TOOL)	Facilities (FCIL), Development Platform (DEVP), Programming Language (PRGL), Type of Implementation (TYOI), Clean Code Development (CCOD), Defects in Third Party Tools (DTPT), Technology Newness (TNEW), Introduction of New Technology (INTY)

Tabelle 4: Korrektheitsmetriken

COCOMO	AS
Percentage of Relative Error (PRE)	Magnitude of Relative Error (MRE)

Abkürzungsverzeichnis

a Produktivitätsfaktor in COCOMO

A Produktivitätsfaktor in WebMo

ACAP Analyst Capability

AEXP Application Experience

AP Application Points

AS Agile Softwareentwicklung

b Projektgrößenskalierungsexponent in COCOMO

B Projektgrößenskalierungsexponent in WebMo

BKGE Breakage

c Produktivitätsfaktor in COCOMO

CCOD Clean Code Development

CFP COSMIC Function Points

CHRC Changing Requirements from Customers

CK Chidamber- und Kemerer-Metriken

CMMI Capability Maturity Model

COCOMO Constructive Cost Model

COCOMO 81 COCOMO 81

COCOMO 2.0 COCOMO 2.0

COCOMO 3.0 COCOMO 3.0

Ada COCOMO Ada COCOMO

CPLX Product Complexity

CSKW Client-Specific Knowledge

d Projektgrößenskalierungsexponent in COCOMO

DEPD Dependencies

DEVP Development Platform

DNKW Domain Knowledge

DSI Delivered Source Instructions

DTPT Defects in Third Party Tools

DVEE Developers Estimation Experience

EXTC Expected Team Change

FAIL Impact of Software Failure

FCIL Facilities

FEDC Features Defined by the Customer

FP Function Points

IMES Impact on Existing System

INTY Introduction of New Technology

IPCP Individual Personal Capability

KS Klassische Softwareentwicklung

KT Kostentreiber

LCYR Lack of Clarity in Requirements

LNGD Language Differences

LOTC Lack of Team Cohesion

LTEX Language and Tool Experience

MIFN Missing Functional and Non-Functional Requirements

MM Man-Months

MP Multimedia Points

MRE Magnitude of Relative Error

NREQ New Requirements

OP Object Points

OPSY Operating System

P1 Produktivitätsfaktor in WebMo

P2 Projektgrößenskalierungsexponent in WebMo

PCAM Principal Component Analysis Model

PCAP Programmer Capability

PCON Personal Continuity

PCUS Process Capability & Usage

PDIF Platform Difficulty

PEFF Process Efficiency

PERS Personnel Capability

PEXP Platform Experience

PLAT Platform Constraints

PRE Percentage of Relative Error

PREQ Performance Requirements

PREX Personal Experience

PRGL Programming Language

PRJD Project Domain

PRMS Process Model of the Sites

PSCP Programmer Estimation Capability

PTCP Perfect Team Composition

PVOL Platform Volatility

RBFM Radial Basis Function Model

RCPX Product Reliability and Complexity

RELY Required Software Reliability

REQL Requirements Legibility

REQX Requirements Complexity

RFM Resistance Factor Model

RISK Risk Taking

RUSE Required Reusability

RVOL Requirements Volatility

SCCP Scope Creep

SCED Required Development Schedule

SECU Security Application

SECY Security

SITE: Collocation Multisite Collocation

SITE: Communication Multisite Communication

SLOC Source Lines of Code

SP Story Points

TASI Task Size

TDEV Development Schedule

TEAM Stakeholder Team Cohesion

TIME Execution Time Constraint

TMDY Team Dynamics

TNEW Technology Newness

TOOL Use of Software Tools

TPEX Team's Prior Experience

TRSP Time Response

TRVL Travel

TYOI Type of Implementation

UCP Use Case Points

USDQ User Stories Description Quality

VREQ Volatility of Requirements

WebMo Web Model

WO Web Objects

XPPR XP Practices

Tabellenverzeichnis

Literaturverzeichnis

Abrahamsson, Pekka u. a. [2007]. „Effort Prediction in Iterative Software Development Processes - Incremental Versus Global Prediction Models“. In: *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, Madrid, Spanien, September 20-21, 2007*, S. 344–353.

Alshayeb, Mohammad und Wei Li [2003]. „An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes“. In: *IEEE Transactions on Software Engineering* SE-29.11, S. 1043–1049.

Azevedo Santos, Mariana de u. a. [2013]. „Improving the Management of Cost and Scope in Software Projects Using Agile Practices“. In: *International Journal of Computer Science & Information Technology (IJCSIT)* 5.1.

Benediktsson, Oddur und Darren Dalcher [2004]. „Project Effort Estimation: Or, When Size Makes a Difference“. In: *Proceedings of the 11th European Conference on Software Process Improvement, EuroSPI 2004, Trondheim, Norwegen, November 10-12, 2004*. Hrsg. von Torgeir Dingsøyr. Bd. 3281. Lecture Notes in Computer Science. Springer, S. 171–183.

Boehm, Barry W. [1981]. *Software Engineering Economics*. Englewood Cliffs, N J: Prentice-Hall.

Boehm, Barry W., Bradford Clark u. a. [1995]. „Cost Models for Future Software Life Cycle Processes: COCOMO 2.0“. In: *Annals of Software Engineering* 1.1, S. 57–94.

Boehm, Barry W. und Walker Royce [1987]. „Ada COCOMO and the Ada Process Model“. In: *Proceedings of the 3rd COCOMO User Group Meeting, Software Engineering Institute (SEI), Carnegie Mellon University, Pittsburgh, PA, USA*, S. 1–34.

Boehm, Barry W. und Ricardo Valerdi [2008]. „Achievements and Challenges in Cocomo-Based Software Resource Estimation“. In: *Software, IEEE* 25.5, S. 74–83.

Boehm, Barry W. und Ray W. Wolverton [1980]. „Software Cost Modeling: Some Lessons Learned“. In: *Journal of Systems and Software* 1, S. 195–201.

Briand, Lionel C., Khaled El Emam und Frank Bomarius [1998]. „COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment“. In: *Proceedings of the 1998 International Conference on Software Engineering, ICSE 1998, Kyoto, Japan, April 19-25, 1998*. Hrsg. von Koji Torii, Kokichi Futatsugi und Richard A. Kemmerer. IEEE Computer Society, S. 390–399.

Britto, Ricardo, Emilia Mendes und Jürgen Börstler [2015]. „An Empirical Investigation on Effort Estimation in Agile Global Software Development“. In: *Proceedings of the 10th International Conference on Global Software Engineering, ICGSE 2015, Ciudad Real, Spanien, Juli 13-16, 2015*. IEEE Computer Society, S. 38–45.

Britto, Ricardo, Muhammad Usman und Emilia Mendes [2014]. „Effort Estimation in Agile Global Software Development Context“. In: *Proceedings of the 2014 International Workshops on Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation, XP 2014, Rom, Italien, Mai 26-30, 2014, Revised Selected Papers*. Hrsg. von Torgeir Dingsøyr u. a. Bd. 199. Lecture Notes in Business Information Processing. Springer, S. 182–192.

Čelar, Stipe, Mili Turić und Linda Vicković [2014]. „Method for Personal Capability Assessment in Agile Teams Using Personal Points“. In: *Proceedings of the 22nd Telecommunications Forum Telfor, TELFOR 2014, Belgrad, Serbien, November 25-27, 2014*, S. 1134–1137.

Chidamber, Shyam R. und Chris F. Kemerer [1994]. „A Metrics Suite for Object Oriented Design“. In: *IEEE Transactions on Software Engineering* SE-20.6, S. 476–493.

Clark, Bradford [2016]. „COCOMO III Project Overview“. In: *Proceedings of the 17th Practical Software and Systems Measurement Users Group Workshop, PSM 2016, Crystal City, Virginia, USA, Februar 22-26, 2016*, S. 1–18.

Cohn, Mike [2004]. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Cowderoy, Adrian J. C. [1999]. „Size and Quality Measures for Multimedia and Web-Site Production“. In: *Proceedings of the 14th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA, USA*.

Desharnais, Jean-Marc, Luigi Buglione und Buğra Kocatürk [2011]. „Using the COSMIC Method to Estimate Agile User Stories“. In: *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, PROFES 2011, Torre Canne, Brindisi, Italien, Juni 20-22, 2011*. Hrsg. von Danilo Caivano u. a. New York, NY, USA: ACM, S. 68–73.

Galorath, Inc. [2017]. *Cost Estimation Software for Planning & Tracking Complex Projects – SEER by Galorath*. URL: `http://galorath.com` (besucht am 04. 08. 2017).

Garg, Sakshi und Daya Gupta [2015]. „PCA Based Cost Estimation Model for Agile Software Development Projects“. In: *Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management, IEOM 2015, Dubai, United Arab Emirates (UAE), März 3-5, 2015*. IEEE, S. 1–7.

Hummel, Oliver [2011]. „Grundlagen der Aufwandsschätzung“. In: *Aufwandsschätzungen in der Software- und Systementwicklung kompakt*. Heidelberg: Spektrum Akademischer Verlag, S. 13–34.

Jørgensen, Magne [2005]. „Practical Guidelines for Expert-Judgment-Based Software Effort Estimation“. In: *IEEE Software* 22.3, S. 57–63.

Keaveney, Siobhán und Kieran Conboy [2006]. „Cost Estimation in Agile Development Projects“. In: *Proceedings of the 14th European Conference on Information Systems, ECIS 2006, Göteborg, Sweden, 2006*. Hrsg. von Jan Ljungberg und Magnus Andersson, S. 183–197.

Kemerer, Chris F. [1987]. „An Empirical Validation of Software Cost Estimation Models“. In: *Communications of the ACM* 30.5, S. 416–429.

Lang, Michael, Kieran Conboy und Siobhán Keaveney [2013]. „Cost Estimation in Agile Software Development Projects“. In: *Information Systems Development: Reflections, Challenges and New Directions*. Hrsg. von Rob Pooley u. a. New York, NY: Springer New York, S. 689–706.

Lenarduzzi, Valentina u. a. [2015]. „Functional Size Measures and Effort Estimation in Agile Development: A Replicated Study“. In: *Proceedings of the 16th International Conference on Agile Processes in Software Engineering and Extreme Programming, XP 2015, Helsinki, Finnland, Mai 25-29, 2015*. Hrsg. von Casper Lassenius, Torgeir Dingsøyr und Maria Paasivaara. Bd. 212. Lecture Notes in Business Information Processing. Springer, S. 105–116.

Likert, Rensis [1932]. „A Technique for the Measurement of Attitudes“. In: *Archives of Psychology* 22, S. 5–55.

Mendes, Emilia u. a. [2003]. „A Comparative Study of Cost Estimation Models for Web Hypermedia Applications“. In: *Empirical Software Engineering* 8.2, S. 163–196.

Menzies, Tim u. a. [2017]. „Negative Results for Software Effort Estimation“. In: *Empirical Software Engineering* 22.5, S. 2658–2683.

Moser, Raimund, Witold Pedrycz und Giancarlo Succi [2007]. „Incremental Effort Prediction Models in Agile Development Using Radial Basis Functions“. In: *Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering, SEKE 2007, Boston, Massachusetts, USA, Juli 9-11, 2007*. Knowledge Systems Institute Graduate School, S. 519–522.

Nguyen-Cong, Danh und De Tran-Cao [2013]. „A Review of Effort Estimation Studies in Agile, Iterative and Incremental Software Development“. In: *2013 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, RIVF 2013, Hanoi, Vietnam, November 10-13, 2013*. IEEE, S. 27–30.

Popli, Rashmi und Naresh Chauhan [2014a]. „Cost and Effort Estimation in Agile Software Development“. In: *Proceedings of the 2014 International Conference on Reliability Optimization and Information Technology, ICROIT 2014, Faridabad, Haryana, Indien, Februar 6-8, 2014*. IEEE, S. 57–61.

— [2014b]. „Estimation in Agile Environment Using Resistance Factors“. In: *Proceedings of the 2014 International Conference on Information Systems and Computer Networks, ISCON 2014, Mathura, Indien, März 1-2, 2014*. Hrsg. von Manas Kumar Mishra. IEEE, S. 60–65.

PRICE Systems, LCC [2017]. *Cost Estimation Software | PRICE Systems*. URL: `http://www.pricesystems.com` (besucht am 04. 08. 2017).

Putnam, Lawrence H. [1978]. „A General Empirical Solution to the Macro Software Sizing and Estimating Problem“. In: *IEEE Transactions on Software Engineering* SE-4.4, S. 345–361.

Reifer, Donald J. [2000]. „Web Development: Estimating Quick-to-Market Software“. In: *IEEE Software* 17.6, S. 57–64.

Santana, Célio u. a. [2011]. „Using Function Points in Agile Projects“. In: *Proceedings of the 12th International Conference on Agile Processes in Software Engineering and Extreme Programming, XP 2011, Madrid, Spanien, Mai 10-13, 2011*. Hrsg. von Alberto Sillitti u. a. Bd. 77. Lecture Notes in Business Information Processing. Springer, S. 176–191.

Schweighofer, Tina u. a. [2016]. „How is Effort Estimated in Agile Software Development Projects?“ In: *Proceedings of the 5th Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, Budapest, Ungarn, August 29-31, 2016*. Hrsg. von Zoran Budimac, Zoltán Horváth und Tamás Kozsik. Bd. 1677. CEUR Workshop Proceedings. CEUR-WS.org, S. 73–80.

Tanveer, Binish, Liliana Guzmán und Ulf M. Engel [2016]. „Understanding and Improving Effort Estimation in Agile Software Development: An Industrial Case Study“. In: *Proceedings of the 2016 International Conference on Software and Systems Process, ICSSP 2016, Austin, Texas, USA, Mai 14-15, 2016*. Hrsg. von Dewayne E. Perry und David Raffo. ACM, S. 41–50.

Usman, Muhammad und Ricardo Britto [2016]. „Effort Estimation in Co-located and Globally Distributed Agile Software Development: A Comparative Study“. In: *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, IWSM-MENSURA 2016, Berlin, Deutschland, Oktober 5-7, 2016*. Hrsg. von Jens Heidrich und Frank W. Vogelezang. IEEE Computer Society, S. 219–224.

Usman, Muhammad, Emilia Mendes und Jürgen Börstler [2015]. „Effort Estimation in Agile Software Development: A Survey on the State of the Practice“. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE 2015, Nanjing, China, April 27-29, 2015*. Hrsg. von Jian Lv, Jason He Zhang und Muhammad Ali Babar. ACM, 12:1–12:10.

Usman, Muhammad, Emilia Mendes, Francila Weidt u. a. [2014]. „Effort Estimation in Agile Software Development: A Systematic Literature Review“. In: *Proceedings of the 10th International Conference on Predictive Models in Software Engineering, PROMISE 2014, Torino, Italien, September 17, 2014*. Hrsg. von Stefan Wagner und Massimiliano Di Penta. ACM, S. 82–91.

Bamberger Beiträge zur Wirtschaftsinformatik

Nr. 1 (1989)	Augsburger W., Bartmann D., Sinz E.J.: Das Bamberger Modell: Der Diplom-Studiengang Wirtschaftsinformatik an der Universität Bamberg (Nachdruck Dez. 1990)
Nr. 2 (1990)	Esswein W.: Definition, Implementierung und Einsatz einer kompatiblen Datenbankschnittstelle für PROLOG
Nr. 3 (1990)	Augsburger W., Rieder H., Schwab J.: Endbenutzerorientierte Informationsgewinnung aus numerischen Daten am Beispiel von Unternehmenskennzahlen
Nr. 4 (1990)	Ferstl O.K., Sinz E.J.: Objektmodellierung betrieblicher Informationsmodelle im Semantischen Objektmodell (SOM) (Nachdruck Nov. 1990)
Nr. 5 (1990)	Ferstl O.K., Sinz E.J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)
Nr. 6 (1991)	Augsburger W., Rieder H., Schwab J.: Systemtheoretische Repräsentation von Strukturen und Bewertungsfunktionen über zeitabhängigen betrieblichen numerischen Daten
Nr. 7 (1991)	Augsburger W., Rieder H., Schwab J.: Wissensbasiertes, inhaltsorientiertes Retrieval statistischer Daten mit EISREVU / Ein Verarbeitungsmodell für eine modulare Bewertung von Kennzahlenwerten für den Endanwender
Nr. 8 (1991)	Schwab J.: Ein computergestütztes Modellierungssystem zur Kennzahlenbewertung
Nr. 9 (1992)	Gross H.-P.: Eine semantiktreue Transformation vom Entity-Relationship-Modell in das Strukturierte Entity-Relationship-Modell
Nr. 10 (1992)	Sinz E.J.: Datenmodellierung im Strukturierten Entity-Relationship-Modell (SERM)
Nr. 11 (1992)	Ferstl O.K., Sinz E. J.: Glossar zum Begriffsystem des Semantischen Objektmodells
Nr. 12 (1992)	Sinz E. J., Popp K.M.: Zur Ableitung der Grobstruktur des konzeptuellen Schemas aus dem Modell der betrieblichen Diskurswelt
Nr. 13 (1992)	Esswein W., Locarek H.: Objektorientierte Programmierung mit dem Objekt-Rollenmodell
Nr. 14 (1992)	Esswein W.: Das Rollenmodell der Organsiation: Die Berücksichtigung aufbauorganisatorische Regelungen in Unternehmensmodellen
Nr. 15 (1992)	Schwab H. J.: EISREVU-Modellierungssystem. Benutzerhandbuch
Nr. 16 (1992)	Schwab K.: Die Implementierung eines relationalen DBMS nach dem Client/Server-Prinzip
Nr. 17 (1993)	Schwab K.: Konzeption, Entwicklung und Implementierung eines computergestützten Bürovorgangssystems zur Modellierung von Vorgangsklassen und Abwicklung und Überwachung von Vorgängen. Dissertation

Nr. 18 (1993) Ferstl O.K., Sinz E.J.: Der Modellierungsansatz des Semantischen Objektmodells

Nr. 19 (1994) Ferstl O.K., Sinz E.J., Amberg M., Hagemann U., Malischewski C.: Tool-Based Business Process Modeling Using the SOM Approach

Nr. 20 (1994) Ferstl O.K., Sinz E.J.: From Business Process Modeling to the Specification of Distributed Business Application Systems - An Object-Oriented Approach -. 1st edition, June 1994

Ferstl O.K., Sinz E.J. : Multi-Layered Development of Business Process Models and Distributed Business Application Systems - An Object-Oriented Approach -. 2nd edition, November 1994

Nr. 21 (1994) Ferstl O.K., Sinz E.J.: Der Ansatz des Semantischen Objektmodells zur Modellierung von Geschäftsprozessen

Nr. 22 (1994) Augsburger W., Schwab K.: Using Formalism and Semi-Formal Constructs for Modeling Information Systems

Nr. 23 (1994) Ferstl O.K., Hagemann U.: Simulation hierarischer objekt- und transaktionsorientierter Modelle

Nr. 24 (1994) Sinz E.J.: Das Informationssystem der Universität als Instrument zur zielgerichteten Lenkung von Universitätsprozessen

Nr. 25 (1994) Wittke M., Mekinic, G.: Kooperierende Informationsräume. Ein Ansatz für verteilte Führungsinformationssysteme

Nr. 26 (1995) Ferstl O.K., Sinz E.J.: Re-Engineering von Geschäftsprozessen auf der Grundlage des SOM-Ansatzes

Nr. 27 (1995) Ferstl, O.K., Mannmeusel, Th.: Dezentrale Produktionslenkung. Erscheint in CIM-Management 3/1995

Nr. 28 (1995) Ludwig, H., Schwab, K.: Integrating cooperation systems: an event-based approach

Nr. 30 (1995) Augsburger W., Ludwig H., Schwab K.: Koordinationsmethoden und -werkzeuge bei der computergestützten kooperativen Arbeit

Nr. 31 (1995) Ferstl O.K., Mannmeusel T.: Gestaltung industrieller Geschäftsprozesse

Nr. 32 (1995) Gunzenhäuser R., Duske A., Ferstl O.K., Ludwig H., Mekinic G., Rieder H., Schwab H.-J., Schwab K., Sinz E.J., Wittke M: Festschrift zum 60. Geburtstag von Walter Augsburger

Nr. 33 (1995) Sinz, E.J.: Kann das Geschäftsprozeßmodell der Unternehmung das unternehmensweite Datenschema ablösen?

Nr. 34 (1995) Sinz E.J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme - Entwicklung, aktueller Stand und Trends -

Nr. 35 (1995) Sinz E.J.: Serviceorientierung der Hochschulverwaltung und ihre Unterstützung durch workflow-orientierte Anwendungssysteme

Nr. 36 (1996) Ferstl O.K., Sinz, E.J., Amberg M.: Stichwörter zum Fachgebiet Wirtschaftsinformatik. Erscheint in: Broy M., Spaniol O. (Hrsg.): Lexikon Informatik und Kommunikationstechnik, 2. Auflage, VDI-Verlag, Düsseldorf 1996

Nr. 37 (1996) Ferstl O.K., Sinz E.J.: Flexible Organizations Through Object-oriented and Transaction-oriented Information Systems, July 1996

Nr. 38 (1996) Ferstl O.K., Schäfer R.: Eine Lernumgebung für die betriebliche Aus- und Weiterbildung on demand, Juli 1996

Nr. 39 (1996) Hazebrouck J.-P.: Einsatzpotentiale von Fuzzy-Logic im Strategischen Management dargestellt an Fuzzy-System-Konzepten für Portfolio-Ansätze

Nr. 40 (1997) Sinz E.J.: Architektur betrieblicher Informationssysteme. In: Rechenberg P., Pomberger G. (Hrsg.): Handbuch der Informatik, Hanser-Verlag, München 1997

Nr. 41 (1997) Sinz E.J.: Analyse und Gestaltung universitärer Geschäftsprozesse und Anwendungssysteme. Angenommen für: Informatik '97. Informatik als Innovationsmotor. 27. Jahrestagung der Gesellschaft für Informatik, Aachen 24.-26.9.1997

Nr. 42 (1997) Ferstl O.K., Sinz E.J., Hammel C., Schlitt M., Wolf S.: Application Objects – fachliche Bausteine für die Entwicklung komponentenbasierter Anwendungssysteme. Angenommen für: HMD – Theorie und Praxis der Wirtschaftsinformatik. Schwerpunkheft ComponentWare, 1997

Nr. 43 (1997): Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using the Semantic Object Model (SOM) – A Methodological Framework - . Accepted for: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1997

Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using (SOM), 2nd Edition. Appears in: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1998

Nr. 44 (1997) Ferstl O.K., Schmitz K.: Zur Nutzung von Hypertextkonzepten in Lernumgebungen. In: Conradi H., Kreutz R., Spitzer K. (Hrsg.): CBT in der Medizin – Methoden, Techniken, Anwendungen -. Proceedings zum Workshop in Aachen 6. – 7. Juni 1997. 1. Auflage Aachen: Verlag der Augustinus Buchhandlung

Nr. 45 (1998) Ferstl O.K.: Datenkommunikation. In. Schulte Ch. (Hrsg.): Lexikon der Logistik, Oldenbourg-Verlag, München 1998

Nr. 46 (1998) Sinz E.J.: Prozeßgestaltung und Prozeßunterstützung im Prüfungswesen. Erschienen in: Proceedings Workshop „Informationssysteme für das Hochschulmanagement". Aachen, September 1997

Nr. 47 (1998) Sinz, E.J.:, Wismans B.: Das „Elektronische Prüfungsamt". Erscheint in: Wirtschaftswissenschaftliches Studium WiSt, 1998

Nr. 48 (1998) Haase, O., Henrich, A.: A Hybrid Respresentation of Vague Collections for Distributed Object Management Systems. Erscheint in: IEEE Transactions on Knowledge and Data Engineering

Nr. 49 (1998) Henrich, A.: Applying Document Retrieval Techniques in Software Engineering Environments. In: Proc. International Conference on Database and Expert Systems

Applications. (DEXA 98), Vienna, Austria, Aug. 98, pp. 240-249, Springer, Lecture Notes in Computer Sciences, No. 1460

Nr. 50 (1999) Henrich, A., Jamin, S.: On the Optimization of Queries containing Regular Path Expressions. Erscheint in: Proceedings of the Fourth Workshop on Next Generation Information Technologies and Systems (NGITS'99), Zikhron-Yaakov, Israel, July, 1999 (Springer, Lecture Notes)

Nr. 51 (1999) Haase O., Henrich, A.: A Closed Approach to Vague Collections in Partly Inaccessible Distributed Databases. Erscheint in: Proceedings of the Third East-European Conference on Advances in Databases and Information Systems – ADBIS'99, Maribor, Slovenia, September 1999 (Springer, Lecture Notes in Computer Science)

Nr. 52 (1999) Sinz E.J., Böhnlein M., Ulbrich-vom Ende A.: Konzeption eines Data Warehouse-Systems für Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule" im Rahmen der 29. Jahrestagung der Gesellschaft für Informatik, Paderborn, 6. Oktober 1999

Nr. 53 (1999) Sinz E.J.: Konstruktion von Informationssystemen. Der Beitrag wurde in geringfügig modifizierter Fassung angenommen für: Rechenberg P., Pomberger G. (Hrsg.): Informatik-Handbuch. 2., aktualisierte und erweiterte Auflage, Hanser, München 1999

Nr. 54 (1999) Herda N., Janson A., Reif M., Schindler T., Augsburger W.: Entwicklung des Intranets SPICE: Erfahrungsbericht einer Praxiskooperation.

Nr. 55 (2000) Böhnlein M., Ulbrich-vom Ende A.: Grundlagen des Data Warehousing. Modellierung und Architektur

Nr. 56 (2000) Freitag B, Sinz E.J., Wismans B.: Die informationstechnische Infrastruktur der Virtuellen Hochschule Bayern (vhb). Angenommen für Workshop "Unternehmen Hochschule 2000" im Rahmen der Jahrestagung der Gesellschaft f. Informatik, Berlin 19. - 22. September 2000

Nr. 57 (2000) Böhnlein M., Ulbrich-vom Ende A.: Developing Data Warehouse Structures from Business Process Models.

Nr. 58 (2000) Knobloch B.: Der Data-Mining-Ansatz zur Analyse betriebswirtschaftlicher Daten.

Nr. 59 (2001) Sinz E.J., Böhnlein M., Plaha M., Ulbrich-vom Ende A.: Architekturkonzept eines verteilten Data-Warehouse-Systems für das Hochschulwesen. Angenommen für: WI-IF 2001, Augsburg, 19.-21. September 2001

Nr. 60 (2001) Sinz E.J., Wismans B.: Anforderungen an die IV-Infrastruktur von Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule 2001" im Rahmen der Jahrestagung der Gesellschaft für Informatik, Wien 25. – 28. September 2001

Änderung des Titels der Schriftenreihe *Bamberger Beiträge zur Wirtschaftsinformatik* in *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* ab Nr. 61

Note: The title of our technical report series has been changed from *Bamberger Beiträge zur Wirtschaftsinformatik* to *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* starting with TR No. 61

Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik

Nr. 61 (2002) Goré R., Mendler M., de Paiva V. (Hrsg.): Proceedings of the International Workshop on Intuitionistic Modal Logic and Applications (IMLA 2002), Copenhagen, July 2002.

Nr. 62 (2002) Sinz E.J., Plaha M., Ulbrich-vom Ende A.: Datenschutz und Datensicherheit in einem landesweiten Data-Warehouse-System für das Hochschulwesen. Erscheint in: Beiträge zur Hochschulforschung, Heft 4-2002, Bayerisches Staatsinstitut für Hochschulforschung und Hochschulplanung, München 2002

Nr. 63 (2005) Aguado, J., Mendler, M.: Constructive Semantics for Instantaneous Reactions

Nr. 64 (2005) Ferstl, O.K.: Lebenslanges Lernen und virtuelle Lehre: globale und lokale Verbesserungspotenziale. Erschienen in: Kerres, Michael; Keil-Slawik, Reinhard (Hrsg.); Hochschulen im digitalen Zeitalter: Innovationspotenziale und Strukturwandel, S. 247 – 263; Reihe education quality forum, herausgegeben durch das Centrum für eCompetence in Hochschulen NRW, Band 2, Münster/New York/München/Berlin: Waxmann 2005

Nr. 65 (2006) Schönberger, Andreas: Modelling and Validating Business Collaborations: A Case Study on RosettaNet

Nr. 66 (2006) Markus Dorsch, Martin Grote, Knut Hildebrandt, Maximilian Röglinger, Matthias Sehr, Christian Wilms, Karsten Loesing, and Guido Wirtz: Concealing Presence Information in Instant Messaging Systems, April 2006

Nr. 67 (2006) Marco Fischer, Andreas Grünert, Sebastian Hudert, Stefan König, Kira Lenskaya, Gregor Scheithauer, Sven Kaffille, and Guido Wirtz: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems, April 2006

Nr. 68 (2006) Michael Mendler, Thomas R. Shiple, Gérard Berry: Constructive Circuits and the Exactness of Ternary Simulation

Nr. 69 (2007) Sebastian Hudert: A Proposal for a Web Services Agreement Negotiation Protocol Framework . February 2007

Nr. 70 (2007) Thomas Meins: Integration eines allgemeinen Service-Centers für PC-und Medientechnik an der Universität Bamberg – Analyse und Realisierungs-Szenarien. February 2007 (out of print)

Nr. 71 (2007) Andreas Grünert: Life-cycle assistance capabilities of cooperating Software Agents for Virtual Enterprises. März 2007

Nr. 72 (2007) Michael Mendler, Gerald Lüttgen: Is Observational Congruence on μ-Expressions Axiomatisable in Equational Horn Logic?

Nr. 73 (2007) Martin Schissler: out of print

Nr. 74 (2007) Sven Kaffille, Karsten Loesing: Open chord version 1.0.4 User's Manual. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 74, Bamberg University, October 2007. ISSN 0937-3349.

Nr. 75 (2008) Karsten Loesing (Hrsg.): Extended Abstracts of the Second *Privacy Enhancing Technologies Convention* (PET-CON 2008.1). Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 75, Bamberg University, April 2008. ISSN 0937-3349.

Nr. 76 (2008) Gregor Scheithauer, Guido Wirtz: Applying Business Process Management Systems – A Case Study. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 76, Bamberg University, May 2008. ISSN 0937-3349.

Nr. 77 (2008) Michael Mendler, Stephan Scheele: Towards Constructive Description Logics for Abstraction and Refinement. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 77, Bamberg University, September 2008. ISSN 0937-3349.

Nr. 78 (2008) Gregor Scheithauer, Matthias Winkler: A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 78, Bamberg University, October 2008. ISSN 0937-3349.

Nr. 79 (2008) Christian Wilms: Improving the Tor Hidden Service Protocol Aiming at Better Performances. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 79, Bamberg University, November 2008. ISSN 0937-3349.

Nr. 80 (2009) Thomas Benker, Stefan Fritzemeier, Matthias Geiger, Simon Harrer, Tristan Kessner, Johannes Schwalb, Andreas Schönberger, Guido Wirtz: QoS Enabled B2B Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 80, Bamberg University, May 2009. ISSN 0937-3349.

Nr. 81 (2009) Ute Schmid, Emanuel Kitzelmann, Rinus Plasmeijer (Eds.): Proceedings of the ACM SIGPLAN Workshop on *Approaches and Applications of Inductive Programming* (AAIP'09), affiliated with ICFP 2009, Edinburgh, Scotland, September 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 81, Bamberg University, September 2009. ISSN 0937-3349.

Nr. 82 (2009) Ute Schmid, Marco Ragni, Markus Knauff (Eds.): Proceedings of the KI 2009 Workshop *Complex Cognition*, Paderborn, Germany, September 15, 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 82, Bamberg University, October 2009. ISSN 0937-3349.

Nr. 83 (2009) Andreas Schönberger, Christian Wilms and Guido Wirtz: A Requirements Analysis of Business-to-Business Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 83, Bamberg University, December 2009. ISSN 0937-3349.

Nr. 84 (2010) Werner Zirkel, Guido Wirtz: A Process for Identifying Predictive Correlation Patterns in Service Management Systems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 84, Bamberg University, February 2010. ISSN 0937-3349.

Nr. 85 (2010) Jan Tobias Mühlberg und Gerald Lüttgen: Symbolic Object Code Analysis. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 85, Bamberg University, February 2010. ISSN 0937-3349.

Nr. 86 (2010) Werner Zirkel, Guido Wirtz: Proaktives Problem Management durch Eventkorrelation – ein *Best Practice* Ansatz. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 86, Bamberg University, August 2010. ISSN 0937-3349.

Nr. 87 (2010) Johannes Schwalb, Andreas Schönberger: Analyzing the Interoperability of WS-Security and WS-ReliableMessaging Implementations. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 87, Bamberg University, September 2010. ISSN 0937-3349.

Nr. 88 (2011) Jörg Lenhard: A Pattern-based Analysis of WS-BPEL and Windows Workflow. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 88, Bamberg University, March 2011. ISSN 0937-3349.

Nr. 89 (2011) Andreas Henrich, Christoph Schlieder, Ute Schmid [eds.]: Visibility in Information Spaces and in Geographic Environments – Post-Proceedings of the KI'11 Workshop. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 89, Bamberg University, December 2011. ISSN 0937-3349.

Nr. 90 (2012) Simon Harrer, Jörg Lenhard: Betsy - A BPEL Engine Test System. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 90, Bamberg University, July 2012. ISSN 0937-3349.

Nr. 91 (2013) Michael Mendler, Stephan Scheele: On the Computational Interpretation of CKn for Contextual Information Processing - Ancillary Material. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 91, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 92 (2013) Matthias Geiger: BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 92, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 93 (2014) Cedric Röck, Simon Harrer: Literature Survey of Performance Benchmarking Approaches of BPEL Engines. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 93, Bamberg University, May 2014. ISSN 0937-3349.

Nr. 94 (2014) Joaquin Aguado, Michael Mendler, Reinhard von Hanxleden, Insa Fuhrmann: Grounding Synchronous Deterministic Concurrency in Sequential Programming. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 94, Bamberg University, August 2014. ISSN 0937-3349.

Nr. 95 (2014) Michael Mendler, Bruno Bodin, Partha S Roop, Jia Jie Wang: WCRT for Synchronous Programs: Studying the Tick Alignment Problem. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 95, Bamberg University, August 2014. ISSN 0937-3349.

Nr. 96 (2015) Joaquin Aguado, Michael Mendler, Reinhard von Hanxleden, Insa Fuhrmann: Denotational Fixed-Point Semantics for Constructive Scheduling of Synchronous Concurrency. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 96, Bamberg University, April 2015. ISSN 0937-3349.

Nr. 97 (2015) Thomas Benker: Konzeption einer Komponentenarchitektur für prozessorientierte OLTP- & OLAP-Anwendungssysteme. Bamberger Beiträge zur Wirtschafts-informatik und Angewandten Informatik Nr. 97, Bamberg University, Oktober 2015. ISSN 0937-3349.

Nr. 98 (2016) Sascha Fendrich, Gerald Lüttgen: A Generalised Theory of Interface Automata, Component Compatibility and Error. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 98, Bamberg University, March 2016. ISSN 0937-3349.

Nr. 99 (2014) Christian Preißinger, Simon Harrer: Static Analysis Rules of the BPEL Specification: Tagging, Formalization and Tests. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 99, Bamberg University, August 2014. ISSN 0937-3349.

Nr. 100 (2016) Cedrik Röck, Stefan Kolb: Nucleus - Unified Deployment and Management for Platform as a Service. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 100, Bamberg University, March 2016. ISSN 0937-3349.

Nr. 101 (2016) Michael Mendler, Partha S. Roop, Bruno Bodin: A Novel WCET Semantics of Synchronous Programs. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 101, Bamberg University, June 2016. ISSN 0937-3349.

Nr. 102 (2017) Joaquín Aguado, Michael Mendler, Marc Pouzet, Partha Roop, Reinhard von Hanxleden: Clock-Synchronised Shared Objects for Deterministic Concurrency. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 102, Bamberg University, July 2017. ISSN 0937-3349.

Nr. 103 (2018) Eugene Yip, Erjola Lalo, Gerald Lüttgen, Michael Deubzer, Andreas Sailer: Optimized Buffering of Time-Triggered Automotive Software. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 103, Bamberg University, September 2018. ISSN 0937-3349.

Nr. 104 (2018) Daniel Hallmann: Die COCOMO-Modelle im Licht der agilen Softwareentwicklung. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 104, Bamberg University, October 2018. ISSN 0937-3349.