Kosten-/Aufwandsabschätzung bei komplexen Software Projekten als Basis moderner IT-Governance

Bernhard Peischl und Dietmar Wuksch

bernhard.peischl@soft-net.at

dietmar.wuksch@cicero-consulting.com

Softnet Austria, Cicero Consulting GmbH

Abstract: Software ist heute ein wesentlicher Innovationstreiber in Wirtschaft und Gesellschaft. Es ist daher ein wesentliches Ziel der IT Governance sicherzustellen, dass Investitionen in Software zielgerichtet und mit möglichst geringen Risiken erfolgen. Die fundierte Abschätzung von (komplexen) Projekten kann dazu einen wesentlichen Beitrag leisten. Wir motivieren, dass hierzu Kennzahlen aus dem Bereich Prozesse (z.B. Entwicklungsphase oder Prozessreife), Produktkennzahlen (z.B. Testumfänge, Qualitätsstandards, alg. Komplexität) und Modellkennzahlen in die Schätzung einfließen können und schlagen vor, diese Kennzahlen in Entwicklung und Betrieb mit der Hilfe von Software Cockpits zu operationalisieren. Dies unterstützt den Aufbau einer geeigneten Projektdatenbank für Benchmarking von Projekten und erleichtert damit die systematische Anwendung von Software Aufwandschätzungen.

1 Einleitung

Seit dem breiten Aufkommen von Computern sind nun sechs Jahrzehnte ins Land gezogen, seit vier Jahrzehnten verfügen wir über Mikroprozessoren und in den letzten zwei Jahrzehnten hat das Internet und die damit verbundenen Möglichkeiten kontinuierlich an Bedeutung gewonnen. Wir verfügen heute über die notwendige Technologie, die es ermöglichen wird, ganze Industrien durch den Einsatz von Software zu transformieren.

In den nächsten zehn Jahren werden mehr al 5. Mrd. Menschen Smartphones benutzen – mit der Möglichkeit jederzeit und nahezu überall das Potential des Internets auszuschöpfen. Moderne Software Werkzeuge, internet-basierte Services und die neuen Möglichkeiten des Cloud Computing machen es möglich, ohne große Investitionen in die Infrastruktur, Geschäftsprozesse umzubauen oder vollkommen zu revolutionieren. Beispiele aus der jüngsten Vergangenheit, zeigen klar, wie Software ganze Geschäftsbereiche oder Industrien transformiert:

Der größte Buchhändler ist heute Amazon. Nicht nur Bücher werden Online verkauft, fast alles kann heute bei Amazon Online erstanden werden. Mit dem Produkt „Kindle Digital Books“ hat es Amazon sogar geschafft das Buch selbst in das digitale Zeitalter zu transformieren. Mit dem gleichen Spirit hat Apple’s iTunes traditionelle Platten- oder CD-Läden nahezu überflüssig gemacht.

Auch der Bereich Fotografie gibt Zeugnis davon, wie Software als Geschäftstreiber eine ganze Branche nahezu transformiert. Kaum ein Smartphone hat heute keine software-gesteuerte Kamera, Fotos sind Objekte geworden, die digital gespeichert, und global verbreitet (geteilt um den Begriff aus der Social Media Welt zu benutzen) werden. Flickr ist schon lange in die Fußstapfen von Kodak getreten, erst unlängst musste Kodak Chapter 11[1] anmelden.

Skype ist zu einer der größten Plattformen für Internettelefonie geworden und damit auch Konkurrenz für klassische Telefonanbieter. Werbung auf Goolge ist heute eine ernsthafte Konkurrenz zu klassischem Marketing, und LinkedIn ist der mit Abstand am Schnellsten

[1]Chapter 11 ist ein Abschnitt des Insolvenzrechts der Vereinigten Staaten. Der Begriff bezeichnet in der angelsächsichen Finaz- und Rechtssprache die Insolvenz eines Unternehmens. Quelle: http://www.forbes.com/sites/ericsavitz/2012/01/19/kodak-files-chapter-11/

wachsende Personalrecruiter, kaum ein Personalberater kann es sich heute leisten, darin nicht nach qualifizierten Leuten Ausschau zu halten.

Diese Liste lässt sich beliebig fortsetzen und verdeutlicht klar, dass Software zunehmend als Geschäftstreiber agiert, mit dem Potential ganze Branchen zu transformieren. Aus diesem Grund ist es von entscheidender Bedeutung, die über das Internet vernetzten Systeme und Ihre Interaktionsdynamik zu beherrschen, dies betrifft die Entwicklung genauso wie den Betrieb.

2 Kostentreiber bei IT Projekten und Produkten

Die Entwicklung komplexer, verteilter und oft auch offener IT-Systemen ist dabei oft nicht nur technisch herausfordernd, sondern stellt auch hohe Anforderungen an die IT Governance. Durch die steigende Bedeutung von IT und ihre zunehmende Verbreitung innerhalb von Organisationen birgt Risiken, welche gesteuert und überwacht werden müssen. IT Governance konzentriert sich explizit auf IT Systeme, deren Performanz und das Risikomanagement. Die primären Ziele von IT Governance bestehen darin, sicherzustellen, dass Investitionen in IT einen Unternehmenswert erzeugen und die mit der IT verbundenen Risiken möglichst reduzieren.

Im Rahmen einer Entwicklung ist so z.B. die Entscheidung ob eine Software(teil) selbst entwickelt wird, oder von einem Fremdhersteller zugekauft (oder gemietet) werden kann von entscheidender Bedeutung. Die in der Software Entwicklung übliche Verschmelzung von Planung und Entwicklung erhöht oft die Kosten, mindert die Qualität und trägt auch nicht gerade zur Transparenz der durchgeführten Projekte bei[2].

Weitere Kostentreiber sind die ungenauen (oder nicht klar spezifizierten) Anforderungen bei Projektstart, politische Aufwandschätzungen und ein sehr hoher Zeit- und Kostendruck. Oft fehlt es auch an einem gut etablierten Projektcontrolling und es mangelt in der Praxis an einfach zu operationalisierenden Kennzahlen, die eine Objektivierung z.B. von Projektfortschritt und konkreten Qualitätsattributen (funktionale Korrektheit, Vollständigkeit des Funktionsumfanges, Performanz, Benutzerfreundlichkeit, etc.) erlauben. Eine Vorkalkulation im Projektmanagement bzw. eine Nachkalkulation in Form einer Revision wird oft nur halbherzig durchgeführt, womit auch keine Basis für den Aufbau historischer Daten (im Sinne eines Benchmarkings) zur Verbesserung der Aufwandschätzung gegeben ist. Abbildung 1 illustriert beispielhaft welche Herausforderungen in der Praxis auftreten können.

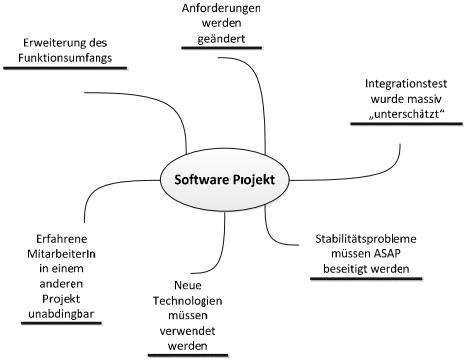

Abbildung 1: Beispielhafte Kostentreiber bei einem IT-Projekt**.**

[2] Hier wird von größeren und komplexen Projekten nach Wasserfall oder V-Modell ausgegangen.

3 Aufwandschätzung für Software-Projekte

Die bisher angeführten Faktoren machen die Aufwandschätzung in einer frühen Phase des Projektes sehr schwierig. Eine jüngst veröffentlichte Studie der Universität Oxford und der Unternehmensberatung McKinsey & Company (Bloch et al. 2012) berichtet, dass 17% aller IT-Projekte die die Existenz des Unternehmens gefährden, und dass das durchschnittliche IT Projekt 45% über Budget mit einem Zeitverzug von 7% abgeschlossen wird. Die Projektaufwandschätzung ist auch deshalb von zunehmender Bedeutung, weil ca. 60% der in Europa durchgeführten Softwareprojekte Festpreisprojekte sind, d.h. z.B. kann der Testaufwand nicht gesondert in Rechnung gestellt werden. Um die Einflussfaktoren, die diese grobe Abweichung von den geplanten Werten verursachen erkennen zu können, illustriert Abbildung 2 den typischen Ablauf einer Abschätzung im Sinne einer Vorkalkulation. Dabei gilt es zu berücksichtigen, dass ein es während der Projektumsetzung in der Regel zu einer schleichenden Erweiterung des Funktionsumfanges um ca. 2% pro Monat kommt[3].

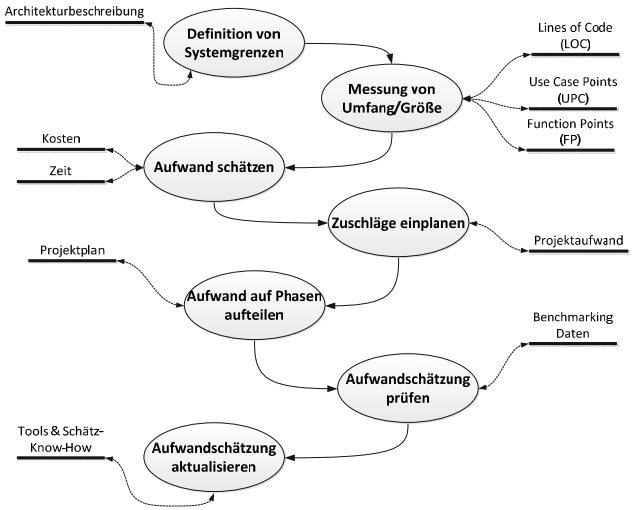

Abbildung 2: Ablauf einer Aufwandschätzung bei einer Schätzklausur und zugehörige Ergebnisse.

Großen Einfluss auf die Schätzung haben also Kenngrößen für Prozesse (wie z.B. die Produktivität von Teams oder die Projektdauer) und Produktkennzahlen (wie z.B. Umfang gemessen in Codezeilen (LOC) oder Funktionspunkten (FP), die Qualität z.B. in Form der Anzahl der Post-Release Bugs oder die Systemkomplexität). Zur genaueren Abschätzung des Testaufwandes sind auch Kennzahlen aus dem Bereich der Modellierung von Software relevant (z.B. Use Case Points).

3.1 Prozesskennzahlen und Aufwandsabschätzung

Unabhängig von der konkreten Schätzmethode die angewendet wird, ist das Ergebnis der Schätzung umso besser, je besser die zugrundeliegenden Informationen sind: Die Function-

[3]Erfahrungswerte aus Gesprächen mit Ziviltechnikern aus dem IT-Bereich (Forum Compliance and IT Governance, Wien).

Point Methode (McConnel 2006), die Object-Point Methode (Antoniol et al. 1999), Use-Case Points (Fronhoff et al. 2006), COCOMO I (Boehm 1981) und COCOMO II (Boehm et al. 2000) und auch die Expertenschätzung (McConnel 2006, Bundschuh and Deckers 2008) sind nur so leistungsfähig, wie dies die zugrundeliegenden Eingangsgrößen in diese Methoden zulassen. Gerade in der sehr frühen Projektphase (oder noch zur Zeit der Angebotslegung) sind diese Informationen nur ungenau vorhanden. Dieser Umstand wird auch als „Cone of Uncertainty" bezeichnet. Abbildung 3 zeigt wie der Unsicherheitsfaktor über die Zeit von anfänglich 4 auf 0 am Ende des Projekts abnimmt.

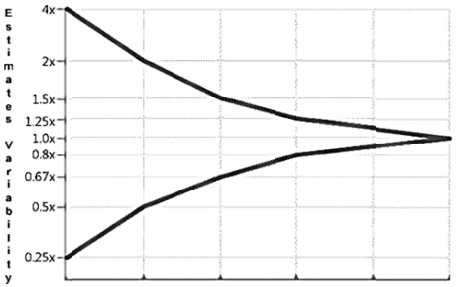

Abbildung 3: Cone of Uncertainty bei der Software Aufwandschätzung.

Aber nicht nur die Projektphase beeinflusst die Genauigkeit der Schätzung, der Schätzfehler hängt auch von der Prozessreife ab. Da z.B. ab einem CMMI Level von 3 („Definierter Prozess") die Aufzeichnung von historischen Projektdaten üblich ist, ist es einfach nachvollziehbar, dass die Schätzung mit dieser Datenbasis entsprechen genauer durchgeführt werden kann. Abbildung 4 zeigt eine schematische Darstellung der Abnahme des Schätzfehlers aufgrund der Verwendung von historischen Daten.

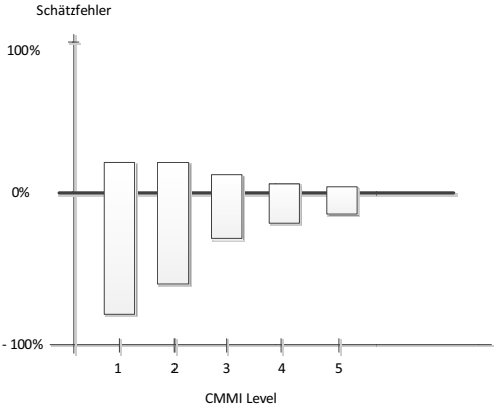

Abbildung 4: Reduktion des Schätzfehlers mit der Hilfe von historischen Daten.

3.2 Produktkennzahlen und Aufwandsabschätzung

Die Bewertung der Aufwände hängt neben der kürzlich besprochenen Phase im Entwicklungsprozess und der Prozessreife im Allgemeinen auch vom Produktumfeld (bzw. im Projektgeschäft vom Projektumfeld) ab. Insofern muss auch eine Anpassung diesbezüglich erfolgen. Folgende Faktoren können dabei den Aufwand/die Kosten beeinflussen:

- Entwicklungsumgebung (PC, Host, Client/Server, Cloud)
- Neuentwicklung bzw. Weiterentwicklung eines Legacy-Systems
- Qualitätsstandards (Testanforderungen, Entwicklungsmethodik)
- Technische Komplexität des Produktes (Standards, defensive Programmierung, Algorithmen, Schnittstellen zu anderen Services und Systemen)
- Technische oder gesetzliche Auflagen (Zertifizierung, Compliance)
- Testumfänge (Testautomatisierung, Entwurf von Testfällen, Testausführung, Bug Management)
- Verwendete Technologien und Programmiersprachen (Java, .Net, C#, Business Komponenten wie z.B. Enterprise Java Beans, Datenbanktechnologie etc.)
- Zielplattform (intern/externes Cloud-Service, Web Service am eigenen Server, etc.)

Als Software Schätzverfahren (McConnel 2006, Bundschuh and Deckers 2008) entwickelt wurden, wurde die Notwendigkeit den Aufwand für das Testen (Testfallentwurf, Testautomatisierung und Auswertung) nicht ausreichend berücksichtigt (Nageswaran 2001). So berücksichtigt die Funktionspunktmethode (McConnel 2006, Bundschuh and Deckers 2008) lediglich das White-Box Testen (Beizer 1990), lässt aber System- und Akzeptanztests (Beizer 1990) außen vor. Da der Testaufwand bis zu 40% des Gesamtprojektaufwandes ausmachen kann (Boehm 1981, Pol et al. 2002), ist es unabdingbar diese Aufwände in der Schätzung zu berücksichtigen. Dies kann z.B. mit der Testpunktanalyse (TPA) erfolgen.

Die Testpunktanalyse (TPA, Koomen et al. 2007) ermöglicht die objektive Schätzung eines System- oder Abnahmetest (Entwicklertests werden wie o.a. mit der FPA abgeschätzt). Um eine TPA durchführen zu können, muss der Umfang der Software bekannt sein. Zu diesem Zweck werden die Ergebnisse der FPA herangezogen. Die Testpunktanalyse lässt sich auch anwenden, wenn die Anzahl der aufzubringenden Teststunden im Voraus festgelegt ist. Durch Ausführung einer Testpunktanalyse können mögliche Risiken klar aufgezeigt werden, indem man die Schätzung aus der objektiven TPA mit der feststehenden Teststundenzahl vergleicht. Eine Testpunktanalyse kann auch angewandt werden, um die relative Bedeutung der verschiedenen Funktionen zu berechnen. Auf der Grundlage dieses Ergebnisses kann die verfügbare Testzeit so optimal wie möglich eingesetzt werden. Des Weiteren kann die Testpunktanalyse benutzt werden, um eine Testschätzung in einem frühen Stadium zu erstellen (Koomen et al. 2007).

Da der TPA eine FPA (Bestimmung des Softwareumfanges) vorangehen muss, ist für die FPA eine ebenso detaillierte Anforderungsspezifikation erforderlich, da man die Bandbreite der Schätzung in Hinblick auf die erzielbare Genauigkeit in Grenzen halten muss. In frühen Phasen des Entwicklungsprozesses bietet es sich daher an, auf Entwurfsdokumente zurückzugreifen. Bei einem objekt-orientierten Entwurf setzt sich die Methode der Use Case Points (UPC) zunehmend durch. Damit sind wir – neben der Dimensionen Produktmetrik und Prozessmetrik – bei einer dritten Dimension angelangt, die Aufwände und damit Abschätzungen beeinflusst - der Modellmetrik.

3.3 Modellkennzahlen und Aufwandsabschätzung

Eine Möglichkeit Projektaufwände in einer sehr frühen Phase abzuschätzen ist die Use Case Point (UCP) Methode (Fronhoff et al. 2006). UCP ist eine Top-Down Schätzmethode, welche

die funktionalen Anforderungen mit der Hilfe von Anwendungsfällen (Use Cases) charakterisiert und den Use Cases eine Maßzahl für deren Komplexität in Form von Use Case Points zuordnet. Der geschätzte Projektaufwand ist proportional zu den Use Case Points. Da Use Cases in unterschiedlicher Granularität beschrieben werden können, hängt das Schätzergebnis stark von der Form der Use Cases ab. In der Unified Modeling Language (UML) wird ein Use Case wie folgt festgelegt (OMG 2001):

„The use case construct is used to define the behavior of a system or other semantic entity

without revealing the entity's internal structure. Each use case specifies a sequence of actions, including variants, that the entity can perform, interacting with actors of the entity."

Diese Definition ist sehr generell und lässt daher viel Spielraum in der Interpretation z.B. wird nicht definiert, nach welchen Kriterien der Anfang und das Ende eines Use Cases festgelegt werden. Genauso ist die Frage nach der Beschreibung der unterschiedlichen Varianten z.B. in Hinblick auf Vollständigkeit in unterschiedlicher Art auslegbar. Zur praktischen Nutzbarkeit müssen daher die Definitionen konkretisiert bzw. ergänzt werden. Z.B. haben sich im Bereich der Aufwandschätzung die folgenden Zusätze als sinnvoll erwiesen (Vigenschow and Weiss 2003):

- Ein Use Case hat keine fachliche vorgesehene Unterbrechung
- Ein Use Case hat genau einen fachlichen Auslöser
- Am Ende eines Anwendungsfalls steht mindestens ein Ergebnis von fachlichem Wert

Die Autoren von (Vigenschow and Weiss 2003) berichten, dass konkrete Use Cases trotz dieser Erweiterungen schnell unübersichtlich werden. Bei einer zu erwartenden hohen Anzahl von Use Cases kann daher nur schwer Nutzen daraus gezogen werden, da die Use Cases sich z.B. nach Abstraktionsgrad, Granularität, Blickwinkel, Sprachstil, Verbindlichkeiten und Umfang unterscheiden. Abhilfe kann z.B. durch die essentielle Beschreibung von Use Cases (Constantine 1999) geschaffen werden. Dabei wird ein Use Case auf eine fachliche Absicht reduziert und stellt eine vereinfachte, abstrakte und technologieunabhängige Sicht mit dem Fokus auf die geschäftliche Intention dar. Weiter ist es sinnvoll, die obig angeführten Zusätze im Sinne eines qualitätsgesicherten Modells des zu beschreibenden Systems zu überwachen. Liegen die Use Cases z.B. in Form der UML vor, kann die Einhaltung der Zusätze durch systematische Qualitätssicherung auf der Basis von Modellmetriken (Opoka 2011, www.squam.info) überwacht werden. Weiter unterstützen moderne Werkezeuge bei der Zählung von z.B. Verzweigungen oder anderen für die Abschätzung relevanten Attributen. Dabei können mit Hilfe der OCL Queries auf der UML formuliert und ausgewertet werden (Opoka 2011).

4 Schätzverfahren im Regelkreislauf

Der Prozess der Aufwandsschätzung ist dabei nicht isoliert zu betrachten, vielmehr ist dieser in eine Art Regelkreis eingebettet. Abbildung 6 zeigt die schematische Darstellung eins idealisierten Kreislaufs.

Der Kreislauf beginnt mit der Vermessung von Prozess-, Produkt- oder auch Modelleigenschaften („Messen"). In der Operationalisierung der Messung und im Aggregieren der relevanten Datenbestände kann heute auf Werkzeuge wie Software Cockpits oder Datawarehouse Technologien zurückgegriffen werden (Larndorfer et al. 2010, Münch et al. 2006). Auf der Grundlage der (aktuellen) Datenbestände erfolgt dann eine Schätzung („Schätzen"). Die dabei anzuwendenden Methoden richten sich nach den verfügbaren Daten bzw. darauf basierenden Kennzahlen. Eine Strategie zur Auswahl geeigneter Schätzverfahren muss die Verfügbarkeit der für die Abschätzung relevanten Daten berücksichtigen. Einfache Strategien wurden auch mit Hilfe von Empfehlungssystemen aufgearbeitet (Peischl et al. 2010), jedoch bilden diese

Systeme lediglich die grundlegenden Regeln ab und eignen sich daher vor allem dann, wenn das Know-How bezüglich Schätzungen nur unvollständig vorhanden ist bzw. zur Überprüfung der Plausibilität und zum Ausloten des Verbesserungspotentials.

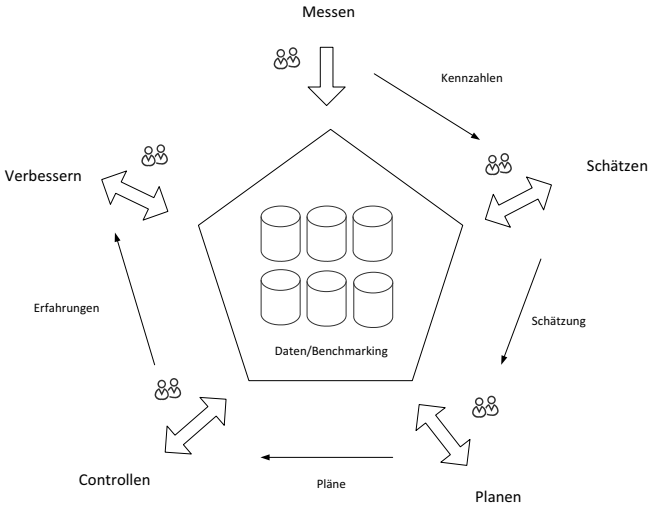

Abbildung 6: Schätzen im Regelkreislauf des Projekts.

In der Praxis ist die Schätzung von Softwareprojekten eine heikle Aufgabe die jedenfalls die Expertise von erfahrenen MitarbeiterInnen erfordert. Daher sollte der Prozess des Schätzens als eine separate Aufgabe im Entwicklungsprozess verankert werden. Auf der Basis der Schätzung müssen dann die Projektpläne („Planen“) adaptiert werden und das Projektcontrolling („Controllen“) gibt nützliche Hinweise und unterstützt damit einen dauerhaften Lern- und Verbesserungsprozess („Verbessern“). Dabei ist es nötig, dass in allen Prozessschritten (Messen, Schätzen, Planen, Controllen und Verbessern) der Datenbestand aktuell gehalten wird, denn dieser stellt das gemeinsame Glied im Regelkreis dar.

4.1 Multidimensionale Sicht auf das Projekt / Produkt

In den vorangehenden Überlegungen haben wir dargestellt, dass für verschiedene Aufgaben wie z.B. Schätzen (aber auch z.B. die Qualitätssicherung) unterschiedliche Dimensionen eines Projekts oder Produkts relevant sind. So fließen in eine Schätzung Kennzahlen bezüglich des Prozesses, des Produkts und der verwendeten Modelle ein. Um diese Vielfalt von Daten im Alltagsgeschäft beherrschbar zu machen, bedarf es einer geeigneten Infrastruktur zum Operationalisieren der anfallenden Daten und zur Berechnung der Kennzahlen.

Für die ganzheitliche Sicht auf die verschiedenen Qualitätsattribute ist es notwendig, Daten aus unterschiedlichen Qualitätsperspektiven/Projektperspektiven und Detailierungsstufen zu betrachten. Konkrete Kennzahlen wie z.B. Durchlaufzeiten von Change Requests, die Testabdeckung oder die Konsistenz von Interfaces lassen sich dann anhand von unterschiedlichen Dimensionen analysieren. Die Daten werden dabei nicht wie in einem relationalen Datenmodell als flache Tabellen sondern in einem multidimensionalen Datenwürfel – dem Cube – vorverarbeitet und gespeichert. Z.B. entsprechen in Abbildung 6 die Dimensionen Produkt, Zeit und Status den Achsen und spannen (im Allgemeinen einen mehrdimensionalen) Würfel auf. Jede Zelle des Würfels enthält dann Werte von Kennzahlen z.B. den Detailierungsrad von Use Cases von Produkt A im Jahr 2010 im Zustand 'Opened'. Mit den OLAP Funktionen

(Codd 1993) kann der Datenwürfel situativ analysiert werden (z.B. können typische Muster erkannt werden etc.). Diese Analysierbarkeit einschließlich des schrittweisen Verfeinerns und Verdichtens (Drill-Down, Roll-Up) unterstützt das menschliche Kalkül in der Entscheidungsfindung bezüglich einer konkreten Fragestellung wie z.B. der Anwendung von Schätzverfahren. So könnte z.B. die mittlere Komplexität von Use Cases im Zustand 'Opened' beim Produkt A im letzten halben Jahr relevant sein. Durch die Einschränkung der Dimension Zeit, Zustand und Produkt ergibt sich der (kleinere, strich-punktiert) dargestellte Datenbereich (siehe Abbildung 7).

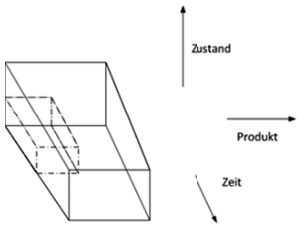

Abbildung 7: Beispielhafte Darstellung eines Cubes mit drei Dimensionen

Um Trends und Zusammenhänge in den Daten erkennen zu können werden im Software Cockpit interaktive Möglichkeiten zum Navigieren im Datenwürfel bereitgestellt z.B. Slicing, Pivotisierung, Drill-Down und Drill-Up. Unter einer Slice verstehen wir einen Unterwürfel unseres Datenwürfels indem eine oder auch mehrere Dimensionen auf einen Wertebereich fixiert sind. Z.B. könnten wir an Kennzahlen bezüglich eines konkreten Produktes interessiert sein, und daher unsere interaktive Analyse durch Anwendung einer Slice auf den relevanten Unterwürfel spezialisieren. Die Dimensionen des so entstandenen Unterwürfels können interaktiv weiter detailliert (Drill-Down) bzw. abstrahiert (Drill-Up) werden. Weiter unterstützt OLAP eine effiziente Pivotisierung z.B. ein Vertauschen der Dimensionen in einer Tabellenansicht.

Abbildung 8 zeigt die Architektur eines Software Cockpits (Lang and Peischl 2011) und stellt eine Möglichkeit der Umsetzung des oben angeführten Regelkreises dar. Das Kernstück ist ein Datawarehouse, dem einerseits ein ausgeflachtes Datenbankschema, in dem die Dimensionen und Fakten gespeichert sind, und andererseits die Definition der Cubes, Metriken und Aggregate, zu Grunde liegen. Die Daten werden unter Verwendung von Adaptern aus den Artefakten gemappt und importiert. Auch Metriken und eventuell notwendige Aggregattabellen werden in dieser Schicht berechnet. Zu den Artefakten zählen Daten aus dem Issue Management, wie z.B Bugzilla, UML Modelle und Source Code aus der Versionsverwaltung, wie beispielsweise SVN.

Im Umfeld der Softwareentwicklung berichten einige Arbeiten über die Einführung von Software Cockpits (Larndorfer et al. 2008, Bennicke et al. 2007, Münch et al., Münch et al. 2006) bei Unternehmen. Meist wird dabei auf den Prozess der Einführung, die technische Konzeption bzw. die Aggregation und Abstraktion der im Entwicklungs- oder Wartungsprozess anfallenden Datenbestände fokussiert. Auch die methodische Entwicklung von Kennzahlsystemen bzw. Qualitätsmodellen wird adressiert. Leider gibt es kaum Arbeiten die von den Effekten der Einführung auf die IT-Governance (und auch die Software Qualität) berichten. In Hinblick auf Schätzmethoden ist vor allem der Aufbau einer Benchmarking Datenbank für Kostenschätzungen eine Herausforderung.

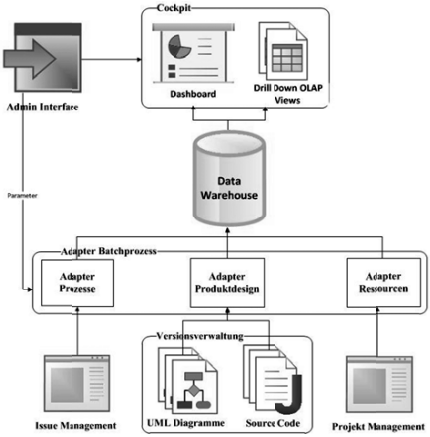

Abbildung 8: Architektur eines Software Cockpits

4.2 Benchmarking – Lokale vs. globale Projektdaten

Durch die Verfügbarkeit von Datensammlungen zu Projekten wie ISBSG (www.isbsg.org) oder CSBSG (www.totalmetrics.com/function-points-groups/csbsg) kann eine Benchmarking durchgeführt werden. Jedoch hat sich in einigen Studien gezeigt, dass im Bereich der Aufwandschätzung die Verwendung von globalen (d.h. z.B. verschiedenen Branchen oder Unternehmen) Datensammlungen über Projektverläufe kritisch ist und nicht immer zum gewünschten Erfolg führt. Neuere Arbeiten demonstrieren, dass lokale (d.h. z.B. selbst aufgebaute Datenbestände, eigene Projekthistorie etc.) die besseren Schätzungen und Prädiktionen ermöglichen (Lum et al. 2006, Menzies et al. 2007). Werden globale Daten verwendet, kann es sein, dass der Projektkontext jenen des relevanten Projekts nicht trifft. Die aus solchen Datenbeständen abgeleiteten Schätzungen treffen nicht die Realität und die von den Schätzungen abgeleiteten Maßnahmen verfehlen ihr Ziel. Andrerseits ist der Aufbau einer lokalen Projektdatenbank mit Investitionen und Aufwänden verbunden, was anfänglich zur Verwendung der global verfügbaren Daten über Projektabschätzungen verleitet.

5 Schlussbemerkungen

Dieser Artikel motiviert die Rolle der Informationstechnologie als Innovationstreiber in Wirtschaft und Gesellschaft. Informationstechnologie und Software nimmt heute nicht ausschließlich den Stellenwert einer Supportfunktion ein und ist in vielen Lebens- und Geschäftsbereichen ein Enabler für Innovationen. Daher ist es heute auch ein primäres Ziel der IT Governance sicherzustellen, dass Investitionen in die IT unmittelbar einen Wert für das Unternehmen erzeugen und möglichst geringe Risiken tragen. Die Abschätzung der Kosten eines Softwareprojektes oder Produktes ist nach wie vor eines der Hemmschuhe in der modernen IT Governance, oft wird eine Vorkalkulation im Projektmanagement bzw. eine Nachkalkulation in Form einer Revision nur halbherzig durchgeführt.

In diesem Artikel motivieren wir die Aufwandschätzung von Softwareprojekten und die wesentlichen Problempunkte dabei, gehen kurz auf die heute verfügbaren Methoden ein und illustrieren welche Kostentreiber es in der heutigen Softwareentwicklung gibt. Wir motivieren,

dass Kennzahlen aus dem Bereich Prozesse (z.B. Entwicklungsphase oder Prozessreife), Produktkennzahlen (z.B. Testumfänge, Qualitätsstandards, alg. Komplexität) und Modellkennzahlen in die Schätzung einfließen können und schlagen vor, diese Kennzahlen in den Regelkreislauf der Software (Entwicklung und Betrieb) miteinzubeziehen. Insbesondere motivieren wir die Verwendung von historischen Projektdaten (Benchmarking) mit Hilfe eines Software Cockpits (Datawarehouse zur Aggregation von Datenbeständen über den Produktlebenszyklus der Software). Schließlich diskutieren wir kurz Herausforderungen beim Benchmarking (globale vs. lokale Projektdaten).

Danksagung
Die hier präsentierten Arbeiten wurden im Rahmen des Kompetenznetzwerkes Softnet Austria II (www.soft-net.at, COMET Programm der Forschungsförderungsgesellschaft) erarbeitet und werden vom Bundesministerium für Wirtschaft Familie und Jugend (bmwfj), dem Land Steiermark, der Steirischen Wirtschaftsförderungsgesellschaft mbH (SFG), und der Stadt Wien durch das Zentrum für Innovation und Technologie (ZIT) unterstützt.

Literaturverzeichnis

(Antoniol et al. 1999) Giuliano Antoniol, Chris Lokan, Gianluigi Caldiera, and Roberto Fiutem, A Function Point-Like Measure for Object-Oriented Software, Empirical Softw. Eng. 4, 263-287, 1999.

(Beizer 1990) B. Beizer, Software Testing Tchniques, John Wiley & Sons, ISBN 0-442-20672-0.

(Bennicke et al. 2007) M. Bennicke, F. Steinbrückner; M. Radicke, & J.-P. Richter, Das sd&m Software Cockpit: Architektur und Erfahrungen., in Rainer Koschke; Otthein Herzog; Karl-Heinz Rödiger & Marc Ronthaler, ed., 'GI Jahrestagung (2)' , GI, , pp. 254-260, 2007.

(Bloch et al. 2012) M. Bloch, S. Blumberg, J. Laartz, Delivering large-scale IT projects on time, on budget, and on value, McKinsey and Company 2012.

(Boehm 1981) B.W. Boehm, Software Economics, Prentice Hall, 1981.

(Boehm 2000) Barry W. Boehm, Donald J. Reifer, Ellis Horowitz, Chris Abts, A. Winsor Brown, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

(Bundschu and Deckers 2008) M. Bundschuh, C. Deckers, The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, Springer Verlag, 2008.

(Codd 1993) E.F. Codd, S.B. Codd, C.T. Salley, Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate, Codd & Date, 1993.

(Fronhoff et al. 2006) Frohnhoff, S.; Jung, V.; Engels, G.: Use Case Points in der industriellen Praxis, In Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress, Abran, A. et al. Eds., pp. 511-526, 2006, Shaker Verlag.

(Koomen et al. 2007) Tim Koomen; Leo Van der Aalst; Bart Broekman; Michiel Vroon, TMap® Next: Praktischer Leitfaden für ergebnisorientiertes Softwaretesten, dPunkt Verlag, 978-3-89864-461-7.

(Larndorfer et al. 2010) Stefan Larndorfer, Rudolf Ramler, Clemens Buchwiser, Experiences and Results from Establishing a Software Cockpit at BMD Systemhaus, Software Engineering and Advanced Applications, Euromicro Conference, pp. 188-194, 2009 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009.

(McConnel 2006) Steve McConnel, Software Estimation: Demystifying the Black Art, Redmond, Wa., Microsoft Press, 352 pages, 2006.

(Münch et al. 2006) J. Münch, J. Heidrich, F. Simon, C. Lewerentz, B. Siegmund, R. Bloch, B. Kurpicz, M. Dehn, Soft-Pit: Ganzheitliche Projekt-Leitstände zur ingenieurmäßigen Software-Projektdurchführung, Proceedings of the status conference of the German research program Software Engineering 2006, June 26-28, 2006.

(OMG 2001) OMG UML Specification 1.4, 2001 (siehe http://www. omg.org/uml).

(Opoka 2011) Joanna Chimiak-Opoka, Measuring UML models using metrics defined in OCL within the SQUAM framework. In Proceedings of the 14th international conference on Model driven engineering languages and systems (MODELS'11), 47-61, 2011, Springer-Verlag.

(Peischl et al. 2010) B. Peischl, M. Zanker, M. Nica and Wo. Schmid, Constrained-based Recommendation for Software Effort Estimation, Journal of Emerging Technologies in Web Intelligence, Special Issue: Recommender Systems for Web Intelligence, Vol 2, No 4, Academy Publisher, 2010.

(Pol et al. 2002) Martin Pol, Tim Koomen, Andreas Spillner: Management und Optimierung des Testprozesses, ISBN 978-3-89864-951-3.

(Vigenschow and Weiss 2003) U. Vigenschof , Ch. Weiss, Das Essenzschritt-Verfahren: Aufwandsschätzung auf der Basis von Use Cases, März/April 2003.