Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 84 (1993)

Heft: 25

Artikel: Fehlerfreie Software : ein realistisches Ziel?

Autor: Schaltegger, Peter

DOI: https://doi.org/10.5169/seals-902768

Moderne elektronische Geräte beinhalten meist Hard- und Softwarefunktionen. Während sich zur Sicherstellung der Hardwarequalität mehr oder weniger einheitliche Konzepte durchgesetzt und bewährt haben, haben die erst nach 1985 entstandenen Software-QS-Konzepte noch nicht dieselbe Verbreitung gefunden. Dass ein Nachholbedarf besteht, belegen einige Softwareausfälle, die in letzter Zeit Schlagzeilen gemacht haben. In diesem Beitrag wird gezeigt, wie die Zahl der Programmfehler reduziert werden kann, wobei ein Vorgehen zur Produktsicherung als auch ein prozessorientiertes Vorgehen dargestellt werden.

Fehlerfreie Software – ein realistisches Ziel?

■ Peter Schaltegger

Mit der Qualität und Zuverlässigkeit der Hardware befasst sich die Industrie seit Jahren. Zwei Stichworte dazu sind Komponentenprüfung und Zuverlässigkeitsanalysen. Gewisse Konzepte, wie zum Beispiel die Verlagerung von der Eingangsprüfung beim Gerätehersteller zur Ausgangsprüfung beim Komponentenhersteller, haben sich durchgesetzt. Über Qualitätssicherungskonzepte für Software aber, sowie darüber, wie man bei einer allfälligen Einführung vorzugehen hat, besteht nicht dieselbe Klarheit. Dabei liegen in der Elektronikindustrie die Kosten für die Softwareentwicklung in derselben Grössenordnung wie für die Hardwareentwicklung. Andererseits kann man doch feststellen, dass das Bewusstsein wächst – und von Entwicklungsleitern namhafter Unternehmen geteilt wird –, dass die Software-Qualitätssicherung schon bald von grosser Wichtigkeit sein wird.

Qualitätsrelevante Unterschiede bei der Behandlung von Hardware und Software

Jeder Hardwareabnehmer verlangt von seinem Lieferanten eine hohe Qualität und Zuverlässigkeit der Produkte. Durch den Erwerb des Qualitätssicherungs-Zertifikats, aber auch durch Abgabe von Zuverlässigkeitsanalysen oder zumindest durch Angabe eines MTBF-Wertes wird den Wünschen der Abnehmer Rechnung getragen. Nach einem Ausfall interessiert sich der Abnehmer nicht dafür, ob die Hardware oder die Software schuld ist; er verlangt, dass möglichst wenig Ausfälle auftreten. Es liegt also nicht zuletzt im eigenen Interesse des Lieferanten, das oben angetönte Ungleichgewicht zwischen Soft- und Hardwarequalität sowie Zuverlässigkeit zu beheben. Softwarezuverlässigkeit darf nicht erst wichtig sein, wenn ein Softwarefehler einen Ausfall erzeugt.

Im folgenden soll zunächst auf Unterschiede und Gemeinsamkeiten von Hardware und Software hingewiesen werden. Anschliessend wenden wir uns der Fehlerfreiheit von Software zu. Welche Massnahmen können kurz- und langfristig das Ziel der Fehlerfreiheit näherbringen? Gerade bei produktintegrierter Software ist die Fehlerfreiheit besonders wichtig. In der Produktehaftpflicht ist die Fehlerfreiheit ein zentraler Begriff. Auch Softwarefehler dürfen nicht zu einem Sicherheitsrisiko werden.

Hardwaresysteme bestehen aus elektronischen Komponenten, Lötstellen, Leiterbahnen, Steckern usw. Die Aufgliederung in Komponenten, Baugruppen und Gesamtsysteme ist daher naheliegend. Entsprechend gibt es Komponenten-, Baugruppen- und Systemtests. Für die Komponentenqualität ist primär der Komponentenhersteller verantwortlich. Die Komponenten sind fast immer Grossserienprodukte von recht konstanter Qualität. Die Gerätequalität hängt von den eingekauften Komponenten, der Entwicklung und der Fertigung ab.

Adresse des Autors:
Peter Schaltegger, lic. math., Schaltegger Zuverlässigkeitssicherung, 8126 Zumikon.

Glossar	
Case	Computer-Aided Software Engineering
FMEA	Failure Modes and Effect Analysis
KLOC	1000 Lines of Code
MTBF	Mean Time Between Failures

Tabelle I

Software besteht aus Codezeilen, Moduln, dem Gesamtprogramm und der Dokumentation. Die Programmqualität hängt ausschliesslich von der Entwicklung ab. Zugekauft werden allenfalls Hilfsprogramme, nicht aber Komponenten. Jede Codezeile kann einen Fehler aufweisen, da sie von einem Programmierer geschrieben wurde. Auf Modul- und Programmebene besteht zwar eine gewisse Analogie zu Prints und reinen Hardwaresystemen. Modul- und Systemtests haben sich eingebürgert. Hinzu kommt allenfalls ein Abnahmetest für den Kunden. Doch der Faktor Mensch spielt weiterhin eine grössere Rolle als bei der Hardware, bei der die Komponentenherstellung und Printbestükkung stark automatisiert sind.

Beschränkung auf die Fehlerfreiheit

Da die Qualitätssicherung ein sehr breites Gebiet ist, soll bei der folgenden Betrachtung eine Beschränkung auf die Fehlerfreiheit vorgenommen werden; viele Autoren sprechen dann von Softwarezuverlässigkeit. In der Zuverlässigkeitssicherung beschränkt man sich auf das Merkmal der Funktionsfähigkeit. Sie kann durch Ausfälle eingeschränkt oder vernichtet werden. Ausfälle sind auf Fehler zurückzuführen. Ein gradueller Unterschied zwischen Hardware und Software besteht darin, dass die Hardware durch physikalisch-chemische Prozesse beeinträchtigt werden kann, was auch durch Prüfungen des Herstellers nicht gänzlich vermieden werden kann. Da Software immateriell ist, besteht dieses Problem nicht. Fragen der Sicherheit werden hier nicht speziell behandelt, obwohl Sicherheitsanforderungen die Bedeutung der Fehlerfreiheit wesentlich erhöhen.

Bei Software kann man sich auf die Entwicklung konzentrieren. Es entsteht ein Prototyp. Ist er fehlerfrei, so sind alle Kopien auch fehlerfrei. Die Produktion, das Kopieren ist ein problemloser Vorgang. Allerdings ist die Kennzeichnung des Status des Prototypen nicht trivial. Da der Einfluss der Mitarbeiter grösser ist und Menschen Fehler machen können, enthalten Programme immer Fehler. Case-Werkzeuge, wie beispielsweise Debugger, sind Ansätze in Richtung Teilautomation. Die Fehlerbehebung ist deshalb auf absehbare Zeit ein wichtiges Element der Software-Zuverlässigkeitssicherung. Information und Schulung der Mitarbeiter haben deshalb einen hohen Stellenwert.

Zusammenhang zwischen Fehlern und Ausfällen

Bei Zuverlässigkeitsanalysen von Hardware wird angenommen, dass ein Komponentenausfall zu einem Geräteausfall führt, was nicht immer zutrifft. Das führt übrigens dazu, dass die Zahl der Ausfälle im Feld tiefer liegt, als nach der Zuverlässigkeitsanalyse zu erwarten wäre. Mit einer Analyse der Ausfallarten (FMEA) kann dieser Effekt allerdings berücksichtigt werden. Die Auswirkungen von Softwarefehlern sind mannigfaltiger als die von Hardwarefehlern. Ein Softwarefehler kann zum Beispiel zu einem formalen Fehler auf einer Anzeige, er kann aber auch zu einem Vollausfall führen. FMEA-Analysen werden übrigens auch für Software vorgeschlagen. Grundsätzlich kann man sich nicht auf bestimmte Arten von Softwarefehlern beschränken. Software ist nicht stetig, worauf in [1] hingewiesen wird. Der Begriff Fehler ist bei Software zentral. Softwarefehler dürfen allerdings nicht mit Programmierfehlern gleichgesetzt werden. Softwarefehler entstehen schon in der Anforderungs- und Konzeptphase.

Flexibilität

Ein Vorteil der Software besteht in der grösseren Flexibilität, das heisst in ihrer Änderbarkeit. Die Kehrseite zeigt sich darin, dass Änderungen an der Software deren Qualität beeinträchtigt. Fehlerkorrekturen können vor der Freigabe gemacht werden; nachher sind sie – besonders bei produktintegrierter Software – unerwünscht.

Qualitätsbewusstsein, Arbeitsweise

Bei der Hardwareentwicklung ist in den letzten Jahren das Qualitätsbewusstsein gestiegen; man spricht zurecht von Engineering. Bei der Softwareentwicklung haben sich erst Elemente des Engineerings eingebürgert, wobei sich die Softwareunternehmen diesbezüglich mehr als die Hardwarehersteller unterscheiden. Realistischerweise muss man sogar feststellen, dass das Qualitätsbewusstsein noch wenig entwickelt ist. Das heisst nun nicht, dass alle Software von fragwürdiger Qualität ist. Aber das Bewusstsein, dass jede Phase so zu bearbeiten ist, dass möglichst wenig Fehler entstehen, ist noch zuwenig ausgeprägt. Wer denkt bei der Formulierung des Pflichtenhefts ernsthaft an mögliche Fehler? Zum Teil ist auch der Arbeitsstil im Softwaregewerbe wenig geeignet, Fehler zu vermeiden. Natürlich können Fehler im nachhinein durch Tests weitgehend eliminiert werden, so dass das Endprodukt trotzdem befriedigt; der Testaufwand ist aber ein wesentlicher Kostenfaktor.

Die Software-Qualitätssicherung hat zum Ziel, Qualität *in jeder Phase* zu gewährleisten. Qualitätsprüfungen und Korrekturmassnahmen sollen reduziert werden. Auf das Testen von Software wird man allerdings nicht so bald verzichten können, denn welcher Software-Projektleiter könnte behaupten, dass sein Programm fehlerfrei sei, oder angeben, wieviele Fehler es enthält. Erfahrungswerte über die Zahl der Fehler, die ein Programm nach Freigabe noch enthält, sprechen von 2 bis 10 Fehlern pro 1000 Zeilen ausführbaren Codes (KLOC). Auf jeden Fall besteht ein Bedarf, die Zuverlässigkeit von Programmen beurteilen zu können.

Software-Fehlerbehebung

Wie bei der Hardware unterscheidet man auch bei der Software zwischen Engineering und Qualitätssicherung, obwohl zwischen den beiden Tätigkeiten Wechselwirkungen bestehen. Das Software-Engineering wird sich in Richtung Case weiterentwickeln, wobei die Anliegen der Qualitätssicherung noch vermehrt miteinbezogen werden müssen.

Die Software-Qualitätssicherung stellt Kataloge von Merkmalen zur Verfügung. Zu Beginn eines Softwareprojektes ist ein Anforderungsprofil aufzustellen, in dem die Wichtigkeit der verschiedenen Qualitätsmerkmale festgelegt wird; immer wichtig wird das Merkmal der Fehlerfreiheit sein. Qualitätsmerkmale können quantifizierbar

Begriffe und Abkürzungen nach IEEE-Standard 982.2	
Irrtum (Error):	Menschliche Handlung, die zu einer Abweichung vom angestrebten Soll-Zustand (Softwarefehler) führt.
Defekt:	Eine Anomalie eines Produktes (oder Dokuments)
Fehler (Fault):	Eine Bedingung, die bei einer funktionalen Einheit dazu führt, dass diese ihre Funktion nicht mehr ausführen kann.
Ausfall (Failure):	Die Beendigung der Fähigkeit einer funktionalen Einheit, ihre benötigte Funktion auszuführen.
Metrik:	Eine quantitative Abschätzung des Grades, den ein Softwareprodukt oder -prozess bezüglich eines Merkmals aufweist.

Tabelle II

oder nichtquantifizierbar sein. Tendenziell sind quantitative Merkmale, welche durch Kennzahlen ausgedrückt werden, zu bevorzugen (Fehler sind quantifizierbar). Betreffend Metriken wird auf [1] verwiesen.

Zuverlässigkeitsanalysen für Software?

Der Gedanke liegt nahe, – analog wie bei der Hardwaretechnik – auch in der Softwareentwicklung mit Zuverlässigkeitsanalysen zu arbeiten. Leider ist dies nicht möglich. Die Zuverlässigkeitsanalysen von Hardware basieren auf Datensammlungen mit Komponentenausfallraten. Ausfallraten zum Beispiel für Codezeilen zu ermitteln, ist doch wohl illusorisch. Es muss also ein anderer Ansatz gefunden werden.

Fehlerentdeckung und -behebung

Da die Fehler den entscheidenden Faktor darstellen, muss bei der Fehlerentdeckung und -behebung angesetzt werden. Der IEEE-Standard 982.2 [2] unterscheidet zwischen Irrtum (Error), Fehler (Fault) und Ausfall (Failure). Ein Irrtum ist eine menschliche Handlung, die zu einer Abweichung vom angestrebten Soll-Zustand führt. Ein Fehler ist ein Codefehler. Wird die fehlerhafte Codezeile ausgeführt, so verursacht sie einen Ausfall, das heisst eine Abweichung vom Soll-Zustand. Der Fehler bezieht sich auf den Zustand des Programms; der Ausfall ist ein zeitabhängiges Ereignis beim Programmablauf.

Irrtümer können in jeder Entwicklungsphase entstehen und sich in einem Programmfehler äussern. Programmfehler verursachen Ausfälle, wobei man unter Ausfall die Beeinträchtigung einer wesentlichen Programmfunktion versteht. Was als Ausfall anzusehen ist, muss von Fall zu Fall im Detail festgelegt werden. Jeder Ausfall geht zwar auf mindestens einen Programmfehler zurück, aber nicht jeder Fehler führt zu einer Auswirkung, welche das Kriterium eines Ausfalls erfüllt. Da komplexe Programme eine enorme Zahl von Ablaufpfaden aufweisen, werden nicht alle Fehler in der Testphase entdeckt. Die Fehlerwirkung kann auch von den Inputdaten abhängen. Die Ausfälle sind deshalb zeitabhängig und scheinbar zufällig verteilt, obwohl die Fehler natürlich deterministisch sind. Es kann auch Fehler geben, die nie zu einem Ausfall führen. Ein Programm weist erst nach der Integration die volle Funktionalität auf, weshalb Ausfälle erst beim Systemtest beurteilt werden können. Das Ziel der Software-Zuverlässigkeitsprüfung ist das Vermeiden von Ausfällen.

Fehleraufzeichnung

Die Phase des Systemtests wird in verschiedene Testabschnitte und -läufe unterteilt. Die Ausfälle werden protokolliert und die verursachenden Fehler gesucht und korrigiert. Beim Testen wird man eine Abnahme der Zahl gefundener Fehler pro Testabschnitt feststellen, was einen Reifeprozess zum Ausdruck bringt. Als Kriterium für den Abbruch der Testphase wird das Unterschreiten einer Zahl gefundener Fehler pro Zeitabschnitt vorgeschlagen. Die Fehlerfunktion, welche die kumulierten Fehler im Testverlauf darstellt, legt es nahe, das Unterschreiten einer Zahl gefundener Fehler pro Zeiteinheit als Kriterium für den Abbruch der Testphase zu verwenden. Die Auswertung der Fehlerprotokollierung liefert Kennzahlen über die Reduktion der Fehlerdichte pro 1000 Zeilen Code. Die Kennzahl gibt für spätere Projekte einen Hinweis, wieviele Fehler zu erwarten sind. Allenfalls können die Fehler nach der Schwere ihrer Auswirkung unterschieden werden (gravierende, formale Fehler).

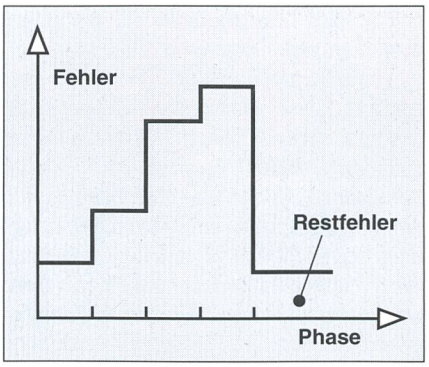

Bild 1a Fehlerentwicklung, wenn nur der Systemtest durchgeführt wird

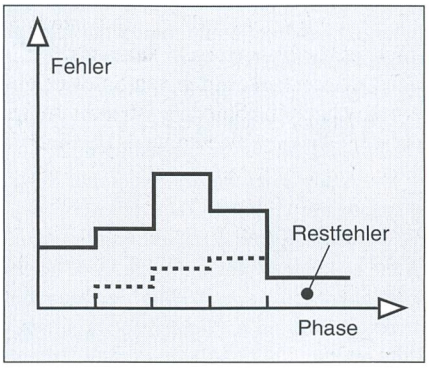

Bild 1b Fehlerentwicklung beim prozessorientierten Vorgehen

Die Fehlererfassung muss durch die Metrik der Testabdeckung ergänzt werden. Bei geringer Testabdeckung findet man weniger Fehler. Man wird bald das Kriterium für den Testabbruch erreichen. Nur bei hoher Testabdeckung kann der Schluss gezogen werden, dass das Programm weitgehend fehlerfrei ist. Die Testabdeckung gibt auch einen Hinweis auf die Zahl der Restfehler, die nicht gefunden wurden. Schon bei der Testplanung muss man sich mit der Testabdeckung befassen. Einerseits soll möglichst das ganze Programm getestet werden, andererseits aber eine grössere Testredundanz vermieden werden.

Die Kostenfrage

Nun wird die Frage auftauchen, ob ein systematisches Testen die Programmkosten nicht wesentlich erhöht. Darauf lässt sich antworten, dass das Testen unvermeidlich ist und bereits heute recht viel kostet, dass aber in der Testphase vielerorts noch ein erhebliches Effizienzsteigerungs-Potential vorliegt. Mit einem systematischen Vorgehen kann man bei gleichem Aufwand mehr Testabdeckung und -tiefe erreichen. Da ein Ausfall beim Testen (inkl. Fehlerkorrektur) weit weniger kostet, als ein Ausfall im Feldeinsatz, ist das Testen sehr lohnend, solange relevante Fehler gefunden werden. Immerhin, professionelles Testen kann den Testaufwand erhöhen, und Einsparungen werden erst später durch Reduktion von Garantie- und Wartungskosten realisiert.

Fehlerdatenerfassung

Bei der Einführung der Fehlerdatenerfassung wird empfohlen, dass man beim Systemtest beginnt und eventuell später den Modultest einbezieht. Häufig wird ein Modul von einem einzelnen Programmierer erstellt. Es könnte der Verdacht aufkommen, dass die Fehlerdatenerfassung der individuellen Qualifikation dient. Das darf natürlich nicht sein. Der Programmierer sollte die Gelegenheit haben, Fehler selbst zu eliminieren. Die Fehlerbehebung ist auf der Modulstufe noch etwas kostengünstiger als auf der Systemstufe.

Testteam

Der Systemtest sollte nicht durch das Entwicklerteam, sondern durch ein Testteam durchgeführt werden. Eine solche Arbeitsteilung existiert ja auch bei der Hardware (Prüffeld). Das Programmtesten stellt andere Anforderungen als das Programmentwikkeln. Zudem ist es auch aus psychologischen Gründen schwierig, ein Programm zu entwickeln und nachher kritisch zu testen. Es gibt unter anderem Fehler, die auf eine falsche Interpretation des Pflichtenhefts zurückzuführen sind. Diese würden durch den Programmierer wohl kaum entdeckt.

Weiteres Vorgehen

Zwei wichtige Metriken wurden erwähnt: die Fehlerdichte und die Testabdeckung. Später sollten je nach Anforderungen weitere Metriken und Indikatoren hinzukommen. Eine Liste von Metriken mit Kurzbeschreibungen enthält der IEEE-Standard 982.2 [2]. Die Auswahl hängt von der Art der Software ab. Zudem muss das Kosten-Nutzen-Verhältnis beachtet werden. Eine sehr sorgfältige Auswahl wird empfohlen.

Vorgehen zur Fehlerreduktion in Programmen

Software-Qualitätssicherung als Rahmen

Software-Engineering befindet sich in rascher Entwicklung und Case-Werkzeuge finden immer weitere Verbreitung. Die Software-Qualitätssicherung wird durch den Zwang zur Zertifizierung gefördert. Die Produktehaftung ist ein weiteres Stichwort, wobei in diesem Zusammenhang der Fehlerfreiheit besondere Bedeutung zukommt. Das Ziel kann zunächst durch Konzentration auf die Produktqualität und später durch den Ausbau zu einer prozessorienten Fehlerbeseitigung erreicht werden.

Vorgehensweise

Eine erste Voraussetzung ist die Fehlerdatenerfassung. Sie beginnt mit der Ausfallprotokollierung und soll zunächst in der Phase des Systemtests eingeführt werden. Die Anzahl Ausfälle pro Zeiteinheit, zum Beispiel pro Testtag, werden erfasst und graphisch aufgetragen. Die Ausfälle werden nach Schwere unterschieden. Aus der Protokollierung muss klar hervorgehen, wie sich ein Ausfall geäussert hat und wo bzw. wann er aufgetreten ist. Wenn möglich soll ein Beleg den Ausfall illustrieren. Zusammen mit dem Programmierer muss dann der Fehler im Code gesucht werden.

Die Protokollierung muss durch die Testabdeckung ergänzt werden. Sie kann funktions- oder strukturbezogen vorgenommen werden. Die Teilfunktionen werden aufgelistet. In der Testplanung wird festgehalten, wie vorzugehen ist. Während den Tests wird protokolliert, welche Teilfunktionen getestet wurden. Die Testabdeckung ist der Quotient getestete Funktionen zu allen Funktionen. Für die Testabdeckung ist eine Zielvorgabe zu machen. Sie hängt ab vom Risiko, das man zu tragen bereit ist, und vom Grad der Detaillierung der Testplanung. Auch der Testaufwand in Mannstunden soll erfasst werden. Gegen Ende der Testphase soll eine Abschätzung der Restfehler vorgenommen werden. Das Vorgehen kann dem IEEE-Standard 982.2 entnommen werden. Die Restfehlerabschätzung kann anfänglich zu einer Ernüchterung führen.

Schliesslich wird ein Freigabekriterium festgelegt. Berücksichtigt werden die Ausfälle, die pro Zeiteinheit noch gefunden wurden, die erreichte Testabdeckung und allenfalls der Testaufwand. Zur Beurteilung der Erfahrungswerte aus den Tests können auch Normwerte aus der Literatur beigezogen werden. Der Entscheid über die Freigabe wird der Projektleiter zusammen mit dem Testteam treffen. Durch die Berücksichtigung von Metriken wird der Entscheid auf eine objektive Grundlage gestellt, ein Vorgehen, das den Projektleiter erheblich entlastet.

Worin liegt der Sinn des oben geschilderten Vorgehens? Sein Hauptvorteil liegt darin, dass man über die Fehlerhäufigkeit genauere und objektive Informationen erhält. Damit kann man das Risiko, das verbleibende Fehler darstellen, erst richtig einschätzen. Die Erfahrung zeigt, dass die Beurteilungssicherheit den Freigabeentscheid wesentlich erleichert. Da die Tests solange weitergeführt werden, bis die Kriterien erfüllt sind, wird eine hohe Zuverlässigkeit des Programms erreicht. Zudem werden Erfahrungswerte für zukünftige Projekte gewonnen. Die Fehlerdichte bei Beginn und am Ende des Systemtests kann abgeschätzt werden. Für die Anzahl der verbliebenen Fehler wird die Restfehlerabschätzung verwendet. Sie gibt nur die Grössenordnung an. Daraus ist ersichtlich, welcher Teil aller Fehler durch die Tests eliminiert wurde.

Die Analyse der Art der Fehler und wo sie entstanden sind, liefert klare Hinweise, wie die Fehlerhäufigkeit reduziert werden kann. Wenn neue Methoden und Werkzeuge eingesetzt werden, kann mit Hilfe der Metriken deren Einfluss auf die Fehlerhäufigkeit beurteilt werden.

Information und Schulung

Information und Schulung waren bereits bei der Einführung der Hardware-Qualitätssicherung ein Muss. Wie sollte es da bei der Software anders sein? Insbesondere der Projektleiter muss von der Notwendigkeit und vom Nutzen der Massnahmen überzeugt sein. Bei der Einführung sollte er durch einen internen oder externen Berater unterstützt werden. Das ganze Projektteam muss rechtzeitig über die Einführung informiert werden. Motivation ist dringend nötig; teilweise müssen vorgefasste Meinungen über die Qualitätssicherung abgebaut werden. Am besten sind die Voraussetzungen, wenn im ganzen Softwarebereich eine gute Qualitätsmotivation besteht. Die Schulung betrifft vor allem das Testteam, das sich mit der Fehlerdatenerfasssung befasst.

Prozessorientierte Fehlerbehebung

Produkt- und Prozessorientierung

In der Qualitätssicherung unterscheidet man allgemein produkt- und prozessorientierte Massnahmen. Grundsätzlich muss der Prozess der Softwareentwicklung beherrscht werden, damit eine hohe Produktqualität erreicht werden kann. Die Abnehmer der Programme sind an der Produktqualität interessiert. Die prozessorientierte Qualitätssicherung ist wegen der Neuerungen bei Methoden und Werkzeugen eine langfristige Aufgabe, die schrittweise zu realisieren ist. Case-Werkzeuge werden dazu einen wesentlichen Beitrag leisten. Als erste Massnahme wurde im letzten Abschnitt der Ausbau der Testphase vorgeschlagen. Damit kann das Ziel der Produktefehlerfreiheit weitgehend erreicht werden, wobei allenfalls ein höherer Aufwand für die Fehlerbehebung in Kauf zu nehmen ist. Der Vorteil der Prozessorientierung besteht darin, dass die Fehler phasenweise eliminiert werden, das heisst, dass ein Teil der Fehler früher beseitigt wird. Je früher ein Fehler korrigiert wird, desto billiger. Ein anderer Aspekt ist allerdings ebenso wichtig: in jeder Phase werden bestimmte Arten von Fehlern eliminiert. Dadurch wird der Systemtest entlastet. Das Spektrum der Fehler ist enger. Auf diese Weise erreicht man eine grössere Testtiefe. Schliesslich können bei Konzentration auf den Systemtest konzeptionelle Fehler kaum noch eliminiert werden. Bei phasenweisem Vorgehen gibt es ein Entwurfsreview.

Phase	Massnahme
Anforderungen	Pflichtenheftreview
Entwurf	Entwurfsreview
Implementierung	Modultests
Integration	Systemtests
Betrieb	Fehlerkorrektur

Tabelle III Reviews und Tests nach Phasen

Prozessorientiertes Konzept für nahezu fehlerfreie Programme

Die Reduktion der Fehler muss in allen Phasen der Softwareentwicklung erfolgen. Dazu ist ein Konzept von Reviews und Tests erforderlich. Mängel sollen schon im Pflichtenheft beseitigt werden. Es sind dies Lücken, Widersprüche, Fehler und Unklarheiten. Sie können in späteren Phasen zu Fehlern und, auch wenn sie entdeckt werden, zu Mehraufwand führen. In der Entwurfsphase sind konzeptionelle Mängel besonders schwerwiegend, weil sie in späteren Phasen kaum noch zu beseitigen sind.

Ausgangspunkt ist ein einfaches Phasenmodell. Reviews sind am Schluss der Anforderungs- und Entwurfsphase, Modultests

Vorgehensschritte bei Reviews

1. Der Reviewer übernimmt die Unterlagen.
2. Der Reviewer prüft die Unterlagen.
3. Die gefundenen Mängel werden besprochen.
4. Reviewer und Ersteller analysieren die Ursachen.
5. Der Ersteller überarbeitet sein Programm.
6. Reviewer überprüft die Unterlagen nochmals, allein oder mit dem Ersteller.
7. Die Unterlagen werden freigegeben.

Tabelle IV

nach der Implementierung, der Systemtest nach der Integration vorgesehen (Tab. III), wobei zu beachten ist, dass der Kürze halber nur die Überprüfungsinstrumente (Reviews, Tests) erwähnt sind. Die Überprüfung der Dokumentation soll hervorgehoben werden, denn sie ist für die Fehlerbehebung in der Betriebsphase von grundlegender Bedeutung.

Während Modultests nicht wesentlich vom Systemtest verschieden sind, sind Reviews wesentlich anders geartet. Gelegentlich besteht eine Skepsis gegenüber der Wirksamkeit von Reviews. Sie müssen gut vorbereitet und systematisch durchgeführt werden. Die Vorgehensschritte sind in Tabelle IV angegeben. Werden Mängel in einzelnen Phasen festgestellt, so müssen Massnahmen veranlasst werden, die jeweils von spezifischen Gegebenheiten abhängen. Es ist klar, dass letztlich Faktoren wie professionelles Projektmanagement, Methoden, Werkzeuge usw. die Qualität erhöhen.

Änderungen als Fehlerquelle

Es ist eine Erfahrungstatsache, dass Änderungen jeder Art Fehler erzeugen. Auch die Fehlerbehebung ist mit Änderungen verbunden. Erhebungen haben gezeigt, dass bei Änderungen die Fehlerdichte bis zu fünfmal so hoch liegt, wie bei der Erstprogrammierung. Das ist ein weiteres Argument für die prozessorientierte Fehlerbehebung, wo die Fehlerdichte dauernd tief gehalten wird (Bild 1). Das heisst aber auch, dass Pflichtenheftänderungen minimal gehalten werden müssen. Änderungen produzieren nicht nur Fehler, sondern auch Kosten. Dabei verleitet die Änderbarkeit der Software geradezu dazu, Änderungen in allen Phasen vorzunehmen.

Neben der Fehlerfreiheit gibt es weitere Qualitätsmerkmale, die gewährleistet werden müssen. Dafür ist ein analoges Vorgehen, wie das hier beschriebene, zu entwikkeln. Nach Möglichkeit sind Metriken zu verwenden. Es gibt aber auch Qualitätsmerkmale wie die Robustheit, für die keine Metriken existieren. In diesen Fällen gibt es oft Indikatoren, die Hinweise auf die Merkmalserfüllung geben. Die weitere Entwicklung, die etwa in den USA schon absehbar ist, dürfte in Richtung aussagefähiger Metriken gehen, die allerdings auch höhere Anforderungen an die Fehlerdatenerfassung stellen und einige Erfahrungen voraussetzen. Beispiele solcher Metriken sind die Ausfallrate und der mittlere Ausfallabstand, also Metriken, die für Hardware bereits eingeführt sind. Die Fehlerdatenerfassung wird in Zukunft vermehrt durch Case-Werkzeuge unterstützt.

Literatur

[1] Software Metriken, *R. Schild,* Bulletin SEV/VSE 84(1993)25, S. 15–20.

[2] IEEE Standard 982.2, 1988: Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software (in Überarbeitung). IEEE Standard Office, 345 East 47th Street, New York N.Y. 10017-2394.

Logiciel sans fautes – un but réaliste?

Beaucoup d'appareils contiennent du hardware et du logiciel. Il y a des concepts pour l'assurance de la qualité du hardware, qui sont assez répandus et éprouvés. Après 1985 de nouveaux concepts pour le logiciel ont été élaborés, qui ne sont pas encore très connus. L'état actuel n'est pas tout à fait satisfaisant. Quelques pannes, qui ont gagnées une grande publicité, prouve ce fait. On présente deux méthodes pour diminuer le nombre des fautes dans les programmes, une méthode orientée vers l'assurance de la qualité du produit et une autre pour améliorer le processus du développement du logiciel.