Qualitätssicherung

Andreas Zeller

Lehrstuhl Softwaretechnik
Universität des Saarlandes, Saarbrücken







Diversifizierende Testverfahren

Diversifizierende Testverfahren

- betrachten verschiedene *Programmvarianten*
- erwarten, dass *beobachtbare Abweichungen* zwischen Original und Varianten erzeugt werden.

N-Versionen-Programmierung: Varianten entstehen durch *Mehrfachimplementierung* derselben Spezifikation

Keine Verhaltensabweichungen ⇒ alles ok

Mutationstesten: Varianten entstehen durch systematische Anwendung von *Transformationsregeln*

Keine Verhaltensabweichungen ⇒ mangelnde Güte der Testfälle







N-Versionen-Programmierung

Auch *back-to-back testing* genannt

- Programm wird nach gegebener Spezifikation von N getrennten Teams implementiert und getestet ($N \geq 3$)
- Die Teams dürfen nicht in Kontakt stehen
- Hardware u.U. gleichfalls N-fach beschafft
- Die N Versionen laufen parallel. Bei nicht identischen Resultaten findet Mehrheitsabstimmung statt (u.u. Komparator-Hardware)

Annahme: Programmfehler in den Varianten sind *statistisch unabhängig*







Probleme

- N-fache Kosten
- Grundlegende Annahme nicht gewährleistet:
  - Fehler in der Spezifikation werden von allen Programmierern gleich (falsch) implementiert
  - systematische Fehler: z.B. alle Implementierer haben zufällig denselben falschen Algorithmus gelernt
  - Untersuchungen haben gezeigt, dass Programmfehler *eben nicht* statistisch unabhängig sind







Mutationstesten

Grundideen:

- Programme sind „fast richtig“ (*competent programmer hypothesis*)
- Kleine Änderungen am Programm sollten bei hinreichend umfassenden Testdatensätzen zu beobachtbaren Verhaltensänderungen führen
- Die möglichen kleinen Änderungen (*Mutationen*) kann man systematisch erzeugen und die Mutanten mit den Testdatensätzen füttern
- Ein perfekter Testdatensatz sollte alle Mutanten erkennen
- Anteil aufgedeckter Mutanten kann als Testgütemaß verwendet werden.







Mutationsoperatoren

Berechnungsfehler

- Ändern von arithmetischen Operatoren ($+$ statt $-$)
- Löschen von arithmetischen (Teil-) Ausdrücken
- Ändern von Konstanten

Schnittstellenfehler

- Vertauschen / Ändern von Parametern
- Aufruf anderer Prozeduren eines Moduls







Mutationsoperatoren (2)

Kontrollflussfehler

- Ersetzen von logischen (Teil-) Ausdrücken durch *true* und *false*
- Ändern von logischen und relationalen Operatoren (AND statt OR, $\leq$ statt $<$)
- Aufruf anderer Prozeduren
- Löschen von Anweisungen
- Einfügen von HALT-Anweisungen







Mutationsoperatoren (3)

Initialisierungsfehler

- Ändern von Konstanten
- Löschen von Zuweisungen / Initialisierungsanweisungen

Datenflussfehler

- Durchtauschen von Variablen in einem Sichtbarkeitsbereich
- Änderungen in der Indexberechnung

Jeder Mutant enthält *genau eine* Abweichung!







Validierung der Testsuite

Hauptanwendung von Mutationstesten:

Bestimmen der Güte einer Testsuite

Schon bei kleiner Mutantenzahl muss ein hoher Prozentsatz Mutanten (>90%) entdeckt werden, ansonsten ist die Testsuite zu klein – es fehlen Testfälle, die die Mutanten abdecken.

„Der Testdatensatz hat nur 60% aller Mutanten gekillt. Wir brauchen also noch mehr Testfälle“

Schätzung der Restfehlerzahl

Gesucht: Restfehlerzahl E

Es werden N Mutationen eingeführt

Es werden M Fehler entdeckt; davon seien X Mutanten.

Mithin gilt ungefähr

$$\frac{X}{M} = \frac{N}{E+N}$$

also

$$E = (M - X) \cdot \frac{N}{X}$$

Analogie: Ich setze in einem Teich $N = 100$ markierte Forellen aus. Später fische ich $M = 10$ Forellen; davon sind $X = 2$ markiert. Also schätze ich die Zahl der nicht-markierten Forellen auf $E = (10 - 2) \cdot 100/2 = 400$.

Mit Vorsicht zu genießen, da Werte stark von N abhängen.

Eigenschaften des Mutationstestens

Vorteile

- ✔ Verfahren zur Beurteilung von Testdatensätzen
- ✔ weitgehend automatisierbar
- ✔ explizite Fehlerorientierung
- ✔ Modellierung anderer Verfahren möglich (z.B. Anweisungsüberdeckung, special value testing)

Nachteile

- ✘ es werden nur „einfache“ Fehler erzeugt
- ✘ Es wird angenommen, dass komplexe Fehler Kombinationen einfacher Fehler sind
- ✘ Diese Hypothese ist nicht bewiesen







Cleanroom Software Development

Man kann auch *ganz ohne Testen* Software entwickeln:

Cleanroom Software Development ist ein Organisationsmodell zur Produktion hochwertiger Software (insbesondere hohe Zuverlässigkeit und Wartbarkeit)

Drei getrennte Teams: Spezifikation, Implementierung, Test







Prinzipien

- Inkrementelles Entwurfsmodell: Entwicklung ausführbarer Teilprodukte
- Einsatz formaler Spezifikations-, Entwicklungs- und Validierungsmethoden
- Implementierung *ohne Programmausführung*: Implementierungsgruppe darf lediglich statische Methoden (Inspektion, [Syntax]analyse, formale Verifikation) einsetzen, aber nicht Testen
- Kein Modultest; Integrationstest nach statistischen Verfahren durch unabhängige Testgruppe







Ergebnisse

- Missverständnisse zwischen Entwicklern und Auftraggeber werden seltener (formale Spezifikation erzwingt gedankliche Durchdringung der Anforderungen)
- Inkrementelles Entwurfsmodell führt zu kontinuierlichen Projektfortschritten und besserer Projektsteuerung
- Korrektheit wird nicht „hineingetestet“ sondern „hineinentwickelt“
- Sehr zuverlässige Software bei kaum erhöhten Entwicklungskosten
- In großen Projekten (IBM, NASA) erfolgreich eingesetzt

Qualitätssicherung durch Menschen

Wir betrachten nun Verfahren, in denen *Menschen* die Qualität sichern:

- Eine *Programminspektion* ist ein formales Verfahren, in dem ein Programm durch andere Personen als den Autor auf Probleme untersucht wird.
- *Review* und *Walkthrough* sind vereinfachte Formen der Inspektion.
- *Pair Programming* ist eine Form des kontinuierlichen Gegenlesens.







Ablauf der Programminspektion

1. Eine Inspektion wird durch den Autor ausgelöst, sein Produkt zu überprüfen. Dies geschieht typischerweise zur Freigabe für eine weitere Entwicklungsaktivität.
2. Das Programm wird von mehreren Gutachtern beurteilt, wobei jeder Gutachter sich auf einen oder mehrere Aspekte konzentriert.
3. Jeder Gutachter prüft das Produkt anhand von Referenz-Dokumenten (etwa die Spezifikation) und notiert Erkenntnisse
4. In einer gemeinsamen Sitzung aller Gutachter mit einem ausgebildeten Moderator werden gefundene und neu entdeckte Fehler protokolliert.
   Lösungen werden nicht diskutiert.

Ergebnis der Programminspektion

1. Ergebnis ist ein formalisiertes Inspektionsprotokoll mit Fehlerklassifizierung
2. Außerdem werden Statistiken (*Inspektionsmetriken*) über die Fehlerhäufigkeit erstellt, die zur Verbesserung des Entwicklungsprozesses dienen
3. Der Autor überarbeitet das Produkt anhand der neuen Erkenntnisse.
4. Der Moderator gibt das Produkt frei oder weist es zurück







Details der Programminspektion

- Der Inspektionsaufwand (individuelle Prüfzeiten, Zeit für die gemeinsame Sitzung) muss von vorneherein eingeplant sein
- Inspektionen haben hohe Priorität; d.h. sie sind kurzfristig einzuberufen
- Inspektionsergebnisse dürfen nicht zur Beurteilung von Mitarbeitern eingesetzt werden
- Vorgesetzte und Zuhörer dürfen an den Inspektionen nicht teilnehmen







Was bringt die Programminspektion?

- Prüfaufwand liegt bei 15 bis 20% des Erstellungsaufwands
- 60 bis 70% der Fehler können in einem Dokument gefunden werden
- Nettonutzen bei 20% in der Entwicklung, 30% in der Wartung

„Der *Return on investment* ist bei Inspektionen wesentlich besser als bei anderen Investitionen.“ (Balzert)







Varianten der Programminspektion

Ein *Review* ist eine weniger formale manuelle Prüfmethode (kein definierter Ablauf, informales Inspektionsprotokoll).

- Nutzen bei Fehlersuche ähnlich wie bei formalen Inspektionen
- Verbesserungen im Entwicklungsprozess nur indirekt

Ein *Walkthrough* ist eine weiter abgeschwächte Form, in der der Autor das Prüfobjekt Schritt für Schritt vorstellt. Die Gutachter stellen spontane Fragen, und versuchen so, Probleme zu identifizieren.

- Geringer Aufwand
- aber wesentlich schlechterer Nutzen
- geeignet für „unkritische“ Dokumente

Pair Programming

Im *pair programming* (Paar-Programmierung) arbeiten *zwei* Programmier Seite an Seite an *einem* Rechner, um Programme zu entwerfen, implementieren und zu testen.

Ein Programmierer „fährt“ (hält die Tastatur oder macht Notizen), der andere ist „Beifahrer“ (begutachtet die Arbeit). Der Beifahrer nimmt *aktiv* teil am Geschehen. Diese Rollen werden kontinuierlich gewechselt.

Durch das kontinuierliche Begutachten werden Fehler und Alternativen frühestmöglich erkannt

Kontinuierliches Review-Verfahren!







Ergebnisse des Pair Programming

Erste Studien zeigen, dass sich der doppelte Personalaufwand auszahlt. Paare . . .

- . . . benötigen einen 100% höheren Personalaufwand
- . . . reduzieren die Codierungszeit hingegen um 40–50%
- . . . verbessern die Codequalität (z.B. 94,4% bestandene Testfälle statt 78,1% bei Individuen)







Grundregeln des Pair Programming

- Beide Programmierer sind gleichermaßen verantwortlich für ihre Arbeit.
- Es gibt keine individuelle Schuld (kein „*Du* hast hier aber einen Fehler gemacht“, sondern „*Wir* haben alle Tests bestanden“).
- Weil der Partner dabei ist, muss man sich auf das Ziel konzentrieren (und nicht auf die e-mail, das Chat-Fenster, die Bundesliga-Übertragung. . .)
- *Egoless programming* ist hilfreich (der eigenen Schwächen bewußt sein, keine übermäßige Identifizierung mit dem „eigenen“ Code)
- Pausen einplanen! Pair programming ist anstrengend.

Pair programming ist eine Technik des *extreme programming*.

Verbesserung der Prozessqualität

Wir kommen nun zu den *konstruktiven Verfahren* der Qualitätssicherung:

- Qualitätssicherung mit ISO 9000
- Totales Qualitätsmanagement (TQM)
- Capability Maturity Model (CMM)
- ISO 15504 (SPICE)







Qualitätssicherung mit ISO 9000

Das *ISO 9000*-Normenwerk legt für das Auftraggeber-Lieferantenverhältnis einen allgemeinen, organisatorischen Rahmen zur Qualitätssicherung fest

Das *ISO 9000-Zertifikat* eines Unternehmens bedeutet, dass die eingesetzten Verfahren der ISO 9000-Norm entsprechen.







Wichtige Teile

ISO 9000-1 Allgemeine Einführung und Überblick

ISO 9000-3 Anwendung von ISO 9001 auf Software

ISO 9001 Modelle der Qualitätssicherung in Design/Entwicklung, Produktion, Montage und Kundendienst

ISO 9004 Verbesserung und Aufbau eines Qualitätsmanagement-Systems.







Erforderliche Dokumente in ISO 9000-3

Vertrag Auftraggeber-Lieferant u.a. Annahmekriterien, Problembehandlung, Tätigkeiten des Auftraggebers

Spezifikation u.a. Anforderungen, Leistung, Ausfallsicherheit

Entwicklungsplan u.a. Zielfestlegung, Projektmittel, Entwicklungsphasen, Management, Projektplan

Qualitätssicherungsplan u.a. Qualitätsziele (in messbaren Größen), Vorgaben und Ergebnisse der Entwicklungsphasen, Testmaßnahmen

Testplan u.a. Testfälle, Testdaten, Kriterien für Vollständigkeit

Wartungsplan u.a. Umfang, Unterstützung

Konfigurationsmanagementplan u.a. Werkzeuge, Methoden

Tätigkeiten in ISO 9000-3

Konfigurationsmanagement Identifikation und Rückverfolgbarkeit der Änderungen, Lenkung von Änderungen, Statusberichte

Lenkung der Dokumente

Qualitätsaufzeichnungen

Messungen und Verbesserungen am Produkt und am Prozess

Festlegung von Regeln, Praktiken und Übereinkommen für ein Qualitätssicherungsssytem

Nutzung von Werkzeugen und Techniken, um den Qualitätssicherungs-Leitfaden umzusetzen

Tätigkeiten in ISO 9000-3 (2)

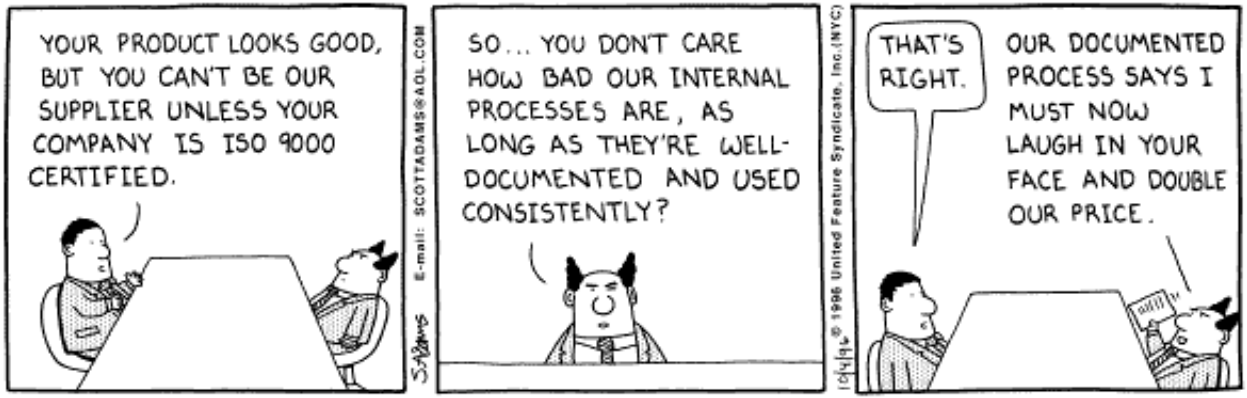
Unterauftragsmanagement

Einführung und Verwendung beigestellter Software-Produkte

Schulung aller Mitarbeiter, die qualitätsrelevante Tatigkeiten durchführen sowie Verfahren zur Ermittlung des Schulungsbedarfs.

Vorsicht

- Normiert sind nur die betrieblichen Abläufe
- Der Einsatz von Qualitätssicherung ist noch keine Garantie für qualitativ hochwertige Produkte



Zertifizierung

Software-Unternehmen, die ein Qualitätsmanagementsystem gemäß diesem Normenwerk besitzen, können sich ein *ISO 9001-Zertifikat* verleihen lassen, das die Qualität der eingesetzten Verfahren bescheinigt.

Alle betroffenen Bereiche eines Unternehmens werden von einer unabhängigen *Zertifizierungsstelle* systematisch daraufhin beurteilt,

- ob die notwendigen Qualitätsmanagementmaßnahmen *festgelegt* sind,
- ob sie *wirksam* sind, und
- ob sie *nachweislich durchgeführt* werden.







Qualitätsauditing

Die Beurteilung heißt *Qualitätsauditing* und findet in Form von *Interviews* statt.

Fragen eines Auditing

- Ist das Qualitätsmanagementsystem hinreichend schriftlich festgelegt und verständlich dargestellt?
  - Gibt es ein Qualitätsmanagement-Handbuch?
  - Welche ergänzenden Dokumente gibt es?
- Besteht eine Verbindlichkeitserklärung für das Qualitätsmanagement-Handbuch?
- Wie werden die Mitarbeiter über die sie betreffenden Regelungen informiert oder geschult?

Das Auditing muss jährlich erneuert werden.

Ein erteiltes Zertifikat eignet sich gut für die Werbung







Totales Qualitätsmanagement (TQM)

Totales Qualitätsmanagement (TQM) ist ein umfassendes Konzept, das das gesamte Unternehmen mit allen Mitarbeitern in die Qualitätsverbesserung einbezieht:

- Japanischer Ansatz
- *Qualität aus der Sicht des Kunden* ist das oberste Ziel
- Die Software-Entwicklung ist eher *kundengetrieben* als *technikgetrieben*







Grundprinzipien des TQM

Prinzip des Primats der Qualität (*Quality first*)

Jeder Mitarbeiter soll seine Arbeit beim ersten Mal und jedes Mal erneut wieder *richtig tun*.

Qualitätsmängel werden sofort beseitigt: Ein Programmierer, der Mängel am Entwurf feststellt, soll die Entwicklung sofort stoppen können.

Prinzip der Zuständigkeit aller Mitarbeiter

Jeder Mitarbeiter versteht Qualität als integralen Bestandteil seiner Arbeit

Keine unabhängige Qualitätssicherungsabteilung







Grundprinzipien des TQM (2)

Prinzip der ständigen Verbesserung (*Kaizen*)

Motto: „Jeder Tag bringt eine konkrete Verbesserung im Unternehmen“

Gefördert durch Team-Arbeit, ständiges Lernen, offenes Klima

Prinzip der Kundenorientierung

Kundennutzen und Kundenzufriedenheit stehen im Mittelpunkt

Entwicklung muss eng mit Marketing und Kundendienst zusammenarbeiten







Grundprinzipien des TQM (3)

Prinzip des internen Kunden-Lieferanten-Verhältnisses

Jeder Mitarbeiter, der für andere Mitarbeiter eine Leistung erbringt, ist ein Lieferant

Der Erfolg eines Teams wird gemessen an der Zufriedenheit der internen wie auch externen Kunden

Prinzip der Prozessorientierung

Fehler werden als Defizite des Entwicklungsprozesses betrachtet

Produktprüfung dient zur Überprüfung der Prozessqualität







TQM in der Praxis – bei der Firma Bosch

12 Leitsätze zur Qualität

1 Wir wollen zufriedene Kunden. Deshalb ist hohe Qualität unserer Erzeugnisse und unserer Dienstleistungen eines der obersten Unternehmensziele. Dies gilt auch für Leistungen, die unter unserem Namen im Handel und im Kundendienst erbracht werden.

2 Den Maßstab für unsere Qualität setzt der Kunde. Das Urteil des Kunden über unsere Erzeugnisse und Dienstleistungen ist ausschlaggebend.

3 Als Qualitätsziel gilt immer »Null Fehler« oder »100% richtig«.

4 Unsere Kunden beurteilen nicht nur die Qualität unserer Erzeugnisse, sondern auch unserer Dienstleistungen. Lieferungen müssen pünktlich erfolgen.

5 Anfragen, Angebote, Muster, Reklamationen usw. sind gründlich und zügig zu bearbeiten. Zugesagte Termine müssen unbedingt eingehalten werden.

6 Jeder Mitarbeiter des Unternehmens trägt an seinem Platz zur Verwirklichung unserer Qualitätsziele bei. Es ist deshalb Aufgabe eines jeden Mitarbeiters, vom Auszubildenden bis zum Geschäftsführer, einwandfreie Arbeit zu leisten. Wer ein Qualitätsrisiko erkennt und dies im Rahmen seiner Befugnisse nicht abstellen kann, ist verpflichtet, seinen Vorgesetzten unverzüglich zu unterrichten.

7 Jede Arbeit sollte schon von Anfang an richtig ausgeführt werden. Das verbessert nicht nur die Qualität, sondern senkt auch unsere Kosten. Qualität erhöht die Wirtschaftlichkeit.

8 Nicht nur die Fehler selbst, sondern die Ursachen von Fehlern müssen beseitigt werden. Fehlervermeidung hat Vorrang vor Fehlerbeseitigung.

9 Die Qualität unserer Erzeugnisse hängt auch von der Qualität der Zukaufteile ab. Fordern Sie deshalb von unseren Zulieferern höchste Qualität und unterstützen Sie diese bei der Verfolgung der gemeinsamen Qualitätsziele.

10 Trotz größter Sorgfalt können dennoch gelegentlich Fehler auftreten. Deshalb wurden zahlreiche erprobte Verfahren eingeführt, um Fehler rechtzeitig entdecken zu können. Diese Methoden müssen mit größter Konsequenz angewendet werden.

11 Das Erreichen unserer Qualitätsziele ist eine wichtige Führungsaufgabe. Bei der Leistungsbeurteilung der Mitarbeiter erhält die Qualität der Arbeit besonderes Gewicht.

12 Unsere Qualitätsrichtlinien sind bindend. Zusätzliche Forderungen unserer Kunden müssen beachtet werden.

Das Capability Maturity Model (CMM)

Referenzmodell für die *Beurteilung von Software-Lieferanten*

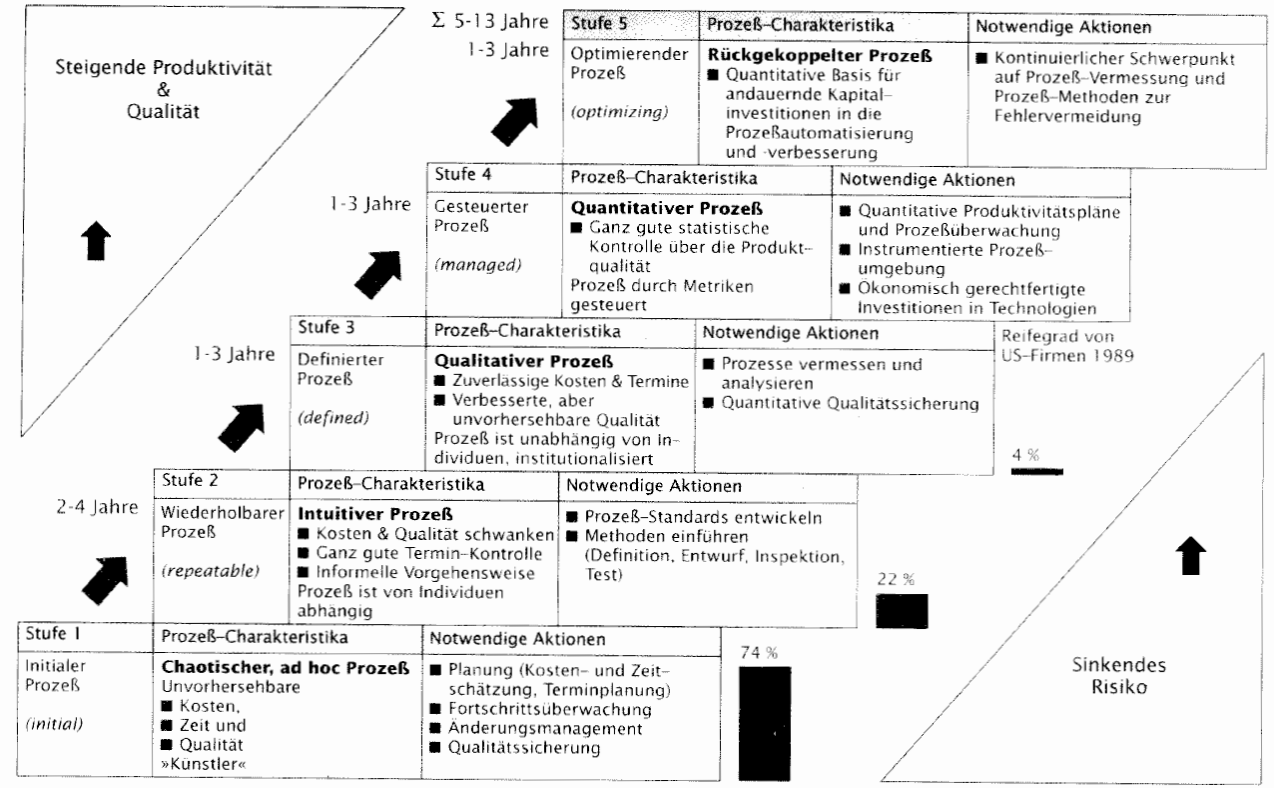
- 1987 von der *Carnegie Mellon University* auf Grund eines Fragebogens entwickelt
- Teilt Software-Entwicklungsprozesse in fünf *Reifegrade* ein
- Jede Qualitätsstufe beschreibt einen bestimmten Reifegrad (*maturity*) im Entwicklungsprozess
- Jede Stufe setzt voraus, dass die Anforderungen der unteren Stufen erfüllt sind



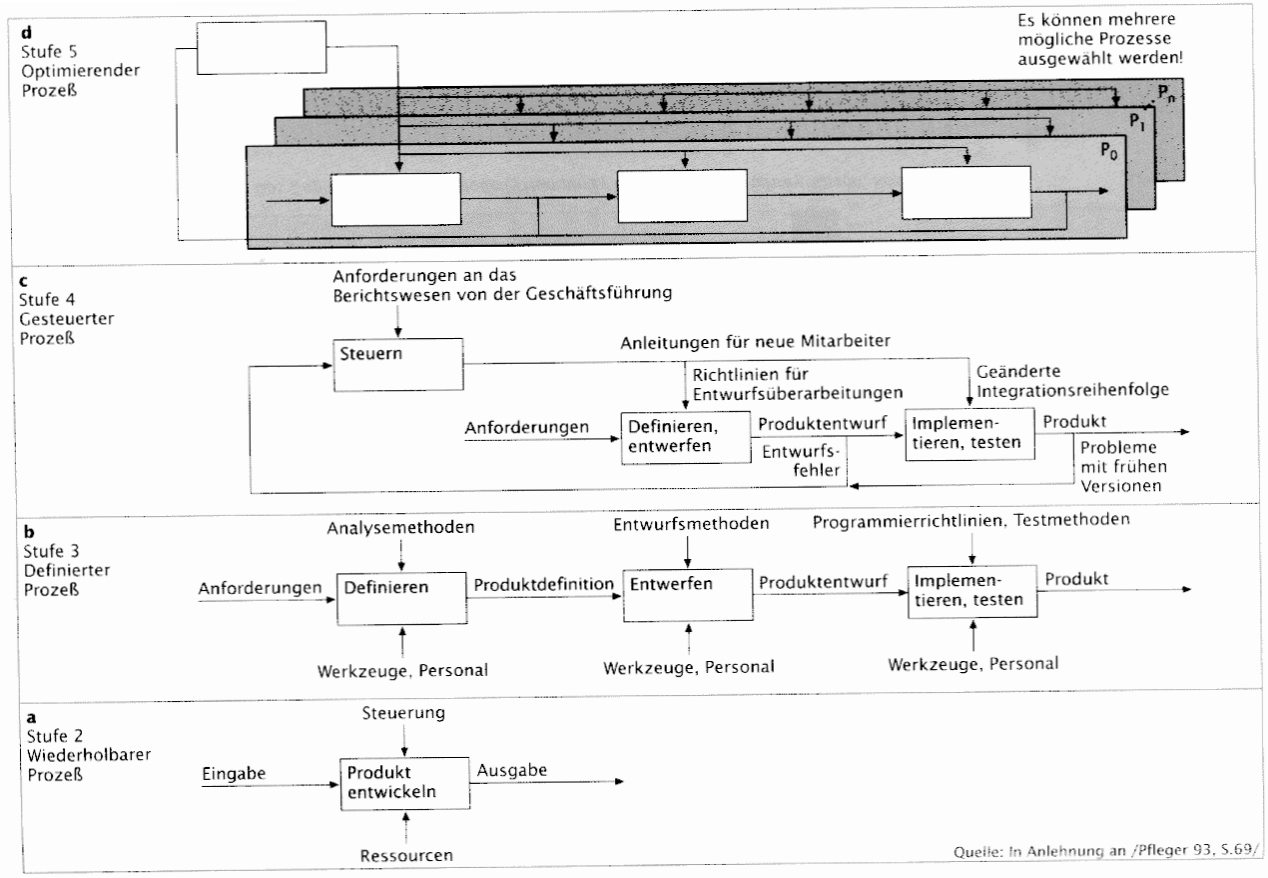




Stufen des CMM



Prozesse des CMM



Stufen des CMM (2)

Von unten nach oben:

- Ein Prozess auf Stufe 1 ist *chaotisch*; er kann nicht dargestellt werden
- Auf Stufe 2 sind immerhin bereits *strukturierte Anforderungen* an den Prozess definiert
- Auf Stufe 3 ist der Prozess bereits klarer definiert; es gibt *individuelle Prozessaktivitäten*
- Auf Stufe 4 gibt es eine zentrale Steuerung, die die Prozessmaße als *Rückkopplung* enthält
- Auf Stufe 5 werden diese Prozessmaße benutzt, um den Prozess *dynamisch* in Abhängigkeit vom Entwicklungsfortschritt zu ändern







Assessment

Ausgangspunkt für eine Prozessverbesserung ist ein *Assessment*, eine Befragung der Mitarbeiter aus Management, Entwicklung und Qualitätssicherung.

Es wird sowohl die dokumentierte Prozessdefinition als auch ihre Umsetzung in die Praxis bewertet







Vom Chaos auf Stufe 2

Anforderungsmanagement Gemeinsames Verständnis zwischen Kunde und Projektteam über die Anforderungen herstellen

Projektmanagement Projektpläne einführen

Projektverfolgung und -Überwachung Transparenter Entwicklungsfortschritt, um frühzeitig Korrekturmaßnahmen einzuleiten

Unterauftragsmanagement Qualifizierte Unterlieferanten auswählen und effektiv steuern und überwachen

Qualitätssicherung Transparenter Prozess und transparente Produkte

Konfigurationsmanagement Integrität der Produkte während ihrer Lebenszyklen sicherstellen







ISO 15504 (SPICE)

Die ISO-Norm 15504 (*SPICE – software process improvement and capability determination*) ist *vergleichbar mit dem CMM-Modell*.

ISO 15504

- bietet gemeinsame Basis für verschiedene Modelle zur Prozessverbesserung
- definiert Mindestanforderungen für Standortbestimmungen (*Assessments*).
- berücksichtigt explizit Kundenorientierung
- orientiert sich an CMM und ISO 9000







SPICE-Reifegrade

In ISO 15504 gibt es ähnliche *Reifegrade* wie bei CMM

Der Reifegrad bezieht sich jedoch auf *Prozesse* statt auf Unternehmen

Zusätzlicher Reifegrad „Durchgeführter Prozess“ – zwischen CMM-Stufe 1 und CMM-Stufe 2:

- Der *Zweck* des Prozesses wird *erfüllt* (im Gegensatz zum chaotischen Prozess)
- Der Prozess wird nicht nicht geplant oder gesteuert
- Besonders für kleine Organisationen sinnvoll







Verfahren im Vergleich: ISO 9001

Der Schwerpunkt der *ISO 9001*-Zertifizierung ist der Nachweis eines Qualitätsmanagementsystems entsprechend der Norm.

- Solide Grundlage für Verhältnis Auftraggeber-Lieferant
- Rein technisch orientiert
- Im wesentlichen statisch







Verfahren im Vergleich: CMM

CMM konzentriert sich auf die Qualitäts- und Produktivitätssteigerung des gesamten Software-Entwicklungsprozesses.

- Speziell auf Software-Technik ausgelegt
- Schwerpunkte in Qualitätssicherungssystem und Metriken
- Technische Ansätze, Prozessdefinition und Metriken helfen auch, die ISO 9001-Zertifizierung zu erlangen.
- Dynamisch: ständige Verbesserung
- Primat der Qualität und Kundenorientierung sind unterrepräsentiert
- Nachfolger ISO 15504 integriert CMM und ISO 9000

Verfahren im Vergleich: TQM

TQM ist eine ganzheitliche, umfassende Unternehmensphilosophie, die *Qualität aus der Sicht des Kunden* als oberstes Ziel verfolgt.

- Berücksichtigt soziale Aspekte
- ISO 9000 und CMM sind Bausteine im TQM-Ansatz
- Subsumiert alles einschließlich klassischer Management-Aufgaben
- Nicht so konkret fassbar wie ISO 9000 oder CMM







Qualitätssicherung: Ein Fazit