von christian kamm, johannes siedersleben, daniel schick und alexandre saad

SYSTEMATISCHE AUFWANDSSCHÄTZUNG FÜR SOFTWARE IM FAHRZEUG

Software im Automobil entwickelt sich zum wettbewerbs-differenzierenden Produkt. Die systematische Aufwandsschätzung für Software im Fahrzeug ist daher für die Automobilhersteller wichtig: Man benötigt sie für die stabile Planung von Fahrzeugprojekten und die Entwicklung von neuen Geschäftsmodellen zwischen Automobilhersteller und Zulieferindustrie. In dem Artikel wird ein Verfahren vorgestellt, mit dem sich der Entwicklungsaufwand für Software im Fahrzeug systematisch schätzen lässt

Wie schätzt man den Entwicklungsaufwand für softwarebasierte Funktionen im Auto? Welche Faktoren sind aufwandsrelevant und welche Anforderungen ergeben sich daraus an das Lastenheft? Welche Schätzverfahren verwendet man in anderen Bereichen des Software-Engineerings und in welchem Umfang lassen sich diese auf die Entwicklung von Software im Fahrzeug übertragen?

Diese Fragen beschäftigen derzeit nicht nur die Einkaufsabteilungen, auch in der Vorentwicklung der Automobilhersteller wird nach praxistauglichen Antworten gesucht.

Paradigmenwechsel: Software als Produkt

Was treibt die *Automobilhersteller (OEMs*[1]*)* um, sich mit diesen Fragen zu beschäftigen? Vor wenigen Jahren hatten Fahrzeuge nur wenige, kaum vernetzte elektronische Steuergeräte, wie elektrische Fensterheber oder Sitzverstellung. Jedes Gerät erfüllte eine klar abgegrenzte Funktion. Zulieferer entwickelten die Steuergeräte, die Integration des Gesamtsystems beim OEM war aufgrund der geringen Vernetzung ohne Probleme möglich. Soft- und Hardware der Steuergeräte bildeten eine Einheit und die Hardwarekosten bestimmten den Preis. Die Kosten der Softwareentwicklung wurden auf die Stückzahl umgelegt und waren für den OEM in der Regel nicht transparent.

Dieser Ansatz reicht heute nicht mehr aus, denn die Bordnetze haben sich weiterentwickelt. In Oberklassewagen kommunizieren nahezu hundert Steuergeräte über fünf verschiedene Bus-Systeme. Daher ist die Integration für alle Automobilhersteller eine echte Herausforderung – das belegen die jüngsten Presseberichte.

Wie begegnen OEMs dieser Herausforderung? Die wichtigsten Ideen sind offene Systemarchitekturen mit standardisierten Schnittstellen, hardwareunabhängigen Softwareschichten und einfach migrierbaren Funktionen. Zentrale Verarbeitungseinheiten mit offener Softwarearchitektur ermöglichen neue Funktionen auf vorhandener Hardware – die Software-Tankstelle wird Realität.

Für die Entwicklung von Elektronik im Auto bedeutet das einen Paradigmenwechsel: Software im Auto ist nicht mehr bloßes Beiwerk von Steuergeräten, sondern ein Produkt mit eigenem Wert und eigenem Preis. Die Aufgaben von Zulieferer und OEM werden neu verteilt, neue Geschäftsmodelle werden benötigt. Für die Automobilhersteller bedeutet das, dass sie sich zunächst Transparenz über die Softwareentwicklungskosten im Fahrzeug verschaffen müssen.

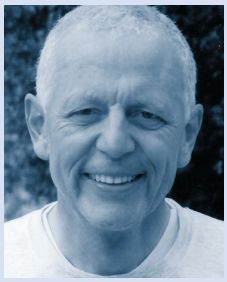
Systematische Aufwandsschätzung im Überblick

Softwarefunktionen haben einen Wert und einen Preis – aber wie wird der ermittelt? Hier ein Beispiel: Moderne Autos speichern die Einstellungen von Sitz, Rückspiegel, Klimaanlage und anderen Geräten für mehrere Fahrer. Bisher besitzt jedes einzelne Steuergerät eine solche Memory-Funktion. Was würde die Entwicklung einer Software

[1]) *OEM = Original Equipment Manufacturer*

die autoren

Christian Kamm (E-Mail: christian.kamm@sdm.de) ist seit sechs Jahren Berater bei der sd&m AG. Software im Automobil ist seit einem Jahr sein Schwerpunktthema.



Prof. Dr. Johannes Siedersleben (E-Mail: johannes.siedersleben@sdm.de) ist Professor für Informatik an der FH Rosenheim und wissenschaftlicher Leiter von sd&m Research.

Daniel Schick (E-Mail: daniel.schick@bmw-carit.de) ist Softwareentwickler bei der BMW Car IT GmbH. Sein Arbeitsschwerpunkt ist das Prototyping neuartiger Systemfunktionen im Fahrzeug.

Dr. Alexandre Saad (E-Mail: alexandre.saad@bmw-carit.de) ist seit drei Jahren stellvertretender Geschäftsführer der BMW Car IT GmbH, einer Tochter der BMW Group.

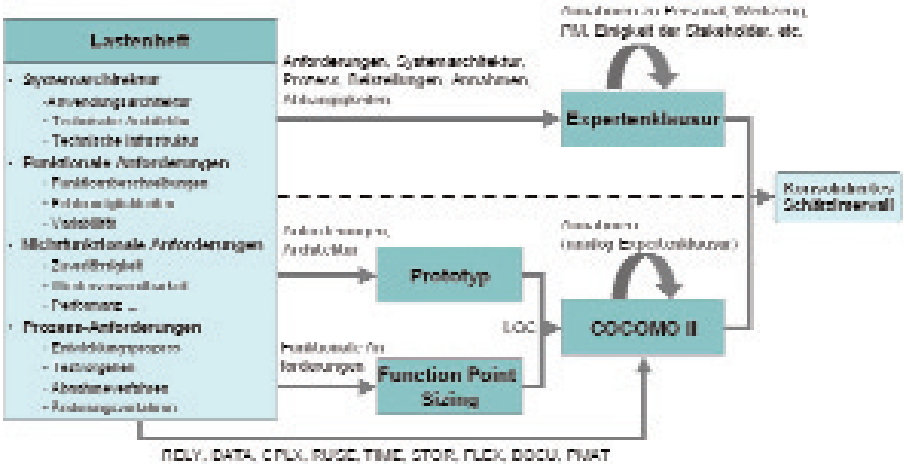


Abb. 1: Schätzverfahren im Überblick

kosten, die diese Einstellwerte zentral verwaltet und nicht mehr getrennt in jedem Gerät?

Dies war der Ausgangspunkt eines gemeinsamen Projekts der Firmen BMW Car IT und sd&m: Den Aufwand für diese Komfort-Funktion haben wir nach allen Regeln der Kunst geschätzt und dabei die Besonderheiten von Software im Auto soweit wie möglich berücksichtigt. Wir beschreiben im Folgenden, welche Verfahren wir angewandt haben und was dabei herausgekommen ist.

Unser Vorgehen ruht auf drei Säulen:

- Lastenheft,
- Expertenklausur und
- algorithmische Schätzung mit *Function Point Sizing* (vgl. [IFPUG]) und *COCOMO-II*-Bewertung (*Constructive Cost Model*, vgl. [Boe00]).

Das Lastenheft ist die Grundlage jeder Schätzung. Es enthält mindestens alle funktionalen Anforderungen; im letzten Abschnitt wird beschrieben, was im Lastenheft sonst noch stehen sollte. Die Expertenklausur ist das wichtigste und zuverlässigste Element unserer Schätzung. Das Ergebnis der Experten validieren wir durch ein algorithmisches Verfahren (siehe Abb. 1). Im Abschnitt „Expertenklausur" wird beschrieben, wie man dabei vorgeht.

Das Ergebnis der Experten validieren wir durch ein algorithmisches Verfahren. Im Abschnitt „COCOMO II" wird beschrieben, wie wir dazu *Function Point Sizing* und COCOMO II miteinander kombinieren. COCOMO II gestattet es, nicht funktionale Faktoren (Zuverlässigkeit, Performanz etc.) und Prozessfaktoren (z. B. Einigkeit der Stakeholder bezüglich der Projektziele) zu berücksichtigen, die gerade bei Software im Auto eine wichtige Rolle spielen.

Wenn die beiden Schätzungen konvergieren, ist das Ergebnis vertrauenswürdig. Wenn nicht, analysiert man die Zahlen und versucht festzustellen, wo unterschiedliche Annahmen oder Einschätzungen vorliegen. Das vorgeschlagene Schätzverfahren ist aufwändig, denn es setzt ein ausführliches Lastenheft voraus, und es funktioniert nur mit qualifizierten Experten. Aber das ist kein Mangel des Verfahrens: Ohne ein detailliertes Lastenheft sollte man niemals schätzen und komplexe Entwicklungsprojekte sollte man nur mit qualifizierten Experten angehen – die braucht man sowieso. Insofern erzeugt unser Schätzverfahren keinen Extra-Aufwand, sondern fordert nur Selbstverständlichkeiten.

Brutto oder Netto: Was schätzen wir überhaupt?

Wir schätzen den Entwicklungsaufwand in *Bearbeitertagen (BT)* und *Bearbeitermonaten (BM)*, nicht in Euro. Die Frage der Preisfindung – wie viel darf der Zulieferer verlangen, wie viel muss der OEM bezahlen – steht auf einem ganz anderen Blatt. Der geschätzte Aufwand in BM ist Grundlage für die Preisfindung, aber nicht mehr. Kostensätze, Gewährleistung, Exklusivitätsrechte und Alleinstellung bestimmen letztlich den Preis.

Erste Voraussetzung für ein geordnetes Schätzverfahren sind klare Begriffe, denn sonst schätzt jeder etwas anderes. Die Schätzeinheit ist der normale Bearbeitertag (1 BT = 8h), der das tägliche Rauschen (Mail, Telefonate, Kaffeepause etc.) bereits enthält.

Der *Nettoaufwand* ist der Aufwand für die eigentliche Erstellung der Software und spiegelt die produktive Leistung des Entwicklungsteams wider. Zum Nettoaufwand addieren sich *Querschnittsaufwand* für Projektleitung, Kommunikation und Technik (z. B. Konfigurationsmanagement) und *Nebenaufwand* für Einarbeitung, Reise, Teamaufbau. So ergibt sich der *Bruttoaufwand* als Summe von Netto-, Querschnitts- und Nebenaufwand.

Expertenklausur

Expertenklausuren sind keine ominösen Orakelrunden, in denen sich die Teilnehmer fest auf den Bauch klopfen und Aufwandszahlen generieren. Stattdessen schätzen Experten in einem strukturierten Prozess den Bruttoaufwand in den im Folgenden beschriebenen drei Phasen. Ein Moderator führt die Gruppe durch diese Phasen.

Phase 1: Stückliste erstellen

Die Stückliste ist die Grundlage der Expertenschätzung. Sie enthält alle Aufwandsposten: Geschätzt wird nur, was in der Stückliste steht, alles andere fällt unter den Tisch. Die Experten braucht man also schon für die Stückliste, nicht erst für die Schätzung selbst. Die Stückliste enthält:

- alle Aufwandsposten des Nettoaufwands, die sich im Wesentlichen aus der Liste der geplanten Komponenten ergeben (wir schätzen den Aufwand pro Komponente und Projektphase),
- alle Posten für den Querschnittsaufwand und
- alle Posten für den Nebenaufwand.

Phase 2: Schätzung in der Gruppe

Schätzungen verschiedener Experten sind nur vergleichbar, wenn alle von denselben Randbedingungen ausgehen. Daher sind diese normiert: Wir unterstellen stabile Anforderungen, klare und konsistente Ziele der Stakeholder, einen kompetenten Projektleiter, einen erfahrenen Chefarchitekten und ein ausgebildetes Team. Zudem

Methode	Beschreibung	Aufwand	Knackpunkt
Einzelschätzung	Experte definiert Arbeitspakete und schätzt Aufwand	gering	Experte
Mehrfachbefragung	Mehrere Experten führen unabhängig und einmalig eine Einzelschätzung durch	mittel	Experten
Delphi	Strukturierte anonyme Mehrfachbefragung in mehreren Iterationen	groß bis sehr groß	Experten
Schätzklausur	Strukturierte, kollektive Mehrfach-befragung mit Gruppendynamik auf Basis einer Stückliste	mittel	Experten, Stückliste
Prozentsatzmethode	Über Phasen-Kennzahlen wird der Gesamtaufwand bestimmt	gering	Nur zur Plausi-bilisierung
Vergleichsmethode	Über ähnliche Projekte wird per Analogie-schluss auf den Aufwand für das aktuelle Projekt geschlossen	mittel	Ähnliche Projekte
Function Point	1.Unbewertete FP (~Größe der Software) 2.Bewertete FP (+/- 35%) 3.Aufwandsermittlung auf Basis einer Erfahrungsdatenbank (Aufwand dafür: sehr groß!)	mittel bis groß	Erfahrungs-DB
COCOMOII	1.Sizing (SW-Größe in LOC) 2.Rating der Faktoren über Tabellen 3.Berechnung des Schätzintervall	Sizing:mittel Rating:gering	Sizing

Tabelle 1: Gängige Schätzverfahren

erläutert der Moderator alle Annahmen über Systemarchitektur, Phasenergebnisse und Entwicklungsprozess, denn all das beeinflusst den Aufwand.

Nur so erhält man vergleichbare Schätzungen. Widrige Umstände wie technische Risiken oder Unerfahrenheit werden später in der Nachkalibrierung berücksichtigt.

Der Moderator führt das Schätzverfahren anhand der Stückliste: Jedes Element der Stückliste wird für sich geschätzt. Dazu schreibt jeder Experte seine Zahlen für sich auf eine Karte. Der Moderator sammelt die Karten ein und publiziert die Ergebnisse (Metaplan oder Excel). Die Gruppe diskutiert die Schätzungen mit dem Ziel, pro Komponente und Entwicklungsphase einen Wert zu finden, den alle vertreten. Dabei entscheiden die besseren Argumente – das arithmetische Mittel ist keine Hilfe.

Es ist wesentlich, die Verhältnisse der Komponenten im Auge zu behalten. Dies ist wichtig, da im Anschluss an eine Expertenschätzung oft noch mit einem Faktor kalibriert wird, der die Abweichung der realen Projektsituation von den normierten Annahmen berücksichtigt.

Phase 3: Plausibilisieren

Über verschiedene Methoden (Prozentsatz- oder Vergleichsmethode, siehe Tabelle 1) plausibilisiert man das Ergebnis im Anschluss an die Expertenklausur. Bei unerklärbaren Unstimmigkeiten wird die Schätzung wiederholt.

Wir sichern das Ergebnis der Expertenklausur über eine algorithmische Schätzung mit *Function Point Sizing* und COCOMO II ab.

COCOMO II

Parallel zur Expertenklausur wenden wir COCOMO II in drei Schritten an:

- *Sizing:* Schätzung der *Lines of Code (LOC)* mit Hilfe von unbewerteten *Function Points* oder auf Basis eines Prototypen.
- *Rating:* Ermittlung der 17 Aufwands- und fünf Prozessfaktoren, die COCOMO II vorsieht.
- Anwendung der empirischen COCOMO II-Formel.

COCOMO II liefert ein Konfidenz-Intervall für den Bruttoaufwand (vgl. [Boe00], [Chu98-a]), aber keine Aufschlüsselung nach Netto-, Querschnitts- und Nebenaufwand.

Sizing

Die *Function-Point*-Methode gestattet es, die Größe eines Softwaresystems in LOC unabhängig von der genutzten Technik zu messen (vgl. [Gar00]). Aber dazu benötigt man die Beschreibung der Funktionen und der Daten – im Grunde also die komplette Spezifikation – und wir haben nur ein Lastenheft.

Daher ziehen wir eine Heuristik zu Rate, die eine grobe Funktions- und Datenbeschreibung einfordert und als Ergebnis eine Schätzung für die Zahl der unbewerteten *Function Points* liefert. Diese Heuristik hat sich bei betrieblichen Informationssystemen gut bewährt. Wir wenden sie unmittelbar auf Software im Auto an, sehen aber noch zusätzlichen Verifikationsbedarf. Über die Heuristik oder die detaillierten Zählregeln der International Function Point User Group (IFPUG) (vgl. [IFPUG]) erhält man die Anzahl der unbewerteten *Function Points*. Über Umrechnungstabellen (vgl. [Boe00], S.20) lässt sich daraus die Anzahl der LOC für die geforderte Programmiersprache bestimmen. Die Anzahl der LOC lässt sich auch auf der Basis eines Prototypen hochrechnen – eine weitere Plausibilisierung des Software-*Sizings*.

Rating

COCOMO II verwendet 17 Aufwandsfaktoren (*Effort Multipliers, EM_i*) und 5 Prozessfaktoren (*Scaling Factors, SF_i*). Man bestimmt sie mit Hilfe von Entscheidungstabellen (vgl. [Boe00], S. 30 ff). Für jeden Faktor gibt es sechs Kategorien von *sehr niedrig* bis *extrem hoch*. Hier zwei Beispiele:

- Der Aufwandsfaktor *Zuverlässigkeit (RELY)* ist sehr niedrig, wenn ein Softwarefehler nur leichte Unannehmlichkeiten erzeugt; er ist normal, wenn er wieder herstellbare Datenverluste verursacht; und er ist sehr hoch, wenn Lebensgefahr droht.
- Der Aufwandsfaktor *Anwendungserfahrung* (APEX) ist sehr niedrig, wenn das Team im Schnitt nicht mehr als zwei Monate Erfahrung mitbringt; er ist sehr hoch, wenn das Team mindestens sechs Jahre Erfahrung hat.

COCOMO II-Formel

Die COCOMO II-Formel basiert auf jahrelanger empirischer Forschung in Zusammenarbeit mit 26 führenden Wirtschaftsunternehmen auf der Basis von vielen Hundert evaluierten Projekten (vgl.

[Boe00], [COCO], [Chu98-a], [Chu98-b]). COCOMO II ermittelt den Bruttoaufwand aus der Größe der Software (gemessen in LOC), den Aufwands- und den Prozessfaktoren:

$$A = C \cdot Size^{0,91+0,01\cdot\sum_{i=1}^{5} SF_i} \cdot \prod_{i=1}^{17} EM_i$$

In der Formel ist *A* der Bruttoaufwand in BM, *C* eine jährlich aktualisierte Kalibrierungskonstante (in diesem Jahr ist C=2,94), und *Size* die Größe der Software in Kilo LOC.

Der geschätzte Bruttoaufwand *A* liegt mit einer Wahrscheinlichkeit von 52% in einem 30%-Intervall um den tatsächlichen Aufwand (vgl. [Chu98-a], [Chu98-b]). Durch firmenspezifische Kalibrierung kann man diese Wahrscheinlichkeit auf ca. 65% verbessern (vgl. [Chu98-a]). Die COCOMO-Schätzung liefert also ein Konfidenz-Intervall für den Bruttoaufwand. Die untere Grenze des Konfidenz-Intervalls nennen wir die optimistische Schätzung, die obere Grenze die pessimistische Schätzung.

Ein Werkzeug der University of Southern California („USC-COCOMOII.2000.0", vgl. [Boe00]) übernimmt die Rechenarbeit und ermittelt aus den LOC und den gewichteten Faktoren den Aufwand in Bearbeitermonaten. Die Bewertung der Faktoren mit Hilfe dieser Tabellen ist einfach.

Ergebnisse

Im Pilotprojekt haben wir das zweigleisige Verfahren aus Expertenklausur und COCOMO II wie beschrieben durchgeführt.

Die Ergebnisse beider Schätzungen decken sich erstaunlich gut: Der Bruttoaufwand der Expertenklausur ist nahezu identisch mit der pessimistischen Schätzung von COCOMO II. Die optimistische Schätzung von COCOMO II entspricht etwa dem geschätzten Nettoaufwand der Expertenklausur. Daher erscheint uns die Schätzung der Expertenklausur plausibel, zumal wir für die Bestimmung der erwarteten Anzahl an LOC sowohl die Heuristik als auch einen Prototypen herangezogen haben.

Dieses Verfahren hat auch in anderen Projekten im Bereich der betrieblichen Informationssysteme gut funktioniert.

Wenn die beiden Schätzverfahren divergieren, wird man auf keinen Fall den Mittelwert bilden, sondern die Abweichung analysieren und den Grund dafür herausfinden. So kann man falsche Annahmen oder grobe Fehleinschätzungen frühzeitig erkennen und korrigieren.

Einflussfaktoren

Welche Faktoren sind aufwandsrelevant und wie wirken sich die Besonderheiten von Software im Auto aus? Die Antworten von COCOMO II decken sich mit unseren Erfahrungen (siehe auch Tabelle 2):

- Die **Personalfaktoren** sind am wichtigsten – das gilt bei Software im Auto genauso wie bei betrieblichen Informationssystemen. Die Erfahrung der Entwickler mit Anwendung, Programmiersprache und Technik ist durch nichts zu ersetzen. Ein eingespieltes, erfahrenes und konstant besetztes Idealteam ist zwei- bis dreimal so produktiv wie ein durchschnittliches Team mit normaler Fluktuation.
- **Produktfaktoren** sind automobilspezifisch. Die Anforderungen an die Qualität von Software im Auto sind hoch, denn Softwarefehler verursachen hohe Rückrufkosten, im schlimmsten Fall sind Menschenleben gefährdet. Dies multipliziert den Aufwand etwa mit dem Faktor drei. Die Ursachen hierfür im Einzelnen sind: Hardwareschnittstellen sind schwer zu bedienen, die Echtzeitanforderungen sind hoch. Software soll über Produktlinien hinweg wiederverwendet werden und die Ansprüche an die Dokumentation der Software sind hoch.
- **Plattformfaktoren** sind ebenfalls spezifisch für Software im Auto. Hardware und Betriebssysteme im automobilen Bereich sind bislang wenig standardisiert. CPU- und Hauptspeicherrestriktionen bestimmen nach wie vor das Softwaredesign von eingebetteten Systemen im Auto: Das bedeutet eine Verdopplung des Aufwands gegenüber Standard-Plattformen.
- **Prozessfaktoren** spielen sowohl im Auto als auch bei betrieblichen Informationssystemen eine wichtige Rolle: Stabilität der Anforderungen, Verständnis und Konsistenz der Projektziele, Reife des Entwicklungsprozesses, aktives Risikomanagement und Kooperation aller Projektbeteiligten beeinflussen den Entwicklungsaufwand überproportional. In COCOMO II modelliert man diese Faktoren schwach exponenziell zur Softwaregröße. Als Faustregel gilt: Stabile Anforderungen und ein reifer Entwicklungsprozess können den Aufwand auf die Hälfte reduzieren: Mit Anforderungs-, Risiko- und Projektmanagement lässt sich richtig Geld sparen. Das wäre auch ein Grund die organisatorische Kluft zwischen OEM und Zulieferer zu überbrücken. Entwicklung in gemischten Teams – schon lange üblich in der Business-IT – könnte ein gangbarer Weg dafür sein.

COCOMO2 Faktor 2-3 für safety-kritische Anwendungen	COCOMO2 Expertenteam doppelt so produktiv wie Nominalteam	COCOMO2, sd&m Unsicherheitsfaktor 1,5 bei früher Schätzung auf Basis Lastenheft
COCOMO2 50% Einsparung durch stabile Anforderungen und reifen Prozess im Vergl. zum Worst Case	COCOMO2 Faktor 1,5 für wiederverwendbare Komponenten	sd&m Kennzahlen: Fachkonzept: 30% 35% IV-Konzept: 10%-15% Implementierung: 35%-40% Integration: 15% 20%
Tom de Marco, sd&m $Team_{Min} = \sqrt{Aufwand\ in\ BM}$ Team=SQR(16,3)=4,03	COCOMO2 $Dauer = 2,5 \cdot Aufwand[BM]^{0,35}$ Dauer=2,5*16,3EXP0,35=6,6	Balzert, COCOMO2 Durchschnittliche Produktivität (gesamt): ~ 350 LOC/BM LOC=350*16,3=5700

Tabelle 2: Faustregeln

Fazit

Systematische Aufwandsschätzung für Software im Auto ist möglich. Die Expertenklausur ist in der Kombination mit COCOMO II eine zuverlässige Methode. Idealerweise führt man sie im gemischten Team durch. Dies erfordert einen Kulturwechsel in der Zusammenarbeit zwischen Zulieferer und OEM und trägt zur Öffnung der Grenzen im Entwicklungsprozess bei. Die Business-IT zeigt, dass dies in der Praxis erfolgreich funktioniert.

Beim OEM kann man die algorithmische Schätzung außerdem zum Plausibilisieren von Aufwandsschätzungen und zur Risikoabschätzung einsetzen. „Was-wäre-wenn"-Analysen lassen sich anhand der COCOMO II-Faktoren gut quantifizieren.

Das Lastenheft sollte neben den funktionalen Anforderungen noch folgende Elemente enthalten: alle nicht funktionalen Anforderungen, die Definition des Entwicklungsprozesses und die grobe fachliche und technische Softwarearchitektur (vgl. [Sie02]).

Nur so lassen sich die heutigen und künftigen Integrationsaufgaben erfolgreich meistern. ■

Literatur & Links

[BoeOO] B.W. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall, 2000

[Chu98-a] S. Chulani, B. Clark, B. Boehm, Calibrating the COCOMO II Post Architecture Model, USC CSE, 1998

[Chu98-b] S. Chulani, B. Steece, A Bayesian Software Estimating Model Using a Generalized g-prior Approach, USC CSE, 1998

[COCO] COCOMO II-Projektseite der University of Southern California, siehe: sunset.usc.edu/research/COCOMOII/

[GarOO] D. Garmus, D. Herron, Function Point Analysis: Measurement Practices for Successful Software Projects, Addison-Wesley, 2000

[IFPUG] International Function Point User Group, siehe: www.ifpug.org/

[Sie04] J. Siedersleben, Moderne Softwarearchitektur, dPunkt, 2004