Software-Qualitätsmanagement in der Praxis

Software-Qualität durch Führung und Verbesserung von Software-Prozessen

von
Ernest Wallmüller

überarbeitet

Software-Qualitätsmanagement in der Praxis – Wallmüller

schnell und portofrei erhältlich bei beck-shop.de DIE FACHBUCHHANDLUNG

Hanser München 2001

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 446 21367 8

Inhaltsverzeichnis: Software-Qualitätsmanagement in der Praxis – Wallmüller

1 Von Software und deren (Un-)Qualität

Wir leben heute in einer Informationsgesellschaft, in der Computer und Informationssysteme eine zentrale Rolle spielen. Software ist ein wesentlicher Bestandteil industrieller Anlagen sowie alltäglicher Produkte, ein unverzichtbares Hilfsmittel in der Verwaltung, in Banken und im Versicherungswesen, im Handel sowie in anderen Branchen. Die Abhängigkeit des Einzelnen von Software-Systemen nimmt ständig zu.

Neben der zunehmenden Bedeutung von Software-Systemen gibt es eine steigende Anzahl von Berichten über Mängel bei der Entwicklung und beim Einsatz dieser Systeme. Spektakuläre Fälle von fehlgeschlagenen Projekten und fehlerhafter Systeme waren beispielsweise [John95]:

- das PKW-Zulassungsprojekt des Staates Kalifornien mit einem Verlust von 54 Millionen Dollar nach 6 Jahren
- das Autovermietungs- und Hotelvermittlungsprojekt der American Airlines mit 165 Millionen Dollar Verlust nach 7 Jahren
- das Gepäckverteilungssystem auf dem Flughafen Denver, das nie fertig wurde und der Stadt Denver einen täglichen Verlust von 1,1 Millionen Dollar bescherte
- das zentrale Steuerverwaltungssystem der U. S. Internal Revenue Service, das schon seit 8 Jahren entwickelt wird, mehr als 200 Millionen Dollar gekostet hat und immer noch nicht funktioniert.

Die US-Bundesverwaltung hat besonders viele Misserfolge vorzuweisen. Weltbekannt ist der Bericht des obersten Rechnungshofes über neue Projekte, bei denen 27 % der bezahlten Software nie geliefert wurden, 52 % nie funktionierten und 18 % erst nach einer aufwändigen Sanierung zum Einsatz kamen. Lediglich 3 % der in Auftrag gegebenen Software erfüllten die vereinbarten Vertragsbedingungen [GAO86].

Diese Liste von fehlgeschlagenen Projekten ließe sich beliebig erweitern. Einerseits stellen wir fest, dass durch die zunehmende Computerdichte immer mehr Software zum Einsatz gebracht wird; dadurch wird unsere Abhängigkeit von Software-Systemen, die steuernde und regelnde Funktionen ausüben, immer größer. Andererseits erkennen wir aber auch, dass es zu wenig gesundes Misstrauen und konstruktive Kritik gegenüber dem Einsatz von Software-Systemen gibt, wahrscheinlich bedingt durch die weit verbreitete Fortschrittsgläubigkeit. Einer der Gründe dafür ist, dass viel zu wenig über fehlgeschlagene Informatikprojekte berichtet wird. Informatikspezialisten sprechen ungern über Misserfolge in ihrer Arbeit. Der Stand des Software- und Projekt-Engineering in Wirtschaft, Industrie und öffentlicher Verwaltung ist aus der Sicht einer ingenieurmäßigen Informatik gegenwärtig noch als unbefriedigend zu bezeichnen. Eines der Kennzeichen dafür ist das Fehlen einer systematischen, auf Methoden und Werkzeuge gestützten Vorgehensweise bei der Software-Entwicklung bzw. bei der Projektabwicklung. Ja, es gibt vielerorts noch die Situation, dass die Entwicklung von Software „auf Zuruf“ erfolgt.

Aufgrund dieser Fakten ist es angebracht, sich verstärkt um die Gestaltung der Software-Qualität zu kümmern und geeignete Prinzipien, Methoden, Techniken und Werkzeuge dafür bereitzustellen. Die Erwartungen, die man an Software-Ersteller und Informatikprojektleiter in allen Bereichen unserer Gesellschaft hat, sind sehr hoch. Einerseits werden von den Software-Entwicklern immer kürzere Entwicklungszeiten gefordert und andererseits eine strikte Einhaltung der Budgets für die Entwicklungsprojekte verlangt. Dass Programme mit zufriedenstellender Produktqualität geliefert werden müssen, ist eine oft implizite Anforderung, die erst bei Nichterfüllung offenkundig wird.

Die Ursachen für die gegenwärtigen Probleme in der Software-Entwicklung sind vielschichtig. Forschungsergebnisse zu diesen Ursachen wurden u. a. von Ewusi-Menach [Ewus97] veröffentlicht. Das Ziel seiner Studie war, die Ursachen für Projektabbrüche zu erforschen. Hauptgründe für den Abbruch von Projekten sind nach Ewusi-Menach:

- unklare Zielsetzung,
- falsche Projektteambesetzung,
- unzulängliche Qualitätssicherung,
- fehlendes technisches Know-how,
- unzureichende Berücksichtigung der Ausgangssituation und
- mangelnde Beteiligung der Anwender.

Auch Tom DeMarco beschreibt die Situation von unklaren Zielsetzungen in seinem Buch „Peopleware“ [DeMa99]. Ohne Übereinstimmung über die zu erzielenden Ergebnisse hat es keinen Sinn, ein Projekt überhaupt anzufangen.

Dass bei komplexen Entwicklungsprojekten die Rollen richtig besetzt werden, sollte eine Selbstverständlichkeit sein. Um so erstaunlicher ist es, dass die Besetzung von Projektpositionen allzu oft eine negative Auslese ist. Entweder passen die Menschen nicht zu den Rollen, die ihnen zugewiesen werden, oder sie passen nicht zueinander. Deshalb sind Fehlbesetzungen, vor allem in der Projektleitung, oft schuld an dem Misserfolg vieler Projekte.

Software-Entwicklungsprojekte geraten häufig außer Kontrolle. Statt kontinuierlich fortzuschreiten, treten sie auf der Stelle oder – schlimmer noch – ihr Zustand verschlechtert sich mit fortschreitender Projektlaufzeit. Die Qualität der Ergebnisse sinkt und die Komplexität der gewählten Lösung steigt. Ohne gewisse Kontrollmechanismen, wie z. B. Assessments, Reviews und Audits, wird eine Verschlechterung der Projektleistung nicht transparent.

Fehlendes technisches Know-how ist ein Dauerzustand in der Informatik, denn kaum ein Wissensgebiet ändert sich so schnell wie dieses. Bis die Software-Techniker mit einer Technik vertraut sind, ist sie vielfach schon überholt, d. h. die Lern- und Erfahrungszeit, die notwendig ist, um eine Technik wirklich zu beherrschen, wird immer kürzer.

Dass die jeweilige Ausgangssituation zu wenig berücksichtigt wird, ist typisch für den falschen Hochmut der Informatiker und ihrer Manager. Sie glauben, es sei nicht not-

wendig, sich mit einer gewachsenen Ist-Situation auseinander setzen zu müssen. Für sie gilt das Soll-Modell. Spätestens dann, wenn es darum geht, den Anschluss an die vorhandenen Systeme zu bewerkstelligen, rächt sich dieses Versäumnis. Auch in der Software-Welt gibt es keine Stunde Null. Alles Neue muss auf dem Alten aufsetzen.

Die letzte Ursache – mangelnde Beteiligung der Anwender – ist oft unvermeidlich. Anwender sind zu sehr mit dem existierenden System beschäftigt, um sich mit dem neuen zu befassen. Wer sie an neuen Projekten beteiligen will, muss sie erst aus dem Tagesgeschäft befreien, und dies erfordert mehr Kosten und eine längerfristige Planung. Auch große Anwenderorganisationen tun sich schwer damit, ihre Anwender für Projektarbeit zu gewinnen, vor allem die Schlüsselpersonen, auf die es schließlich ankommt.

Diese oben angeführten Fakten führen zur Erkenntnis, dass die Planung und Erstellung von Software systematisch und ingenieurmäßig zu erfolgen hat und dass Qualität und deren Management ein Entwicklungsziel sein müssen.

1.1 Das Engineering von Software und Projekten

Die Entwicklung und Pflege von Software-Systemen erfordert ein umfangreiches Ausmaß an Ressourcen. Diese Ressourcen sind ökonomisch zu verbrauchen, um die beabsichtigten Ziele zu erreichen und die Planung zu erfüllen. Bauer postuliert Software Engineering als Disziplin, die mit ingenieurmäßigen Mitteln und ökonomischem Vorgehen dem Entwickler hilft, qualitativ hochwertige Software zu erstellen und zu pflegen [Naur69].

Neben Software Engineering braucht es aber auch ein wirksames Software-Management [Phil98], das insbesondere in der Praxis unterschätzt wird bzw. in akademischen Kreisen oft für überflüssig eingestuft wird. Wirksames Software-Management erfordert einen erfahrenen Projektleiter, einen handhabbaren Projektmanagementansatz und eine disziplinierte ziel- und ergebnisorientierte Projektführung, die in der täglichen Arbeit ihre Anwendung findet. Von herkömmlichen Führungskonzepten unterscheidet sich das Software-Management durch

- eine projektadäquate Organisation,
- ein projektspezifisches Entwicklungsvorgehen,
- eine projektbezogene Planung,
- einen laufenden Soll-/Ist-Vergleich von Software-Prozessen und
- ein definiertes Projektende.

Software-Management schafft eine Balance zwischen Software-Mitarbeitern, Software-Managern, Software-Prozessen und dem Software-Produkt. Dazu ist das Projektgeschehen transparent zu gestalten. Die Verwendung von Werkzeugen zum Software-Konfigurationsmanagement und von Entwicklungsstandards bilden dabei eine wesentliche Unterstützung.

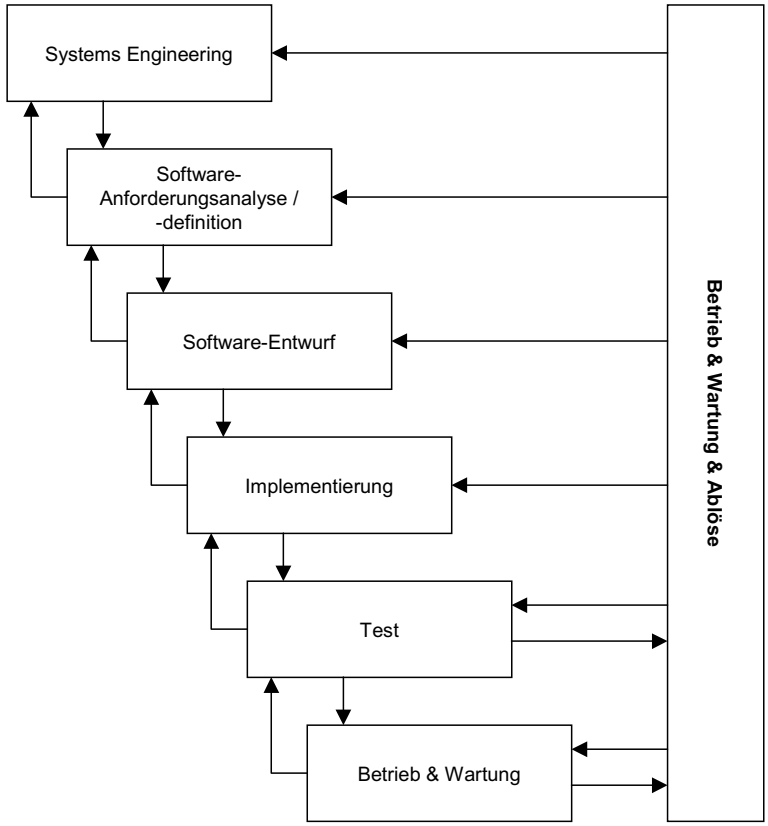

Abb. 1.1: Klassisches Lifecycle-Modell eines Software-Systems

Software-Systeme werden erfahrungsgemäß viele Jahre in Betrieb gehalten, ehe sie vollständig abgelöst werden. Wir sprechen in diesem Zusammenhang vom Lifecycle eines Software-Systems (einer Applikation). Um die Erstellung und Pflege in geordneten Stufen abzuwickeln, wurden Modelle und Hilfsmittel entwickelt. Da Software-Systeme in der Regel in umfassendere Systeme eingebettet sind, gibt es bei jedem Entwicklungs- oder Pflegeprozess auch eine systemtheoretische Sicht, die durch die Hilfsmittel des Systems Engineering ([Daen94], [Thom93], [Stev98]) beschrieben wird. Durchdachte Lifecycle-Modelle stützen sich daher auf die vom Systems Engineering bekannten Erfahrungswerte und berücksichtigen die Aufgaben des Systems Engineering durch eine eigene Phase (siehe Abbildung 1.1).

Beim klassischen Lifecycle-Modell werden folgende Aktivitäten ausgeführt:

- *Systems Engineering*
  Da Software-Systeme immer Teil von größeren Systemen sind, beginnt der Zyklus bei der Analyse der Systemanforderungen und einer Machbarkeitsstudie auf System-

ebene. Von besonderer Bedeutung sind Schnittstellenaspekte der Software-Systeme zur Hardware (z. B. bei integrierten Systemen, aber auch bei Echtzeitsystemen), zur Organisation aus betriebswirtschaftlicher Sicht (z. B. bei großen kommerziellen interaktiven Informationssystemen) und zu Datenbasen.

- *Software-Anforderungsanalyse und -Anforderungsdefinition*
  Der eigentliche Zyklus für die Software-Entwicklung beginnt bei der Sammlung, der Analyse und der Definition der Anforderungen an das Software-System. Um die Ziele und den Zweck des Systems zu verstehen, benötigt der Entwickler genaue Kenntnisse der Anforderungen, die in Form von funktionaler und nicht funktionaler Anforderungen spezifiziert werden. Das Dokumentieren dieser Anforderungen und das Prüfen durch den Auftraggeber sind wichtige Voraussetzungen für die Sicherstellung einer qualitativ guten Software- und Systementwicklung. Gute Fachkenntnisse über die Applikation tragen zu einem besseren Verständnis der Benutzer und ihrer Anforderungen wesentlich bei.
- *Software-Entwurf*
  Der Entwurf ist die Brücke zwischen Anforderungen und der implementierten Lösung. Wichtige Aufgaben sind der Entwurf der Datenstrukturen, der Software-Architektur und der Schnittstellen der Software-Bausteine (Module). Ein guter Entwurf ist nur durch Kreativität und Disziplin realisierbar. Die Nachvollziehbarkeit von Entwurfsentscheidungen erleichtert sowohl die Prüfung der Qualität als auch die Wartung der Software.
- *Implementierung*
  Der Entwurf muss in eine maschinenlesbare Form transformiert werden. Der Mechanisierbarkeitsgrad dieser Tätigkeitengruppe ist am größten, und es ist damit zu rechnen, dass in Zukunft diese Phase durch Generator- und Transformationswerkzeuge automatisiert wird.
- *Testen*
  Durch Testen wird geprüft, ob das implementierte Software-System die geforderten Anforderungen erfüllt und ausreichendes Vertrauen in das System gerechtfertigt ist, um es in der Benutzerumgebung einzuführen und zu betreiben.
- *Betrieb und Wartung*
  Wir zählen die Einführung und die Schulung der Benutzer zu den ersten Aktivitäten in der Betriebs- und Wartungsphase. Diese Phase ist geprägt von den Änderungen, die durch die Benutzer und die Betriebsumgebung, aber auch durch mangelnde Qualität hervorgerufen werden. Die gegenwärtige Situation in der Informatik-Praxis ist gekennzeichnet durch eine größer werdende Menge von Software, die permanent zu warten ist.

Die Probleme mit dem klassischen Lifecycle sind seit längerem bekannt:

- Bei Iterationen in diesem Modell kommt es oft zu erheblichen Qualitätsmängeln (z. B. Nachführen der Dokumentation).

- Anforderungen sind nur begrenzt erfassbar und spezifizierbar. Es gibt Applikationen und Benutzer, die nur vage Anforderungsspezifikationen zulassen. Dies führt zu erheblichen Unsicherheiten bei der Projektführung. Qualitätsprobleme sind dann meist die Folge.
- Je länger es dauert, bis der Auftraggeber bzw. die Benutzer ein ablauffähiges Modell der Applikation (einen Prototyp) zur ersten Prüfung erhalten, umso größer ist das Risiko, die gesetzten Ziele zu verfehlen.

Einen Ausweg aus diesem Dilemma bietet die Verwendung von iterativen, inkrementellen Vorgehensmodellen (siehe Abbildung 1.2). Bei dieser Vorgehensweise wird angenommen, dass die Anforderungen unvollständig sind, sich während des Projekts teilweise ändern können und durch die gewonnenen Erkenntnisse im Laufe der Zeit vervollständigt werden. Statt zu versuchen, das System in einem Guss zu erstellen, wird die Anwendung schrittweise entwickelt, d. h. das System wird als eine Reihe von ausführbaren Programmen (Releases, Iterationen) entwickelt. Die Programme werden dabei nicht weggeworfen, sondern jede Iteration baut auf den Ergebnissen der vorangegangenen Iteration auf. Das Vorgehen ist inkrementell, weil die Funktionalität mit jeder Iteration zunimmt. Jede Iteration besteht aus einem Entwurfs-, Implementierungs-, Test- und Integrationsteil. Iteratives Vorgehen wird erst durch den Einsatz von objektorientierter

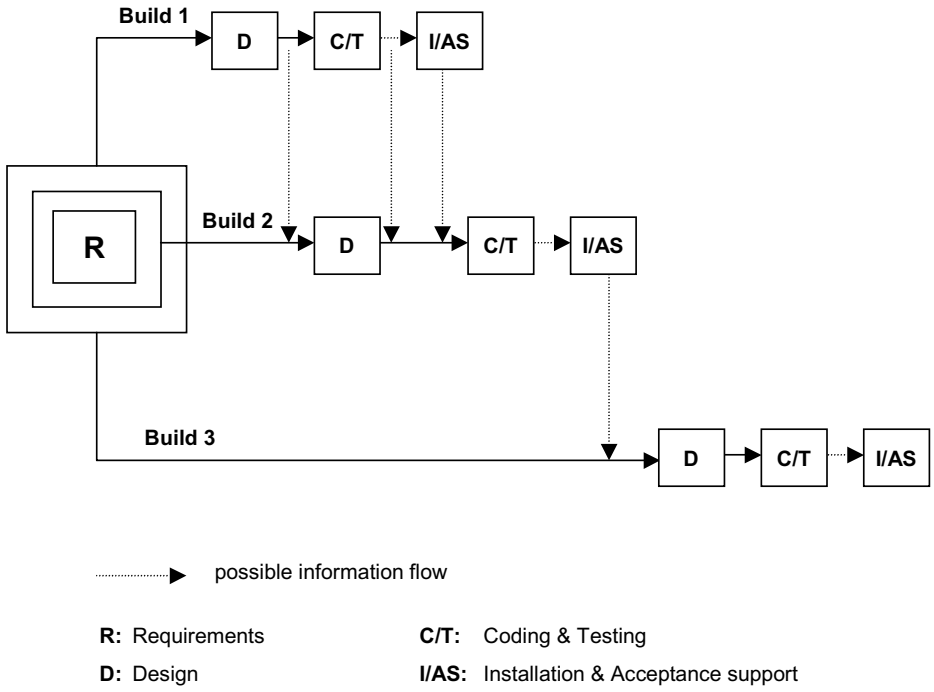

Abb. 1.2: Iteratives, inkrementelles Vorgehen

Technik sinnvoll. Durch die gute Kapselung gibt es klare Schnittstellen zwischen den Komponenten. Teile können getrennt getestet oder durch teilweise funktionsfähige Implementierungen ersetzt werden. In späteren Iterationen werden die internen Implementierungen vervollständigt.

Software-Entwicklung zählt zu den komplexesten Aufgaben, die Menschen zu beherrschen versuchen. Deshalb braucht es für die Software-Entwicklung ein umfassendes Qualitätsmanagement. Software-Qualitätsmanagement ist eine Managementdisziplin, die versucht, den Faktor Qualität durch geeignete Vorgehensweisen (Prozesse), Hilfsmittel und Verfahren sowie durch geplante und systematische Anwendung des Systems, Projekt- und Software Engineering im Lifecycle von Software-Produkten bewusst zu gestalten.

1.2 Grundlagen des Software-Qualitätsmanagements

Im Folgenden werden die verschiedenen Qualitätsaspekte näher untersucht, um zu einer differenzierten und umfassenden Sichtweise auf Software-Qualität zu kommen. Grundsätzlich wird zwischen der Qualität des Produkts und der Qualität des Entwicklungs-, Betriebs- und Wartungsprozesses unterschieden [Masi88]. Masing bezeichnet die Gesamtheit aller Tätigkeiten zur Produktherstellung als den Herstellungsprozess (bei Software-Produkten sprechen wir vom Entwicklungsprozess). Jede einzelne Tätigkeit muss bestimmte Forderungen erfüllen. Diese Forderungen haben die Form von Anweisungen oder Aufgabenspezifikationen. Das Ausmaß der Übereinstimmung von Anweisung und Ausführung einer Tätigkeit definiert deren Qualität. Damit wird die Qualität des Prozesses mess- und analysierbar. Masing schränkt ein, dass eine bloße Übereinstimmung von Anweisung und Ausführung das Qualitätsurteil des Marktes nicht berührt oder gar vorwegnimmt.

Wir gehen von folgendem Sachverhalt aus: die (Qualitäts-)Ziele für das Software-Produkt bestimmen die (Qualitäts-)Ziele für den Entwicklungsprozess. Die Qualität des Entwicklungsprozesses (Reife, Prozessfähigkeit) hat wiederum entscheidenden Einfluss auf die Qualität des Produkts (siehe Abbildung 1.3). Dieses Beziehungsgefüge ist Ausgangspunkt für das prozessorientierte Qualitätsmanagement (siehe Kapitel 2).

Software-Entwicklung, -Beschaffung, -Betrieb und -Wartung sind sehr komplexe Prozesse, die den Einsatz verschiedenster Subprozesse und der dazugehörigen Hilfsmittel erfordern, um ein Produkt kunden- und anforderungsgerecht zu entwickeln, zu beschaffen, zu betreiben oder zu warten. Die Prozesse, die wir dazu benötigen, sind in einigen Referenzmodellen für Software-Prozesse wie z. B ISO/IEC 12207 (Software Lifecycle Processes) oder dem CMM-Prozessmodell dargestellt. Der Standard ISO/IEC 12207 und die dazugehörige Richtlinie ISO 15271 definieren den Software-Engineering-Prozess, der von der Beschaffung bis zur Entsorgung (retirement) des Software-Pro-

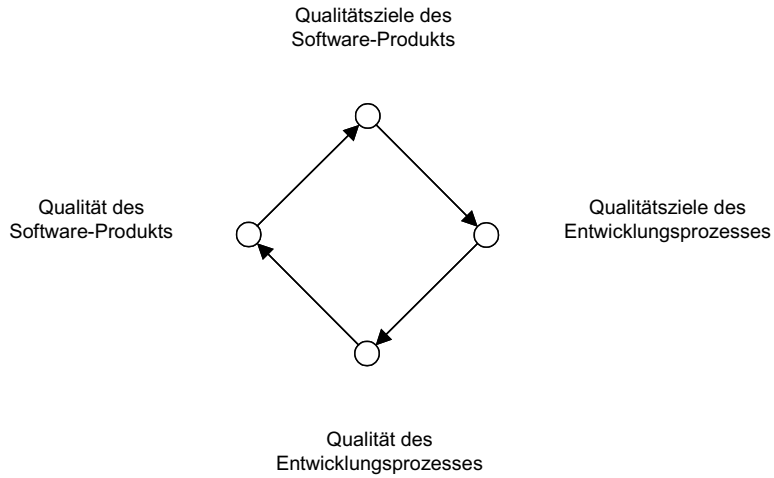

Abb. 1.3: Qualitativer Zusammenhang von Entwicklungsprozess und Software-Produkt

dukts reicht, aus unterschiedlichen Sichten wie z. B. Management, Vertrag, Betrieb, Engineering und Support (siehe Abbildung 1.4).

Die Prozesse werden dazu in die drei Lifecycle-Prozesse Primary, Supporting und Organization eingeteilt (siehe Abbildung 1.5).

In Abbildung 1.5 werden die Software-Lifecycle-Prozesse und ihre Beziehungen aus der unterschiedlichen Sicht ihrer Verwendung diskutiert. Die wichtigsten Sichten sind Vertrags-, Management-, Betriebs-, Engineering- und Supportsicht. Betriebsprozesse werden gegenwärtig aus der Sicht des Service Management organisiert. Ein internationaler Standard dafür ist ITIL [Vogt00].

1.2.1 Was ist Software-Qualität?

Grundlegende Arbeiten zu dieser Frage hat D. A. Garvin von der Harvard-Universität geliefert [Garv84]. Er unterscheidet aufgrund von empirischen Untersuchungen fünf Ansätze, um zu einer Qualitätsvorstellung zu kommen.

W1) Der „transzendente“ Ansatz
Danach ist Qualität universell erkennbar und ein Synonym für kompromisslos hohe Standards und Ansprüche an die Funktionsweise eines Produkts. Die Vertreter dieses Ansatzes glauben allerdings, dass sich Qualität in diesem Sinne nicht präzise definieren oder messen lässt. Ebenso meinen sie, dass Qualität nur durch Erfahrung bewertet werden kann. Der Begriff der Qualität kann genauso wenig implizit definiert werden wie z. B. jener der Schönheit. Vertreter dieses Ansatzes meinen, es genüge, ein Software-Produkt am Bildschirm durch seine Handhabung zu erleben.

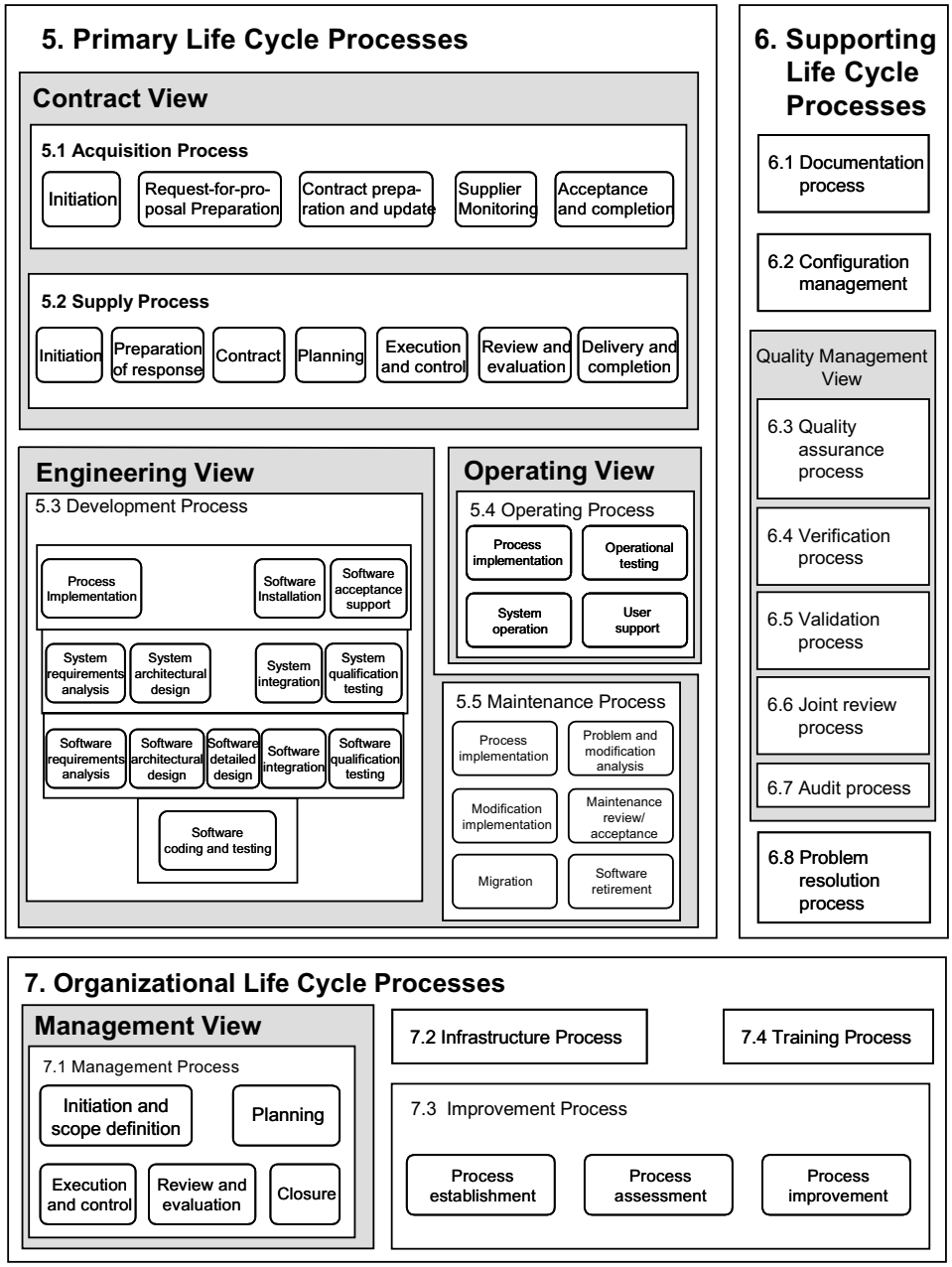

Abb. 1.4: Software-Lifecycle-Prozesse, Sichten und Aktivitäten nach ISO/IEC 12207

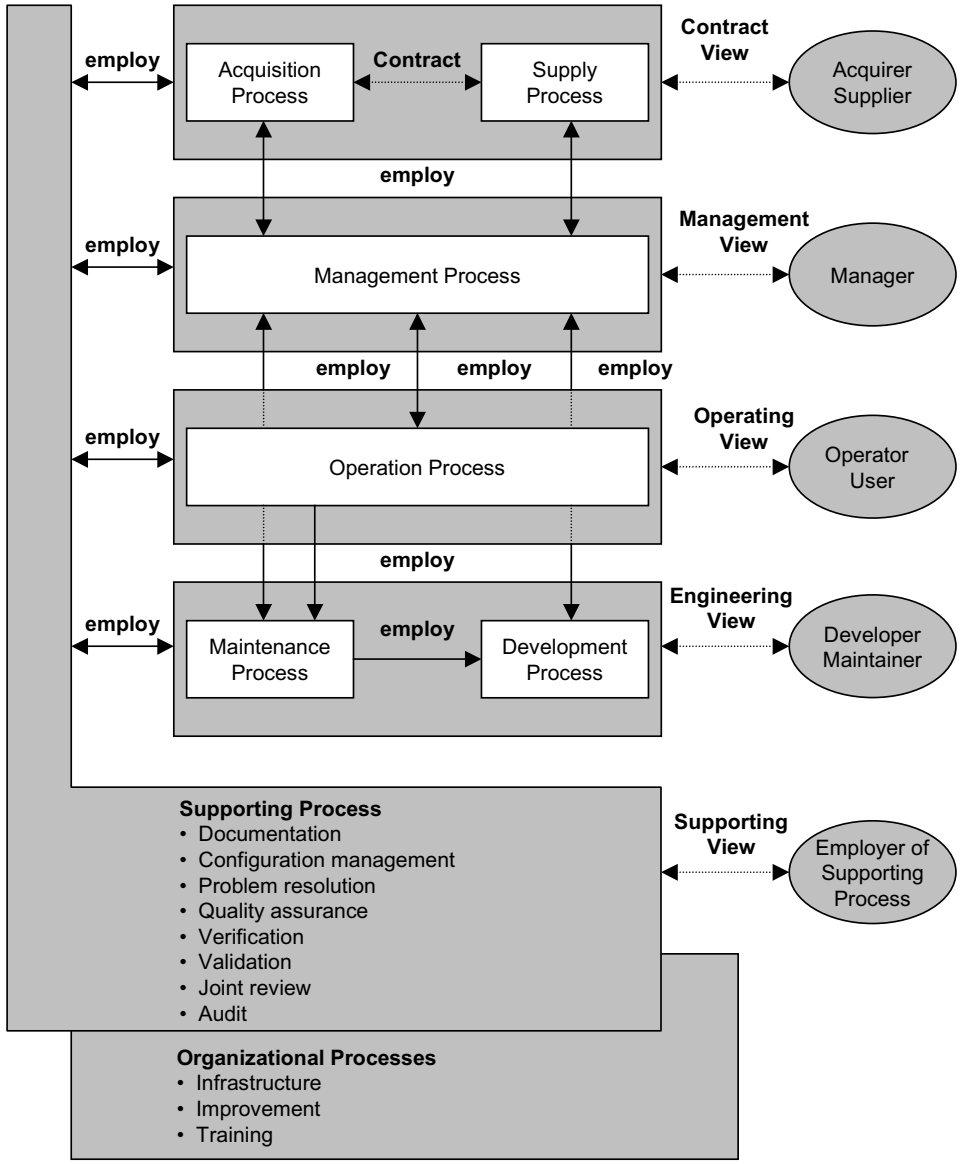

Abb. 1.5: Rollen und Beziehungen in ISO/IEC 12207

W2) Der produktbezogene Ansatz

Die Vertreter dieses Ansatzes glauben, dass die Qualität präzise messbar ist. Danach spiegeln Qualitätsdifferenzen Unterschiede in der vorhandenen, beobachtbaren Quantität bestimmter Eigenschaftsausprägungen wider, die in einem Produkt festgestellt werden können. Dieser Ansatz erlaubt es im Prinzip, eine Rangordnung von verschiedenen Software-Produkten der gleichen Kategorie anzugeben.

In den 60er Jahren des letzten Jahrhunderts haben amerikanische Auftraggeber von Großprojekten begonnen, Software-Bewertungsmethoden zu entwickeln, um bei der Beschaffung und Entwicklung unterschiedliche Produkte von Software-Lieferanten besser beurteilen zu können. Es wurden Merkmalsmodelle wie z. B. jenes von McCall [McCa77] und Bewertungsverfahren entwickelt, die es erlauben, Produkte qualitativ zu evaluieren und zu klassifizieren.

W3) Der anwenderbezogene Ansatz

Hierbei liegt die Auffassung vor, dass die Qualität durch den Produktbenutzer festgelegt wird und weniger durch das Produkt selbst. Danach haben verschiedene Produktbenutzer unterschiedliche Wünsche und Bedürfnisse, und diejenigen Produkte, die diese Bedürfnisse am besten befriedigen, werden als qualitativ hochstehend angesehen. Beispielsweise werden bei der Einführung des Unternehmensportals mySAP.com im Finanzbereich eines Unternehmens zuerst einmal die Rollen (z. B. Senior Management, Kostenstellenverantwortliche, Controller) definiert. Durch eine HTML-basierte Web-Oberfläche wird den unterschiedlichen Benutzerrollen der Zugriff auf SAP-Funktionen ermöglicht. Das „Look and Feel“ der Webseiten kann rollen-, unternehmens- und branchenspezifisch den Anforderungen der Nutzer an ihren Arbeitsplatz angepasst werden.

W4) Der prozessbezogene Ansatz

Nach diesem Ansatz ist Qualität gleichzusetzen mit der Einhaltung von Spezifikationen, mit der Idealvorstellung, eine Tätigkeit zur Produkterstellung gleich das erste Mal richtig auszuführen. Diese Vorstellung von Qualität ist auf die heutige Wirtschaft und Industrie ausgerichtet. Im Mittelpunkt steht der Produktionsprozess, der kontrolliert wird, um die Ausschuss- und Nachbearbeitungskosten zu verringern. Dabei spielt der Automatisierungsgrad eine große Rolle. Insbesondere Roboter und Automaten sollen gewährleisten, dass Produktionsprozesse soweit als möglich mängelfrei und reibungslos abgewickelt werden. Für Software-Projekte bedeutet dies, dass durch geeignete Auswahl und Anpassung (Tailoring) von Prozess-/Vorgehensmodellen und durch ausreichenden Einsatz prüfender (analytischer) Qualitätsmaßnahmen wie z. B. durch Reviews Prozesse so angewandt werden, dass Prozess- und Produktspezifikationen erfüllt werden. Der Vorteil dieses Ansatzes ist, dass durch Prozesskostenrechnung, Quality Function Deployment (QFD) und Kundenorientierung auch die Eigenschaften der Ansätze von W2, W3 und W5 erfüllt werden können.

W5) Der Preis-Nutzen-bezogene Ansatz
Bei diesem Ansatz wird ein Bezug zwischen Preis und Qualität hergestellt. Ein Qualitätsprodukt ist in dieser Denkweise ein Erzeugnis, das einen bestimmten Nutzen zu einem akzeptablen Preis oder eine Übereinstimmung mit Spezifikationen zu akzeptablen Kosten erbringt. Beispielsweise möchte der Finanzvorstand eines Energieversorgungsunternehmens mit einer neuen Informatik-/Software-Lösung für das Rechnungswesen eine jährliche Einsparung von ca. 15 Millionen Euro realisieren. Die neue Lösung kostet ca. 60 Millionen Euro und er rechnet mit einer Amortisierung dieser Investition nach 4 Jahren.

Neben diesen Ansätzen, die versuchen, den Begriff Qualität durch Umschreibung zu verdeutlichen, gibt es auch eine Reihe von Versuchen, den Begriff Qualität zu definieren. Je nach Präferenz und individueller Auffassung des Einzelnen werden dabei verschiedene Akzente gesetzt. Im Folgenden werden verschiedene Qualitätsbegriffe diskutiert und ihre Anwendbarkeit für Software-Produkte geprüft.

In der Norm ISO 8402 heißt es:

> *„Qualität ist die Gesamtheit von Merkmalen einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen."*

Diese Definition geht von einem strengen Kunden-/Lieferantenverhältnis aus. Es lassen sich aus dieser Definition zwei wesentliche Schlussfolgerungen ziehen:

- Die Eignung ein und derselben Sache kann für verschiedene Verwendungen unterschiedlich sein. Beispielsweise ist ein komplexes, alles an Funktionalität bietendes Konfigurationsmanagement-Tool für kleine Software-Projekte (Aufwand < 6 Personenmonate) schlechter geeignet als ein einfacheres Tool, das weniger an Funktionalität bietet.
- Die Erfordernisse ergeben sich aus dem Verwendungszweck des Software-Produkts. Ein Air-Traffic-Control-System hat ca. 6000 Anforderungen. Zur Prüfung dieser Anforderungen ist ein Anforderungsdatenbanksystem wie z. B. DOORS oder C.A.R.E. einzusetzen. Mit diesen Werkzeugen ist eine ausreichende Prüf- und Verfolgbarkeit der Anforderungen möglich.

Ein Produkt oder eine Tätigkeit (z. B. eine Dienstleistung) hat verschiedene Eigenschaften, von denen nicht alle die Qualität konstituieren. Eigenschaften, die die Qualität festlegen, sind jene, die für den Produktbenutzer oder den Dienstleistungsnehmer relevant sind, d. h. die vorgegebenen Erfordernisse erfüllen.

Die IEEE-Norm für Software-Qualität (IEEE Std 729-1983) hebt die Erwartungen der Kunden hervor:

> ***Software quality:***
>
> *(1) The totality of features and characteristics of a software product that bear on its ability to satisfy given needs; for example, conform to specifications.*
>
> *(2) The degree to which software possesses a desired combination of attributes.*

(3) The degree to which a customer or user perceives that software meets his or her composite expectations.

(4) The composite characteristics of software that determine the degree to which the software in use will meet the expectations of the customer.

Wenn wir uns nicht nur mit der Bewertung von fertigen Software-Produkten zufrieden geben, sondern Qualität auch konstruktiv realisieren wollen, müssen wir einen Bewertungsansatz für den Entwicklungs- und Pflegeprozess aufstellen. Das bedeutet, dass wir Merkmale und Bewertungsmaßstäbe auch für Zwischen-/Endprodukte und für deren Entwicklungsaktivitäten in allen Phasen des Prozesses benötigen.

Gleichzeitig sollte uns bewusst sein, dass Qualität nichts Absolutes ist, sondern immer relativ zu gegebenen Erfordernissen gesehen werden muss. Daraus leiten wir ab, dass eine Qualitätsbewertung immer einen Vergleich beinhaltet zwischen den aus den gegebenen Erfordernissen abgeleiteten Qualitätsvorgaben (Soll-Werte) und den tatsächlich erreichten Ausprägungen der Merkmale (Ist-Werte).

Boehm stellt zur Problematik der Bestimmung der Qualität eines Software-Produktes sinngemäß folgende drei Fragen [Boeh78]:

1. *Problem der Definition von Software-Qualität*
   Ist es überhaupt möglich, Definitionen der Eigenschaften und Merkmale eines Software-Produkts aufzustellen, die messbar sind und sich nicht überschneiden?
2. *Problem der Qualitätsprüfung*
   Wie gut kann man die Qualität eines Software-Produkts bzw. die Eigenschaften und Merkmale messen, welche die Qualität des Software-Produkts bestimmen?
3. *Problem der Qualitätslenkung*
   Wie kann man Informationen über die Qualität des Produkts zur Verbesserung des Produkts im Lifecycle verwenden?

Zunächst einmal spricht er das Problem der Eindeutigkeit und Relevanz von Prozess- und Produktmerkmalen an. Jeder von uns definiert und interpretiert heute (Qualitäts-)Merkmale anders. Eine Klärung der Begriffe hinsichtlich der Produktmerkmale erfolgt durch die Norm ISO 9126. In dieser Norm werden Merkmale für die Evaluation von Software-Produkten und ein Prozess zur Evaluation vorgestellt. Die in der Norm definierten Merkmale sind Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Portabilität.

In der zweiten Frage taucht das Problem der Vollständigkeit und Prägnanz der Qualitätsmerkmale auf. Hier geht es mit anderen Worten darum, wie gut wir einzelne Anforderungen mit einem vorgegebenen Instrumentarium formulieren können, ob es dabei Lücken gibt oder ob gewisse Anforderungen mehrfach durch verschiedene Qualitätsmerkmale abgedeckt werden.

In der dritten Fragestellung klingt durch, dass Qualität eine Steuerungsgröße für den Prozess ist. Entscheidend ist die Rückkopplung der Ergebnisse von Qualitätsprüfungen mit der Prozesssteuerung.

Eine Voraussetzung für ein qualitätsbewusstes Umgehen mit Software ist, dass diese als Produkt gesehen wird. Wir meinen damit, dass eine aktuelle und ausreichende Dokumentation, in der neben Programmen auch die Daten beschrieben werden, existiert, um überhaupt von einem Produkt sprechen zu können.

Ein Software-Produkt (siehe IEEE Std 729, Glossar der Software-Engineering-Terminologie) besteht aus den Teilen Sourcecode, Objektcode und Dokumentation.

Differenzierte und den Erfordernissen von Software-Produkten angepasste Abgrenzungen des Qualitätsbegriffs bieten Qualitätsmodelle. Die bekanntesten sind jene von McCall [McCa77], Boehm [Boeh76], Willmer [Will85], NEC [Azum85] und Siemens ([Asam86], [Zopf88]). Der Autor hat im Rahmen eines Projektes, bei dem ein Qualitätssystem aufgebaut und eingeführt wurde, ebenfalls ein Qualitätsmodell erstellt [Wall87]. Es wird im Folgenden SPARDAT-Modell genannt. Qualitätsmodelle tragen wesentlich zur Vereinheitlichung der verschiedenen Vorstellungen über Software-Qualität bei. Durch ihre strukturelle Zerlegungssystematik wird der nebelhafte Begriff Software-Qualität konkretisiert. Die am weitesten entwickelten Modelle, beispielsweise das von McCall und von NEC, erlauben es, Qualitätsplanung und Qualitätsbewertungen damit durchzuführen. Wir diskutieren Qualitätsmodelle in Abschnitt 1.4 ausführlich.

1.2.2 Wo entstehen Software-Qualität bzw. Software-Mängel?

Die Qualität eines Software-Produkts wird bestimmt durch den Prozess, in dem es entwickelt wird, und die Merkmale, die es besitzt. Der Entwicklungsprozess kann zeitlich durch Iterationen oder Phasen strukturiert werden, wir sprechen dann von einem Phasen- oder Vorgehensmodell (siehe Abschnitt 3.2). Die Qualität muss in jeder dieser Phasen entstehen. Wichtig ist, dass an jede Phase Anforderungen (bezogen auf die Phasenergebnisse, aber auch auf die Aktivitäten) gestellt werden und diese am Ende der Phase auf Erfüllung überprüft werden. Für das Management ist das Erreichen von Meilensteinen eine wichtige Prozessanforderung. Beispielsweise ist die Fertigstellung eines Dokuments (z. B. Pflichtenheft) am Ende einer Phase ein Meilenstein. Die Gesamtqualität eines Produktes setzt sich nun stufenweise aus der Qualität der Phasenergebnisse und der Erfüllung der Prozessanforderungen zusammen.

Wie bereits weiter oben erwähnt, spielt der Entwicklungsprozess und insbesondere seine Qualität eine wesentliche Rolle bei der Entstehung der Produktqualität. In diesem Zusammenhang sprechen wir auch von Prozessqualität. Voraussetzungen für gute Prozessqualität sind das Ausmaß des systematischen, methodischen Entwickelns (beispielsweise das Einhalten von Entwicklungsstandards), die Sorgfalt des Projektmanagements, die Qualifikation und der Ausbildungsstand der Mitarbeiter sowie die Beschaffenheit der eingesetzten Hilfsmittel.

Wie und wann entstehen Fehler im Prozess? Nach Mizumo [Mizu83] kommt es beim Entstehen von Fehlern zu einem Summationseffekt (siehe Abbildung 1.6). Bei einem Projekt beginnt man normalerweise mit der Erfassung, Analyse und Definition der

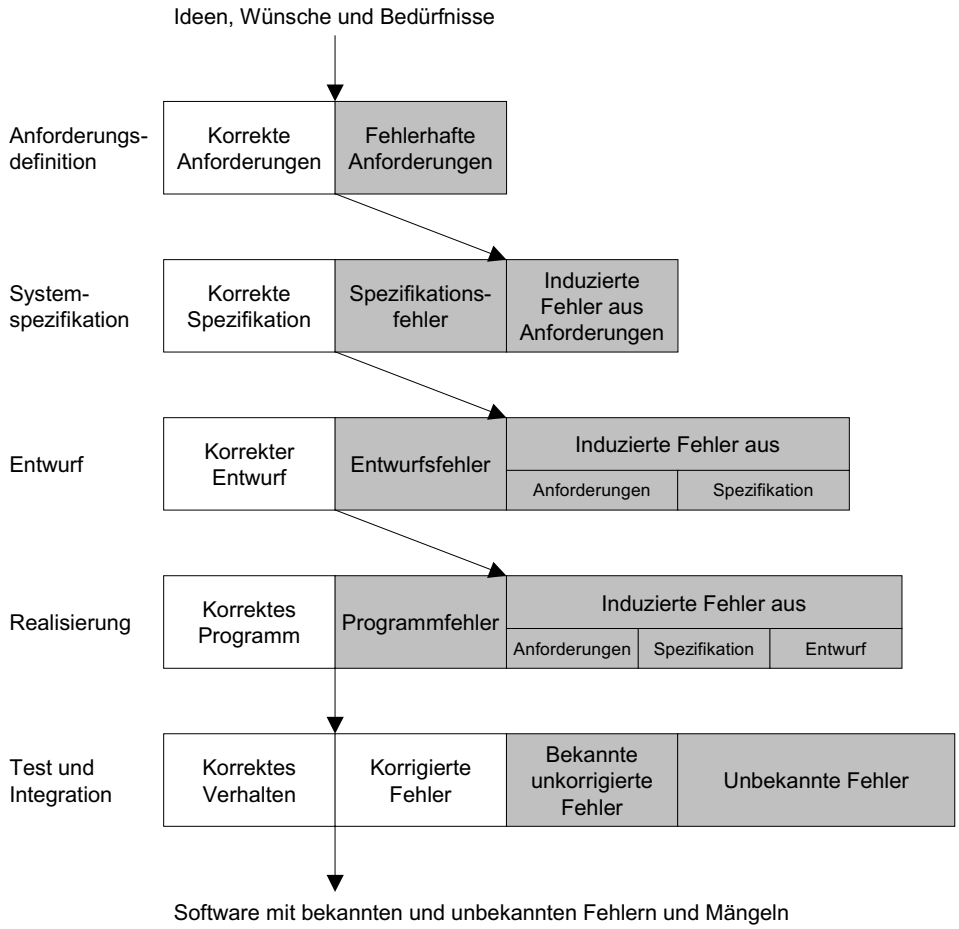

Abb. 1.6: Summationseffekt von Fehlern

Anforderungen. Diese Anforderungen werden in Form einer Anforderungsspezifikation festgehalten. Erfahrungsgemäß ist ein Teil dieser Spezifikation korrekt, ein anderer Teil ist mit Fehlern behaftet.

In der nächsten Phase wird der Entwurf erstellt. Ergebnis ist eine Entwurfsdokumentation. Ein Teil des Entwurfs ist nun korrekt, ein anderer Teil ist mit Fehlern behaftet, die im Entwurf entstanden sind, und ein dritter Teil des Entwurfs basiert auf der fehlerhaften Spezifikation.

Der nächste Schritt ist die Programmierung. Ein Teil der Programme ist korrekt, ein anderer Teil ist mit Fehlern behaftet. Ein weiterer Teil der Programme basiert auf fehlerhaftem Entwurf, fehlerhafter Spezifikation und fehlerhaften Anforderungen.

In der nachfolgenden Integrations- und Testphase haben wir folgende Situation: ein Teil der Programme funktioniert korrekt. Ein anderer Teil der Programme enthält korrigierbare Fehler, die auch korrigiert werden. Ein weiterer Teil enthält nicht korrigierbare Fehler und wiederum ein anderer enthält versteckte Fehler. Zusammenfassend lässt sich sagen, dass zu diesem Zeitpunkt ein unvollkommenes Software-Produkt vorliegt. Diese Unvollkommenheit ist durch einen Summationseffekt der aufgetretenen Fehler entstanden, der in gesamter Tragweite erst gegen Ende des Projekts zum Vorschein kommt.

Dieser Effekt tritt auch bei anderen technischen Produktentwicklungen auf. Dort hingegen gibt es seit langem in der Entwicklung integrierte Zwischenprüfungen.

Durch geeignete Qualitätsprinzipien kann dieser Effekt verringert werden. Eines der wichtigsten Prinzipien ist, bereits zu Beginn des Lifecycle Prüfungen (z. B. der Anforderungsspezifikation) einzusetzen sowie alle Prozessergebnisse zu prüfen. Wenn dies nicht geschieht, kann nach [Boeh00] die Behebung der Qualitätsmängel sehr teuer werden (siehe Abbildung 1.7), da mit zunehmender Verweilzeit eines Fehlers im Produkt seine Behebungskosten steigen. Auf Qualitätsprinzipien wird in Abschnitt 1.2.3 detailliert eingegangen.

Eine Hauptquelle für Mängel und Fehler ist die Planung. Wir meinen damit die Planung des Projekts als Ganzes und des Entwicklungsprozesses im Speziellen. Nach ei-

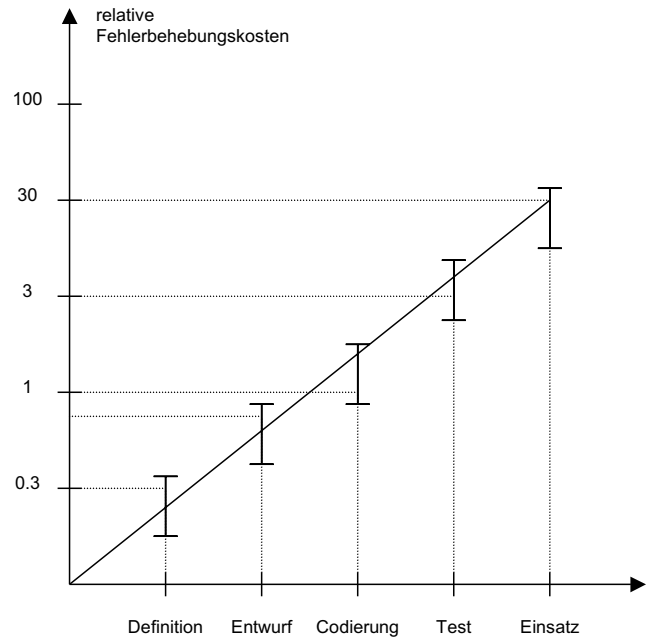

Abb. 1.7: Relative Fehlerbehebungskosten nach Boehm

ner Untersuchung von Thayer u. a. über Probleme des Managements von Software-Projekten [Thay81] betreffen 60 % der Probleme die Projektplanung und 20 % die Projektkontrolle. Der Rest sind diverse Führungsprobleme sowie Probleme, die das Verständnis der technischen Projektarbeit angehen. Wir sehen daraus, welchen Stellenwert die Planung hat, um ein Projekt erfolgreich durchzuführen.

Wir starten den Software-Entwicklungsprozess mit ganz bestimmten Prozess- und Produktanforderungen, wie beispielsweise der Anforderung nach einer vorgegebenen Produktqualität, oder der Anforderung, das Projekt termingerecht und kostengerecht, d. h. im Budgetrahmen, fertig zu stellen. Wie können wir den Entwicklungsprozess so gestalten, dass wir diese Anforderungen erfüllen können? Die Antwort liegt im Vorbereitungsprozess zum Projekt (für Software-Lieferanten im Offertprozess oder für Inhouse-Entwicklungen im Projektantragsprozess), im Teil Planung des Projektmanagementprozesses und im eigentlichen Start-up-Prozess des Projektes, der erst dann gestartet werden kann, wenn das Projekt genehmigt wurde.

Eines der wesentlichsten Hilfsmittel, um eine qualitätsorientierte Projektplanung durchführen zu können, ist der Projektmanagementplan, der aus verschiedenen Teilplänen besteht (siehe [Früh87], IEEE Std 1058.1-1987). Die wichtigsten Inhalte dieses Planes sind in Abbildung 1.8 enthalten.

Wir stellen fest, dass die Vorbereitung zum Projekt (wie kommt es überhaupt zum Projekt und wie gut ist die Offerte/der Projektantrag erstellt und geprüft?) und das eigentliche Aufstarten des Projekts sehr kritische und qualitätsrelevante Elemente sind und mit besonderer Achtsamkeit und Sorgfalt durchzuführen sind.

1.2.3 Prinzipien des Software-Qualitätsmanagements

Die Software-Erstellung unterliegt wie jeder andere industrielle Produktions- und Entwicklungsprozess dem Zwang zu hoher Produktivität. Dies wiederum bedingt eine Berücksichtigung der Ziele Termintreue, minimale Kosten und zufriedenstellende Produktqualität. Wir sprechen dann von einer wirtschaftlichen Software-Erstellung, wenn diese Ziele verwirklicht worden sind. Um die Ziele, wie die Erfüllung der Kosten-, Termin- und Produktqualitätsanforderungen, zu erreichen, müssen gewisse Prinzipien beachtet werden. Diese Prinzipien leiten sich aus den besprochenen Phänomenen des Entwicklungs- und Wartungsprozesses und den Erfahrungen der Software-Krise ab [DGQ95].

Folgende Prinzipien sind zur Beeinflussung der Qualität gegenwärtig bekannt:

P1) *Konkrete operationalisierbare Qualitätsmerkmale*
Entscheidend für die Qualität ist, dass wir am Beginn des Projektes für das zu planende Produkt die wesentlichen Qualitätsmerkmale aus der Sicht des Auftraggebers und Benutzers erstellen. Zur Auffindung der Merkmale kann QFD (siehe Kapitel 7) eingesetzt werden. Wichtig ist, dass diese Qualitätsmerkmale

1 Einführung
1.1 Projektüberblick
1.2 Evolution des PMP
1.3 Referenzierte Dokumente
1.4 Definitionen und Abkürzungen

2 Projektorganisation
2.1 Prozessmodell
2.2 Organisationsstruktur
2.3 Grenzen und Schnittstellen des Projekts
2.4 Projektverantwortlichkeiten

3 Führungsprozess
3.1 Ziele und Prioritäten
3.2 Annahmen, Abhängigkeiten und Einschränkungen
3.3 Risikomanagement
3.4 Projektkontrolle und Berichtswesen
3.5 Mitarbeitereinsatzplan
3.6 Qualitätsmanagement
3.7 Konfigurationsmanagement
3.8 Change-Management

4 Technischer Prozess
4.1 Methoden, Werkzeuge und Techniken
4.2 Software-Dokumentation
4.3 Projektunterstützungsfunktion/Project Office
4.4 Projektarchiv

5 Arbeitspakete, Zeitplan und Budget
5.1 Arbeitspakete
5.2 Ergebnisse
5.3 Abhängigkeiten
5.4 Ressourcen-Anforderungen
5.5 Budget und Ressourcen-Zuweisung
5.6 Zeitplan

6 Eskalationsprozedur
6.1 Eskalationskriterien
6.2 Eskalationsgremium

7 Offene Punkte

8 Unterschriften

Abb. 1.8: Muster einer Inhaltsstruktur eines Projektmanagementplans (PMP) nach IEEE Std 1058.1

konkret und möglichst quantifizierbar sind. Es wird dadurch auch leichter möglich, zwischen verschiedenen Projektkategorien, wie zum Beispiel Standardprojekten und Sonderprojekten mit speziellen Anforderungen, zu unterscheiden. Es ist ebenso von Bedeutung, dass es gelingt, für einzelne Phasen konkrete Qualitätsmerkmale zu finden. Diese Merkmale können sowohl prozess- als auch produktorientiert sein. Beispielsweise ist es sinnvoll, für die Fertigstellung eines Pflichtenheftes bestimmte Qualitätsmerkmale von vornherein festzulegen und vorzugeben (siehe dazu Qualitätsmerkmale für Anforderungsdefinitionen der SAQ [SAQ88]). Aus der Sicht der Prozessqualität ist es beispielsweise wichtig, für das Testen ausreichende Testzeit und Testressourcen bereitzustellen. Generell ist das rechtzeitige Bereitstellen ausreichender Ressourcen ein Indikator für gute Prozessqualität.

P2) Produkt- und projektabhängige Qualitätsplanung

Es wird heute vielfach zu wenig untersucht, was der eigentliche Verwendungszweck des geplanten und zu entwickelnden Software-Produktes ist, wie die Lebensdauer dieses Produktes gesehen wird und wer die potenziellen Benutzer sind. Alle diese Faktoren beeinflussen die Ausprägung der Qualitätsanforderungen an ein Projekt und an ein Produkt wesentlich. Je nach Ausprägung dieser Qualitätsanforderungen sind geeignete Software-Engineering-Methoden und -Werkzeuge, aber auch Qualitätsmaßnahmen auszuwählen. Beispielsweise sind für ein Kundeninformationssystem einer Bank die Handhabbarkeit, die Verfügbarkeit und die Zuverlässigkeit wichtige Qualitätseigenschaften. Durch Ergonomie-Reviews sowie spezielle Last- und Stresstests kann die Ausprägung dieser Merkmale geprüft werden.

Bei der Qualitätsplanung sind auch die Projektrisiken zu berücksichtigen. Erfahrungen der Praxis zeigen, dass die Qualitätssicherungspläne bei Nichtberücksichtigung der Projektrisiken, wie beispielsweise Komplexität und Neuigkeitsgrad des Produkts, wertlos werden. Folgende Risikofaktoren haben Einfluss auf die Produkt- und Qualitätssicherungsplanung:

- Größe des Projekts (Anzahl der Mitarbeiter, Aufwand an Personenjahren),
- Qualifikation der Mitarbeiter,
- Komplexität und Neuigkeitsgrad des Vorhabens,
- Umfang der Anforderungen,
- Grad der Unbestimmtheit der Anforderungen,
- Arbeitsteilung zwischen den Mitarbeitern und Anzahl externer Entwickler,
- Zeitdruck.

Diese Risikofaktoren liegen in unterschiedlicher Ausprägung bei einem Projekt vor. Bei der Projekt- und Qualitätsplanung sind jene Faktoren zunächst zu identifizieren, die ein potenzielles Risiko in sich bergen. Bei vorliegendem Risiko sind entsprechende Qualitätsmaßnahmen zu wählen und die Projektplanung (Planungshorizont, Ressourcen) ist zu verändern bzw. zu verfeinern.

P3) Rückkopplung der Ergebnisse der Qualitätsprüfungen
Diese Rückkopplung ist ein wesentlicher Bestandteil für eine erfolgreiche Qualitätslenkung. Durch sie können Soll-Ist-Abweichungen des geplanten Qualitätsniveaus erkannt werden. Das Prinzip dabei ist, dass Informationen, die bei Qualitätsprüfungen gewonnen wurden, zu entsprechenden Korrekturmaßnahmen im Entwicklungsprozess führen. Reviews (Inspektionen und Walkthroughs) und Audits (zur Prüfung des Entwicklungsprozesses oder des Managementprozesses) liefern die zur Korrektur des Entwicklungsprozesses notwendigen Informationen.

P4) N-Augenkontrolle bei Qualitätsprüfungen
Bei diesem wichtigen Prinzip werden die kognitiven Fähigkeiten einzelner Personen durch Gruppenprozesse, wie beispielsweise gemeinsame Analyse eines Dokuments, besser genutzt. Dass Menschen Fehler machen, ist eine Tatsache. Es ist daher auch natürlich, die intellektuellen Fähigkeiten der Menschen zur positiven Qualitätsbeeinflussung zu nutzen. Neben den bereits im vorhergehenden Punkt genannten Prüfmethoden geht es hier um informelle Prüfungen. Davon spricht man beispielsweise dann, wenn ein Entwickler eines Zwischen- oder Endprodukts einer Phase seinen Kollegen bittet, sich dieses Dokument anzusehen und Korrektur zu lesen.

P5) Maximale konstruktive Qualitätsmaßnahmen
Während es bei der Qualitätsprüfung darum geht, die vorhandene bzw. nicht vorhandene Qualität zu bestimmen, ist das Ziel der konstruktiven Qualitätsmaßnahmen, Fehler im Entwicklungsprozess zu vermeiden. Die Devise lautet also: „Keine Fehler machen ist besser als Fehler entdecken bzw. beheben." Durch geeignete vorbeugende Maßnahmen im Entwicklungsprozess wird die Produktqualität direkt verbessert und erhöht. Erfahrungen zeigen auch, dass durch vorbeugende Maßnahmen der Aufwand der Qualitätsprüfungen, beispielsweise des Testens, sich erheblich reduzieren lässt. Ein weiterer positiver Aspekt dieses Prinzips ist, dass durch konstruktive Qualitätsmaßnahmen Qualitätsprüfungen erst ermöglicht werden. Beispielsweise kann durch eine geeignete Schnittstellenspezifikation für Module, in der Vorbedingungen, Nachbedingungen, der Effekt des Moduls und Fehlerbedingungen enthalten sind, das spezifikationsorientierte Testen wesentlich erleichtert werden. Konstruktive Qualitätsmaßnahmen und Qualitätsprüfungen ergänzen einander und bringen so einen effektiven Nutzen.

P6) Frühzeitige Entdeckung und Behebung von Fehlern und Mängeln
Wie Boehm [Boeh00] gezeigt hat (siehe Abbildung 1.7), steigen die Kosten für die Entdeckung und Behebung eines Fehlers exponentiell mit der Dauer zwischen dem Zeitpunkt des Entstehens und dem Zeitpunkt der Entdeckung des Fehlers. Ein Fehler in der Anforderungsdefinition, der erst im Betrieb des Produkts beim Endbenutzer entdeckt wird, kostet ca. 100mal mehr, als wenn dieser Fehler bei einem Anforderungsreview in der Anforderungsphase entdeckt wor-

den wäre. Die Strategie muss also sein, Fehler möglichst früh zu erkennen und zu beheben. Durch dieses Prinzip ist auch eine verbesserte Qualitätslenkung gegeben, und der Verbreitungseffekt von Fehlern (siehe Abbildung 1.6) wird ebenfalls reduziert.

P7) Integriertes Qualitätsmanagement
Qualitätsmanagement sollte in den gesamten Erstellungsprozess integriert sein. Das führt auch dazu, dass Qualitätsmaßnahmen organisatorisch mit den sonstigen Entwicklungsmaßnahmen geplant werden. Die Grundidee dabei ist, dass jede Entwicklungsaktivität aus einem konstruktiven und einem analysierenden/prüfenden Teil besteht. Beide Teile schlagen sich in entsprechender Dokumentation nieder. Beispielsweise werden Anforderungen durch eine geeignete Anforderungsdefinitionsmethode erfasst und strukturiert. Durch eine informelle oder formelle Anforderungsprüfung werden anschließend die Mängel in den Anforderungen beseitigt. Diese Prüfung wird auch dadurch erleichtert, dass durch den Einsatz einer Methode zur Anforderungsdefinition gleichzeitig Dokumentation entsteht (Anforderungsspezifikation). Durch dieses Prinzip wird auch das Qualitätsniveau zu jedem Zeitpunkt sichtbar. Insbesondere ist dies für die frühen Phasen wichtig. Es ist eine Voraussetzung für die realistische Beurteilung des Projektfortschritts und die Feststellung, ob die Qualitätsziele erreicht wurden.

P8) Unabhängigkeit bei Qualitätsprüfungen
Viele der Software-Entwicklungsmaßnahmen sind konstruktiver und produktiver Natur, d. h. auch, dass der Entwickler Urheber vieler Arbeitsprodukte ist. Bei Qualitätsprüfungen ist die Situation eine andere, da es hier darum geht, Fehler und Mängel aufzudecken. Ja, es wird vielerorts gefordert, dass Qualitätsprüfungen mit einer sehr kritischen, wenn nicht destruktiven Einstellung durchzuführen sind. Wenn nun ein Entwickler seine eigenen Arbeitsergebnisse prüfen muss, so führt dies in der Regel zu Spannungen und psychologischen Problemen. Dieser prinzipielle Interessenskonflikt lässt sich nur durch eine unabhängige Qualitätsprüfung beheben. Ziel dabei ist, dass nur die Ist-Qualität festgestellt wird und die Fehlerbehebung in einem nachfolgenden Schritt durchgeführt wird. Durch dieses wichtige Prinzip können die Folgen einer Betriebsblindheit oder mangelnder Objektivität behoben werden. Ein Nachteil besteht darin, dass wenn nicht das Produkt beurteilt wird, sondern die Person, dies zum Boykott weiterer Qualitätsprüfungen führt.

P9) Bewertung der eingesetzten Qualitätsmaßnahmen
In bestimmten Zeitabständen ist es notwendig, die eingesetzte Qualitätsmanagement-Organisation und deren Maßnahmen zu überprüfen. Wir sprechen dann von internen oder externen Qualitätsaudits. Externe Audits werden von externen Beratern durchgeführt, die das vorhandene (Qualitäts-)Managementsystem und die eingesetzten Qualitätsmaßnahmen überprüfen. Diese Audits führen in der Regel zu Korrekturen des Managementsystems und der eingesetzten

Qualitätsmaßnahmen. Dadurch wird auch die Rentabilität der in ein Managementsystem getätigten Investition gesteigert. Ein weiterer Grund, Qualitätsaudits durchzuführen, besteht darin, dass laufend neue Software-Entwicklungstechnologie auf den Markt kommt und in den Projekten eingeführt wird. Dies hat zur Folge, dass die Qualitätsmaßnahmen an diese neue Technologie angepasst werden müssen.

Um diese Untersuchungen durchzuführen, ist das Vorhandensein einer entsprechenden Qualitätsdatenerfassung und eines Berichtsystems notwendig (siehe Kapitel 6).

1.2.4 Begriffe und Definitionen

Wir wollen zunächst einige Begriffe im Zusammenhang mit Qualitätsmanagement definieren. Wesentlich ist zu erkennen, dass Qualität sowohl auf Produktebene als auch auf Prozessebene betrachtet werden muss. Unter Produkt werden auch Zwischen- und Endergebnisse von Phasen des Entwicklungsprozesses, wie beispielsweise Modulentwürfe, verstanden.

Qualitätsmanagement (QM)

Nach der Norm ISO 8402 vom März 1992 verstehen wir unter Qualitätsmanagement Folgendes:

Alle Tätigkeiten der Gesamtführungsaufgabe, welche die Qualitätspolitik, Ziele und Verantwortungen festlegen, sowie diese durch Mittel der Qualitätsplanung, Qualitätslenkung, Qualitätssicherung und Qualitätsverbesserung im Rahmen des Qualitätsmanagementsystems verwirklichen.

Qualitätsmanagement gehört in den Verantwortungsbereich aller Führungskräfte. Bei der Umsetzung des Qualitätsmanagements, die unter dem Gesichtspunkt der Wirtschaftlichkeit erfolgt, sind alle Mitarbeiter der Organisation involviert.

Seghezzi [Segh92] rundet die obige Definition praxisnahe ab, indem er dem Qualitätsmanagement vier Fachfunktionen und eine Führungsfunktion zuordnet (siehe Abbildung 1.9).

Von den vier Fachfunktionen hat die Qualitätsplanung die Aufgabe, Bedürfnisse zu ermitteln und diese in Form von Produkt- und Prozessmerkmalen umzusetzen. Im Rahmen der Qualitätslenkung (Prozessmanagement) sind sämtliche Prozesse einer Unternehmung so durchzuführen und zu beherrschen, dass die Spezifikationen eingehalten werden und fehlerfreie Produkte entstehen. Die Qualitätssicherung nimmt ausschließlich die Aufgabe wahr, Qualitätsrisiken zu ermitteln und Maßnahmen zu treffen, um sie zu vermindern oder zu eliminieren. Schließlich werden durch die Qualitätsförderung (Qualitätsverbesserung) Anstrengungen unternommen, die Qualität der Produkte, der Prozesse und somit des Unternehmens zu steigern, um damit die Wettbewerbsfähigkeit zu erhöhen.

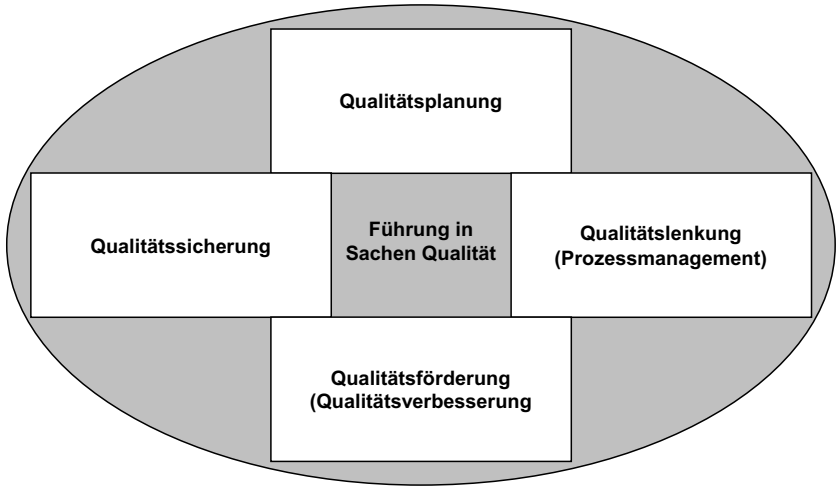

Abb. 1.9 Funktionen des Qualitätsmanagements

Diese vier Fachfunktionen lassen sich nicht einzelnen Organisationseinheiten zuordnen, sondern sind als Querschnittsaufgaben auf eine Vielzahl von Unternehmenseinheiten verteilt. Damit tritt auch eine beachtliche Schnittstellenproblematik auf. Aus diesem Grund definiert Seghezzi eine fünfte Funktion, die Führung in Sachen Qualität, die weit bedeutsamer geworden ist, als dies in der Vergangenheit der Fall war.

Qualitätsmanagement gibt uns Antworten auf folgende Fragen:

- Wie erreichen wir Kundenzufriedenheit?
- Wie finden wir einfach, rasch und kostengünstig die tatsächlichen Anforderungen an unsere Produkte/Dienstleistungen?
- Wie finden wir die erforderlichen Prozesse/Aktivitäten?
- Wie gestalten wir die Prozesse/Aktivitäten optimal?
- Wie finden wir die richtigen Ressourcen (Menschen, Wissen, Sachmittel)?
- Wie finden wir die richtigen Lieferanten – und wie steuern wir sie?
- Wie gehen wir mit Veränderungen um?
- Wie gestalten wir den kontinuierlichen Verbesserungsprozess?

Qualitätsmanagement wird nur dann akzeptiert, wenn es sich rentiert. Die Aufwendungen für Qualitätsmanagement richten sich nach den jeweiligen

- Fehlerkosten (eher bei Routineprozessen),
- Risiken (in Software-Projekten),
- Chancen (bei Innovationsvorhaben).

Qualitätsmanagement sorgt also für angemessene Aktivitäten, um das Optimum zu realisieren.

Qualitätspolitik

Darunter verstehen wir die grundlegenden Absichten und Zielsetzungen einer Organisation in Hinblick auf Qualität, wie sie von ihrer Leitung erklärt werden. Beispiele dafür sind: Kundenorientiertheit, schnelles Reagieren auf Marktbedingungen durch Einführung neuer Produkte, Erzeugung von Produkten mit hohem Wert, umfassendes und leistungsstarkes Kundenservice.

Qualitätspolitik ist eine zentrale Aufgabe des Top- und des mittleren Managements. Es besteht die Gefahr ist, dass sie als Alibifunktion wahrgenommen wird und im Tagesgeschäft untergeht.

Qualitätsmanagementsystem (QMS)

Darunter verstehen wir die Aufbau- und Ablauforganisation, die Zuständigkeiten und die Mittel für die Durchführung des Qualitätsmanagements (siehe dazu Kapitel 6).

Qualitätsmanagementsysteme bilden die Grundlage für alle Maßnahmen und Strategien zur Erreichung von Business Excellence (Total Quality). Es gibt verschiedene Ausprägungsstufen solcher Systeme. Die erste Stufe ist eine projektübergreifende und firmenspezifische Ausprägung. Die zweite Stufe ist eine projektspezifische Ausprägung, d. h. für jedes Projekt gibt es eine Qualitätsmanagementorganisation. Die dritte Stufe ist ein auf die jeweilige Phase bezogenes Qualitätsmanagement. Die Abbildungen 1.2 und 1.4 geben Beispiele dafür an.

Qualitätsplanung

Bei der Qualitätsplanung geht es darum, festzulegen, welche Anforderungen an den Prozess und das Produkt in welchem Umfang realisiert werden sollen. Ausgangspunkt dafür sind Kundennutzenanalysen, die methodisch über QFD (siehe Kapitel 7) realisiert werden. Es sind Produkt- und Prozessmerkmale auszuwählen, zu klassifizieren und zu gewichten. Die Quantifizierbarkeit der Merkmale spielt eine wichtige Rolle, da nur durch sie eine objektive Planerfüllung nachweisbar ist. Ist die Quantifizierung möglich, so sind die Zielwerte vorzugeben. Parallel dazu sind Mess- und Bewertungshilfsmittel (z. B. Qualitätskenngrößen) bereitzustellen, um die Prüfung der geplanten Qualität zu ermöglichen. Die Qualitätsplanung muss mit dem Auftraggeber bzw. den zukünftigen Benutzern eines Software-Systems abgesprochen werden.

Qualitätslenkung

Darunter verstehen wir die Steuerung, die Überwachung und die Korrektur der Realisierung einer Arbeitseinheit mit dem Ziel, die vorgegebenen Anforderungen zu erfüllen.

Im Mittelpunkt der Qualitätslenkung steht die Realisierung konstruktiver Maßnahmen wie z. B. die Anwendung von Software-Engineering-Methoden und -Techniken, aber auch deren Schulung. Die Aufgaben der Qualitätslenkung werden über die Prozesse des Projektmanagements realisiert.

Qualitätsprüfung/Qualitätsbeurteilung

Darunter verstehen wir das Feststellen, inwieweit eine Einheit (ein Prüfobjekt) die vorgegebenen Anforderungen erfüllt.

Wir unterscheiden zwischen statischen und dynamischen Prüfungen. Beispiele für statische Prüfungen sind Reviews (Inspektionen, Walkthroughs) und Audits. Zu den dynamischen Prüfungen gehören Tests und Zählungen von Prüfmerkmalen durch Werkzeuge (beispielsweise statische Analysatoren). Wir gehen im Kapitel 4 auf beide Kategorien von Prüfungen im Detail ein. Eine weitere Art von Qualitätsprüfungen sind Mängel- und Fehleranalysen, die auf Mängelkatalogen und Problemberichten beruhen. Sie geben Antworten auf folgende Fragen: In welcher Phase kommen welche Fehlertypen am häufigsten vor? Wie viele noch nicht behobene Fehler existieren für ein Produkt? Sie sind die Basis für weitere Verbesserungen des Entwicklungsprozesses.

Qualitätsplan

Der Qualitätsplan (Qualitätsmanagementplan, früher auch Qualitätssicherungsplan) ist das zentrale Hilfsmittel, mit dem die Produkt- und Prozessqualität geplant, gelenkt und kontrolliert wird. Er enthält alle bewusst gewählten Qualitätsmaßnahmen für ein Software-Projekt und sichert die Erreichung der Zielsetzung des Projekts. Somit ist er auch der schriftliche Nachweis der Qualitätslenkung. Wir gehen in Abschnitt 1.2.5 näher auf den Qualitätsplan ein.

1.2.5 Der Qualitätsplan

Ein Qualitätsplan (Q-Plan, QM-Plan, QS-Plan) wird immer für ein spezifisches Software-Projekt geschrieben und enthält die dafür geplanten qualitätsrelevanten Maßnahmen (Qualitätsmaßnahmen). Der Qualitätsplan kann und soll auf die anderen mitgeltenden Projektpläne und -dokumente Bezug nehmen (siehe Abbildung 1.10).

Ausgangspunkt für die Erstellung des Qualitätsplans sind die identifizierten Projekt- und Produktrisiken, der Projekttyp (z. B. Einführung von Standardsoftware, Echtzeitsoftware, Multimedia etc.) und die Komplexität der Aufgabenstellung. Der Qualitätsplan sollte nach Dunn [Dunn93] folgende Themen (Was, Wie, Wer) behandeln:

- Messungen
- Trendanalyse
- Werkzeugzertifizierung und -eignung
- Reviews
- Audits
- Testprozess
- Defektermittlung und -management
- Konfigurationsmanagement
- Zertifizierung von fertiggestellten Aufgaben
- Qualifikation von Ressourcen, Hilfsmitteln etc.

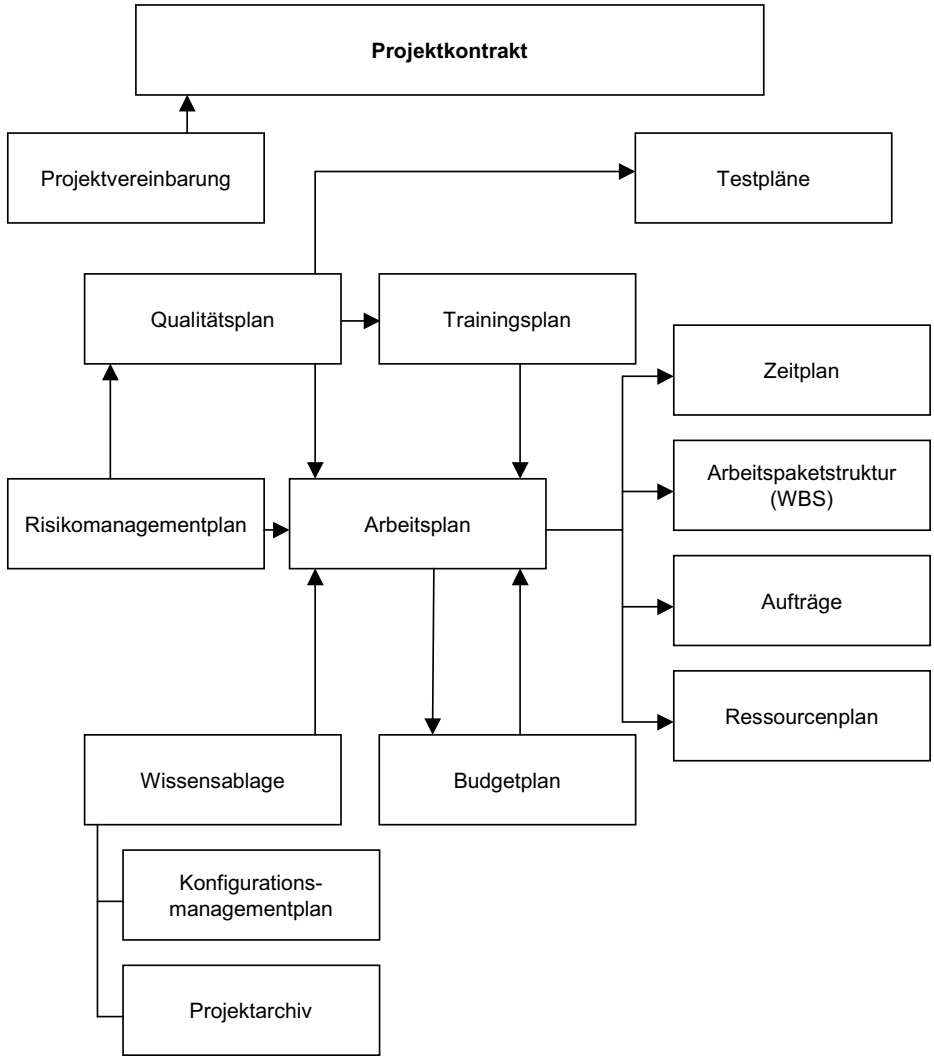

Abb. 1.10: Zusammenhang verschiedener Projektdokumente

- Dokumentation und deren Lenkung
- Produktanalysen
- Sicherstellen von Korrekturmaßnahmen
- Lieferantenmanagement, insbesondere deren Auswahl und Monitoring
- Sicherstellen von Reportingmaßnahmen, insbesondere von Qualitätsprotokollen

Wir stellen im Folgenden die Norm ANSI/IEEE Std 730-1989 vor, eine international akzeptierte Norm, die zur Erstellung von Qualitätsplänen dient:

QP1) Ziele der Qualitätsmaßnahmen für das spezielle Projekt

In diesem Abschnitt des Plans werden folgende Fragen näher erläutert: Welche Software-Produkte werden durch den Q-Plan abgedeckt? Wofür wird diese Software verwendet? Wie kritisch ist der Einsatz der Software? Warum wird ein Qualitätsplan geschrieben? Gibt es dafür externe oder interne Anforderungen? Was ist das Basisdokument für diesen Qualitätsplan (beispielsweise die IEEE-Norm, ein firmeninternes oder externes Dokumentenmuster)? Was sind die Gründe für eventuelle Abweichungen vom Basisdokument, und wo sind diese Abweichungen beschrieben?

QP2) Referenzierte Dokumente

Dieser Abschnitt enthält eine vollständige Liste aller Dokumente, die im Qualitätsplan erwähnt werden. Es ist auch anzugeben, von wo diese Dokumente zu beziehen sind und wer für sie verantwortlich ist.

QP3) Management

Hier werden die Organisation, die Aufgaben und die Verantwortlichkeiten der Projektführung beschrieben. Die Aufbauorganisation soll durch ein Organisationsstrukturdiagramm dargestellt und durch eine schriftliche Erläuterung ergänzt werden. Diese Erläuterung soll Folgendes enthalten:

- eine Beschreibung jedes Elements der Aufbauorganisation, das qualitätsrelevante Aufgaben durchführt;
- delegierbare Verantwortlichkeiten;
- Verantwortlichkeiten im Berichtswesen;
- Identifikation jener Elemente, die für die Produktfreigabe verantwortlich sind;
- Identifikation jener Elemente, die den Q-Plan prüfen;
- alle Verfahren, die zur Konfliktlösung zwischen den organisatorischen Elementen herangezogen werden;
- Größe und Umfang der Qualitätsorganisation;
- alle Abweichungen von der durch die Organisation festgelegten Qualitätspolitik, den Maßnahmen und den Standards für das Qualitätsmanagement.

Alle Elemente in dieser Aufbauorganisation sollten vollständig beschrieben werden, damit die Aufgaben, die im Q-Plan verzeichnet sind, direkt Elementen der Organisation zugeordnet werden können.

Die Aufgabenbeschreibung des Qualitätsmanagements, insbesondere die Reihenfolge der Aufgaben, soll den ganzen Software-Lifecycle umfassen. Dazu gehört unter anderem, wer den Q-Plan veröffentlicht, verteilt, wartet und durchsetzt. Jede Qualitätsmanagementaufgabe soll durch Auslöse- und Beendigungsbedingungen definiert sein.

Die Beschreibung der Verantwortlichkeiten enthält Hinweise, welche Qualitätsmanagementelemente für welche Qualitätsmanagementaufgaben verantwortlich sind.

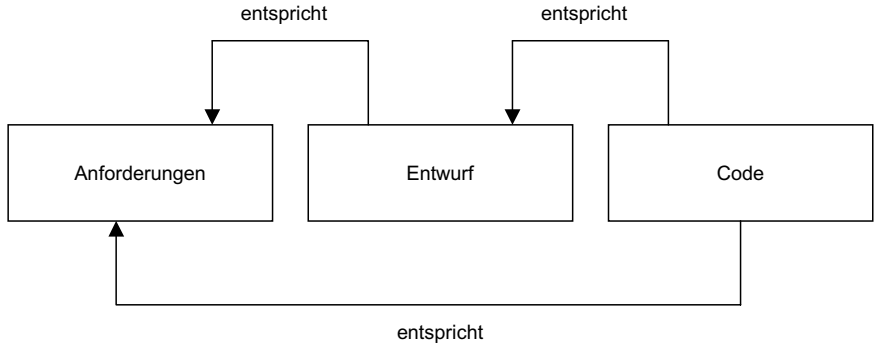

Abb. 1.11: Prüfziele des Prüfplans

QP4) Software-Dokumentation
Hier werden alle Dokumente spezifiziert, die für den Entwicklungs- und Wartungsprozess gefordert werden. Weiters werden alle Reviews und Audits angegeben, die die Angemessenheit und die Qualität der Dokumentation feststellen. Eine Minimalforderung an Dokumentation umfasst:

- Software-Anforderungsspezifikation,
- Software-Entwurfsbeschreibung,
- Software-Prüfplan,
- Software-Prüfbericht,
- Benutzerdokumentation.

Der Prüfplan enthält die Methoden und Hilfsmittel, mit denen der Nachweis erbracht wird, dass der Entwurf den Anforderungen und der Quellcode dem Entwurf und den Anforderungen entspricht (siehe dazu Abbildung 1.11).

Der Prüfbericht schildert das Ergebnis der Ausführung des Prüfplans. Insbesondere werden die Ergebnisse aller Reviews, Audits und Tests des Prüfplans beschrieben.

QP5) Normen, Verfahren und Konventionen
Hier werden alle Normen, Verfahren und Konventionen angeführt, die angewandt werden sollen. Ebenso sollen die Organisationselemente spezifiziert werden, die für die Durchsetzung, Bewertung und Pflege der Normen, Verfahren und Konventionen zuständig sind.

Ein Minimum an Normen, Verfahren und Konventionen soll vorhanden sein für

- die Anforderungsspezifikation,
- den Entwurf,
- die Implementierung (speziell Codierung und Kommentierung),
- das Testen und
- die Dokumentation.

QP6) Reviews und Audits

Hier wird angegeben, welche fachtechnischen oder managementorientierten Reviews und Audits wann durchgeführt werden. Wichtig ist auch anzuführen, wie deren Abschluss zu erfolgen hat.

Als Minimum sind folgende Prüfungen durchzuführen:

- Software-Anforderungsreview,
- Grobentwurfsreview,
- Feinentwurfsreview,
- Review des Prüfplans (Angemessenheit und Vollständigkeit der gewählten Prüfmethoden),
- Audit der Systemfunktionen (Code gegen Anforderungen prüfen),
- Auslieferungsaudit (Konsistenz der Software und der Dokumentation prüfen),
- Entwicklungsprozessaudits,
- Managementreview zur Bewertung der Ausführung des Q-Plans,
- Review der Benutzerdokumentation.

QP7) Software-Konfigurationsmanagement

Hier wird beschrieben, welche Methoden und Hilfsmittel zur Identifikation von Software-Produktelementen, zur Überwachung und Realisierung von Änderungen, sowie zur Aufzeichnung und Berichterstattung des Änderungszustands bei der Realisierung eingesetzt werden. Es kann auch auf einen eigenen Konfigurationsmanagementplan verwiesen werden.

QP8) Problemmeldewesen und Korrekturmaßnahmen

Hier wird beschrieben, welche Verfahren und Hilfsmittel für die Berichterstattung, Verfolgung und Lösung von Software-Problemen eingesetzt werden. Ebenso wird festgelegt, wer in der bestehenden Organisation für die Durchführung dieser Verfahren zuständig ist.

QP9) Software-Engineering-Konzept

Es wird beschrieben, welches Software-Engineering-Konzept, insbesondere welche konstruktiven Software-Engineering-Maßnahmen (Methoden, Hilfsmittel und Werkzeuge) eingesetzt werden und warum. Dazu ist es auch ratsam, Referenzen zum Methoden- und Verfahrenshandbuch anzugeben. Ebenso sind die dafür verantwortlichen Stellen zu bezeichnen.

QP10) Code-Kontrolle

Dieser Abschnitt muss die anzuwendenden Verfahren und Einrichtungen definieren, die zum Unterhalt und zur Speicherung kontrollierter Versionen der identifizierten Software dienen. Dies kann im Zusammenhang mit einer Programmbibliothek erfolgen.

QP11) Diverses

Unter diesem Punkt wird formuliert, was nicht unter QP1) bis QP10) fällt. Beispielsweise werden folgende Sachverhalte beschrieben:

- *Datenträger-Kontrolle*
  Dieser Abschnitt muss die anzuwendenden Verfahren und Einrichtungen festlegen, um die Programm-Datenträger vor unbefugtem Zugriff oder unbeabsichtigten Schäden oder Beeinträchtigungen zu schützen.
- *Lieferanten-Kontrolle*
  In diesem Abschnitt werden die Vorkehrungen festgehalten, die sicherstellen, dass die von externen Lieferanten entwickelte Software die festgelegten technischen Anforderungen erfüllt. In der Regel werden jene Qualitätsmaßnahmen angegeben, die der Lieferant von Fremdleistungen zu erfüllen hat.
- *Verwaltung, Pflege und Archivierung von Dokumenten*
  Es müssen Methoden und Einrichtungen angegeben werden, um die Dokumentation zusammenzustellen, zu schützen, zu pflegen und zu archivieren. Ebenso sind die Organisationseinheiten zu benennen, die für obige Maßnahmen verantwortlich sind.
- *Training*
  Hier werden die notwendigen Aus- und Fortbildungsmaßnahmen identifiziert und beschrieben.
- *Risikomanagement*
  Es müssen Methoden und Werkzeuge angegeben werden, um die Projekt-/Produktrisiken zu identifizieren, zu bewerten, zu vermeiden und zu monitoren.

Bei einer kritischen Durchsicht der IEEE-Norm fallen folgende Punkte auf: Es fehlt der Nachweis des Nutzens für den Auftraggeber sowie die Angabe von Qualitätskosten, die Qualitätsmaßnahmen unweigerlich nach sich ziehen. Trotz allem wurde und wird diese Norm international stark eingesetzt.

1.2.6 Klassifikation der Qualitätsmaßnahmen

Wir unterscheiden vier Kategorien von Qualitätsmaßnahmen:

- organisatorische,
- konstruktive,
- analytische und
- psychologisch-orientierte Maßnahmen.

Es ist sinnvoll zu unterscheiden, auf welcher Ebene diese Maßnahmen definiert werden. Folgende Ebenen lassen sich unterscheiden: projektübergreifend (global, unternehmensweit), projektbezogen und phasenbezogen.

QM1) Organisatorische Qualitätsmaßnahmen
Dabei geht es um den Aufbau, die Einführung und die Pflege eines (Qualitäts-)Managementsystems. Dieses System wird auf drei Ebenen wirksam, nämlich projektübergreifend, projektspezifisch und phasenspezifisch (siehe dazu Kapitel 6).

QM2) Konstruktive Qualitätsmaßnahmen
Unter konstruktiven Qualitätsmaßnahmen verstehen wir all jene, die der Qualitätsgestaltung dienen. Sie sind präventiv und sollen das Entstehen von Fehlern und Qualitätsmängeln von vornherein durch Vorgabe von geeigneten Prinzipien, Methoden, Techniken, Formalismen und Werkzeugen verhindern. Sie schließen auch alle Maßnahmen zur Fehlerbehebung ein. In Kapitel 3 wird auf konstruktive Qualitätsmaßnahmen näher eingegangen.

QM3) Analytische Qualitätsmaßnahmen
Unter analytischen Qualitätsmaßnahmen verstehen wir all jene Maßnahmen, die zur Erkennung und Lokalisierung von Mängeln und Fehlern im weitesten Sinne dienen, d. h. alle Maßnahmen, die der Bewertung der Qualität dienen. In Kapitel 4 gehen wir auf analytische Qualitätsmaßnahmen näher ein.

QM4) Psychologisch-orientierte Qualitätsmaßnahmen
Diese Kategorie von Maßnahmen betrifft den Menschen als Benutzer, Entwickler, Projektleiter oder Projektmanager. Es werden Maßnahmen unterschieden, die die Arbeit des Einzelnen bzw. die Teamarbeit betreffen. Beispielsweise führt der Autor bei Großprojekten in der Startphase des Projekts regelmäßig Teambildungsmaßnahmen durch. Dabei wird das Projektteam ein bis zwei Tage zu einer Veranstaltung möglichst in der Natur eingeladen, bei der sich die Projektteammitglieder kennenlernen und in einer spielerischen Art vertraut machen (z. B. Riverrafting, Abseilen bei einer Kletterwand, Floß bauen und damit einen See überqueren).

Bei Software-Projekten besteht immer die Gefahr, dass technische Aspekte überbetont werden. Untersuchungen haben gezeigt, dass die Fähigkeiten derer, die im Software-Entwicklungsprozess involviert sind, stark variieren. Beispielsweise wurden die Fähigkeiten des Erfassens und Analysierens von Anforderungen bzw. des Codierens näher untersucht. Dabei stellte man fest, dass es Leistungsunterschiede im Bereich 1 zu 5, bei der Codierung aber auch von 1 zu 30 gibt, d. h., dass bestimmte Entwickler bis zu 30mal produktiver sind als andere. Im Vergleich zu anderen technischen Disziplinen sind die Auswirkungen der verschiedenen Leistungsunterschiede im Entwicklungsprozess nicht sofort feststellbar. Wir müssen daher sorgfältig das Umfeld des Software-Entwicklungs- und Pflegeprozesses betrachten und mögliche Störquellen beseitigen.

Beispiele:

- individuelle Fähigkeiten und Erfahrungen der Entwickler nutzen;
- ein ganzes und identifizierbares Stück einer Aufgabe vollenden;
- Wichtigkeit der Arbeit betonen;
- die Möglichkeit, den Entwicklern Freiräume zu lassen;
- Erfolgserlebnisse einplanen;
- einen Führungsstil entwickeln, der sowohl leistungs- als auch mitarbeiterbezogen ist.

Die Berücksichtigung obiger Aspekte führt zu folgenden Ergebnissen: die Mitarbeiter sind stark motiviert, sie sind zufrieden mit ihrer Arbeit, eine erhöhte Qualität ihrer Arbeit ist festzustellen, und es gibt weniger Abwesenheit und Fluktuation.

Viele Informatikvorhaben sind nur mehr in Form von Teamarbeit zu bewältigen. Die Qualität der Ergebnisse, die durch Teamarbeit erstellt werden, wird durch eine Reihe von Größen, wie z. B. Unternehmenskultur, Kommunikations- und Führungsverhalten beeinflusst.

Voraussetzungen für eine funktionierende Teamarbeit sind:

- Jeder muss die Bedeutung seiner Arbeit kennen.
- Jeder hat das Bedürfnis, das Ergebnis seiner Arbeit darzustellen. Dies hat das Team zu ermöglichen.
- Jeder erwartet Belohnung (Feedback), die es zu vergeben gilt.
- Teamarbeit soll nur in Kleingruppen bis maximal fünf Personen durchgeführt werden.
- Ein Projekt ist zeitlich zu begrenzen (maximal 2 Kalenderjahre, ideal 9 bis 12 Kalendermonate).
- Die Arbeitsstätte soll so ausgestattet sein, dass sowohl informale als auch formale Kommunikation möglich ist.
- Es muss ein repressionsfreies Arbeitsklima geschaffen werden.

Das Gebiet der psychologisch-orientierten Qualitätsmaßnahmen ist im Rahmen der Informatik noch wenig untersucht worden. Informatiker sind in der Regel von der Bedeutung ihrer technischen Arbeit so überzeugt, dass für sie Experten anderer Fachgebiete, wie z. B. Arbeitspsychologen, als nicht geeignete Partner eingestuft werden und deren Disziplin als nicht wissenschaftlich fundiert bezeichnet wird. Dieses Faktum ist an vielen Orten heute leider anzutreffen und hindert den Fortschritt auf diesem Gebiet sehr (siehe dazu Abschnitt 3.7).

1.3 Zählen, Messen und Verbessern

Im Folgenden geben wir einen Überblick zum gegenwärtigen Stand des Software Engineering auf dem Gebiet des Messens und Verbesserns von Software-Produkten und -Prozessen. In diesem Abschnitt soll die Bedeutung des Zählens, des Messens und des Bewertens näher untersucht werden.

1.3.1 Bedeutung des Messens

In jeder Disziplin, die ingenieurmäßig vorgeht, spielen die Begriffe Zählen und Messen eine wichtige Rolle. Um die Bedeutung und Problematik des Messens in den Naturwissenschaften und der Technik zum Ausdruck zu bringen, lassen wir einige prominente Vertreter aus diesen Bereichen zu Wort kommen: