Oliver Hummel

Aufwands-schätzungen

in der Software- und Systementwicklung

kompakt

Spektrum
AKADEMISCHER VERLAG

Umrechnung eines (Unadjusted) Function Point in Lines of Code

Die Werte in der Literatur schwanken teilweise erheblich und sind daher nur als *ungefähre Richtgröße* zu verstehen: Es bietet sich an, entweder den Mittelwert direkt zu verwenden oder aber den Mittelwert zusammen mit Minimum und Maximum als Ausgangspunkt für eine Drei-Punkt-Schätzung zur Berechnung eines wahrscheinlichsten Erwartungswertes zu verwenden (vgl. Seite 28).

Sprache	Minimum	Mittelwert	Maximum
ABAP	16	18	20
Makroassembler	90	210	400
Ada	40	70	150
C	25	150	300
C#	40	55	70
C++	20	60	150
Cobol	10	100	350
Delphi	unbekannt	ca. 25	unbekannt
Fortran	30	90	200
Java	40	55	200
Javascript	45	55	65
Lisp	25	65	80
Pascal	50	90	125
Perl	10	20	30
PHP	unbekannt	ca. 65	unbekannt
Smalltalk	15	25	50
SQL	10	20	30
Visual Basic	15	30	40

Quellen: [McConnell 06], [Stutzke 05], [Boehm 00], IFPUG und QSM [WWW] sowie eigene Recherchen.

Aufwandsschätzungen in der Software- und Systementwicklung kompakt

Werke der „kompakt-Reihe“ zu wichtigen Konzepten und Technologien der IT-Branche:

- ermöglichen einen raschen Einstieg,
- bieten einen fundierten Überblick,
- sind praxisorientiert, aktuell und immer ihren Preis wert.

Bisher erschienen:

- Heide Balzert
  UML kompakt, 2. Auflage
- Andreas Böhm / Elisabeth Felt
  e-commerce kompakt
- Christian Bunse / Antje von Knethen
  Vorgehensmodelle kompakt, 2. Auflage
- Holger Dörnemann / René Meyer
  Anforderungsmanagement kompakt
- Christof Ebert
  Outsourcing kompakt
- Christof Ebert
  Risikomanagement kompakt
- Karl Eilebrecht / Gernot Starke
  Patterns kompakt, 3. Auflage
- Andreas Essigkrug / Thomas Mey
  Rational Unified Process kompakt, 2. Auflage
- Peter Hruschka / Chris Rupp / Gernot Starke
  Agility kompakt, 2. Auflage
- Oliver Hummel
  Aufwandsschätzungen in der Software- und Systementwicklung kompakt
- Arne Koschel / Stefan Fischer / Gerhard Wagner
  J2EE/Java EE kompakt, 2. Auflage
- Michael Kuschke / Ludger Wölfel
  Web Services kompakt
- Torsten Langner
  C# kompakt
- Pascal Mangold
  IT-Projektmanagement kompakt, 3. Auflage
- Michael Richter / Markus Flückiger
  Usability Engineering kompakt, 2. Auflage
- Thilo Rottach / Sascha Groß
  XML kompakt: die wichtigsten Standards
- SOPHIST GROUP / Chris Rupp
  Systemanalyse kompakt, 2. Auflage
- Gernot Starke / Peter Hruschka
  Software-Architektur kompakt
- Ernst Tiemeyer
  IT-Controlling kompakt
- Ernst Tiemeyer
  IT-Servicemanagement kompakt
- Ralf Westphal
  .NET kompakt
- Ralf Westphal / Christian Weyer
  .NET 3.0 kompakt

Oliver Hummel

Aufwandsschätzungen in der Software- und Systementwicklung kompakt

Spektrum
AKADEMISCHER VERLAG

Autor
Prof. Dr. Oliver Hummel
Institut für Informatik und Wirtschaftsinformatik
Universiät Mannheim
E-Mail: hummel@informatik.uni-mannheim.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

11 12 13 14 15 5 4 3 2 1

Planung und Lektorat: Dr. Andreas Rüdinger, Barbara Lühker
Herstellung und Satz: Crest Premedia Solutions (P) Ltd, Pune, Maharashtra, India
Umschlaggestaltung: SpieszDesign, Neu-Ulm
Titelbild: © SpieszDesign

ISBN 978-3-8274-2751-9

Vorwort

Jeder, der regelmäßig an Softwareprojekten beteiligt ist, wird früher oder später mit dem Thema Aufwandsabschätzung in Berührung kommen, denn ohne akkurate Aufwandsschätzungen kann ein Projektplan niemals zuverlässig funktionieren. Gleichzeitig wird dieses Gebiet in Ausbildung und Studium aber meist nur gestreift und ist entsprechend in der Praxis oft von einem Hauch des Geheimnisvollen umgeben. Auch in meinem Studium und in den darauf folgenden Jahren meiner Tätigkeit an einer Universität kam ich nur sehr oberflächlich damit in Kontakt, und selbst der darauf folgende Wechsel in die Industrie vertiefte meine Kenntnisse nur wenig, führte mir aber deutlich vor Augen, dass Aufwandsschätzungen dort – sofern überhaupt vorgesehen – meist nur wenig planvoll angegangen werden. Daher nutzte ich nach meiner Rückkehr in die Wissenschaft meine Faszination für dieses Thema und die Möglichkeit, systematische Aufwandsabschätzungen als wesentlichen Bestandteil in eine weiterführende Software-Engineering-Vorlesung aufzunehmen.

Obgleich sich themenbedingt mathematische Formeln und teilweise umfangreiche Tabellen nicht ganz vermeiden lassen, soll Ihnen das vorliegende Buch den Stand der Softwaretechnik und meine Erfahrungen aus Wissenschaft und Praxis in kompakter und unterhaltsamer Form vermitteln. Egal ob Sie Auftraggeber, Auftragnehmer oder Projektmanager eines Softwareprojekts oder einer der (oft leidtragenden) Mitarbeiter in einem solchen sind (bzw. werden wollen), ich habe mich bemüht, den Inhalt so aufzubereiten, dass er nicht nur einfach nachzuvollziehen ist, sondern auch möglichst direkt für die tägliche Arbeit verwendet werden kann. Auf der Homepage zu diesem Buch [WWW] finden Sie zudem zahlreiche nützliche Verweise auf weiterführende Informationen, die dieses Buch innerhalb seines bewusst kompakten Rahmens nicht mehr aufzunehmen vermochte.

Ich denke daher, Ihnen eine interessante und kurzweilige Lektüre zusammengestellt zu haben, die Sie dabei unterstützen kann, sogenannte „Todesmarsch-Projekte“ [Yourdon 04] frühzeitig zu erkennen und ihnen erfolgreich aus dem Weg zu gehen. Eventuell helfen Ihnen die vorgestellten Techniken sogar dabei, die Planung des einen oder anderen Projekts positiv zu beeinflussen, auch wenn das erfahrungsgemäß sehr viel Überzeugungsarbeit erfordert. Was auch immer Ihre

Erfahrungen in der Praxis sein werden, ich freue mich über Anregungen und Feedback zum Buch ebenso wie über lesenswerte Anekdoten aus der täglichen Praxis. Ich jedenfalls hoffe, Sie empfinden dieses Werk als eine lohnenswerte oder (nach dem nächsten Schätzworkshop) vielleicht sogar als eine weiterempfehlenswerte Lektüre.

Viel Spaß beim Lesen wünscht Ihnen Ihr Oliver Hummel.

Inhalt

Einführung

Seit vielen Jahren gelten die sogenannten **Chaos-Reports** der Standish Group als Gradmesser für die noch immer nicht als überwunden geltende **Softwarekrise**. Gut vier Jahrzehnte nachdem 1968 auf einer NATO-Konferenz im bayerischen Garmisch das Software Engineering, also das ingenieurmäßige Entwickeln von Software, als „Heilmittel" dafür vorgeschlagen wurde, werden nach Standish noch immer nur rund 30 % aller Software-Entwicklungsprojekte innerhalb der vorgesehenen Zeit- und Budgetvorgaben und mit dem zuvor definierten Funktionsumfang abgeschlossen. Etwa weitere 30 % werden ohne nutzbares Ergebnis vorzeitig abgebrochen und etwa die verbleibenden 40 % aller Projekte liefern zwar ein brauchbares System, überschreiten aber bis zur Auslieferung Budget- oder Zeitvorgaben oder beides. Dies noch immer allein als die Auswirkungen von mangelhaften Fähigkeiten in der Softwareentwicklung zu betrachten, widerspricht aber offensichtlich einer Menge eindrucksvoller Beispiele für das Können heutiger Softwareingenieure, die uns im täglichen Leben begegnen: Betriebssysteme wie Windows oder Linux wachsen im weltweiten Zusammenspiel von hunderten oder gar tausenden von Entwicklern, neue Software für Mobiltelefone oder Webapplikationen entsteht in extrem kurzen Zeiträumen, und unzählige eingebettete Softwaresysteme in PKWs, Flugzeugen bis hin zu Haushaltsgeräten funktionieren im Allgemeinen trotz mancher Kinderkrankheiten erstaunlich zuverlässig. Das klingt nun nicht mehr nach der großen Krise der Softwareentwicklung.

Möglicherweise sieht sich die Standish Group genau deshalb einer beständig wachsenden Kritik an ihren Untersuchungen ausgesetzt. Lassen wir einmal die jüngst öfter geäußerten Zweifel bezüglich der angewandten Methodik außen vor, so zeigt doch bereits ein oberflächlicher Blick, dass die Chaos-Reports mitnichten nur die Fähigkeiten von Softwareentwicklern abbilden; sie beleuchten vielmehr, wie das von dem bekannten Softwareexperten Robert Glass zutreffend beobachtet wurde, ganz banal die Fragestellung, wie gut die gemachten Budget- und Zeitvorgaben bei der Systementwicklung eingehalten werden können. Eine mögliche Ursache für deren Nichteinhaltung sind einerseits natürlich mangelhafte Fähigkeiten in der Softwareentwicklung, andererseits könnten aber auch eine überzo-

O. Hummel, *Aufwandsschätzungen in der Software- und Systementwicklung kompakt,* DOI 10.1007/978-3-8274-2752-6_1,

gene Erwartungshaltung und daraus resultierende nicht umsetzbare Projektvorgaben Auslöser dieser Symptomatik sein.

Ein Blick über den Tellerrand auf zahlreiche Megaprojekte der jüngeren Vergangenheit erhärtet diesen Verdacht: seien es die Verzögerungen beim Ausbau der Stadien für die Fußball-WM 2006 oder bei der Auslieferung des Airbus A380, die Schwierigkeiten bei der Umsetzung des Toll-Collect-Systems, die jüngsten Diskussionen um die Kostenexplosionen bei „Stuttgart 21" oder dem europäischen Militärtransportflugzeug A400M; neuartige Großprojekte verschiedenster Art haben offenbar häufig eine „Tendenz zum Scheitern", wie es einmal ein Mitarbeiter eines bekannten Beratungshauses in einem Vortrag formulierte. Ganz offensichtlich haben die Schätzmechanismen im Vorfeld dieser Projekte entweder nicht richtig funktioniert oder wurden gar bewusst ignoriert bzw. manipuliert, um beispielsweise die Finanzierbarkeit nicht in Frage zu stellen. Doch nicht nur in der Öffentlichkeit sichtbare Großprojekte kämpfen mit dieser Symptomatik, im Bereich der Softwareentwicklung tritt sie oft bereits bei alltäglichen Projekten von überschaubarer Größe zu Tage. Kann dies ausschließlich mit den mangelhaften Kenntnissen in der ingenieurmäßigen Softwareentwicklung zu tun haben?

Systementwicklung und Aufwandsschätzung

Eine mögliche Antwort auf diese Frage mag nach dem bisher Gesagten nur noch geringfügig überraschen, aber der bereits genannte Robert Glass leitet aus der weiten Verbreitung ähnlicher Probleme die These ab, dass die Softwarekrise inzwischen zu einer Krise der Aufwandsschätzungen geworden sei [Glass 06]. Zur Begründung führt er an, dass „Schätzungen" meist von nicht entsprechend qualifizierten Personen (z. B. solchen aus dem projektfernen Management oder Marketing) zu einem unpassenden Zeitpunkt (nämlich weit vor Projektbeginn) aufgestellt und im weiteren Projektverlauf nur selten den tatsächlichen Entwicklungen angepasst würden. Aufwandsschätzungen und darauf aufbauende Projektplanungen beruhen also häufig auf unklaren sowie unvollständigen Anforderungen und sind nicht selten stärker von Wunschdenken oder Termindruck als von analytischem Vorgehen getrieben. Erfahrungsgemäß zweifeln viele (Projekt-)Manager in einer solchen Situation dann allerdings eher an der Leistungsbereitschaft

ihrer Mitarbeiter, anstatt ihre Schätzungen bzw. Vorgaben in Frage zu stellen. Um es mit den Worten des bekannten Softwareexperten und Autors Tom DeMarco zu kommentieren: „Wenn ein Termin nicht eingehalten wurde, war der Zeitplan falsch, unabhängig davon, warum der Termin nicht eingehalten werden konnte."

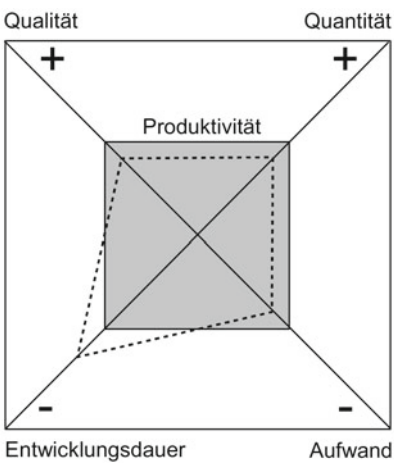
Und weiter: „Der Sinn einer Planung ist es zu planen, nicht Ziele vorzugeben." [DeMarco 01]. Entsprechend muss die Grundlage jeder seriösen Projektplanung natürlich die fundierte Schätzung der zu erwartenden Aufwände sein [Endres & Rombach 03]. Die traditionellen Ingenieurswissenschaften können diese, basierend auf jahrzehntelangen Erfahrungen, sehr genau vorausberechnen (und trotzdem kommt es auch dort immer wieder zu unvorhergesehenen Verzögerungen oder wie oben erwähnt, zu eklatanten Fehleinschätzungen). Sie wissen z. B. an einem gewissen Punkt in einem Bauprojekt sehr genau, wie viele Kubikmeter Erdaushub bewegt oder Tonnen Beton noch angeliefert und verbaut werden müssen und wie hoch Arbeits- und Zeitaufwand dafür üblicherweise sind. In der Softwareentwicklung sind solche Vorhersagen bei Weitem noch keine Routine: zum einen lässt sich die Funktionalität eines Softwaresystems nicht mit Hilfe eines anschaulichen Maßes wie Kilogramm oder Kubikmetern erfassen, zum anderen werden Technologien oft in einem Tempo durch neue ersetzt, dass sie bereits wieder verschwunden sind, ehe entsprechende Erfahrungen mit ihnen hätten gesammelt werden können. Entsprechend umgibt Aufwandsschätzungen noch immer jener Hauch von Hellseherei, der dazu führt, dass selbst erprobte Modelle und systematische Techniken nicht verwendet, sondern Schätzungen am ehesten auf Grundlage von bloßen Vermutungen über den Daumen gepeilt werden. Anekdoten wie die, dass Schätzungen in großen Softwareunternehmen auf ihrem Weg durch die Hierarchieebenen sicherheitshalber mehrmals verdoppelt werden, enthalten sicher mehr als nur das berühmte Körnchen Wahrheit, sind aber auf dem heutigen Stand der Softwaretechnik durchaus vermeidbar.

An dieser Stelle seien noch einige Worte zum Titel dieses Buches erlaubt: dort finden sich mit *Software* und *System* zwei Begriffe, die meistens synonym oder oft auch noch gemeinsam (als Softwaresystem) genutzt werden. Höchste Zeit also, zu klären, ob und wie sich eine mögliche Unterscheidung in diesem Buch wiederfindet. Primär in dessen Fokus liegt die systematische Erstellung von Aufwandsprognosen für die Entwicklung von Software, also von ausführbaren

Programmen mit allen dazu nötigen Modellen, Dokumenten und Testfällen. Von einem Softwaresystem sprechen wir im Rahmen dieses Buches dann, wenn auch noch die für den Betrieb benötigte Hardwareausstattung abgeschätzt werden soll. Entsprechende Schätztechniken für Informationssysteme werden wir gegen Ende des Buches detaillierter diskutieren, zunächst wollen wir uns aber den Grundlagen der Aufwandsschätzung zuwenden, die alleine die (ggf. auch in technische Systeme eingebettete) Software betreffen.

Unterm Rad

Der in der Softwareindustrie herrschende Druck ist zweifellos enorm; egal ob durch beständig zunehmenden Wettbewerb, Wirtschaftskrisen oder Near- und Off-Shoring, Softwareunternehmen sehen sich dem beständigen Zwang ausgeliefert, immer mehr Arbeit mit immer weniger Personal bewältigen zu müssen. In anderen Worten, sie sind gezwungen, ihre Produktivität beständig weiter zu steigern, um wettbewerbsfähig zu bleiben. Grob gesprochen lässt sich die Produktivität innerhalb eines Softwareprojekts als die implementierte Menge an Funktionalität pro dafür investiertem Aufwand (also beispielsweise als Lines of Code pro Personenmonat) definieren. Sie wird innerhalb eines Projekts von vier Parametern, nämlich Qualität, Quantität (also Produktumfang), Entwicklungsdauer (also Zeit) und Aufwand (d. h. Kosten) beeinflusst. Harry Sneed hat die links gezeigte Darstellung dieser Einflüsse als sogenanntes **Teufelsquadrat** geprägt.



Das „Verstellen“ eines Parameters beeinflusst nach dieser Darstellung unweigerlich die jeweils anderen Werte, da die Produktivität innerhalb eines Projekts (in der Abbildung die Fläche des grauen Quadrats) als eine konstante Größe gesehen wird. Zu Beginn ist sie nominal