

FREDERICK P. BROOKS JUN.

VOM MYTHOS DES MANN-MONATS

ESSAYS ÜBER SOFTWARE-ENGINEERING





Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Vom Mythos des Mann-Monats

Frederick P. Brooks jun.

Vom Mythos des Mann-Monats

Essays über Software-Engineering

Übersetzt aus dem Amerikanischen von Arne Schäpers

Bibliografische Informationen Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <<http://dnb.ddb.de>> abrufbar.

ISBN 978-3-95845-877-2

1. Auflage 2019

Authorised translation from the English language edition, entitled MYTHICAL MAN MONTH: ESSAYS ON SOFTWARE ENGINEERING, ANNIVERSARY EDITION; 2nd Edition, 0201835959 by BROOKS, FREDERICK P., published by Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright ©1995 Addison Wesley Longman, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. Electronic German language edition published by mitp Verlags GmbH & Co. KG, Copyright © 2019

Autorisierte Übersetzung der englischsprachigen Ausgabe mit dem Titel MYTHICAL MAN MONTH: ESSAYS ON SOFTWARE ENGINEERING, ANNIVERSARY EDITION; 2nd Edition, 0201835959 von BROOKS, FREDERICK P., erschienen bei Pearson Education, Inc, als Addison Wesley Professional, Copyright © 1995 Addison Wesley Longman, Inc.

Alle Rechte vorbehalten. Kein Teil dieses Werkes darf ohne Genehmigung von Pearson Education reproduziert oder in irgendeiner Form (elektronisch, mechanisch, Fotokopie) oder auf sonst eine Art aufgenommen oder elektronisch gespeichert werden. Elektronische Deutsche Ausgabe von mitp Verlags GmbH & Co. KG
© Copyright 2019 by mitp-Verlag/Frechen,

Printed in Germany

Übersetzung aus dem Amerikanischen: Arne Schäpers
Übersetzung der ursprünglichen Ausgabe von 1975: Arne Schäpers, Armin Hopp
Lektorat: Katja Völpel
Korrektorat: Petra Heubach-Erdmann
Satz und Layout: Arne Schäpers
Umschlaggestaltung: Christian Kalkert

Über den Autor

Frederik P. Brooks jun. ist Professor Emeritus für Computerwissenschaften an der Universität von North Carolina in Chapel Hill und bekannt als „Vater des IBM System/360“, dessen Entwicklung er zunächst als Projektmanager leitete; später war er Leiter des Software-Projekts Operating System/360 während seiner Designphase. Für diese Arbeit erhielt er zusammen mit Bob Evans und Erich Bloch im Jahre 1985 die *National Medal of Technology*. Zuvor war er einer der Architekten des IBM Stretch und des IBM Harvest-Computers.

In Chapel Hill gründete Dr. Brooks das Department of Computer Science, dem er von 1964 bis 1984 vorstand. Er war sowohl Mitglied des National Science Board als auch des Defense Science Board. Seine aktuellen Lehr- und Forschungsgebiete sind Computer-Architektur, molekulare Grafik und virtuelle Umgebungen.



Lernaea-Hydra: Sie wurde in dem Lerna, einem Pfuhle in Argolien auferzogen und verwüstete das flache Land weit und breit, weil sie Menschen und Vieh hinwegraubete. Sie war von einer ungeheuern Größe, hatte neun Köpfe, unter welchen der mittelste unsterblich war. Diese zu erlegen, wurde dem Herkules von dem Eurysiheus anbefohlen, der sich denn mit dem Jolaus auf seinem Wagen dahin machte. Er traf sie auf einem Hügel, bey der Amymone Brunnen, in ihrem Lager an, ließ den Wagen stehn, griff sie mit feurigen Pfeilen an, und zwang sie also, sich hervor zu machen. Er fassete sie darauf selbst mit den Händen: sie schlug sich aber dagegen um so genauer um seine Beine herum. Allein, indem er anfang, mit seiner Keule ihr die Köpfe abzuschlagen, so sah er, daß an eines Stelle sofort zween andere hervorkamen. Dabey kam ihr noch ein ungeheurer Krebs zu Hülfe, der ihn gewaltig in die Füße kipp. Er erschlug also erst diesen und rief darauf den Jolaus zu Hülfe, welcher denn einen nahestehenden Wald anzündete und mit den Bränden sogleich über den Hals der Hydra herfuhr, wenn ein Kopf angeschlagen war. Dadurch verhinderte er dann, dass keine andere an dessen Stelle wuchsen, und Herkules also das Ungeheuer endlich erlegen konnte. Den unsterblichen Kopf nahm er und vergrub ihn in die Erde, legete auch zu mehrer Sicherheit, einen gewaltigen Stein auf denselben. Dabei tunkete er seine Pfeile in ihr Blut, welche sodann ganz unheilbare Wunden machten.

Hederich, Benjamin: Gründliches Mythologisches Lexikon, Leipzig, Gleditsch 1770; Reprgr. Nachdruck, Darmstadt, Wissenschaftliche Buchgesellschaft 1996, S. 1454 f.

Widmung der Ausgabe von 1975

Zwei Menschen gewidmet, die meine Jahre bei IBM in ganz besonderer Weise bereichert haben:

Thomas J. Watson, Jr., dessen liebevolle Sorge um Menschen nach wie vor die Firma durchdringt, und Bob O. Evans, unter dessen kraftvoller Führung Arbeit zum Abenteuer wurde.

Widmung der Ausgabe von 1995

Für Nancy, Gottes Geschenk an mich.

Vorworte der Übersetzer

Als F. P. Brooks diese Serie von Essays schrieb – mittlerweile ist mehr als ein Jahrzehnt vergangen – war die Programmierung von Computern die Domäne eines kleinen Spezialistenzirkels. Auch wenn die Firma Intel bereits im Jahre 1972 den ersten Mikroprozessor auf den Markt gebracht hat: Die Vorstellung eines PC auf jedermanns Schreibtisch wäre selbst vom abgebrühtesten Verkaufsmanager als wilde Phantasterei abgetan worden.

Die Programmierung der ersten Mikrocomputer (des PET 2001 und des Apple II) *war* das Werk von „Garagenduos“, auf die der Autor mit dem nachsichtigen Lächeln eines Großrechnerspezialisten anspielt (Kapitel 1). Auch aus heutiger Sicht ist er dabei vollkommen im Recht.

Unnötig zu sagen, dass sich die Zeiten geändert haben. Für die Wochenmiete von 200 KByte zusätzlichem Speicher einer IBM 360 bekommt man einen kompletten Computer mit 4 MByte RAM, Programmierarbeit ohne symbolische Prozedurnamen ist auch auf einem Homecomputer unvorstellbar geworden – und Betriebssysteme von 200 KByte und mehr sind die Regel.

Die Zeit scheint nur in einem einzigen Punkt stehen geblieben zu sein: Probleme der Organisation großer Programmierprojekte haben sich nicht etwa geändert (geschweige denn: sind gelöst worden) – mit der Popularisierung integrierter Programmpakete, deren reiner Codeumfang bei mehreren hundert KByte liegt, haben sie sich lediglich auf den PC verlagert und dabei vervielfacht. Während die Vorschläge und Überlegungen Brooks' im Großcomputerbereich Allgemeingut sind, ist die Verspätung angekündigter Programme auf dem PC-Markt zur Regel geworden (für die im amerikanischen Sprachgebrauch denn auch der bezeichnende Satz „available real soon now“ entstand).

Einige Aussagen (PL/1 als Programmiersprache der Wahl) und Postulate (interaktive Terminals anstelle von Zeilendruckern, Symbole anstelle von Speicheradressen) sind nur noch vor historischem Hin-

tergrund von Interesse – der Kernpunkt des Buchs, die Organisation großer Programmierprojekte, ist dagegen aktueller denn je.

München, April 1987

Arne Schäpers/Armin Hopp

Ein Buch über Programmierung 16 Jahre später erneut auf dem Tisch zu haben, ist für sich schon absolut einmalig. Und wieder – diesmal nicht zehn, sondern „nur“ acht Jahre nach Erscheinen des Originals – zeigt sich der prophetische Charakter dieses Werks, sei es nun bei der komponentenbasierten Programmierung (Delphi und Visual Basic steckten 1995 in den Kinderschuhen, von C# erst gar nicht zu reden), der Forderung nach anwendungsübergreifenden Scriptsprachen (VBA) oder des zur Einheitlichkeit zwingenden, ins System integrierten Werkzeugkastens, wie ihn Microsoft zur Zeit mit der .NET-Initiative in nie gekannter Geschlossenheit vorexerziert. Eine weitere Ausgabe dieses Werks im Jahre 2015 vorauszusagen – dazu braucht es keine prophetischen Gaben.

München, Juli 2003

Arne Schäpers

Vorwort zum 20. Jubiläum der Originalausgabe

Zu meiner Überraschung und Freude ist *Vom Mythos des Mann-Monats* 20 Jahre lang populär geblieben; über 250 000 Exemplare allein der amerikanischen Originalausgabe wurden gedruckt. Ich werde oft gefragt, zu welchen der von mir im Jahre 1975 geäußerten Meinungen und Vorschläge ich nach wie vor stehe, was sich geändert hat – und was immer noch gilt. Von Zeit zu Zeit bin ich in Vorlesungen auf diese Fragen eingegangen, und der Wunsch, all dies einmal schriftlich niederzulegen, wurde lange gehegt.

Peter Gordon, mittlerweile assoziierter Hersteller bei Addison-Wesley, hat meine Arbeiten seit 1980 mit Geduld und hilfreicher Unterstützung begleitet. Von ihm stammt der Vorschlag einer Geburtstagsausgabe, für die wir beschlossen, das Original (mit Ausnahme trivialer Korrekturen) unverändert wiederzugeben und es um aktuelle Überlegungen zu ergänzen.

Kapitel 16 ist ein Abdruck von „No Silver Bullet: Essence and Accidents of Software Engineering“, eines für die IFIPS 1986 von mir publizierten Papiers, das meine Erfahrungen als Leiter einer Studie des Defense Science Board zu militärischer Software zusammenfasst. Die Koautoren dieser Studie und unser Generalsekretär, Robert L. Patrick, haben Unschätzbare dazu beigetragen, mich wieder auf Tuchfühlung mit großen Software-Projekten in der realen Welt zu bringen. Das Papier wurde 1987 von der Zeitschrift *IEEE Computer* abgedruckt und erreichte so eine breite Leserschaft.

„Keine Silberkugel“ ist ein provokativer Text, der voraussagt, dass es für die nächsten zehn Jahre keine einzige Technologie geben wird, die für sich allein genommen Verbesserungen der Produktivität bei Software um wenigstens eine Größenordnung erreicht. Diese Dekade steht ein Jahr vor ihrem Ablauf, und die Vorhersage scheint sich zu bestätigen. „Keine Silberkugel“ hat mehr und engagiertere Diskussionen in der Fachliteratur angeregt als *Vom Mythos des*

Mann-Monats. Aus diesem Grund geht Kapitel 17 auf die öffentlich geäußerte Kritik ein und stellt die 1986 geäußerten Thesen in einen aktuelleren Kontext.

Bei der Vorbereitung des Rückblicks auf *Vom Mythos des Mann-Monats* und seine Aktualisierung hat mich vor allem erstaunt, wie wenige der von mir aufgestellten Thesen durch spätere Studien der Disziplin und praktische Erfahrungen kritisch hinterfragt, tatsächlich bewiesen oder widerlegt worden sind. Für mich erwies es sich als nützlich, diese Vorschläge und Vorhersagen einmal als dürre Liste zusammenzufassen, bar jeglicher unterstützender Argumentation oder den dazugehörigen Daten. In der Hoffnung, dass diese nackten Statements zur Diskussion anregen, das Beibringen von Fakten provozieren, die sie stützen, ad absurdum führen, durch Aktuelleres ersetzen oder auch nur verfeinern, habe ich dieses Grundgerüst als Kapitel 18 in dieses Buch eingesetzt.

Kapitel 19 stellt die eigentliche Aktualisierung dar. Der Leser sollte sich bewusst sein, dass die dort geäußerten Meinungen nicht annähernd so gut durch eigene Erfahrungen mit täglicher Kärnerarbeit abgesichert sind wie das von mir im originalen Buch Dargelegte: Ich habe an einer Universität gearbeitet, nicht in der Industrie, und kleine Projekte anstelle großer realisiert. Seit 1986 lehre ich Software Engineering, anstatt auf diesem Gebiet praktisch zu forschen. (Tatsächlich haben sich meine universitären Forschungsarbeiten stark auf virtuelle Umgebungen und ihren Einsatz konzentriert.)

Für diese Retrospektive musste ich deshalb die Hilfe von Freunden in Anspruch nehmen, die auch heute noch im Bereich des praktischen Software Engineering arbeiten. Für ihre wunderbare Bereitschaft, mich an ihren Ansichten teilhaben zu lassen, das Manuskript gründlich zu hinterfragen und mein eigenes Wissen auf den neuesten Stand zu bringen, schulde ich vor allem den folgenden Leuten Dank: Barry Boehm, Ken Brooks, Dick Case, James Coggins, Tom DeMarco, Jim McCarthy, David Parnas, Earl Wheeler und Edward Yourdon. Und natürlich Fay Ward, die sich souverän um die Produktion der neuen Kapitel gekümmert hat.

Ich danke Gordon Bell, Bruce Buchanan, Rick Hayes-Roth, meinen Kollegen von der Defense Science Board Task Force on Military Software und vor allem David Parnas für ihre Einsichten und Anregungen zu dem hier als Kapitel 16 abgedruckten Papier, und Rebekah Bierly für dessen technische Produktion. Die Analyse des Software-Problems in Termini von Essenz und Akzidenz wurde übrigens von Nancy Greenwood Brooks inspiriert, die dieselbe Perspektive in einem Papier über die Pädagogik der Suzuki-Violine verwendete.

Die hauseigene Politik von Addison-Wesley hat es mir nicht gestattet, im Vorwort der Ausgabe von 1975 den Verlagsmitarbeitern zu danken, die damals Schlüsselrollen innehatten. Speziell die Beiträge zweier Menschen sollten nicht unerwähnt bleiben: des damaligen Cheflektors Norman Stanton und des damaligen Art Directors Herbert Boes, der für den eleganten Stil verantwortlich zeichnet, über den eine Rezension schrieb: „große Abstände, sprechender Einsatz von Schriftarten und des Layouts“. Wichtiger war mir sein Vorschlag, jedes Kapitel mit einem eigenen Bild einzuleiten. (Ich hatte ursprünglich nur den Teersumpf und die Kathedrale vom Reims.) Die Suche nach weiteren Bildern hat mich ein zusätzliches Jahr Arbeit gekostet, für den Ratschlag werde ich aber ewig dankbar sein.

Soli Deo gloria – Allein Gott die Ehre.

Chapel Hill, N.C.
März 1995

F.P.B. Jr.

Vorwort zur ersten Ausgabe

Das Management großer Computerprojekte zeigt mehr Gemeinsamkeiten mit dem anderer vergleichbar großer Unternehmen, als die meisten Programmierer glauben. Andererseits gibt es aber auch eine Anzahl fundamentaler Unterschiede, die die meisten Projektmanager kaum erwarten werden.

Inzwischen weiß man schon einiges über dieses Problemfeld. Es wurden verschiedene Symposien abgehalten, einige Sitzungen auf AFIPS-Konferenzen beschäftigten sich ebenfalls mit diesem Thema. Parallel dazu sind einige Bücher und Aufsätze in die Diskussion eingebracht worden. Allerdings kann man nicht sagen, dass sich schon zu diesem Zeitpunkt Umrisse abzeichnen, die die Abfassung eines Lehrbuchs rechtfertigen würden. Ich fühle mich dennoch berufen, mit diesem Buch meinen eigenen Standpunkt darzulegen.

Obwohl mein ursprüngliches Betätigungsfeld auf dem Gebiet der Computerprogrammierung lag, war ich während der Jahre 1956–1963, in denen das autonome Kontrollprogramm und der Hochsprachencompiler entwickelt wurden, hauptsächlich mit Hardware-Architektur beschäftigt. Als ich dann 1964 Manager des Operating System/360 wurde, fand ich eine im Zuge des Fortschritts stark veränderte Programmierwelt vor. Die Leitung der Entwicklung des OS/360 war für mich eine große Erfahrung, obwohl wir natürlich auch frustrierende Rückschläge erlitten. Das Team hat einschließlich meines Nachfolgers F. M. Trapnell allen Grund, auf das Geleistete stolz zu sein – sie haben ein ausgezeichnetes System auf die Beine gestellt. Es hat sowohl in seinem Design als auch in der Ausführung unbestreitbare Vorzüge und ist deshalb auf dem Markt sehr erfolgreich. Bestimmte Ideen (hervorzuheben sind hier besonders die Umleitungsmöglichkeiten der Ein- und Ausgabe sowie die Verwaltung externer Funktionsbibliotheken) stellten seinerzeit technische Neuerungen dar, die heute fast überall Verwendung finden. Das System

ist mittlerweile recht zuverlässig, hinlänglich effizient und sehr vielseitig.

Trotzdem kann das Ergebnis all der Anstrengungen nicht ganz überzeugen, wird doch jedem Benutzer des OS/360 schnell klar, um wie viel besser es sein könnte. Die Mängel in Ausführung und Design liegen hauptsächlich im Betriebssystem selbst, weniger in den Compilern. Die meisten Unzulänglichkeiten haben ihre Wurzeln in den Jahren 1964–1965 und fallen folglich unter meine Verantwortung. Überdies konnte der projektierte Zeitrahmen nicht eingehalten werden, wodurch das Produkt mit Verspätung auf den Markt kam. Es brauchte mehr Speicherplatz als eigentlich geplant, kostete ein Mehrfaches der veranschlagten Summe und gab in den ersten Versionen kein gutes Bild ab. 1965 verließ ich IBM, um nach Chapel Hill an die Universität von North Carolina zu gehen, wie es schon vereinbart worden war, als ich OS/360 übernahm. Dort begann ich, meine gesammelten Erfahrungen zu analysieren. Es ging mir darum, herauszufinden, welche administrativen und technischen Lehren man daraus ziehen könnte. Eine Klärung der bemerkenswerten Unterschiede, die bei der Entwicklung der System/360-Hardware im Gegensatz zu der OS/360-Software zu Tage traten, lag mir dabei besonders am Herzen. Dieses Buch ist also eine reichlich verspätete Reaktion auf Tom Watsons Gretchenfrage, warum die Kunst des Programmierens denn eine so schwierige sei.

Bei der Suche nach einer Antwort habe ich aus langen Gesprächen mit R. P. Case, dem stellvertretenden Manager der Jahre 1964–1965, und F. M. Trapnell, dem Projektleiter von 1965–1968, viele Erkenntnisse gewonnen. Meine Schlussfolgerungen konnte ich schließlich mit den Leitern anderer ähnlich großer Programmierprojekte diskutieren. Unter ihnen waren F. J. Corbato vom M.I.T., John Harr und V. Vyssotsky von den Bell Telephone Laboratories, A. P. Ershov von den Computerlaboratorien der Sibirischen Abteilung der Akademie der Wissenschaften der UdSSR und A. M. Pietrasanta von IBM.

Die Sammlung meiner Erkenntnisse findet sich in den folgenden Essays, die für Berufsprogrammierer, Manager und besonders für Manager von Berufsprogrammierern bestimmt sind.

Obwohl dieses Buch aus voneinander unabhängigen Essays besteht, ist es doch um einen zentralen Gedanken herum aufgebaut, der sich besonders in den Kapiteln 2 bis 7 wiederfindet. Kurz gesagt bin ich zu der Überzeugung gelangt, dass sich bei großen Programmierprojekten wegen des Maßes an Arbeitsteilung andere Probleme aufwerfen als bei kleineren. Deswegen muss ein Hauptbestandteil der Programmierarbeit in der Bewahrung eines einheitlich durchgehenden Produktkonzepts selber liegen. Die genannten Kapitel untersuchen sowohl die Schwierigkeiten beim Versuch, eine solche Struktur durchgängig zu bewahren, als auch die dazu notwendige Methodik. Die folgenden Kapitel behandeln dann weitere Aspekte des Managements der Software-Entwicklung.

Die Literatur für dieses Problemfeld ist zwar nicht gerade dünn gesät, aber weit gestreut. Deswegen habe ich versucht, Hinweise zu geben, die bestimmte Punkte zusätzlich beleuchten und dem interessierten Leser weitere Arbeiten zum Thema zugänglich machen sollen. Viele meiner Freunde haben das Manuskript gelesen und konnten mir mit ihren Kommentaren oft weiterhelfen; wo mir diese wertvoll erschienen, aber den Fluss des Texts behinderten, habe ich sie in die Anmerkungen aufgenommen.

Da dieses Buch eine Sammlung von Essays enthält und keinen durchgehenden Text, wurden alle Hinweise und Anmerkungen ans Ende verbannt, wodurch der Leser gezwungen ist, sie zunächst außer Acht zu lassen.

Zu tiefstem Dank verpflichtet bin ich Miss Sara Elizabeth Moore, Mr. David Wagner und Mrs. Rebecca Burris für ihre Hilfe bei der Erstellung des Manuskripts und Professor Joseph C. Sloane für seine Ratschläge zur Illustration des Buchs.

Vorwort des Verlags

Im Vorwort zur 20-jährigen Jubiläumsausgabe schreibt der Übersetzer Arne Schäpers im Jahr 2003: »Eine weitere Ausgabe dieses Werks im Jahre 2015 vorauszusagen – dazu braucht es keine prophetischen Gaben.«

Nun, ganz so hat es nicht geklappt, denn mittlerweile schreiben wir das Jahr 2019. Aber das vorliegende E-Book, das 15 Jahre nach der ersten deutschen Audio-Veröffentlichung und ganze 44 Jahre nach der Erstausgabe von Brooks' »Vom Mythos des Mann-Monats« nun im mitp-Verlag allen interessierten Lesern neu zur Verfügung gestellt wird, ist und bleibt einfach ein Klassiker.

Schwierigkeiten beim Managen komplexer Projekte hat es zu jeder Zeit gegeben und dies wird wohl auch immer so bleiben. Frederik P. Brooks analysiert und seziert in seinen Essays die Zutaten für ein erfolgreiches Projektmanagement mit sowohl historischem Bezug als auch zeitlosem Scharfsinn.

Ob Sie aus rein geschichtlicher Neugierde den Ausführungen von Brooks folgen wollen oder einige der auch in den heutigen Zeiten von Android und iOS gültigen Kernaussagen auf Ihr Software-Projekt übertragen möchten, Sie werden sicherlich einige wertvolle Ansätze in diesem Buch finden.

Wir wünschen Ihnen ein angenehmes Lese-Erlebnis!

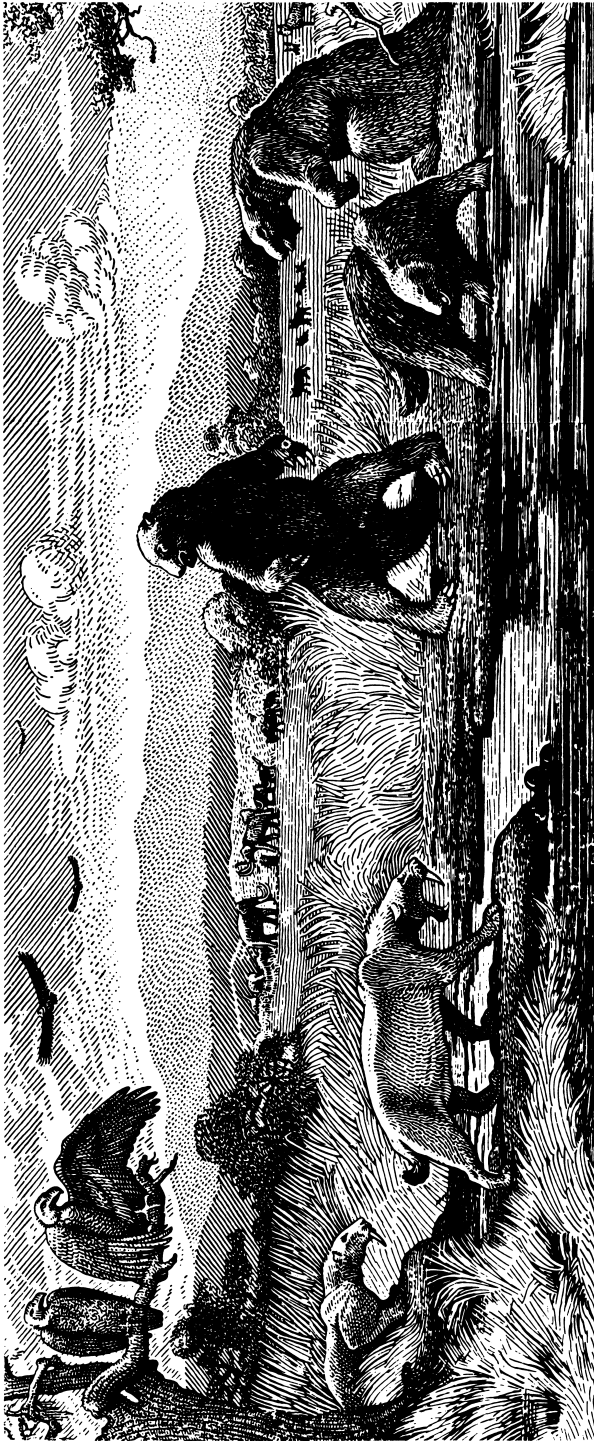
Ihr mitp-Verlag

Inhalt

	Vorworte der Übersetzer.....	ix
	Vorwort zum 20. Jubiläum der Originalausgabe	xi
	Vorwort zur ersten Ausgabe	xv
	Vorwort des Verlags	xxi
Kapitel 1	Der Teersumpf.....	1
Kapitel 2	Der Mythos Mann-Monat	11
Kapitel 3	Das Ärzteteam	29
Kapitel 4	Aristokratie, Demokratie und Systementwicklung	41
Kapitel 5	Das zweite System.....	55
Kapitel 6	Die Wortstafette.....	65
Kapitel 7	Wieso fiel der Turm zu Babel?.....	79
Kapitel 8	Die Praxis als Herausforderung	95
Kapitel 9	Zwei Zentner in einem Ein-Zentner-Sack.....	105
Kapitel 10	Die Dokumenten-Hypothese.....	115
Kapitel 11	Das Pilotprojekt für den Abfalleimer.....	123
Kapitel 12	Gutes Werkzeug.....	135
Kapitel 13	Das Ganze und seine Teile.....	149
Kapitel 14	Die Katastrophe wird ausgebrütet.....	165
Kapitel 15	Das andere Gesicht	177
Kapitel 16	Silberkugeln sind leider aus – Essenz und Akzidenz im Software Engineering	195
Kapitel 17	Keine Silberkugeln: Nachschuss.....	229
Kapitel 18	Vorhersagen des Originals: Wahr oder falsch?.....	255
Kapitel 19	Vom Mythos des Mann-Monats nach 20 Jahren.....	281
	Epilog: Fünfzig Jahre Wunder, Aufregung und Freude..	325
	Anmerkungen und Hinweise.....	327

1

Der Teersumpf



1

Der Teersumpf

Een ship op het strand is een baken in zee.
(Ein gestrandetes Schiff ist eine Bake zur See hinaus.)

Niederländisches Sprichwort

C.R. Knight, Mural of La Brea Tar Pits
Naturhistorisches Museum von Los Angeles, Fotografische Abteilung

Kein Bild aus prähistorischer Zeit ist nur annähernd so beeindruckend wie das vom Todeskampf riesiger Bestien in den urweltlichen Teersümpfen. Vor dem geistigen Auge erscheinen Dinosaurier, Mammuts und Säbelzahniger im Kampf gegen den Sog des Sumpfes. Je heftiger sie sich wehren, desto unnachgiebiger wird der Griff des Teers, und auch all ihre Kraft und Gewandtheit können sie nicht davor bewahren, letztendlich doch zu versinken.

Die Programmierung großer Systeme war während des letzten Jahrzehnts ein solcher Teersumpf, in dem viele große und starke Bestien wild um sich geschlagen haben. Die meisten haben funktionierende Systeme hervorgebracht – wenige haben Zielsetzungen, Zeitvorgaben und Kostenrahmen einhalten können. Ob umfangreich oder schmalbrüstig, in großem oder kleinem Rahmen, ein Team nach dem anderen hat sich in den Sumpf ziehen lassen. Es scheint unmöglich, für dieses Phänomen einen besonderen Grund zu isolieren. Genauso wie die versinkende Bestie, die jede einzelne ihrer Pfoten wieder aus dem Sumpf ziehen kann, schließlich aber doch untergehen muss, ist auch das programmierte Team dem Sog hilflos ausgeliefert. Die Anhäufung gleichzeitig auftretender und sich gegenseitig beeinflussender Faktoren schränkt die Bewegungsfähigkeit immer weiter ein. Jedermann zeigt sich überrascht von der Unzugänglichkeit des Problems, dessen Natur offensichtlich schwer zu ergründen ist. Aber wenn wir es lösen wollen, müssen wir versuchen, es verstehen zu lernen.

So wollen wir damit beginnen, die Kunst (das Handwerk?) des Programmierens von Systemen zu beschreiben, mitsamt den ihr inwohnenden Freuden und Leiden.

Das Produkt „Programmiersystem“

Dann und wann liest man in den Zeitungen Berichte über zwei Programmierer, die in einer umfunktionierten Garage ein bedeutendes Programm entwickelt haben, das die Anstrengungen ganzer Teams in den Schatten stellt. Und jeder Programmierer ist nur allzu gern bereit, derlei Märchen Glauben zu schenken, ist er doch davon überzeugt, er könne *jedes* Programm wesentlich effizienter und schneller

schreiben als die Teams in der Industrie, deren Durchschnittsleistung bekanntlich bei 1000 Befehlen pro Programmierer und Jahr liegt.

Warum werden dann nicht einfach alle in der Industrie beschäftigten Programmerteams durch solche Garagenduos ersetzt? Dazu muss man genauer betrachten, *was* eigentlich produziert wird.

In Bild 1.1 findet sich oben links das *Programm*. In sich selbst ist es vollständig und kann von seinem Autor auf dem System benutzt werden, auf dem es geschrieben wurde. Das ist es, was gewöhnlich in Garagen entwickelt wird, und daran misst der einzelne Programmierer seine eigene Produktivität.

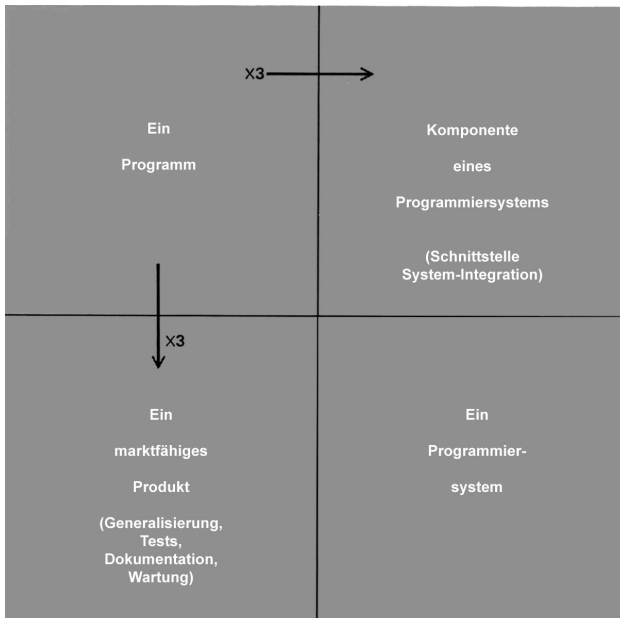


Bild 1.1 Die Evolution eines Programmierprodukts

Es gibt zwei Wege, um ein Programm in etwas Nützlicheres umzuwandeln – beide sind allerdings mit erheblich höheren Kosten verbunden. Diese beiden Möglichkeiten werden im Diagramm durch zwei Linien dargestellt. Unterhalb der horizontalen Linie sehen wir, wie aus einem Programm ein *marktfähiges Produkt* wird. Ein sol-

ches Programm kann von jedermann benutzt, abgefragt, instandgesetzt und erweitert werden. Man kann es in mehreren Systemkonfigurationen und für viele verschiedene Arten von Datensätzen benutzen. Hat man sich ein solches generell verwendbares Produkt zum Ziel gesetzt, muss auch das Programm selber in einer generalisierten Form gehalten sein.

Besonders Umfang und Form der Eingaben verlangen nach weitestgehender Generalisierung, so weit, wie es der zu Grunde liegende Algorithmus vernünftigerweise zulässt. Schließlich muss das Programm einer gründlichen Prüfung unterzogen werden, bis seine Zuverlässigkeit gewährleistet werden kann. Dies bedeutet, dass eine große Anzahl von Prüfverfahren zur Untersuchung des Umfangs der Eingaben und ihrer Grenzen entwickelt, angewandt und festgehalten werden muss. Letztendlich muss das zum marktfähigen Produkt avancierte Programm noch von einer gründlichen Dokumentation begleitet werden, damit ein jeder es verwenden, reparieren und erweitern kann. Als Faustregel gehe ich davon aus, dass ein solches Produkt mindestens das Dreifache eines fehlerbereinigten Programms mit vergleichbarer Arbeitsweise kostet.

Mit dem Sprung über die vertikale Linie wird das Programm zur *Komponente eines Programmiersystems*. Dabei handelt es sich um eine Zusammenstellung interagierender Programme, deren jeweilige Funktionen koordiniert und in eine Struktur eingebracht werden. Aus dieser Montage entsteht dann eine Arbeitseinheit zur Lösung umfangreicher Aufgaben. Damit ein Programm zur Komponente eines Programmiersystems werden kann, muss es so geschrieben sein, dass seine Ein- und Ausgaben syntaktisch und semantisch präzise festgelegten Definitionen entsprechen. Darüber hinaus soll das Programm so angelegt sein, dass es die verfügbaren Ressourcen nur in einem genau festgelegten Ausmaß in Anspruch nimmt – sowohl was Rechenzeit und Speicherplatz als auch die Peripheriegeräte betrifft. Schließlich muss das Programm mit anderen Systemkomponenten getestet werden, und zwar in allen zu erwartenden Kombinationen. Ein solcher Prüfungsvorgang ist aufwendig – einfach schon deshalb, weil die Zahl der Testfälle mit der der möglichen Kombinationen steigt. Er ist zeitaufwendig, da das Zusammenspiel schon feh-