vdf | Praxis und Lehre

Informatik

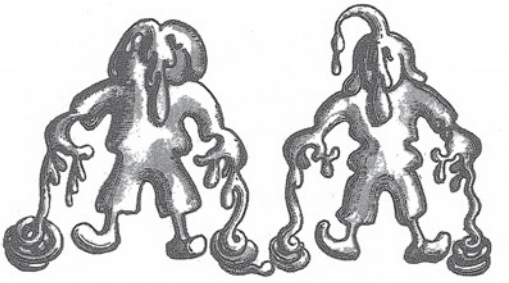
Software-Projektmanagement und -Qualitätssicherung

4. Auflage

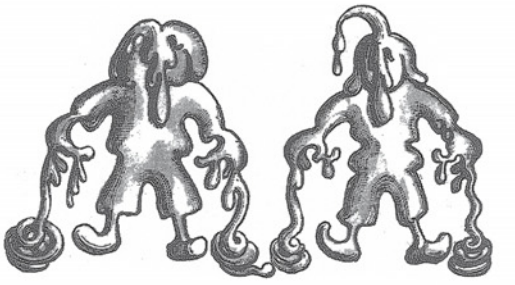
Karol Frühauf
Jochen Ludewig
Helmut Sandmayr

vdf

Software-Projektmanagement und -Qualitätssicherung



4. Auflage

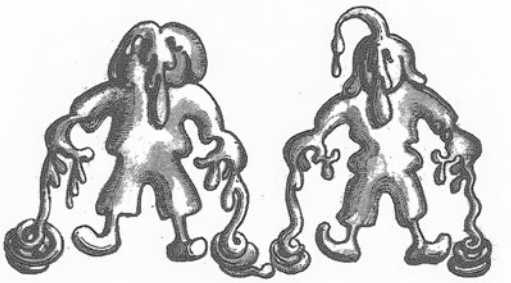


v/d|f

vdf Hochschulverlag AG
an der ETH Zürich

Karol Frühauf
Jochen Ludewig
Helmut Sandmayr

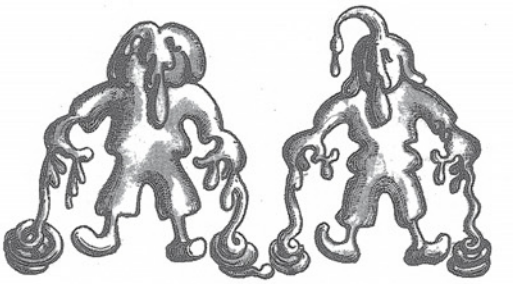
Software-Projektmanagement und -Qualitätssicherung



4. Auflage

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Frühauf, Karol:
Software-Projektmanagement und -Qualitätssicherung / Karol Frühauf ; Jochen Ludewig ; Helmut Sandmayr. – 4., durchges. Aufl.. – Zürich : vdf, Hochsch.-Verl. an der ETH, 2002
(vdf Praxis und Lehre)
ISBN 3-7281-2822-8 (Printversion)
ISBN 3-7281-3948-1 (E-Book)
DOI-Nr. 10.3218/3948-1



ISBN 3-7281-2822-8 (Printversion)
ISBN 3-7281-3948-1 (E-Book)
DOI-Nr. 10.3218/3948-1

1. Auflage 1988
2., durchgesehene Auflage 1991
3., völlig neu überarbeitete Auflage 2000
4., durchgesehene Auflage 2002

Vorwort

Viele Leute kennen die Situation: Man fährt mit dem Auto durch eine Gegend, die man nicht kennt, und man ist in Eile. Statt sich (unter Aufwendung kostbarer Zeit) durch Studium der Karte oder durch Fragen an Passanten zu informieren, fährt man „nach Gefühl" drauflos – und kommt darum erst recht zu spät ans Ziel. Dies ist auch im Software Engineering ein typisches Verhaltensmuster: „We don't have the time to be efficient."

Im Vergleich zur Hardware hat sich die Software-Technologie in den vergangenen dreißig Jahren relativ langsam entwickelt. Es überrascht viele Menschen zu hören, dass die Kosten pro Zeile Programmcode sogar etwa gleich geblieben sind. Der Grund liegt darin, dass Software von Menschen gemacht wird, deren Intelligenz sich kaum ändern lässt. Darum ist eine Erhöhung der Produktivität (fast) nur durch effizienten Einsatz der Denkleistung möglich. Hierzu gibt es vor allem zwei Möglichkeiten:

- Fehlerquellen beseitigen
- nichtkreative Arbeiten automatisieren

Mit unserem Buch verfolgen wir das Ziel, diejenigen, die vom intuitiven zum systematischen Ansatz wechseln wollen, mit dem notwendigen Wissen auszustatten. Dabei erscheint uns vieles, was wir mitteilen, durchaus trivial. Da aber viele Binsenweisheiten in der Praxis ganz offensichtlich ignoriert werden, halten wir auch triviale Feststellungen für notwendig.

Wir gebrauchen für die Berufsbezeichnungen (beispielsweise „Manager", „Programmierer") durchgehend die männliche Form. Dies geschieht, weil sich der Text zum einen so glatter liest, zum anderen aus Bequemlichkeit, aber sicher ohne Absicht der Diskriminierung. Im Gegenteil bedauern wir, dass die Darstellung der Informatik als ein Stück, in dem es nur männliche Rollen gibt, der mitteleuropäischen Realität des Jahres 1999 sehr nahe kommt.

Die Autoren

Dipl. Ing. Karol Frühauf und Dr. math. ETH Helmut Sandmayr betreiben seit 1987 die Beratungsfirma *INFOGEM AG* in Baden, Schweiz.

Karol_Fruehauf@CompuServe.com

Sandmayr@CompuServe.com

Dr. rer. nat. Jochen Ludewig ist ordentlicher Professor für Informatik an der Universität Stuttgart.

Jochen.Ludewig@informatik.uni-stuttgart.de

Umschlagbilder

Max und Moritz haben wir von Wilhelm Busch entlehnt; zum einen sind die beiden offensichtlich von Software umgeben, zum anderen repräsentiert das berühmte Gespann *Management* und *Qualitätssicherung*, die ebenfalls stets zusammengehen sollten. Schließlich lässt sich – leider – auch bei Software häufig die Feststellung verwenden:

Aber hier, wie überhaupt, kommt es anders, als man glaubt.
W. Busch: Plisch und Plum (1882)

Der Elefant am Schluss des Buches ist ein Werk von Imre Sebestyén. Er lässt deutlich erkennen, dass es sich bei Software-Qualitätssicherung nicht um eine leichte Aufgabe handelt.

Die erste Fassung dieses Buches war unseren Kindern gewidmet:

Imre, Linda, Monika und Nora

Sie ist auch in dieser Hinsicht überholt. Die dritte Auflage widmen wir darum jenen jungen Erwachsenen, die einmal Kinder waren:

Imre, Linda, Monika und Nora

Gegenüber der zweiten Auflage von 1991 wurde das Buch weitgehend neu gestaltet, sowohl im Text als auch im Schriftbild. Wieder einmal hat sich dabei gezeigt, dass eine Neuauflage kaum weniger Mühe macht als die Urfassung, denn jeder Gedanke, jedes veränderte Argument droht sich im Netz dessen, was schon auf dem Papier steht, zu verfangen oder gegen die Barrieren zu prallen, die wir in den Köpfen tragen. Auch ein Buch ist eben ein Software-System, und die Software-Wartung ist bekanntermaßen schwierig. Monika Peterhans und Ruedi Schild haben es für uns etwas leichter gemacht – wir danken hierfür.

Da die Arbeit an der dritten Auflage von 2000 schnell belohnt wurde, können wir die vierte mit kleinen Korrekturen auf den Weg bringen.

Wir hoffen, unseren Lesern einen knappen, gut lesbaren Text in die Hand zu geben, der als Lehrbuch und Ratgeber taugt. Für alle verbliebenen oder neu eingeschleppten Fehler, Mängel und Unzulänglichkeiten bitten wir um Nachsicht und um Nachricht!

Baden und Stuttgart, im August 2001
Karol Frühauf, Jochen Ludewig, Helmut Sandmayr

Inhalt

Verzeichnis der Abbildungen

Kapitel 1

Einleitung und Grundlagen

1.1 Zielsetzung der Autoren

Zur Zielgruppe dieses Buches zählen alle diejenigen, die die Aufgabe haben oder bekommen werden, Software-Projekte zu führen oder darin an maßgeblicher Stelle mitzuarbeiten. Dazu gehören offenbar zunächst die Praktiker, dann die Studenten der Informatik und verwandter Fachrichtungen. Die Abgrenzung ist am leichtesten negativ zu formulieren: Zur Zielgruppe gehören weder der Leiter des Riesenprojektes, in dem auch ein wenig Software entsteht, noch der Hilfsprogrammierer, der eine Dateischnittstelle implementiert, ohne das Projekt zu kennen, wohl aber der Software-Projektleiter und jeder, der diese Rolle eines Tages übernehmen könnte.

Unser Buch hat den Zweck, gewisse grundlegende Kenntnisse des Software-Managements zu vermitteln; Software-Qualitätsmanagement gehört dazu. Es wäre vermessen, den Eindruck zu erwecken, dass die Durchführung eines Software-Projekts bereits zur Routine geworden ist und dem Fachmann keinerlei Probleme bietet; ebenso falsch ist es aber, ein solches Unterfangen als völlig unüberschaubares Risiko darzustellen. Wir bemühen uns also zu zeigen, dass man durch Anwendung gesicherter Erkenntnisse vieler Untersuchungen auf dem Gebiet des Software Engineerings ein Software-Projekt (nachfolgend einfach: ein Projekt) systematisch planen, durchführen und kontrollieren kann, gerade so wie ein Projekt auf einem anderen technischen Gebiet.

Zur Projektdurchführung gehört neben Planung, Führung und Kontrolle die technische Arbeit, die verschiedenste Tätigkeiten umfasst und die traditionell in Phasen gegliedert wird. Phasenspezifische Aspekte wie z.B. Spezifikationssysteme oder Teststrategien gehören zweifellos zu den wesentlichen Elementen des Software Engineerings, liegen aber außerhalb der Grenzen, die wir uns für dieses Buch gezogen haben. Auch innerhalb des Buches werden nicht alle Punkte behandelt: Es fehlen z.B. Ausführungen über Zeit-Management, Verhandlungs-, Berichts- und Führungstechnik. Bei den besprochenen Methoden beschränken wir uns nicht auf eine Aufzählung, sondern versuchen Hinweise und Wertungen zu geben, die dem Projektleiter bei der Auswahl für sein Projekt helfen können.

1.2 Begriffe

1.2.1 Rollen im Software-Entwicklungsprozess

Wir unterstellen nachfolgend durchgehend, dass es zwei am Projekt beteiligte Organisationen gibt, nämlich einen *Hersteller*, der die Software entwickelt, und einen *Kunden*, der die Software kauft und nutzt. In der Praxis ist die Situation vielfach weniger klar (siehe Kapitel 1.2.2). Wenn wir vereinfachend von *Unternehmen* sprechen, so sind alle vergleichbaren Organisationen, beispielsweise Behörden, die als Auftraggeber einer Software auftreten, oder Universitätsinstitute, die einen Entwicklungsauftrag übernehmen, eingeschlossen.

Auf der Seite des Herstellers unterscheiden wir für unsere Zwecke nur drei Rollen:

- *Projekteigentümer*: der Repräsentant des Herstellers, unter dessen Aufsicht und Verantwortung das Projekt durchgeführt wird
- *Projektleiter*: der Leiter des Software-Projekts
- *Entwickler*: die Analytiker, Architekten, Programmierer, Tester, usw.

Auf der Seite des Kunden ist für die Diskussion nur die Rolle des *Auftraggebers* relevant; diese Rolle sollte nach Möglichkeit von einer Person und nicht von einer organisatorischen Einheit oder gar einem Ausschuss wahrgenommen werden. Sie vertritt die Anwender, das Management, die Beschaffungsstelle usw. des Kunden.

1.2.2 Projekt

„Projekt" bedeutet im Wortsinn eigentlich „Plan". Als pars pro toto schließt der Begriff aber auch die Durchführung des Plans ein (und auch den leider nicht ungewöhnlichen Fall, in dem gerade der Plan fehlt). Eine ausführliche Definition bietet Thayer (1997):

> *project: A temporary activity that is characterized by having a start date, specific objectives and constraints, established responsibilities, a budget and schedule, and a completion date. If the objective of the project is to develop a software system, then it is sometimes called a software development or software engineering project. (Thayer, 1997)*

Ein *Projekt* ist die Menge aller Tätigkeiten, Interaktionen und Resultate, die mit dem Versuch zusammenhängen, ein bestimmtes Ziel mit begrenzten Mitteln und innerhalb begrenzter Zeit zu erreichen. Bei dem hier zu betrachtenden *Software-Projekt* ist das Ziel die Bereitstellung eines *Software-Systems* oder *-Produkts*.

Für unsere Zwecke ist es sinnvoll, die Projekte grob in drei Kategorien zu ordnen (auch wenn viele Mischungen möglich sind):

A *Auftragsprojekte*
Der Hersteller entwickelt die Software nach den Wünschen eines externen Kunden. Die Verantwortungen sind klar geregelt, zwischen dem Hersteller

und dem Kunden besteht ein Vertrag, in dem Lieferungen mit Zahlungen verknüpft sind. Konflikte werden notfalls durch Gerichte gelöst.

Typischer Inhalt eines solchen Projekts ist eine Industrieanlage, die ein Software-System enthält, im Zeitalter des Outsourcing auch ein Informationssystem.

B *Interne Projekte*
Im Unternehmen wird Software für den eigenen Bedarf entwickelt. Hersteller und Kunde sind in derselben Firma, die Bezahlung erfolgt mit „Scheingeld". Konflikte werden durch den gemeinsamen Vorgesetzten gelöst. Die Motivation für die Abgrenzung des Projekts ist nicht so groß, man sitzt ja im selben Boot und hat Vertrauen zueinander, solange man dadurch „Papier" spart.

Hier geht es meist um Informationssysteme im weitesten Sinne, also um Datenverwaltung oder „EDV", das Resultat dient häufig der Rationalisierung interner Abläufe.

C *Entwicklungsprojekte*
Der Hersteller entwickelt ein Software-Produkt auf eigenes Risiko, um es auf dem Markt anbieten zu können. Die späteren Kunden werden in diesem Fall typisch durch die Marketing-Abteilung oder einen Produkt-Manager repräsentiert; das Geld kommt aus dem Entwicklungsbudget. Konflikte zwischen der Entwicklung und dem Marketing werden von der Geschäftsleitung gelöst.

Typisch für solche Produkte sind Programmpakete, die ein in der Anwendung oft vorkommendes Problem lösen, z.B. Buchhaltungspaket, Textverarbeitung, CASE-Tool, Datensicherungssoftware.

Der Zusammenhang zwischen Projektkategorie und -inhalt kann durch die folgende Tabelle (Abbildung 1.1) dargestellt werden. Darin steht das große X für die typische Situation.

			Typischer Inhalt des Projekts		
	Auftraggeber		Industrie-anlagen	Software-Produkte	Betriebliche Informationssysteme
Kategorie	Wer?	Wo ?			
A	Kunde	extern	X		
B	Management	intern			X
C	Marketing	intern		X	

Abbildung 1.1: Projektkategorien und -inhalte

1.2.3 Software und Software-Produkt

Software entstand als Kunstwort in Anlehnung an das Wort *Hardware* (Eisenwaren). Im IEEE Standard 610.12-1990 ist der Begriff so definiert:

> *software: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.*
> *See also: application software; support software; system software.*
> *Contrast with: hardware.*

Diese Definition wird gern fehlinterpretiert: Die Dokumentation ist nicht *vielleicht* eingeschlossen, sondern *immer*, sofern sie überhaupt existiert. Damit geht der Software-Begriff weit über den Code hinaus; alle Informationen, die zum Rechnerprogramm in Beziehung stehen, gehören dazu, auch wenn sie informal oder chiffriert sind. Einzige Bedingung ist, dass sie in permanenter Form vorliegen: Gedanken oder Gespräche, die in keiner Form aufgezeichnet sind, zählen wir nicht zur Software.

In diesem Buch ist meist nicht von Software allgemein die Rede, sondern von *Software-Produkten*, also von derjenigen Software, die insgesamt ein Produkt oder eine Komponente in einem Produkt oder System bildet.

1.2.4 Software-Qualität

Das Wort Qualität wird – wie viele andere Wörter, z.B. Alter, Höhe – in zwei verschiedenen Färbungen verwendet: Das lateinische Wort *qualitas* (Beschaffenheit) ist beschreibend. So sprechen wir von *ungenügender Qualität* (*geringem Alter, minimaler Höhe*). Im Laufe der Zeit hat sich der *wertende* Gebrauch in den Vordergrund geschoben: „die Qualitäten demonstrieren" heißt nicht einfach „die Beschaffenheit zeigen", sondern impliziert, dass es sich um *hohe* Qualität handelt (vgl.: „Im Alter bekommt man in der Höhe schlecht Luft.").

Beide Auslegungen stehen heute nebeneinander, und es wäre weltfremd, eine davon ausschließen zu wollen. ISO 8402 bietet folgende Definition für Qualität an (Querverweise weggelassen):

> *Qualität: Gesamtheit von Merkmalen (und Merkmalswerten) einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen.*

Wie schwierig es ist, Qualität unverfänglich zu definieren, ist den Vätern und Müttern dieser Norm sehr wohl bewusst gewesen, darum haben sie vier Anmerkungen und eine Fußnote nachgeschoben:

> *Anmerkung 1: In einer vertraglichen Situation oder in einer gesetzlich festgelegten Situation wie etwa auf dem Gebiet kerntechnischer Sicherheit sind Erfordernisse spezifiziert, während in anderen Situationen vorausgesetzte Erfordernisse festgestellt und genau festgelegt werden müssen.*
>
> *Anmerkung 2: In zahlreichen Fällen können sich Erfordernisse im Laufe der Zeit ändern; das bedeutet eine periodische Prüfung der Qualitätsforderung.*

Anmerkung 3: Erfordernisse werden gewöhnlich in Merkmale mit vorgegebenen Werten umgesetzt. Erfordernisse können z.B. Gesichtspunkte der Leistung, Brauchbarkeit, Zuverlässigkeit (Verfügbarkeit, Funktionsfähigkeit, Instandhaltbarkeit), Sicherheit, Umwelt, Wirtschaftlichkeit und Ästhetik mit einbeziehen.

Anmerkung 4: Die Benennung „Qualität" sollte weder als einzelnes Wort gebraucht werden, um einen Vortrefflichkeitsgrad im vergleichenden Sinn auszudrücken, noch sollte sie in einem quantitativen Sinn für technische Bewertungen verwendet werden. Um diese Bedeutungen auszudrücken, sollte ein qualifizierendes Adjektiv benutzt werden. Z.B. können folgende Benennungen verwendet werden:

a) „Relative Qualität", wo Einheiten auf relativer Grundlage nach dem „Vortrefflichkeitsgrad" oder im vergleichenden Sinn geordnet werden (was nicht verwechselt werden darf mit der Anspruchsklasse);

b) „Qualitätslage" in einem quantitativen Sinn (wie bei der Annahmestichprobenprüfung benutzt) sowie „Qualitätsmessgröße", wo genaue technische Bewertungen erfolgen.

Fußnote: „Festgelegte und vorausgesetzte Erfordernisse" sind zwei spezifische Konkretisierungen der Qualitätsforderung. Die Qualitätsdefinition gilt, wie auch die Anmerkung 1 zeigt, für jeden beliebigen Bestandteil und jede beliebige Konkretisierung der Qualitätsforderung, gleichviel wie diese jeweils genannt werden.

Wir werden das Wort Qualität in beschreibendem, nicht in wertendem Sinne benutzen.

1.3 Das Phasenmodell

Das Konzept der Phasen ist zentral für das Projektmanagement. Obwohl die Inhalte der einzelnen Phasen nicht Gegenstand dieses Buches sind, können wir auf die Diskussion der Ideen, die zum Phasenmodell führen, nicht verzichten. Es wurde in Form des sogenannten Wasserfall-Modells 1970 von Royce vorgestellt und vor allem von Boehm seit 1973 verbreitet. Abbildung 1.2 zeigt das Wasserfall-Modell mit der ursprünglichen Terminologie von 1970.

Grundidee des Phasenmodells ist die Feststellung, dass sich Software ebenso wenig wie ein anderes technisches Produkt in *einem* Zuge entwickeln lässt, sondern gewisse Entwicklungsstufen (die *Phasen*) durchläuft. Zwischen der ersten Idee und der endgültigen Außerbetriebnahme der Software durchläuft das Software-Produkt verschiedene Phasen.

Nach jeder Phase wird eine Positionsbestimmung durchgeführt und entschieden, ob und wie der nächste Schritt gewagt werden soll. Diese Entscheidungspunkte werden *Meilensteine* genannt. Die bekannten Modelle des Software-Entwicklungsprozesses verwenden den Begriff Phase mit unterschiedlicher Interpretation.

Im Wasserfall-Modell (Abbildung 1.2) werden die einzelnen Phasen als Tätigkeiten interpretiert. Da es nur bei den einfachsten Programmen möglich ist, die Tätigkeiten streng sequentiell anzuordnen, sind in diesem Modell Zyklen vorgesehen, die aber mit dem Konzept des Meilensteins unverträglich sind.

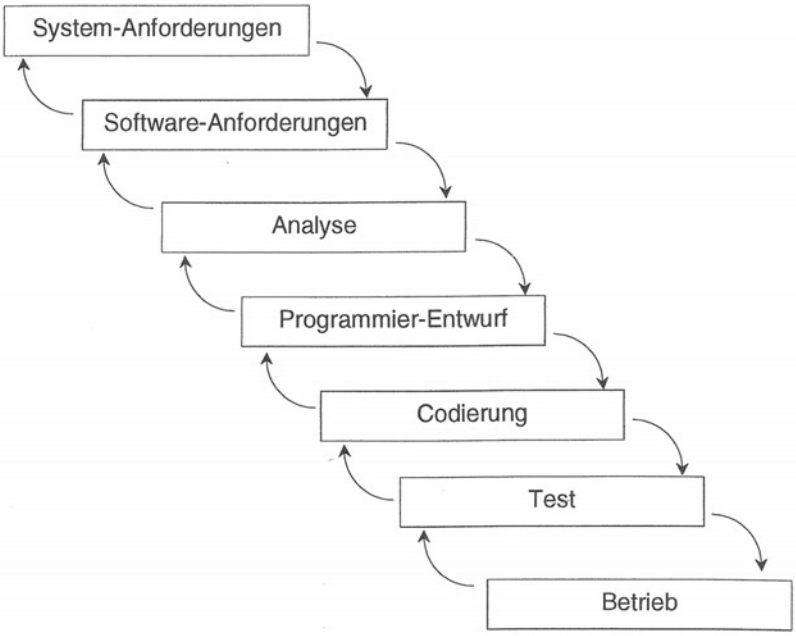


Abbildung 1.2: Das Wasserfall-Modell (Royce, 1970)

Wir verwenden durchgehend den Software-Lebenslauf in der Interpretation als *Kostenmodell*. In der Abbildung 1.3 ist die Kostenverteilung einiger typischer *Tätigkeiten* im Projekt dargestellt (die Verteilung ist nur skizziert, ohne Anspruch auf quantitative Genauigkeit). Aus der Darstellung ist offensichtlich, dass pro Phase verschiedene Tätigkeiten in unterschiedlichem Umfang ausgeführt werden; eine von diesen Tätigkeiten dominiert jeweils und gibt üblicherweise der Phase den Namen.

Ein Meilenstein ist durch *Arbeitsergebnisse* definiert; wenn sie vorliegen und ohne erhebliche Beanstandungen geprüft sind, ist der Meilenstein erreicht. Für jeden Meilenstein ist ein Termin geplant. Die Arbeitsergebnisse sind Dokumente, die in der Phase neu erarbeitet wurden, und Dokumente aus früheren Phasen, die überarbeitet wurden und in neuer Version vorliegen (vgl. Abbildung 1.3). Die Überarbeitung kann durch Fehler in einem Dokument, aber auch durch geänderte Bedürfnisse des Auftraggebers oder durch die im Laufe der Entwicklung gewonnenen Erkenntnisse der Entwickler begründet sein.

Am Meilenstein liegen die Arbeitsergebnisse bereits geprüft und bewertet vor. In den Normen werden häufig Reviews gefordert, an denen Arbeitsergebnisse formell freigegeben werden. Wir bezeichnen sie als *Freigabe-Sitzungen*. In