9. Software-Qualitätsmanagement

9.1 Grundlagen

Werden Produkte in kleinem, überschaubarem Rahmen entwickelt, hergestellt und vertrieben, so erübrigen sich spezielle Maßnahmen zur Sicherstellung der Qualität. Pfuscharbeit hat sofortige Rückwirkungen auf deren Verursacher, einerseits weil Hersteller und Abnehmer einander kennen, andererseits weil es im Kleinbetrieb eine wirksame Kontrolle und häufig ein traditionelles, qualitätsbewusstes Berufsethos gibt. Dies ist z.B. typisch der Fall in kleinen handwerklichen Meisterbetrieben. Qualitätsmanagement im eigentlichen Sinn gab es historisch erst mit der industriellen Massenproduktion, wo die Qualität in der Produktion zunächst durch Kontrolleure und Prüfer, später dann vor allem durch statistische Verfahren sichergestellt wurde.

Qualitätsmanagement in der *Entwicklung* von Produkten ist nochmals wesentlich jünger, da Massenphänomene (Probleme, die sich auch in der Entwicklung nur noch durch Arbeitsteilung und Gruppenarbeit lösen lassen, zunehmende Anonymisierung, Entfremdung von Entwicklern und Produkten) dort erst seit etwa der Mitte des 20. Jahrhunderts aufgetreten sind.

In der zweiten Hälfte des 20. Jahrhunderts werden auch die Prinzipien des heutigen modernen Qualitätsmanagements, insbesondere die Selbstverantwortung der Produzierenden für Qualität und das prozessorientierte Qualitätsmanagement entwickelt. Sie verbreiten sich zuerst in Japan und zwischen 1980 und 1990 auch in den übrigen Industrieländern.

Software-Qualitätsmanagement ist der jüngste Spross, bei dem es speziell um die Qualität von industriell entwickelter Software geht.

9.1.1 Definitionen

Im allgemeinen Sprachgebrauch wird Qualität meistens mit einem hohen Wert auf einer *absoluten* Güteskala assoziiert. Im Gegensatz dazu bezeichnet der industrielle Qualitätsbegriff eine *relative* Güte, die sich an der *Erfüllung zuvor aufgestellter Ziele* bemisst.

Definition 9.1. *Qualität (quality).* Der Grad, in dem ein Satz *inhärenter Merkmale Anforderungen* erfüllt. (ISO 9000:2000).

Inhärente Merkmale (inherent characteristics) sind *kennzeichnende Eigenschaften* einer Einheit (Produkt, Dienstleistung, Prozess, System, Person, Organisation, etc.), welche diese aus sich selbst heraus hat und die ihr nicht explizit zugeordnet sind. Beispiel: Das Material oder die Haltbarkeit eines Kleidungsstücks sind Merkmale, welche dieses aus sich selbst heraus hat, während der Verkaufspreis ein explizit zugeordnetes Merkmal ist. Explizit zugeordnete Merkmale einer Einheit sind daran zu erkennen, dass sie geändert werden können, ohne die Einheit selbst damit zu verändern.

Unter *Anforderung (requirement)* wird hier ein Erfordernis oder eine Erwartung verstanden, das oder die *festgelegt, üblicherweise vorausgesetzt* oder *verpflichtend* ist. Qualität im Sinn von ISO 9000 ist folglich *Zielerfüllung*. Die Ziele (sprich Anforderungen) können *explizit* festgelegt oder *implizit* durch gemeinsame Vorstellungen der Beteiligten gegeben sein.

Die Gesamtheit der qualitätsrelevanten Tätigkeiten in einer Organisation wird als Qualitäts*management* bezeichnet.

Definition 9.2. *Qualitätsmanagement (quality management).* Aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bezüglich Qualität.
Leiten und Lenken bezüglich Qualität umfassen üblicherweise das Festlegen der *Qualitätspolitik* und der *Qualitätsziele*, die *Qualitätsplanung*, die *Qualitätslenkung*, die *Qualitätssicherung* und die *Qualitätsverbesserung* . (ISO 9000:2000).

Hinweis: Bis 1993 Jahren wurde das gesamte Qualitätsmanagement in den Qualitätsnormen als *Qualitätssicherung (quality assurance)* bezeichnet. Seither ist dieser Begriff in den Normen auf die Qualitätsmanagement-Darlegung, d.h. den Nachweis der Qualitätsmaßnahmen beschränkt worden. In der Praxis und auch in Lehrbüchern wird aber immer noch häufig von Qualitätssicherung gesprochen, wenn Qualitätsmanagement gemeint ist.

Qualitätsmanagement braucht einen organisatorischen Rahmen. Dieser wird durch das Qualitätsmanagement*system* gebildet.

Definition 9.3. *Qualitätsmanagementsystem (quality management system).* Managementsystem zum Leiten und Lenken einer Organisation bezüglich der Qualität. (ISO 9000:2000).

9.1.2 Besonderheiten des Software-Qualitätsmanagements

Software nimmt insofern eine Sonderstellung ein, als
- Software ein Produkt ist, bei dem nur die Entwicklung schwierig ist, die Produktion dagegen aus bloßem, völlig unproblematischem Kopieren besteht
- es praktische keine Traditionen für Software-Entwicklung gibt
- Software aufgrund ihres immateriellen Charakters schwierig zu überprüfen ist.

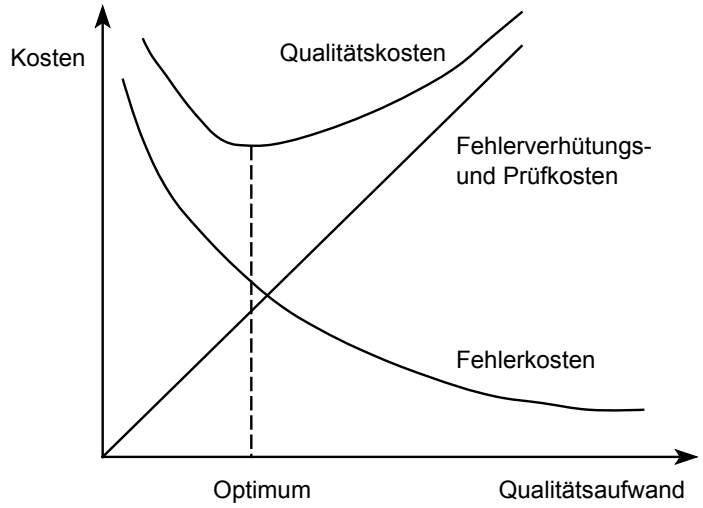
Bezogen auf Qualität heißt dies, dass die Qualität der Entwicklung, d.h. der Produkt*schöpfung* gesichert werden muss, dass auf keine Traditionen und kein Berufsethos für die Herstellung qualitativ hochwertiger Software zurückgegriffen werden kann und dass die Qualitätsprüfung von Software schwierig ist. Vor allem das zweite sorgt für Schwierigkeiten bei der Einführung eines Qualitätsmanagementsystems und bei der notwendigen Verankerung des Qualitätsbewusstseins in den Köpfen der Beteiligten.

Grundsätzlich gibt es aber keinen grundlegenden Unterschied zwischen Software-Qualitätsmanagement und dem Qualitätsmanagement bei der Entwicklung anderer Produkte.

9.1.3 Qualitätskosten

Qualität ist nicht umsonst. Einrichtung und Unterhalt eines Qualitätsmanagementsystems kosten erhebliche Summen. Die Durchführung von Qualitätsmanagement-Maßnahmen in den Projekten belastet diese sowohl terminlich wie kostenmäßig.

Welcher Nutzen steht diesen Kosten gegenüber? Qualitätsmanagement bedeutet Verhinderung bzw. rechtzeitige Entdeckung von Fehlern. Jeder nicht entdeckte oder zu spät entdeckte Fehler ist mit teilweise erheblichen Fehlerbeseitigungskosten und möglichen Folgekosten (z.B. Verlust von Marktanteilen) verbunden. Die Wirtschaftlichkeit des Qualitätsmanagements ergibt sich also aus der Gegenüberstellung von Qualitätsmanagementkosten einerseits und Fehlerkosten andererseits (Bild 9.1).



nach Frühauf, Ludewig, Sandmayr (1988)

BILD 9.1. Qualitätskosten

9.1.4 Grundsätze des Qualitätsmanagements

Im Qualitätsmanagement gelten die folgenden sechs Grundsätze.

Grundsatz 1. Qualität – in der Entwicklung wie in der Produktion – muss *erzeugt* werden, sie kann *nicht erprüft* werden.

Grundsatz 2. Qualität bezieht sich immer sowohl auf die hergestellten Produkte wie auf die Prozesse zur Herstellung (Entwicklung und Produktion) dieser Produkte.

Schlampige Herstellungsprozesse sind eine denkbar schlechte Voraussetzung für die Herstellung guter Produkte. Wer qualitativ hochwertige Produkte will (das, was letztlich zählt), muss daher auch für qualitativ einwandfreie Herstellungsprozesse sorgen.

Grundsatz 3. Die *Qualitätsverantwortung* liegt immer bei den gleichen Leuten, welche auch die Sach-, Termin- und Kostenverantwortung haben, d.h. bei den *Führungskräften* (in der Linie wie in Projektorganisationen) und bei den *Entwicklern*.

Grundsatz 4. Das *Qualitätswesen* (in dem die Qualitätsfachleute zusammengefasst sind, siehe unten) ist verantwortlich für die *Ermittlung (Messung) der Qualität*. Es erbringt Dienstleistungen in allen Belangen der Qualität sowohl für die Entwickler wie für die Führungsverantwortlichen.

Das Qualitätswesen erarbeitet die Kriterien zur Qualitätsmessung, führt die Messungen durch oder ist dabei behilflich. Es erarbeitet für die Führungsverantwortlichen Empfehlungen zu anstehenden Entscheidungen (z.B. Verkaufsfreigabe eines Produkts). Es informiert die Entwickler über die Qualität ihrer Arbeit.

Grundsatz 5. Das Qualitätswesen muss einen unabhängigen Berichterstattungspfad haben, der bis zur Geschäftsleitung geht.

Die Führungsverantwortlichen können Empfehlungen des Qualitätswesens ignorieren, da sie gemäß Grundsatz 3 nicht nur die Termin-, Kosten- und Sachverantwortung, sondern auch die Qualitätsverantwortung haben. Da qualitativ schlechte Produkte jedoch den Ruf eines ganzen Unternehmens ruinieren können, muss das Qualitätswesen die Möglichkeit haben, auf einem ordentlichen Berichtspfad die Geschäftsleitung über Pfuscharbeit zu orientieren.

Grundsatz 6. Die Mitarbeiter müssen über die Qualität ihrer Arbeit orientiert werden. Jeder Vorgesetzte muss die Qualität der Arbeit seiner Mitarbeiter in deren Beurteilung mit einbeziehen.

Mitarbeiter sind nur dann motiviert, Qualitätsarbeit zu leisten, wenn sie wissen und spüren, dass Qualität belohnt und Pfusch bestraft wird.

Will man erreichen, dass Qualitätsmanagement etwas Aufbauendes, ein positiver Teil der Unternehmenskultur ist und nicht etwas Negatives, Destruktives, so ist darauf zu achten, dass nicht nur Schlechtes kritisiert, sondern ebenso auch Gutes gelobt wird.

9.2 Das Qualitätsmanagementsystem

9.2.1 Prinzipien eines modernen Qualitätsmanagementsystems

Heutige Qualitätsmanagementsysteme orientieren sich an der Selbstverantwortung aller Beteiligten sowie an der Kundenzufriedenheit. Dieser Ansatz wurde hauptsächlich von Deming (1986) entwickelt und propagiert. Zur Realisierung wird heute meist ein prozessorientierter, systemischer Ansatz verwendet (Bild 9.2).

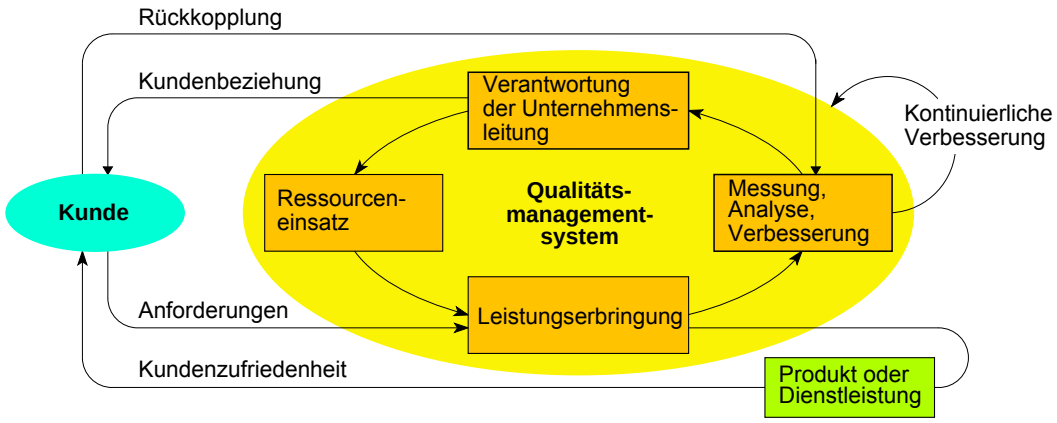


BILD 9.2. Qualitätsmanagementsystem nach ISO 9000:2000

9.2.2 Die Aufbau- und Ablauforganisation

In einem qualitätsbewussten Unternehmen gibt es innerhalb der Unternehmensorganisation eine Sekundärorganisation für Qualität, das so genannte Qualitätswesen, in der die Fachleute für Qualität zusammengefasst sind. An der Spitze der Qualitätsorganisation steht ein Qualitätsleiter, welcher unmittelbar an die Geschäftsleitung berichtet (Bild 9.3). Die Qualitätsbeauftragten in den Stäben sowie die Qualitätsingenieure in den Abteilungen arbeiten je nach Unternehmensgröße haupt- oder nebenamtlich in der Qualitätsorganisation. Im Qualitätswesen ist das *Fachwissen* über Qualität, insbesondere über Qualitätsmanagementverfahren, konzentriert. Die Qualitäts*verantwortung* liegt jedoch nicht allein beim Qualitätswesen, sondern muss vom ganzen Unternehmen getragen werden.

Innerhalb der *Ablauforganisation* muss das Qualitätsmanagementsystem für alle Abläufe die qualitätsrelevanten Kompetenzen, Verantwortlichkeiten und gegenseitigen Beziehungen festlegen. Man ist dabei bestrebt, möglichst wenig Qualitätsaufgaben separat zu regeln, sondern diese weitestgehend in die Prozesse des Unternehmens zu integrieren. Bild 9.3 zeigt eine typische prozessorientierte Ablauforganisation für ein Softwareunternehmen.

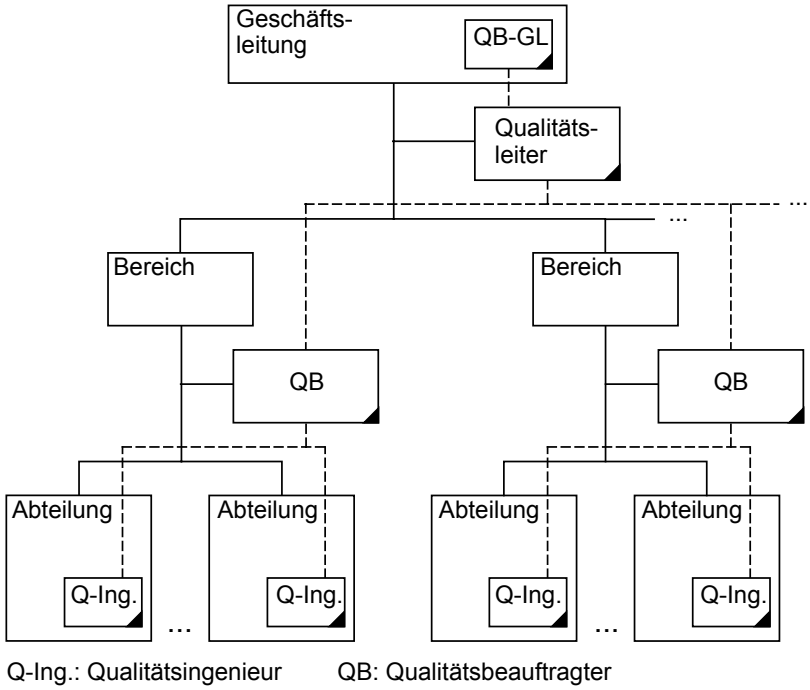


BILD 9.3. Das Qualitätswesen im Unternehmen

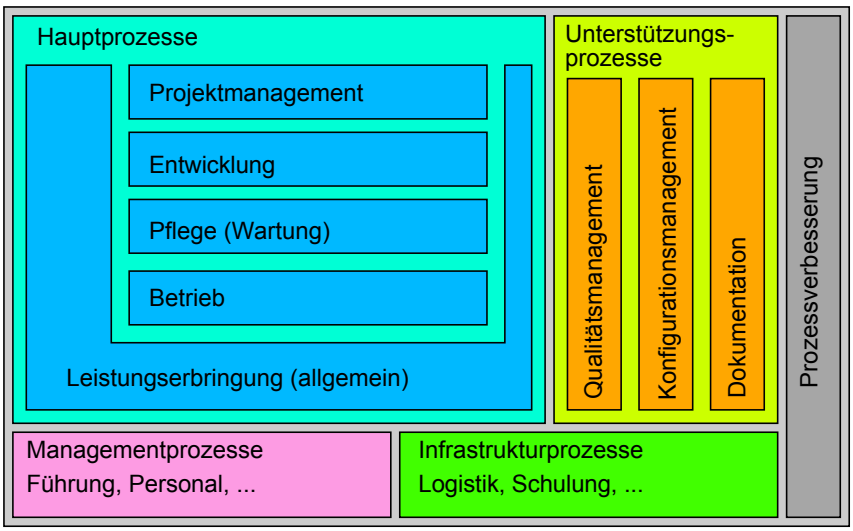


BILD 9.4. Prozessorientierte Ablauforganisation in einem Softwareunternehmen

9.2.3 QM-Verfahren

Die Qualitätsmanagementverfahren kann man in drei Klassen gliedern: Qualitätsplanung, Qualitätslenkung und Qualitätsverbesserung (Bild 9.5).

Zur **Qualitätsplanung** (*quality planning*) im Allgemeinen gehört der Aufbau des Qualitätsmanagementsystems, seine Dokumentation im Qualitätshandbuch und seine Ausprägung im Qualitätswesen. Ferner gehört dazu eine Festlegung von Qualitätsmerkmalen, die bei jeder Produktentwicklung gemessen werden sollen.

Qualitätsplanung im Speziellen ist die Festlegung der Ziele für ein Projekt bzw. Produkt (vgl. Kapitel 2).

Qualitätslenkung (*quality control*) gliedert sich in *konstruktive* und *analytische* Maßnahmen. Letztere werden auch *Qualitätsprüfung* genannt.

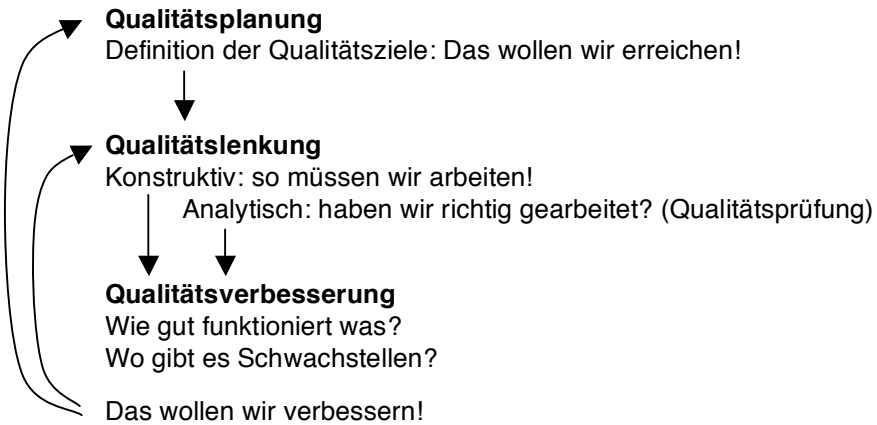


BILD 9.5. Qualitätsmanagement-Maßnahmen

Konstruktive Qualitätslenkung im Allgemeinen findet statt durch Festlegung von Entwicklungs*methoden*, welche die Vorgehensweise definieren, durch *Sprachen*, welche die Ausdrucksweise normieren, und durch *Werkzeuge*, welche den Einsatz von Methoden und Sprachen unterstützen. Zu den Methoden gehören auch *Entwicklungsrichtlinien*.

Maßnahmen zur Ausbildung und zur Vereinheitlichung der Arbeitsweise der Entwickler sind ebenfalls konstruktive Qualitätslenkungs-Maßnahmen.

Konstruktive Qualitätslenkung im Speziellen sind alle Maßnahmen, die in der Führung eines Projekts ergriffen werden, um die Erreichung der gesetzten Ziele sicherzustellen.

Analytische Qualitätslenkung oder **Qualitätsprüfung** wird in Abschnitt 9.3 besprochen.

Qualitätsverbesserung (*quality improvement*) umfasst einerseits die Behebung der bei der Produktprüfung gefundenen Qualitätsmängel. Dies ist notwendig zur Erreichung der geplanten Produktqualität, ist aber häufig nur Symptombekämpfung, nämlich dann, wenn die eigentlichen Fehlerursachen im Entwicklungsprozess oder im Qualitätsmanagementsystem liegen. Qualitätsverbesserung umfasst daher auch die Aufgabe, Modifikationen im Entwicklungsprozess und im Qualitätsmanagementsystem vorzunehmen. Dies geschieht typisch auf der Grundlage von Auswertung von Fehlerursachen, Resultaten von Audits sowie von Messungen.

9.2.4 Dokumentation des Qualitätsmanagementsystems

Das Qualitätsmanagementsystem wird wie folgt dokumentiert:

- Das *Qualitätshandbuch* enthält die Beschreibung des Qualitätsmanagementsystems eines Unternehmens oder Unternehmensbereichs.
  Falls das Unternehmen Kunden hat, die über das Qualitätsmanagement orientiert werden wollen, bevor sie Aufträge erteilen, wird das Qualitätshandbuch in zwei Bände unterteilt. Der erste Band wird auf Anfrage an Kunden abgegeben. Er enthält die Qualitätsmanagement-Organisation und einen Überblick über die Qualitätsmanagement-Maßnahmen.
  Der zweite Band ist das Handbuch der Qualitätsmanagement-Verfahren. Er beschreibt das unternehmensspezifische Qualitätsmanagement-Know-How und ist vertraulich.
- Der *Software-Qualitätsplan* enthält die Vorgaben betreffend Qualität für die Entwicklung von Produkten. In der Regel gibt es einen allgemeinen Rahmenplan, der bei Bedarf projektspezifisch ergänzt wird.
- Detaillierte Entwicklungsrichtlinien und Ausführungsbestimmungen, die den Rahmen eines Qualitätsplans sprengen würden, werden häufig in einem *Software-Entwicklungshandbuch* oder in *Software-Entwicklungsrichtlinien* (Berner et al. 1997) zusammengefasst.

9.2.5 Qualitätssicherung

Die Existenz eines Qualitätsmanagementsystems garantiert noch nicht dessen Wirksamkeit. Die *Darlegung des Qualitätsmanagements*, d.h. alle Tätigkeiten zur Schaffung von Vertrauen, dass die Qualitätsanforderungen erfüllt werden, wird Qualitätssicherung[1)] genannt. Als zentrale Maßnahmen braucht es regelmäßige Überprüfungen, ob das Qualitätsmanagementsystem wie geplant funktioniert und ob die vorgesehenen Qualitätsmaßnahmen wirklich durchgeführt werden. Solche Überprüfungen heißen *Audits*.

Interne Audits werden durch das Qualitätswesen geplant und durchgeführt. Jede Organisationseinheit sollte etwa einmal jährlich auditiert werden. Die Ergebnisse dieser Audits fließen in Maßnahmenpläne zur Verbesserung der Qualität und zur Eliminierung erkannter Schwachstellen ein.

Bei *externen* Audits wird die Überprüfung durch eine externe, auf Qualitätsmanagement spezialisierte Organisation durchgeführt. Ein externer Audit ist unter anderem dann erforderlich, wenn eine Organisation ein *Zertifikat* für ihr Qualitätsmanagementsystem braucht oder will. Dies ist beispielsweise der Fall, wenn ein großer Kunde einen Auftrag davon abhängig macht, dass das Qualitätsmanagementsystem des Auftragnehmers gewissen Mindestanforderungen genügt oder wenn sich die Organisation von der Zertifizierung einen Wettbewerbsvorteil erhofft. Die Zertifizierung erfolgt nach den Vorgaben der Norm ISO 9000.

9.3 Qualitätsprüfung

9.3.1 Allgemeines

Die Qualitätsprüfung oder analytische Qualitätslenkung (vgl. Bild 9.5) stellt fest, ob ein Arbeitsergebnis den Vorgaben entspricht. Letztlich ist zu prüfen, ob ein fertig gestelltes Produkt die an das Produkt gestellten Anforderungen erfüllt. Mit der Prüfung darf jedoch nicht bis zum Abschluss der Entwicklung gewartet werden, weil verfehlte Ziele dann möglicherweise nicht mehr oder nur noch mit hohen Kosten erreichbar sind. *Qualitätsprüfung ist daher eine permanente, die Entwicklung begleitende Aufgabe.* Man unterscheidet zwei Arten der Überprüfung, die Validierung und die Verifikation (Bild 9.6).

Definition 9.4. *Validierung (validation).* Der Prozess der Beurteilung eines Systems oder einer Komponente während oder am Ende des Entwicklungsprozesses, mit dem Ziel, festzustellen, ob die spezifizierten Anforderungen erfüllt sind. (IEEE 610.12).

Validierung ist folglich die Prüfung, ob ein Arbeitsergebnis geeignet ist, die Erwartungen der Endbenutzer zu erfüllen, d.h. ob überhaupt das *richtige Produkt* entwickelt wird.

Definition 9.5. *Verifikation (verification).* 1. Der Prozess der Beurteilung eines Systems oder einer Komponente mit dem Ziel, festzustellen, ob die Resultate einer gegebenen Entwicklungsphase den Vorgaben für diese Phase entsprechen. 2. der formale Beweis der Korrektheit eines Programms. (IEEE 610.12).

Verifikation ist demnach Prüfung, ob ein Arbeitsergebnis den für den betreffenden Arbeitsschritt gegebenen Vorgaben genügt, d.h. ob das Produkt *richtig* entwickelt wird. Der Beweis der Korrektheit von Programmen ist ein Verifikationsverfahren auf der Grundlage formaler mathematischer Mittel.

---

1) Bis 1993 Jahren wurde das gesamte Qualitätsmanagement in den Qualitätsnormen als *Qualitätssicherung (quality assurance)* bezeichnet. Seither ist dieser Begriff auf die hier beschriebene Bedeutung beschränkt. In der Praxis und auch in Lehrbüchern wird aber immer noch häufig von Qualitätssicherung gesprochen, wenn Qualitätsmanagement gemeint ist.

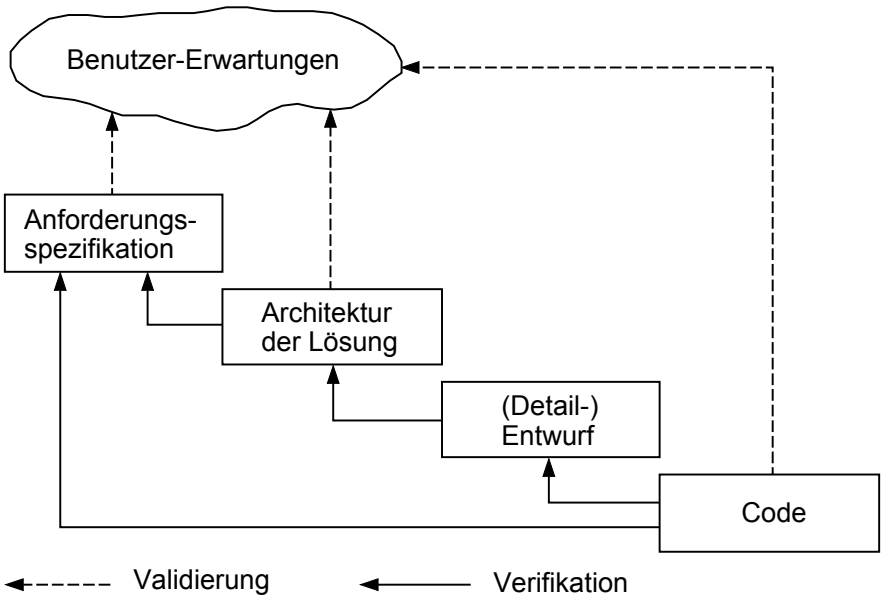


BILD 9.6. Validierung und Verifikation

Zur Validierung können grundsätzlich keine mathematisch formalen Verfahren verwendet werden, da die Benutzererwartungen nicht formalisierbar sind.

Bei der Verifikation der Zwischenprodukte einer Entwicklung dagegen ist es grundsätzlich möglich, mathematisch formale Verfahren zu verwenden, welche die Korrektheit der Entwicklung *beweisen.* Der dafür erforderliche Aufwand ist jedoch so groß, dass die formale Verifikation in der Praxis nur eine Nischenrolle spielt (siehe nächster Abschnitt).

Die meisten Prüfverfahren, die heute in der Praxis eingesetzt werden bzw. eingesetzt werden sollten, sind daher zwar *formalisiert* (d.h. folgen einem formellen Verfahrensprozedere), aber nicht im mathematischen Sinn formal.

9.3.2 Prüfverfahren

In der Praxis werden heute im Wesentlichen fünf Prüfverfahren verwendet: das Review, die statische Analyse, der Test, die Simulation und der Einsatz von Prototypen.

Ein *Review* ist eine formell organisierte Überprüfung eines Arbeitsergebnisses durch eine Gruppe von Gutachtern. Trotz nachgewiesener großer Wirksamkeit werden Reviews heute immer noch viel zu wenig eingesetzt. In Kapitel 9.4 ist daher die Reviewtechnik genauer erläutert.

Jede automatisierte Untersuchung des statischen Aufbaus eines Prüflings auf die Erfüllung vorgegebener Kriterien bezeichnet man als *statische Analyse*. Dazu gehören beispielsweise die automatisierte Überprüfung der Einhaltung von Codierrichtlinien, die Erkennung deklarierter, aber nicht benutzter Daten oder die Erkennung von Zyklen in der Benutzungshierarchie von Modulen.

Ein *Test* ist die Überprüfung eines Programms, ob dieses bei gegebenen Eingaben die erwarteten Resultate liefert. Nach dem heutigen Stand der Technik ist qualitativ hochwertige Software nur mit systematischem Testen (vgl. Kapitel 8.3) erstellbar.

In einer *Simulation* wird ein bestimmter Aspekt eines Systems modelliert und in seinem Verhalten überprüft. Simulationen dienen vor allem dazu, die Erreichung von Leistungsanforderungen zu überprüfen.

Ein *Prototyp* ist eine Vorab-Realisierung eines kritischen Systemteils. Er dient vor allem dazu, Anforderungen zu validieren und Konzepte auf Machbarkeit bzw. Einhaltung der Leistungsanforderungen zu überprüfen (vgl. Kapitel 3 und 4).

Für Spezialanwendungen, zum Beispiel im Bereich sicherheitskritischer Systeme, gibt es weitere Prüfverfahren. Hierzu gehören der *Beweis der Korrektheit* von Programmen mit formalen mathematischen Methoden (formale Verifikation) sowie die Überprüfung von Programmen oder formalen Software-Modellen im Hinblick auf die Gültigkeit wichtiger Eigenschaften (beispielsweise bei einer Liftsteuerung, dass der Lift nie mit offenen Türen fährt). Letzteres wird in der Fachliteratur als *Model Checking* bezeichnet. Alle formalen Prüfverfahren sind sehr aufwendig und erfordern tiefgehende Spezialkenntnisse. Sie werden daher in der Praxis nur dort verwendet, wo die Risiken sehr hoch sind und nicht mit anderen Mitteln ausreichend beherrscht werden können.

9.4 Reviewtechnik

9.4.1 Grundlagen

9.4.1.1 Definition, Ziele

Definition 9.6. *Review*. Eine formell organisierte Zusammenkunft von Personen zur inhaltlichen oder formellen Überprüfung eines Produktteils (Dokument, Programmstück, etc.) nach vorgegebenen Prüfkriterien und -listen.

Ein Review hat zum Ziel, sowohl die Schwachstellen und Mängel wie auch die Stärken des geprüften Produkts aufzuzeigen und damit dessen Qualität festzustellen. Man unterscheidet zwei Formen von Reviews: Inspektionen und Walkthroughs.

Eine *Inspektion* ist ein Review, bei dem der Prüfling von den Gutachtern systematisch Punkt für Punkt auf Stärken und Schwächen abgeklopft wird.

Ein *Walkthrough* ist ein Review, bei dem der Autor oder die Autorin die Funktionsweise des Prüflings Schritt für Schritt beschreibt, während die Gutachter aufmerksam zuhören und überall einhaken, wo sie Mängel entdecken.

In der Regel werden Reviews als Inspektionen durchgeführt. Dementsprechend gelten die Ausführungen in den folgenden Kapiteln primär für Inspektionen. Sie lassen sich sinngemäß aber auch auf Walkthroughs anwenden. Walkthroughs sind dann sinnvoll, wenn das Urteil von Gutachtern gefragt ist, welche das zu prüfende Produkt (oder die Sprache, in der es formuliert ist) nicht so gut kennen, dass sie es ohne Führung inspizieren können.

Neben dem zu prüfenden Material braucht es für ein Review so genannte *Referenzunterlagen*. In einem Code-Review kann beispielsweise die inhaltliche Richtigkeit von Algorithmen und Datenstrukturen nur geprüft werden, wenn als Referenz ein Konzept- oder Entwurfsdokument vorhanden ist, in welchem entsprechende Vorgaben stehen.

Unter Referenzunterlagen versteht man einerseits Unterlagen, welche sachlich-inhaltliche Vorgaben enthalten, die der Prüfling erfüllen muss und andererseits Standards und Prüflisten, anhand derer die Prüfung durchgeführt werden soll.

Reviews können in jedem Stadium der Software-Entwicklung eingesetzt werden. In den frühen Phasen der Entwicklung ist das Review das einzige Mittel für eine fundierte Überprüfung von Resultaten. In der Codierung und Integration werden Reviews als zusätzliches Prüfmittel neben den Tests eingesetzt.

9.4.1.2 Motivation

Sorgfältig durchgeführte Reviews verbessern die Qualität der erzeugten Software erheblich. Fagan (1986) berichtet von zwei Projekten, bei denen in den Code-Reviews 82% bzw. 93% aller Fehler entdeckt wurden, und dies gemessen über die gesamte Lebensdauer der jeweiligen Software! Andere Quellen sprechen von Senkungen der Restfehlerraten (Fehler pro 1000 Zeilen ausgeliefertem Code) um bis zu zwei Drittel.

Solche Qualitätsverbesserungen schlagen nicht nur in höherer Kundenzufriedenheit zu Buche, sondern reduzieren sowohl die Entwicklungskosten (weniger Aufwand für Test und Fehlerbehebung) als auch die Pflegekosten.

9.4.1.3 Positive Nebeneffekte

Werden in einer Organisation regelmäßig Reviews durchgeführt, so hat dies einen ständigen *Wissens- und Kenntnistransfer* von den besseren zu den schlechteren Leuten zur Folge. Jeder Review-Teilnehmer lernt aus den Stärken und Schwächen in den Produkten anderer etwas für die eigene Arbeit. Auf die gleiche Weise wird eine einheitliche Anwendung von Standards und Konventionen innerhalb der Organisation gefördert.

Die in Reviews festgestellten Mängel lassen sich statistisch auswerten. Die Ergebnisse können zur *Qualitätslenkung* verwendet werden. Tritt beispielsweise eine bestimmte Art von Fehlern besonders häufig auf, so können gezielt Maßnahmen ergriffen werden, damit diese Fehler nicht mehr gemacht werden.

9.4.2 Die Durchführung von Reviews

Effektivität und Erfolg von Reviews hängen wesentlich von der richtigen Durchführung ab. Dazu gehört nicht nur die Abwicklung der eigentlichen Review-Sitzung nach besonderen Regeln, sondern ein insgesamt wohlorganisiertes Umfeld.

9.4.2.1 Der Ablauf eines formellen Review

Zu einem formellen Review gehören die nachfolgend beschriebenen fünf Schritte.

1. Planung

Reviews müssen in der Entwicklung eingeplant werden. Der notwendige Aufwand an Zeit und Arbeitskapazität ist zu budgetieren. Die Review-Teilnehmer müssen rechtzeitig eingeladen und mit dem zu prüfenden Material versorgt werden. Je nach Qualitätsanforderungen können die Aufwendungen für Reviews bis zu 15% des gesamten Entwicklungsaufwandes ausmachen.

2. Vorbereitung

Die Teilnehmer an einem Review erhalten das zu prüfende Material sowie die Referenzunterlagen im Voraus und bereiten sich individuell auf die Sitzung vor. Sie arbeiten das Material durch und markieren bzw. notieren alle ihre Feststellungen über Schwächen und Stärken. Die eigentliche Review-Sitzung dient nicht zum *Suchen*, sondern zum *Zusammentragen und Bewerten* von Schwächen und Stärken. Review-Sitzungen mit unvorbereiteten Teilnehmern sind Zeitverschwendung. Vorgegebene Prüflisten und Standards erleichtern die Arbeit.

3. Sitzung

Die Review-Sitzung wird von einem *Moderator* geleitet. Dieser sorgt für einen geordneten Sitzungsablauf und wacht über die Einhaltung der Review-Regeln. Zu den Aufgaben des Moderators gehören außerdem die Einladung zur Sitzung, die Überwachung der rechtzeitigen Verteilung des zu prüfenden Materials sowie die Weiterleitung des Review-Berichts an die verantwortliche Stelle.

4. Review-Bericht

Während der Sitzung wird ein Review-Bericht erstellt, welcher die in der Sitzung genannten und von der Mehrheit der Teilnehmer gebilligten Schwachstellen und Stärken auflistet. Der Review-Bericht soll einen Konsens der am Review Beteiligten ausdrücken; er wird am Ende der Sitzung von allen unterschrieben.

5. Überarbeitung und Nachkontrolle

Der Projektverantwortliche entscheidet aufgrund des Review-Berichts über die durchzuführenden Änderungen und gibt diese in Auftrag. Je nach Umfang dieser Änderungen erfolgt die Nachkontrolle durch ein neues Review oder durch den Projektverantwortlichen.

9.4.2.2 Informelle Reviews

Der mit formellen Reviews verbundene Organisationsaufwand scheint auf den ersten Blick beträchtlich. Man fragt sich, ob spontane, informelle Zusammenkünfte zum Finden, Besprechen oder Lösen von Problemen nicht viel einfacher und nützlicher wären. Die Erfahrung zeigt, dass dies nicht zutrifft. Da informelle Reviews nicht geplant werden, sind sie keine systematische qualitätssichernde Maßnahme, sondern es bleibt weitgehend dem Zufall überlassen, ob ein Produktteil ein Review durchläuft oder nicht. Man hat außerdem festgestellt, dass die Fehleraufdeckungsrate in informellen Reviews deutlich niedriger ist als in formellen Reviews.

Einen Sonderfall bildet die *Selbstinspektion*, wo eine Person ihr eigenes Arbeitsergebnis nach den gleichen strengen Regeln wie bei einer Inspektion durch Dritte inspiziert. Selbstinspektionen betrachten wir daher als Sonderfall der formellen Inspektion. Sie haben den Vorteil, ohne Planungsvorlauf jederzeit möglich zu sein, sind aber weniger effektiv als Inspektionen durch Dritte.

9.4.3 Rollen und Aufgaben der am Review Beteiligten

9.4.3.1 Moderator

Der Moderator organisiert und leitet die Review-Sitzung (vgl. 9.4.2.1, Punkt 3). Er sorgt ferner für die Fertigstellung und Weiterleitung des Review-Berichts. Im Idealfall ist der Moderator ein Mitarbeiter des Qualitätswesens und hat ein spezielles Training für die Leitung von Reviews absolviert.

Die Effektivität eines Reviews hängt stark vom Geschick des Moderators ab. Es geht unter anderem darum,

- eine offene Atmosphäre zu schaffen, in der jeder sich frei äußern kann, ohne Angst sich zu blamieren
- Zurückhaltende Teilnehmer zu Aussagen zu ermuntern
- Dauerredner zu bremsen
- Streit und persönliche Angriffe zu vermeiden bzw. beizulegen
- Abgleiten in nutzlose Detaildiskussionen zu verhindern
- immer wieder Konsens herzustellen
- generell über die Einhaltung der Review-Regeln zu wachen und diese freundlich, aber bestimmt durchzusetzen.

9.4.3.2 Schreiber

Der Schreiber erstellt den Review-Bericht. Dieser entsteht für alle Beteiligten sichtbar während der Sitzung. Technisch geschieht dies, indem der Schreiber beispielsweise am Hellraumprojektor auf Folien schreibt oder indem er an einem PC schreibt, dessen Bildschirminhalt an die Wand projiziert wird. Die übrigen Teilnehmer können so das Geschriebene laufend verfolgen

und Einspruch erheben, wenn sie meinen, es sei etwas nicht adäquat notiert worden. Schreiber und Moderator sollten möglichst nicht die gleiche Person sein.

9.4.3.3 Gutachter

Die übrigen Review-Teilnehmer haben die Aufgabe, das zu prüfende Material in der Vorbereitung zu begutachten und in der Sitzung über ihre Erkenntnisse zu berichten. Die Autoren des zu prüfenden Materials können nicht gleichzeitig Gutachter sein. Die Gutachter sind in der Regel Entwickler aus anderen Projekten der gleichen Abteilung oder des gleichen Bereichs. Gegebenenfalls können für bestimmte Probleme Fachgutachter oder Qualitätsingenieure herangezogen werden. Die Gutachter sollen

- sich sorgfältig vorbereiten
- alle wichtigen Punkte in der Sitzung nennen
- sich nicht persönlich profilieren wollen und nicht alles besser wissen
- den anderen gut zuhören und nicht bereits genannte Dinge erneut aufbringen
- während der Sitzung zu einer positiven, offenen und kooperativen Atmosphäre beitragen.

Der Schreiber kann gleichzeitig auch Gutachter sein.

9.4.3.4 Autor

Der Autor (bzw. ein Vertreter der Autorenschaft) nimmt aus zwei Gründen an der Sitzung teil: Erstens können Unklarheiten und Missverständnisse unter Umständen sofort geklärt und mit einer entsprechenden Notiz im Review-Bericht erledigt werden. Zweitens müssen die Autoren später anhand des Review-Berichts die festgestellten Mängel beheben. Die richtige Interpretation der Befunde im Review-Bericht fällt wesentlich leichter, wenn ein Vertreter der Autorenschaft bei der Abfassung dieses Berichts dabei gewesen ist.

9.4.3.5 Die Qualitätsverantwortung

Die Review-Teilnehmer sind verantwortlich für die *Qualität des Reviews*, d.h. dafür, dass einerseits schlechte Produkte als schlecht erkannt werden und möglichst alle Mängel im Review-Bericht aufgeführt sind, und dass andererseits gute Produkte als gut erkannt und nicht mit gesuchten und nebensächlichen Schwachstellen belastet werden.

Sie sind jedoch *nicht* verantwortlich für die Qualität des *Produkts*. Weder können sie für festgestellte Schwächen haftbar gemacht werden, noch dafür, dass festgestellte Mängel unter Umständen nicht behoben werden.

Die Verantwortung für die Qualität eines Produkts muss bei der selben Person liegen wie die Verantwortung für Sachziele, Kosten und Termine. Diese Person ist verantwortlich sowohl für die Veranlassung der Reviews als auch für die Veranlassung und Nachkontrolle der notwendigen Verbesserungen.

9.4.4 Review-Regeln

- Das zu prüfende Material wird rechtzeitig vor der Sitzung verteilt. Ein „Review“ von Tischvorlagen ist Zeitvergeudung.
- Es muss dafür gesorgt sein, dass die Teilnehmer rechtzeitig eingeladen werden und zu diesem Termin auch frei sind. Sollen Reviews zum festen Bestandteil einer Entwicklungsmethode werden, so empfiehlt es sich, einen Zeitblock auf einen festen Wochentag zu legen (z.B. immer Dienstags 10-12h), in dem Reviews durchgeführt werden und auf den nur in Notfällen andere Termine gelegt werden dürfen.
- Alle kommen vorbereitet zur Sitzung. Der Moderator bricht das Review ab, wenn sich zeigt, dass mehrere Teilnehmer nicht vorbereitet sind.

- An einem Review nehmen mindestens drei und höchstens etwa sieben Personen teil. Bei zu wenig Personen geht der zentrale Effekt des Gruppen-Gutachtens verloren. Ab etwa sieben teilnehmenden Personen tragen weitere Teilnehmer nur noch marginal zur Liste der Befunde bei, folglich wird das Review mit zunehmender Teilnehmerzahl immer unwirtschaftlicher.
- Eine Sitzung dauert maximal 2 Stunden. Bei längeren Sitzungen lässt die Konzentration nach und die Effektivität des Reviews sinkt rapide. Dementsprechend muss der Umfang des zu prüfenden Materials gewählt werden. Richtwerte: für Code-Reviews ca. 150-200 Codezeilen (ohne Kommentare), für Dokument-Reviews ca. 25 Seiten pro Sitzung.
- Die Probleme werden in der Sitzung nur genannt, nicht gelöst. Problemlösungsdiskussionen werden vom Moderator möglichst im Keim erstickt. Eine informelle Nachsitzung (z.B. bei einem gemeinsamen anschließenden Mittagessen) kann solchen Problemlösungsgesprächen Raum bieten.
- Jeder Gutachter nennt mindestens einen positiven und einen negativen Punkt im zu prüfenden Material.
- Stilfragen werden nicht diskutiert. Dort, wo vorgeschriebene Standards auch Stilfragen regeln, werden Abweichungen vom Standard als Mängel genannt, in allen anderen Fällen ist der Stil Ermessenssache der Autoren.
- Das Produkt wird bewertet, nicht deren Produzenten. Reviews sind dazu da, die Qualität von Produkten zu beurteilen. Sie sollen und dürfen nicht dazu verwendet werden, die Fähigkeiten der Personen, welche das Produkt erzeugt haben, zu beurteilen. Dies gilt sowohl für die Review-Teilnehmer wie für die Manager. Die Teilnehmer müssen sich mit persönlichen Beurteilungen zurückhalten, um die positive Atmosphäre während des Reviews nicht zu gefährden und um den Autor, welcher beim Review anwesend ist, nicht über das unvermeidliche Maß hinaus psychisch zu belasten. Die Manager dürfen die Review-Ergebnisse nicht zur Mitarbeiterbeurteilung heranziehen, wenn sie nicht wollen, dass die Reviews massiv an Effektivität verlieren, weil die Gutachter ihre Kollegen, welche das geprüfte Produkt erzeugt haben, schonen.
- Die Prüfung ist umso einfacher und effizienter, je mehr Standards und Prüflisten bereitstehen, nach denen bewertet werden kann. Wo keine solchen vorhanden sind, sollten sie daher unbedingt geschaffen bzw. eingeführt werden.
- Häufig werden beim Review nicht nur Befunde im Prüfling, sondern auch Fehler in den Referenzunterlagen gefunden. Solche Befunde werden ebenfalls erhoben und auf einer separaten Befundliste notiert.

9.4.5 Der Review-Bericht

Für den Review-Bericht werden mit Vorteil vorbereitete Formblätter verwendet. Ein Beispiel, wie solche Formblätter aussehen können, findet sich im Anhang. Das Deckblatt wird vom Moderator vorbereitet. Am Ende der Sitzung wird dort der Gesamtbefund notiert und von allen Beteiligten mit ihrer Unterschrift bestätigt. Daran angeheftet wird die Liste der Befunde.

9.5 Testen

Das Thema Testen wird in diesem Skript derzeit nur summarisch behandelt, da es nach dem bisherigen Curriculum an der Universität Zürich nicht in der Grundvorlesung, sondern in der Kernvorlesung Software Engineering behandelt wurde. Eine Überarbeitung des Kapitels ist für 2006 geplant. Derzeit sei für eine eingehende Behandlung des Themas auf die einschlägige Literatur verwiesen, beispielsweise das Buch von Spillner und Linz (2003).

9.5.1 Grundsätze

Mit Tests werden zwei Ziele verfolgt:

- Im Test sollen möglichst viele der in einem Programm vorhandenen Fehler gefunden werden (Myers, 1979: *Testen* ist der Prozess, ein Programm mit der Absicht auszuführen, Fehler zu finden.)
- Aus der Tatsache, dass das Programm mit einer Menge von Testdaten fehlerfrei läuft, soll mit brauchbarer statistischer Sicherheit darauf geschlossen werden, dass das Programm auch im Einsatz fehlerfrei läuft.

Die *Korrektheit* eines Programms kann durch Testen (außer in trivialen Fällen) *nicht bewiesen* werden. Der Grund dafür ist einfach. Für einen Korrektheitsbeweis müssten alle Kombinationen aller möglichen Werte der Eingabedaten getestet werden. Schon für ein triviales Programm, das lediglich zwei mit 16 Bit repräsentierte ganze Zahlen addiert, wären 2^{32} Fälle zu testen.

Testen setzt voraus, dass die erwarteten Ergebnisse bekannt sind. Folglich muss entweder gegen eine *Spezifikation* oder gegen *vorhandene Testergebnisse* getestet werden. Letzteres ist zum Beispiel der Fall, wenn Tests nach Programm-Modifikationen wiederholt werden (so genannter *Regressionstest*). Unvorbereitete und undokumentierte Tests sind sinnlos.

Nicht alle Eigenschaften eines Programms können mit Testen geprüft werden. Qualitätsmerkmale wie zum Beispiel Portabilität oder Pflegbarkeit sind nicht testbar.

Mit Testen werden nur Fehler*symptome*, nicht aber die Fehler*ursachen* gefunden. Auf jeden Test muss daher eine Phase folgen, in der die Ursachen zu den festgestellten Symptomen gefunden und behoben werden.

9.5.2 Testablauf

In jedem Test gibt es drei Phasen: Die Planung, die Vorbereitung und die Durchführung des Tests.

In der *Testplanung* wird die Teststrategie festgelegt und werden im Rahmen der Projektplanung die notwendigen Arbeiten und die dafür erforderlichen Ressourcen geplant.

In der *Testvorbereitung* werden die zu testenden Testfälle ausgewählt, eine Testumgebung zur Abarbeitung dieser Testfälle bereitgestellt und die Testvorschriften (d.h. die Vorgabedokumente, gemäß denen die Tests durchgeführt werden) erstellt.

Bei der *Testdurchführung* wird zunächst die Testumgebung nach den Angaben der für diesen Test geltenden Testvorschrift eingerichtet. Dann werden die Testfälle der Testvorschrift der Reihe nach ausgeführt. Alle Befunde werden notiert. Während der Testdurchführung wird der Prüfling nicht verändert. Die Testergebnisse werden in einer Testzusammenfassung festgehalten.

Die *Fehlerbehebung* ist kein Bestandteil des Tests. Sie erfolgt nachher, indem die Testbefunde analysiert, die Fehlerursachen bestimmt und die Fehler dann behoben werden.

Es ist sehr wichtig, dass die Aufwendungen für das Testen in jedem Projekt sorgfältig geschätzt und in der Projektplanung berücksichtigt werden. Nur dann ist es möglich, die für das Erreichen der angestrebten Qualität notwendigen Tests auch tatsächlich durchzuführen.

9.5.3 Testverfahren

Man unterscheidet zwei Klassen von Testverfahren. Beim *funktionsorientierten* Test (Black-Box-Test) erfolgt die Auswahl der Testfälle aufgrund der Spezifikation. Die Programmstruktur kann unbekannt sein. Es wird angestrebt, die geforderten Eigenschaften des Prüflings (Funktionalität, Leistungsverhalten, weitere Attribute) des Prüflings möglichst vollständig durch Testfälle abzudecken.

Beim *strukturorientierten* Test (White-Box-Test, Glass-Box-Test) erfolgt die Auswahl der Testfälle aufgrund der Programmstruktur. Es wird angestrebt, die Programmstruktur möglichst vollständig abzudecken (zum Beispiel alle Programmzweige mindestens einmal zu durchlaufen). Die Spezifikation muss ebenfalls bekannt sein, damit festgelegt werden kann, welche Resultate mit den ausgewählten Testdaten zu erwarten sind.

Für Einzelheiten wird auf die Literatur (zum Beispiel Beizer 1995, Frühauf, Ludewig, und Sandmayr 1991, Liggesmeyer 1990, Myers 1979) verwiesen.

Aufgaben

9.1 Sie arbeiten an der Entwicklung eines Software-Produkts mit. Wer ist für die Qualität dieses Produkts verantwortlich?

9.2 Sie sind Projektleiter in einem Software-Entwicklungsprojekt. Soeben haben Sie den Review-Bericht über das Lösungskonzept der zu entwickelnden Software erhalten. Der Bericht weist neben einer Reihe leichter Fehler drei schwere Fehler und eine als kritisch eingestufte Auslassung aus. Sie sind gegenüber dem Zeitplan drei Wochen in Verzug und möchten daher so rasch wie möglich mit dem Detailentwurf und der Codierung beginnen lassen. Wie verhalten Sie sich? Was halten Sie von folgenden Maßnahmen?

a) Sie lassen alle festgestellten Mängel beheben und beginnen nicht mit dem Detailentwurf, bis das Dokument das Nachreview bestanden hat und freigegeben ist.
b) Sie lassen sofort mit Detailentwurf und Codierung beginnen und beschließen, die Fehler und Lücken erst zu beseitigen, wenn man beim Entwerfen bzw. Codieren an diese Stellen kommt.
c) Sie gehen den Review-Bericht durch, markieren die Punkte, die Ihnen schwerwiegend erscheinen und lassen nur diese beheben.
d) Sie veranlassen die Behebung der festgestellten Fehler, lassen aber gleichzeitig mit dem Detailentwurf eines Teilsystems beginnen, in dessen Konzept nur zwei leichte Mängel gefunden wurden.

9.3 Inspizieren Sie die im Anhang 2 gegebene Prozedur. Bereiten Sie sich vor, indem Sie die Prozedur durcharbeiten und Ihre Befunde notieren. Tun Sie sich mit drei oder vier Kolleginnen und Kollegen zusammen und führen Sie eine Review-Sitzung durch. Vergleichen Sie die Befundliste der Sitzung mit Ihren eigenen Befunden.

9.4 Warum sind unvorbereitete und undokumentierte Tests sinnlos?

Zitierte und ergänzende Literatur

Beizer, B. (1995). *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. New York etc.: John Wiley & Sons.

Berner, S., S. Joos, M. Glinz (1997). Entwicklungsrichtlinien für die Programmiersprache Java. *Informatik/Informatique* **4**, 3 (Jun 1997). 8-11.

Deming, W.E. (1986). *Out of the Crisis*. Cambridge, Mass.: M.I.T. Press.

Dunn, R. (1990). *Software Quality - Concepts and Plans*. Englewood Cliffs, N.J.: Prentice-Hall. [in dt. Übersetzung: *Software-Qualität. Konzepte und Pläne*. München: Hanser, 1993]

Fagan, M.E. (1986). Advances in Software Inspections. *IEEE Transactions on Software Engineering*, **SE-12**, 7 (July 1986). 744-751.

Freedman D.P., G.M. Weinberg (1982). *Handbook of Walkthroughs, Inspections and Technical Reviews*. 3rd edition. Boston, Toronto: Little, Brown and Co.

Frühauf, K., J. Ludewig, H. Sandmayr (2000). *Software-Projektmanagement und -Qualitätssicherung*. 3. Auflage. Zürich: vdf Hochschulverlag.

Frühauf, K., J. Ludewig, H. Sandmayr (1991). *Software-Prüfung. Eine Fibel*. Zürich: vdf und Stuttgart: Teubner.

Liggesmeyer, P. (1990). *Modultest und Modulverifikation*. Reihe Angewandte Informatik Bd. 4. Mannheim, etc.: BI-Wissenschaftsverlag.

Liggesmeyer P., H.M. Sneed, A. Spillner (Hrsg.) (1992). *Testen, Analysieren und Verifizieren von Software*. Berlin, etc.: Springer.

Myers, G.J. (1979). *The Art of Software Testing*. New York: John Wiley & Sons. [in dt. Übersetzung: *Methodisches Testen von Programmen*. 4. Auflage. München: Oldenbourg, 1991.]

Russell, G.W. (1991). Experience with Inspection in Ultralarge-Scale Developments. *IEEE Software* **8**, 1 (Jan. 1991). 25-31.

Schnurer, K.E. (1988) Programminspektionen. *Informatik-Spektrum* **11**, 6 (Dez. 1988). 312-322.

Spillner, A., T. Linz (2003). *Basiswissen Softwaretest*. Heidelberg: dpunkt-Verlag.

Normen

IEEE (1990). *Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. IEEE Computer Society Press.

ISO 9000:2000. *Qualitätsmanagementsysteme – Grundlagen und Begriffe*. Deutsche Fassung der Europäischen Norm EN ISO 9000 (deutsch/englisch/französisch).

ISO 9001:2000. *Qualitätsmanagementsysteme – Anforderungen*. Deutsche Fassung der Europäischen Norm EN ISO 9001 (deutsch/englisch/französisch).

ISO 9004:2000. *Qualitätsmanagementsysteme – Leitfaden zur Leistungsverbesserung*. Deutsche Fassung der Europäischen Norm EN ISO 9004 (deutsch/englisch/französisch).

Anhänge:

Anhang 1: Formblätter für Review-Bericht:

- Deckblatt mit Review-Zusammenfassung
- Liste der Befunde

Anhang 2: Unterlagen für die Codeinspektion aus Aufgabe 9.3

Review-Bericht	Review-Nr.: Seite _____ von _____ Datum: Zeit von: bis:

Zu prüfendes Produkt

Nummer / Version	Titel

Referenzunterlagen

Nummer / Version	Titel

Bewertung

- ❑ akzeptiert (kein neues Review erforderlich)
- ❑ nicht akzeptiert (neues Review erforderlich)
- ❑ Review nicht beendet

- ❑ wie es ist
- ❑ kleine Änderungen
- ❑ große Änderungen
- ❑ Überarbeitung

Review-Team

Name	Funktion	Vorbereitungszeit	Unterschrift

Freigabe (nach Nachkontrolle)

Name	Datum	Unterschrift

Review Nr.: Review-Bericht Seite _____ von _____

Liste der Befunde

❑ im zu prüfenden Produkt
❑ in den Referenzunterlagen

Nr.	Ort	Befund	Gewicht

Gewicht: **K**ritischer / **W**ichtiger / **N**ebensächlicher Befund **G**ut

Anhang 2: Unterlagen für die Codeinspektion aus Aufgabe 9.3

1. Zu inspizierendes Codestück (in Modula-2)

```
PROCEDURE KennlinieTabellieren (b:REAL; e:REAL; d: REAL);
(* Tabelliert die verlangte Funktion im Bereich von b bis e *)
VAR lfd:     REAL; (* laufender x-Wert für Fkt.-Berechnung *)
    FktWert: REAL; (* berechneter Funktionswert *)

BEGIN
  (* Titel schreiben, Werte initialisieren: *)
  WriteString ("         x  F(x)=2x**2 e**(x**2) -3");
  WriteLn;
  lfd := b;
  (*Tabellierschleife: *)
  REPEAT
  FktWert := 2.0*3.141*lfd*lfd*exp(lfd*lfd) - 3.0;
  WriteReal (lfd, 10); WriteReal (FktWert, 12); WriteLn;
  lfd := lfd + d; (* laufenden Wert um Schrittweite erhöhen *)
  UNTIL lfd = e;
END KennlinieTabellieren;
```

Hinweise zum Lesen von Programmen in Modula-2

- `WriteString` schreibt eine Zeichenkette auf ein Ausgabemedium.
- `WriteReal (zahl, n)` konvertiert `zahl` in eine Zeichenkette und schreibt diese auf einer Breite von insgesamt `n` Zeichen auf ein Ausgabemedium.
- Die von mehreren Write-Anweisungen erzeugten Ausgaben schließen unmittelbar und ohne Zwischenraum aneinander an.

Durch den Aufruf von `WriteLn` wird eine neue Zeile auf dem Ausgabemedium begonnen.

- Es gibt keinen Operator für Exponentiation. Die Berechnung von x^2 wird als `x*x` programmiert.
- `exp (x)` ist eine eingebaute Standardfunktion, welche die Exponentialfunktion e^x berechnet.

2. Auszug aus dem Entwurfsdokument:

```
...
Prozedur Kennlinie tabellieren:

Die Kennlinie, welche durch die Funktion

f(x) = 2π x^2 e^(x^2)-3

gegeben ist, wird im Bereich (x1,x2) mit der Schrittweite Δx tabelliert. Für
den Fall x2<x1 soll die Tabellierung unterbleiben. Über der Tabelle wird ein
Titel
        x  f(x)=...
 ausgedruckt.
...
```

3. Auszug aus den Codier-Richtlinien:

. . .

- Jede Anweisung beginnt auf einer neuen Zeile.
- Die Körper aller Prozeduren und Blöcke sind jeweils 2 Zeichen einzurücken.
- Namen sind so zu wählen, dass sie möglichst selbstdokumentierend sind.
- Namen, die aus drei und weniger Zeichen bestehen, sollen möglichst nur für lokale Aufgaben (Schleifenindizes o.ä. verwendet werden. Ihre Bedeutung ist bei der Definition zu dokumentieren.
- Konstante Größen sind am Beginn der Programme bzw. Prozeduren als symbolische Konstanten mit einem selbsterklärenden Namen zu vereinbaren.
- Jede Prozedur hat einen Kommentarkopf, in dem die Aufgabe der Prozedur sowie die Bedeutung der Prozedurparameter erklärt sind.

. . .

4. Auszug aus der Prüfliste für Codeinspektionen

. . .

- Übereinstimmung des Codes mit Entwurfsdokument:
  - Schnittstelle?
  - Datenstrukturen adäquat?
  - Algorithmen richtig? effizient?
  - geforderte Ein-/Ausgaben vorhanden?
- alle verwendeten Variablen initialisiert?
- Schleifen:
  - Verhalten am Anfang?
  - Verhalten am Ende; terminiert die Schleife sicher?
  - Weist die Schleife illegale Durchläufe ab?
  - Hat es schleifeninvarianten Code in der Schleife?
  - falls Laufvariablen vorhanden:
    - Laufvariable initialisiert?
    - Fortschaltung der Laufvariablen vorhanden und richtig?
- Ausgabe:
  - Zeilenende mit Zeilenfortschaltbefehl abgeschlossen?
  - tabellarisch aufgebaute Ausgaben sauber untereinander?

. . .