Systementwicklung

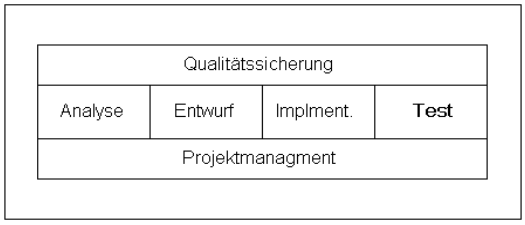
Qualitätssicherung

- eine Einführung -

Uwe H. Suhl und Chris Bizer
SS 2008

Qualitätssicherung

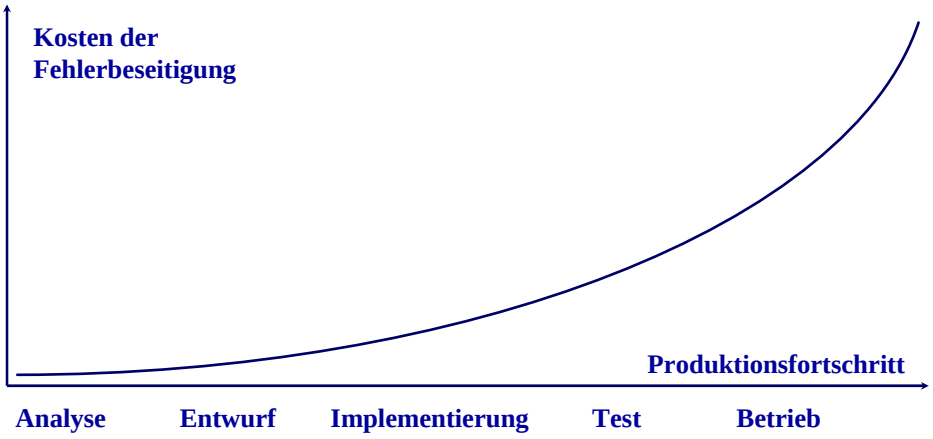
- **erweitert die Sicht des Entwicklers um eine zusätzliche Dimension**



- **Qualitätssicherungsmaßnahmen beschränken sich nicht auf einzelne Phasen.**
- **Qualitätssicherung besteht aus einer Vielzahl von Aufgaben und Tätigkeiten.**
- **QS garantiert nicht die Herstellung hochwertiger Produkte aber die Wahrscheinlichkeit steigt.**

Warum hat Qualitätssicherung eine sehr hohe Bedeutung?

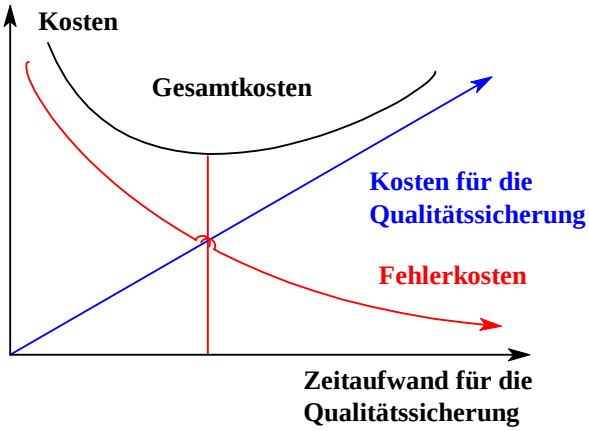
- **Fundamentalprinzip: Fehlervermeidung statt Fehlerbehebung!**



- **Qualitätssicherung ist eine projektübergreifende Aufgabe, die in allen Entwicklungsphasen für alle Artefakte relevant ist**

Fundamentale Zielkonflikte bei der Qualitätssicherung

- **Kosten die durch Softwarefehler entstehen**
- **Kosten die durch Qualitätssicherung entstehen**



Grundlagen des Qualitätsmanagements

- **Definitionen und Begriffe**
  - ***Qualität*: Gesamtheit von Merkmalen einer Einheit (Produkt, Tätigkeit) bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen (ISO 8402)**
  - ***Qualitätssicherung* (QS): Alle geplanten und systematischen Tätigkeiten, die notwendig sind, um ein angemessenes Vertrauen zu schaffen, dass ein Produkt oder eine Dienstleistung gegebene Qualitätsanforderungen erfüllt (DIN)**
  - ***Software-Qualität*: ist die Gesamtheit der Qualitätsmerkmale und deren Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen. (DIN ISO 9126).**
  - ***Qualitätsmanagement* (QM): Alle Tätigkeiten der Gesamtführungsaufgabe, welche die Qualitätspolitik, Ziele und Verantwortlichkeiten festlegen sowie diese durch Mittel wie Qualitätsplanung, Qualitätslenkung, Qualitätsprüfung und Qualitätsverbesserung im Rahmen des Qualitätsmanagementsystems verwirklichen (ISO 8402)**
  - ***Qualitätsmanagementsystem* (QMS): Die Struktur, Verantwortlichkeiten und Mittel zur Verwirklichung des Qualitätsmanagements (ISO 8402)**

Software-Qualitätsmerkmale

- **Man unterscheidet sechs fundamentale Qualitätsmerkmale: *Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit***
- **Leider sind diese Merkmale schlecht quantifizierbar und subjektiv!**

1. ***Funktionalität*: die im Pflichtenheft festgelegten Anforderungen werden durch definierte Funktionen erfüllt. Im einzelnen:**
   - ***Richtigkeit*: Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen, z.B. die benötigte Genauigkeit von berechneten Werten.**
   - ***Angemessenheit*: Eignung der Funktionen für spezifizierte Aufgaben, z.B. aufgabenorientierte Zusammensetzung von Funktionen aus Teilfunktionen.**
   - ***Interoperabilität*: Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken.**
   - ***Ordnungsmäßigkeit*: Erfüllung anwendungsspezifischer Normen, Vereinbarungen, gesetzlichen Bestimmungen und ähnlichen Vorschriften.**
   - ***Sicherheit*: Fähigkeit, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern.**

Software-Qualitätsmerkmale (2)

2. ***Zuverlässigkeit*: Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren. Im einzelnen:**
   - ***Reife*: Geringe Versagenshäufigkeit durch Fehlzustände.**
   - ***Fehlertoleranz*: Fähigkeit, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren.**
   - ***Wiederherstellbarkeit*: Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen, wobei die dafür benötigte Zeit und der benötigte Aufwand zu berücksichtigen sind.**
3. ***Benutzbarkeit*: erforderlicher Benutzungsaufwand und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe. Im einzelnen:**
   - ***Verständlichkeit*: Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen.**
   - ***Erlernbarkeit*: Aufwand für den Benutzer, die Anwendung zu erlernen (z.B. Bedienung, Ein-, Ausgabe).**
   - ***Bedienbarkeit*: Aufwand für den Benutzer, die Anwendung zu bedienen.**

Software-Qualitätsmerkmale (3)

4. **Änderbarkeit: erforderlicher Änderungsaufwand. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen und der funktionalen Spezifikationen einschließen.**
   - ***Analysierbarkeit*: Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.**
   - ***Modifizierbarkeit*: Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen.**
   - ***Stabilität*: Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen.**
   - ***Prüfbarkeit*: Aufwand, der zur Prüfung der geänderten Software notwendig ist.**
5. ***Effizienz*: Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen. Im einzelnen:**
   - ***Zeitverhalten* : Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung.**
   - ***Verbrauchsverhalten*: Anzahl und Dauer der benötigten Betriebsmittel für die Erfüllung der Funktionen.**

Software-Qualitätsmerkmale (4)

6. ***Übertragbarkeit*: Eignung der Software, von einer Umgebung in eine andere übertragen zu werden. Umgebung kann organisatorische Umgebung, Hardware- oder Software- Umgebung einschließen. Im einzelnen:**
   - ***Anpassbarkeit*: Möglichkeiten, die Software an verschiedene, festgelegte Umgebungen anzupassen, wenn nur Schritte unternommen oder Mittel eingesetzt werden, die für diesen Zweck für die betrachtete Software vorgesehen sind (Teilbereich ist die *Portabilität*).**
   - ***Installierbarkeit*: Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist.**
   - ***Konformität*: Grad, in dem die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllt.**
   - ***Austauschbarkeit*: Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand.**

Grundsätze

- **Qualität muss *erzeugt* werden, sie kann *nicht erprüft* werden!**
- **Qualität bezieht sich immer sowohl auf die *Artefakte* wie auf die *Prozesse* zur Herstellung dieser Artefakte.**
- **Die *Qualitätsverantwortung* liegt im Projekt d.h. bei *Führungskräften* und *Entwicklern*. Der Projektleiter braucht einen Gegenpart ohne Termin- und Kostenverantwortung.**
- **Das *Qualitätswesen* ist verantwortlich für die *Ermittlung (Messung) der Qualität*. Es erbringt Dienstleistungen in allen Belangen der Qualität sowohl für die Entwickler wie für die Führungskräfte.**
- **Das Qualitätswesen muss einen unabhängigen Berichterstattungspfad haben, der bis zur Geschäftsleitung reicht..**
- **Die Entwickler müssen über die Qualität ihrer Arbeit informiert werden.**
- ***Nur das Projekt schafft die Qualität, das QMS liefert nur die Leitlinien.***

Qualitätsmanagement

- **Man unterscheidet folgende Qualitätsmanagement -Verfahren:**
  - **Qualitätsplanung: Definition der Qualitätsziele (Das wollen wir erreichen!)**
    - ✓ Aufbau des Qualitätsmanagementsystems und **Dokumentation** im Qualitätshandbuch
    - ✓ die Festlegung von Qualitätsmerkmalen (**QM-Plan**)
  - **Qualitätslenkung: Konstruktive Maßnahmen (So müssen wir arbeiten!)**
    - ✓ Festlegung von *Entwicklungsmethoden*, die das Vorgehen definieren
    - ✓ *Werkzeuge*, die den Einsatz von Methoden und Sprachen unterstützen.
    - ✓ Projekt-Rollenverteilung (Aufstellung)
  - **Qualitätsprüfung: Analytische Maßnahmen (Haben wir richtig gearbeitet?)**
- **Qualitätswesen: ist eine Unternehmensabteilung an deren Spitze ein *QualitätsmanagerIn* steht - berichtet direkt an die Geschäftsleitung**
  - ***QualitätsmanagerIn:*** implementiert die Qualitätspolitik in der Organisation
    - ✓ Ziel ist ständige Qualitätsverbesserung, Verantwortlich für QMS (Zertifizierungen, Dökumentation, ..)
  - ***Qualitätsbeauftragte:*** ist für das QMS in einem Projekt verantwortlich
    - ✓ Erstellt den QM-Plan
    - ✓ Überwacht die Durchführung der QS-Maßnahmen

Qualitätsplanung

- ***Qualitätshandbuch:* Beschreibung des Qualitätsmanagementsystems**
  - **Organisation**
  - **Maßnahmen**
- ***Verfahrenshandbuch:* Beschreibung der zu benutzenden Verfahren**
- ***Software-Qualitätsplan:* enthält die Vorgaben für die Entwicklung von Produkten**:
  - **die *Qualitätsziele* für das Projekt**
    - ✓ **wie hoch muss meine Qualität überhaupt sein**
    - ✓ **was heißt Qualität in diesem Projekt (Anforderungen)**
    - ✓ **was ist besonders wichtig**
    - ✓ **Was wird geprüft?**
    - ✓ **Wie werden die Kriterien geprüft?**
    - ✓ **Wer führt die QS-Maßnahmen durch?**
  - **Einfluss auf *Budget* und *Termine***
  - **die *Organisation* im Projekt**
  - **die Festlegung der Termine von Audits**

Qualitätssicherung

- **sind alle geplanten und systematischen Tätigkeiten, die notwendig sind, um ein angemessenes Vertrauen zu schaffen, dass ein Produkt (Dienstleistung) die gegebenen Qualitätsanforderungen erfüllt.**
- **Man unterscheidet die *analytische* und die *konstruktive* Qualitätssicherung**
- **Die konstruktive Qualitätssicherung betrifft:**
  - **Projektleitung, Projektmanagement, eingesetzte Entwicklungsmethoden, Erfahrungsaustausch, Ausbildungsmaßnahmen,…**
- **die analytische Qualitätssicherung (auch Qualitätsprüfung) stellt fest, ob ein Artefakt den Vorgaben entspricht und ist eine permanente, die Entwicklung begleitende Aufgabe! Sie betrifft die:**
  - **Prozesse und Artefakte (Software und Dokumente)**
  - **Maßnahmen: analytische Beweise (nicht praktikabel), Tests, Reviews, Audits, Simulation**

Qualitätssicherung: Definitionen und Begriffe

- **Zwei Begriffe sind von zentraler Bedeutung: *Validierung* und *Verifikation***
- ***Validierung (Gültigkeit):* 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 beschäftigt sich also mit der Frage: Ist das Softwaresystem für die geplante Anwendung geeignet?**
  - **Validierung beantwortet also die Frage: *Entwickeln wir das richtige System?***
- ***Verifikation:* 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.**
  - **Verifikation beantwortet die Frage: *Entwickeln wir das System richtig?***
  - **Dieser Begriff wird auch benutzt für den Prozess des formalen Beweisens der Korrektheit eines Programms. (IEEE 610.12).**

Methoden der analytischen Qualitätssicherung

- ***Testen*: ist die Überprüfung eines Software-Bausteins, ob dieser bei gegebenen Eingaben die erwarteten Resultate liefert: man unterscheidet *Modul- bzw. Klassentests, Subsystemtests* und *Integrationstests* (Gesamtsystem)**
- ***Review*: Zusammenkunft von Personen zur inhaltlichen oder formellen Überprüfung eines Artefaktes nach vorgegebenen Prüfkriterien; Ziele sind *Schwachstellen, Mängel* und *Stärken* aufzuzeigen**
  - **Ein *Walkthrough* ist ein Review, bei dem die AutorIn die Funktionsweise eines Artefaktes detailliert beschreibt und die Gutachter bei Mängeln einhaken.**
- ***Audit*: ist die regelmäßige Überprüfung, ob das Qualitätsmanagement-system wie geplant funktioniert.**
- ***Simulation*: modelliert einen Aspekt eines Systems und überprüft sein Verhalten.**
- ***Prototyp*: ist eine Vorab-Implementierung eines kritischen Systemteils.**

Analytische QS: Tests

- **Einzige praktikable Methode zur *Verifikation* der erbrachten Leistung**
- ***Testfall*: Inputdaten mit Spezifikation des zu erwarteten Resultates**
- ***Testplan*: die Testfälle werden vorab spezifiziert**
- **Systemtests erfordern eine Balance zwischen Testaufwand und Fehler-freiheit des Systems vor dem Einsatz, d.h. Kosten / Nutzen-Analyse**
- **zwei Test-Extrema können unterschieden werden:**
  1. **Test der externen Spezifikationen bzw. Schnittstellen (System als black box)**
  2. **Testdaten werden an Hand der internen Systemarbeitsweise entworfen, d.h. unter Einbeziehung der Programmlogik, wobei möglichst viele Programmzweige getestet werden**
  - **Vorgehensweise 1. erscheint erstrebenswerter, ist jedoch nicht praktikabel, da hier eine extrem große Anzahl an Testdaten benötigt wird**
  - **Testfälle müssen daher unter Berücksichtigung der internen Architektur konstruiert werden und zwar so, dass möglichst viele Fehler entdeckt werden (Klassenbildung)**
- **Techniken zum Testen von einzelnen Modulen / Klassen werden später thematisiert**

Analytische QS: Tests (2)

- nur die kreativsten (destruktivsten) MA sollen testen! Eigene Programme sollten niemals *verantwortlich* selbst getestet werden!
- Erfahrung: je mehr Fehler in einem Softwarebaustein gefunden werden, desto größer ist die Wahrscheinlichkeit der Existenz weiterer unentdeckter Fehler!
- Systembausteine (Moduln, Klassen, Subsysteme) werden zunächst einzeln getestet; man unterscheidet:
  - Prozedurverifikation: neben Tests erfolgt auch ein Code-Walkthrough durch Dritte
  - Modultest bzw. Klassentests mit Code-Walkthrough
  - Integrationstest nach erfolgreicher Zusammenführung aller Systemkomponenten mit eigenen Testdaten erfolgt und vom Projektteam bzw. einem Testteam durchgeführt wird
  - Abnahmetest (Akzeptanztest) mit echten Daten von Anwendern; bei Bestehen wird das System in den Betrieb übergeleitet
- Zwei Testphilosophien: Bottom-Up und Top-Down-Tests
  - Top-Down-Test: beginnt auf Schichten-Ebene, wobei Moduln zunächst nur als Stümpfe (Stubs), d.h. durch ihre Schnittstellenspezifikation repräsentiert werden
  - Moduln werden getestet, indem die Prozeduren durch Stümpfe ersetzt werden
  - jeder Test besteht aus der Ausführung einer Reihe von Testfällen
- schwierige Entscheidung: wann ist Testen zu beenden?

Übungsaufgabe

- Aus G.J. Myers: Methodisches Testen von Programmen. 5. Auflage, R. Oldenbourg Verlag, München 1995
- Ein Modul ist zu testen, der 3 ganzzahlige Werte einliest und als Längen eines Dreiecks interpretiert. Der Modul liefert einen Return Code zurück, der folgende Werte annehmen kann:
  - -2 syntaktisch fehlerhafte Eingabedaten
  - -1 kein Dreieck
  - 0 ein allgemeines Dreieck
  - 1 ein gleichschenkliges Dreieck
  - 2 ein gleichseitiges Dreieck
  - 3 ein rechtwinkliges Dreieck (ein Winkel ist 90°)
  - 4 ein rechtwinkliges Dreieck, das auch gleichseitig ist
- Entwickeln Sie einen Testplan mit Testfällen, die möglichst alle denkbaren Szenarien abbilden, um zu testen ob der Modul korrekt implementiert wurde!
- Zentrale Frage: Notwenige und hinreichende Bedingung wann (a,b,c) ein Dreieck bilden!
- Weitere Frage: Wann ist ein Dreieck rechtwinklig?
- Welche Tests müsste man zusätzlich machen, wenn die Längen auch rationale bzw. reelle Zahlen sein dürfen (Speicherung als Currency bzw. Floating Point Format)?
- Wie ändern sich die Tests, wenn ein Dreieck über 3 Koordinaten (x,y) def. wird?

Lösung der Aufgabe:

1. **kein Dreieck (1,0,3): eine Seitenangabe = 0 (alle Permutationen)**
2. **kein Dreieck (1,-5,9): eine Seitenangabe < 0 (alle Permutationen)**
3. **kein Dreieck (1,2,3): Summe der beiden kürzeren Seiten = 3. Seitenlänge (alle Permutationen**
4. **kein Dreieck (1,2,4) Summe der beiden kürzeren Seiten < 3. Seitenlänge (alle Permutationen)**
5. **kein Dreieck oder Fehlermeldung (0,0,0) alle Seiten = 0 (2 Seiten = 0?)**
6. **zulässiges ungleichseitiges Dreieck (2,3,4)**
7. **zulässiges gleichseitiges Dreieck (2,2,2)**
8. **zulässiges gleichschenkliges Dreieck (2,2,1)**
9. **weiterer Testfälle mit Permutation für gleichschenklige Dreiecke (1,2,2)**
10. **weiterer Testfälle mit Permutation für gleichschenklige Dreiecke (2,1,2)**
11. **Test mit maximalen Werten (Max_int, Max_int, Max_int) usw.**
12. **Test auf rechtwinkliges Dreieck: (3,4,5)**

Analytische QS : Reviews

- **Planung:** Reviews müssen in der Entwicklung eingeplant werden.
- **Vorbereitung:** Die Teilnehmer an einem Review erhalten die zu prüfenden Artefakte sowie die Referenzunterlagen und bereiten sich individuell auf die Sitzung vor.
- **Review-Sitzung:** wird von einem Moderator geleitet. Er sorgt für einen geordneten Sitzungsablauf und wacht über die Einhaltung der Regeln.
- **Review-Protokoll / Bericht:** fasst die Ergebnisse der Sitzung zusammen.
- **Überarbeitung und Nachkontrolle:** Der Projektleiter entscheidet auf Grund des Review-Berichtes über die durchzuführenden Änderungen.
- **Der Moderator:** Organisiert, leitet die Sitzung und erstellt den Bericht
- **Der Protokollant:** Erstellt das Review-Protokoll, aus dem dann der Review-Bericht erstellt wird
- **Der Autor des Artefaktes: k**lärt Unklarheiten und Missverständnisse und behebt die festgestellten Mängel

Analytische QS: Review - Regeln

- Das zu prüfende Material wird rechtzeitig vor der Sitzung verteilt (keine Tischvorlage) und die Teilnehmer werden rechtzeitig eingeladen.
- Alle kommen vorbereitet zur Sitzung. Der Moderator bricht das Review ab, wenn mehrere Teilnehmer nicht vorbereitet sind.
- An einem Review nehmen 3-7 Personen teil.
- Eine Sitzung dauert maximal 2 Stunden; entsprechend muss der Umfang des zu prüfenden Artefaktes gewählt werden:
  - **Richtwerte: für Code-Reviews ca. 150-200 Codezeilen**
  - **für Dokument-Reviews ca. 25 Seiten pro Sitzung.**
- Die Probleme werden in der Sitzung identifiziert aber nicht gelöst.
- Stilfragen werden nicht diskutiert.
- Das Artefakt wird bewertet, jedoch nicht dessen Autor.
- Häufig werden bei Reviews auch Fehler in den Referenzunterlagen gefunden. Diese werden auf separat notiert.

Qualitätsmodelle und Normen

ISO 9000f (9001-9004)

- **Verbreitet in Europa und USA**
- **Gilt für materielle und immaterielle Produkte wie Dienstleistungen und Software**
- **9000: Einführung in die Normenfamilie**
  9000-3 zusätzliche Richtlinien für Softwareprozesse
- **9001: Gilt für Organisationen, die Produkte im gesamten Lebenszyklus betreuen**
- **9002: Gilt für Organisationen, die Produkte von Produktion bis Wartung betreuen**
- **9003: Gilt für Organisationen die Produkte fertig stellen und Qualität durch Endprüfung sichern**
- **9004: Leitfaden für Aufbau eines QM-Systems Anforderungen wie Produktsicherheit, Wirtschaftlichkeit...**

Audit

- **ist eine Aktivität, bei der sowohl die Angemessenheit und Einhaltung vorgegebener Vorgehensweisen, Anweisungen und Standards als auch deren Wirksamkeit und Sinnhaftigkeit geprüft werden (ANSI- Norm N45.2.10-1973).**
- **Audit versus Review**

Review

- **Durchführung von Personen die in den Entwicklungsablauf integriert sind**
- **Dauert meist nur einige Stunden**

Audit

- **Durchführung von Personen die *nicht* an der Entwicklung beteiligt sind**
- **Mitunter monatelanger Prozess**