Definition Qualität
I. A. Synonym für ein hochwertiges Produkt o. Dienst

Garvins Definitionsansatz der Qualität
1. *Transzendente Ansatz*
- Einzigartig & absolut. Nicht präzise messbar
- Beruht auf Erfahrung & steht für kompromisslose Produktstandards

2. Produktbezogene Ansatz
- Präzise messbare Größe
- Q-unterschiede spiegeln sich in Differenzen von Eigenschaften wieder
- Abbildung anhand von objektiver Merkmale

3. Anwenderbezogene Ansatz
- Liegt im Auge des Betrachters
- Höchste Q. wenn Verbraucherbedürfnisse am besten erfüllt

4. Prozessbezogene Ansatz
- Einhaltung von Spezifikationen
- Abweichungen von Spezifikation -> Verminderung Q.

5. Kosten-Nutzenbezogene Ansatz
- Q. ist hoch, wenn das Erzeugnis eine bestimmte Leistung zu einen akzeptablem Preis bietet

Definition Qualität (ISO9000)
„Der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt“
Inhärente Merkmale: Eigenschaften einer Einheit, die deren Beschaffenheit ausmachen. (Bsp. Funktionalität, Speicherverbrauch)

$$Q = \frac{erfüllter\ Anteil\ der\ Anforderungen}{Gesamheit\ der\ Anforderungen}$$

Definition Qualitätsmanagement
“Qualitätsmanagement umfasst alle aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bezüglich Qualität.”

Qualitätsmanagement
1. Qualitätspolitik
Gibt Unternehmensrichtung vor. Legt strategischen Handlungsrahmen fest

2. Qualitätsziele
Festlegen von nachvollziehbaren und quantifizierbaren Qualitätszielen. Ermittlung der Qualitätsanforderungen.

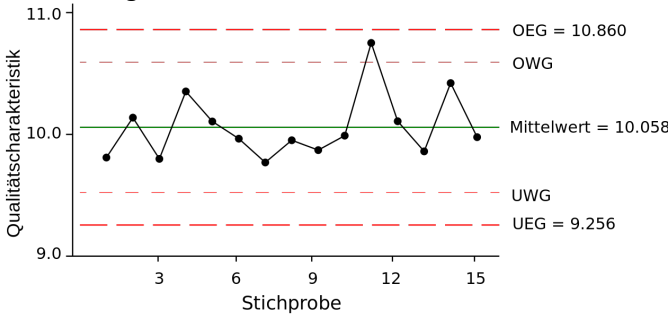
3. Qualitätsplanung
Detaillierte Ermittlung der Qualitätsanforderungen aus den Qualitätszielen. Definieren der Prozesse und Ressourcen, welche die angestrebten Qualitätsziele ermöglichen sollen.

4. Qualitätslenkung
Beinhaltet sämtliche vorbeugende, überwachende und korrigierende Tätigkeiten zur Erfüllung der Qualitätsziele.

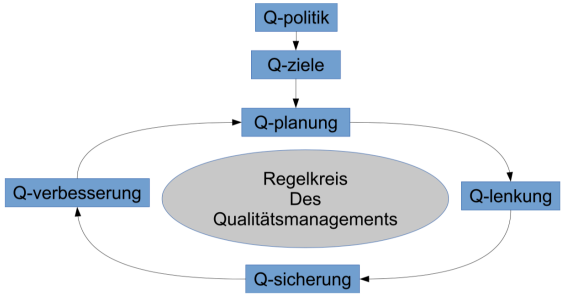
5. Qualitätssicherung
Umfasst alle Tätigkeiten, die Vertrauen schaffen, dass die Qualitätsziele erfüllt werden.

6. Qualitätsverbesserung
Umfasst Tätigkeiten, die den Qualitätsmanagementprozess aus den Erfahrungen heraus verbessern.

Qualitätsregelkarte



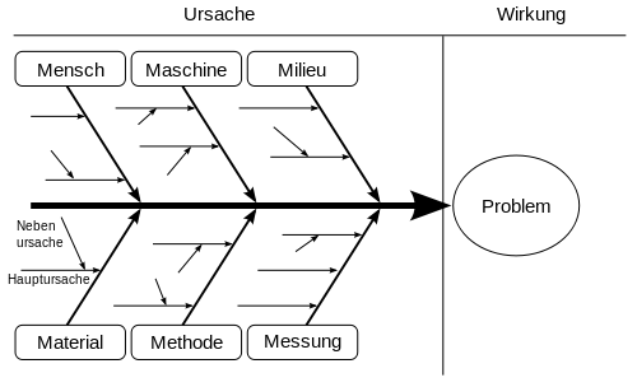
Regelkreis des Qualitätsmanagements (PDCA Zyklus)



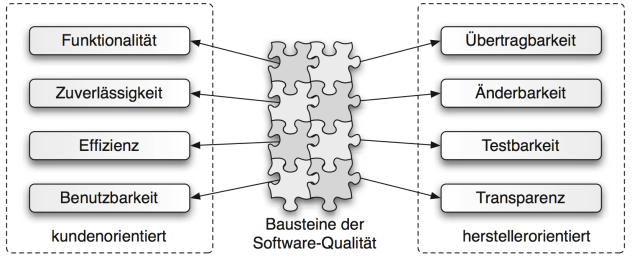
Grundsätze des Qualitätsmanagements
- Q. erzeugen nicht erprüfen
- Q. bezieht sich immer auf Produkte & Prozesse
- Q-verantwortung untrennbar verbunden mit Sach-, Termin- & Kostenverantwortung
- Q-wesen erbringt Dienstleistung & ist verantwortlich für Ermittlung (Messung) der Q.
- Q-wesen benötigt unabhängigen Berichterstattungs-pfad bis zur Geschäftsführung
- Mitarbeiter müssen über die Q. ihrer Arbeit orientiert werden

Totales Qualitätsmanagement
Führungsmethode mit Kundenzufriedenheit als oberstes Unternehmensziel. Qualität im Mittelpunkt des Unternehmens. Alle Mitarbeiter sind ins Qualitätsmanagement einbezogen.

Fischgräten-Diagramm (Ishikawa Diagramm)



Qualitätsziele nach ISO9126



sichtbare Seite — nicht sichtbare Seite

(Eisberg-Modell)

Qualitätsziele mit Erläuterung
Funktionalität:
Korrektheit, Angemessenheit, Sicherheit, Kompatibilität

Zuverlässigkeit:
Reife, Fehlertoleranz, Wiederherstellbarkeit

Effizienz:
Wirtschaftlichkeit, Laufzeitverhalten, Verbrauchsverhalten

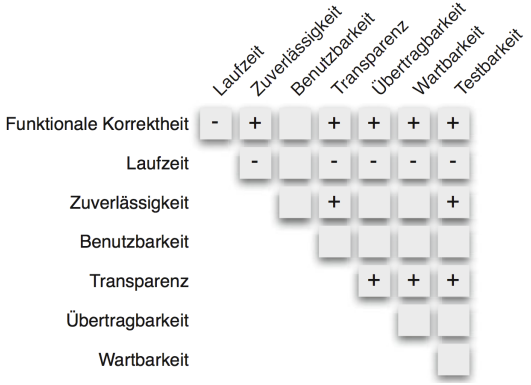
Übertragbarkeit:
Anpassbarkeit, Installierbarkeit, Konformität

Änderbarkeit:
Modularität, Strukturiertheit

Testbarkeit:
Analysierbarkeit

Transparenz:
Modularität

Vereinbarkeit der Qualitätsziele



Güte von Metriken
1. Objektivität
Messung frei von subjektiven Einflüssen.

2. Robustheit
Ergebnisse müssen wiederholbar sein. Objektivität ist notwendig

3. Vergleichbarkeit
Verschiedene Messungen der gleichen Metrik müssen in Relation gestellt werden können

4. Ökonomie
Erhebung muss kostenökonomisch erfolgen

5. Korrelation
Rückschluss auf das im Fokus stehende Merkmal

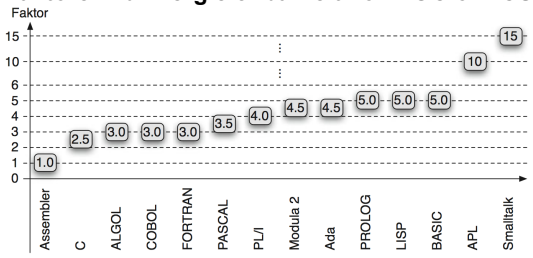
6. Verwertbarkeit
Je nach Ergebnis soll eine Reaktion erfolgen

Lines of Code
LOC = Anzahl aller Codezeilen
NCSS = Anzahl aller Codezeile ohne Kommentare & Leerzeilen

$$Dokumentationsgrad = \frac{NCSS}{LOC}$$

Vorteile / Nachteile:
+ Anwendbar auf alle Programmarten (außer Grafisch)
- Programmierstil hat großen Einfluss auf Metrik
→daher Vergleichbarkeit schlecht
- Vergleich schwierig bei unterschiedlichen Programmiersprachen

Faktoren zur Vergleichbarkeit von LOC o. NCSS



Halstaed-Metrik
$\eta_1 = Anzahl\ unterschiedlicher\ Operatoren$
$\eta_2 = Anzahl\ unterschiedlicher\ Operanden$
$N_1 = Gesamtzahl\ der\ Vorkommen\ aller\ Operatoren$
$N_2 = Gesamtzahl\ der\ Vorkommen\ aller\ Operanden$

$Größe\ des\ Vokabulars\ \eta = \eta_1 + \eta_2$
$Länge\ der\ Implementierung\ N = N_1 + N_2$
$Schwierigkeit\ des\ Programms\ D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$
$Umfang\ in\ Bits\ V = N \times log_2(n)$
$Aufwand\ zum\ verstehen\ E = D \times V$

Vorteile / Nachteile:
+ einfach zu ermitteln
+ für alle Programmiersprachen einsetzbar
+ gutes Maß für Komplexität
- Nur lexikalische Komplexität
- Moderne Programmierkonzepte wie Sichtbarkeit o. Namensräume nicht berücksichtigt
- Aufteilung Operatoren / Operanden ist sprachabhängig

Variante 1

```
1  int ggt1(int x, int y)
2  {
3      while (x != y) {
4
5          if (x > y)
6              x -= y;
7          else
8              y -= x;
9      }
10     return x;
11 }
```

Operatoren			
int	(3×)	(), {}	(5×)
,	(1×)	while	(1×)
!=	(1×)	if else	(1×)
>	(1×)	-=	(2×)
return	(1×)	;	(3×)
ggt1	(1×)		
Operanden			
x	(6×)	y	(5×)

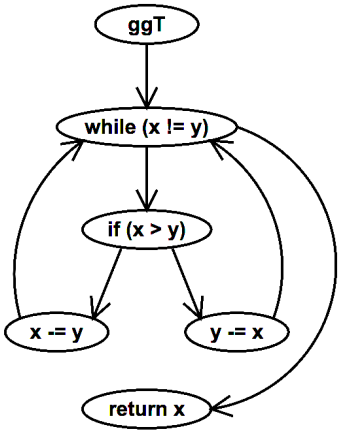
Variante 2

```
1  int ggt2(int x, int y)
2  {
3      int r;
4
5      do {
6          r = x % y;
7          x = y;
8          y = r;
9      } while (y != 0);
10     return x;
11 }
```

Operatoren			
int	(4×)	(), {}	(4×)
,	(1×)	do while	(1×)
!=	(1×)	=	(3×)
%	(1×)	;	(6×)
return	(1×)	ggt2	(1×)
Operanden			
0	(1×)	x	(4×)
y	(5×)	r	(3×)

Zyklomatische Komplexität
$V(G) = e\text{-}n + 2p$
$e = Anzahl\ der\ Kanten$
$n = Anzahl\ der\ Knoten$
$p = Anzahl\ der\ Komponenten$

V(G)	Risiko
1-10	Einfache App. geringes R.
11-20	Komplexere App. mittleres R.
21-50	Komplexe App. hohes R.
>50	Untestbare App. extrem hohes R.



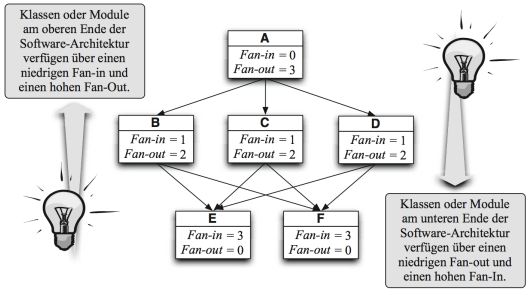
Vorteile / Nachteile:
+ einfach zu ermitteln
+ Gute Korrelation zwischen zyklomatischer Zahl & Verständlichkeit
- Berücksichtigt nur den Kontrollfluss, Datenfluss wird nicht berücksichtig
- Kontrollfluss zwischen Komponenten kann komplex sein
- Ungeeignet für OO-Programme

Objektorientierte Komponentenmetriken
Es werden meist einzelne Klassen betrachtet

Objektorientierte Strukturmetriken
Es werden Beziehungen zwischen Klassen betrachtet

Wichtigste Metrik: Kopplung zwischen Klassen

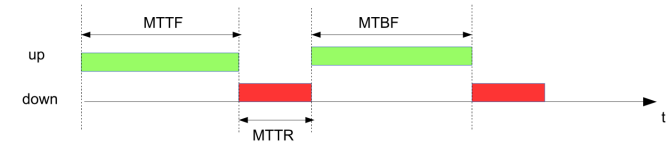
Kenngrößen
$F_{in}(C) = Anzahl\ der\ Klassen\ die\ auf\ C\ zugreifen$
$F_{out}(C) = Anzahl\ der\ Klassen\ auf\ die\ C\ zugreift$



Komplexitätsmaß einer Klasse nach Henry & Kafura
$v_{HK}(C) = (F_{in}(C) \times F_{out}(C))^2$

Komplexitätsmaß einer Klasse nach Henry & Selig
$v_{HS}(C) = v_{internal}(C) \times (F_{in}(C) \times F_{out}(C))^2$
$v_{internal}(C) = internes\ Komplexitätsmaß\ der\ Klasse$
Z.B. Zyklomatische Komplexität

Definition Zuverlässigkeit
Wahrscheinlichkeit dafür, dass ein System unter vorgegebenen Arbeitsbedingungen während einer festgelegten Zeitdauer fehlerfrei funktioniert.



Definition Verfügbarkeit
Verfügbarkeit ist die Wahrscheinlichkeit für doe Funktionsfähigkeit eines Systems zu einem gegebenen Zeitpunkt.

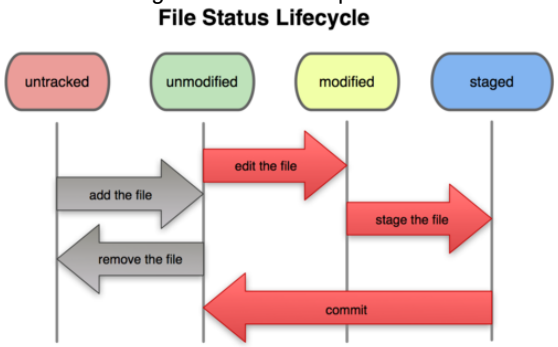
$$A = \frac{MTBF}{MTBF + MTTR}$$

$$MTBF = \frac{MTTR \times A}{(1 - A)}$$

Definition Prozessqualitätsmaßnahmen
Maßnahmen für geregelte Produktentwicklung in Form eines definierten Prozesses.

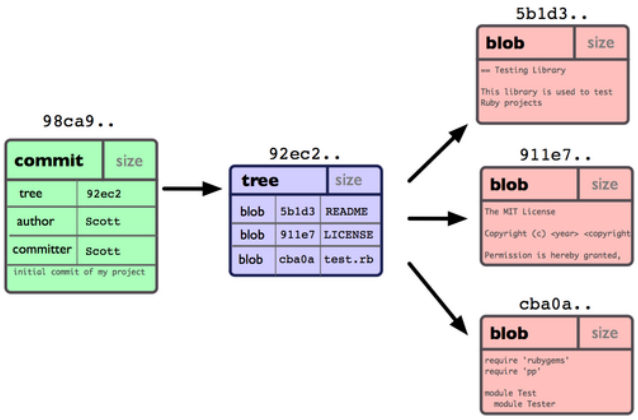
Produktdiversifizierung
Nutzung von Versionsverwaltung, Build- & Testautomatisierung & Defektmanagement

Versionsverwaltung
Transparenz: Wer hat was geändert
Rekonstruktion: Älterer Projektzustand soll jederzeit wiederherstellbar sein
Simultaner Zugriff: Zusammenspiel mehrerer Entwickler

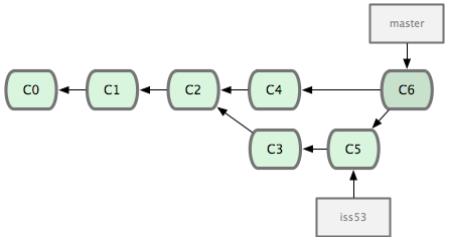


Simultaner Zugriff:
1. Lock & Checkout -> Checkin & Unlock
2. Checkout Merge Checkin

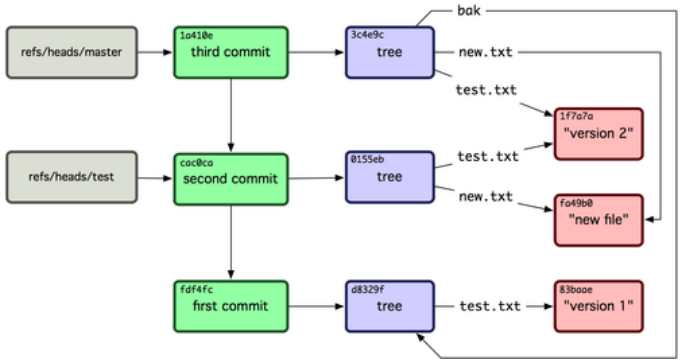
Filesystem in Git



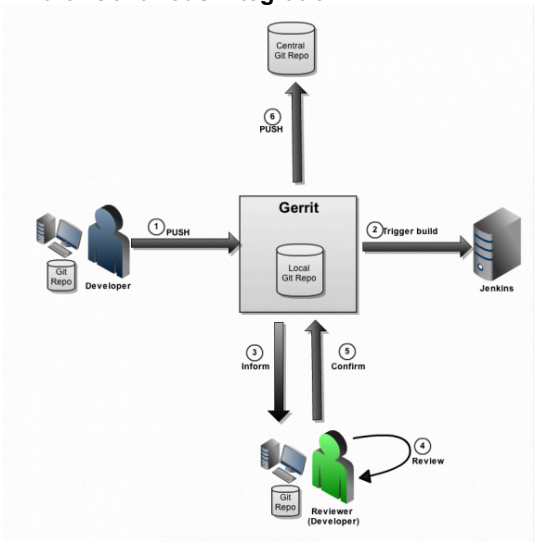
Branching in Git



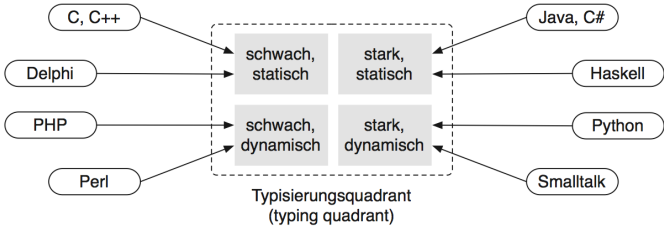
Commits in Git



Kind of Continous Integration



Typisierung

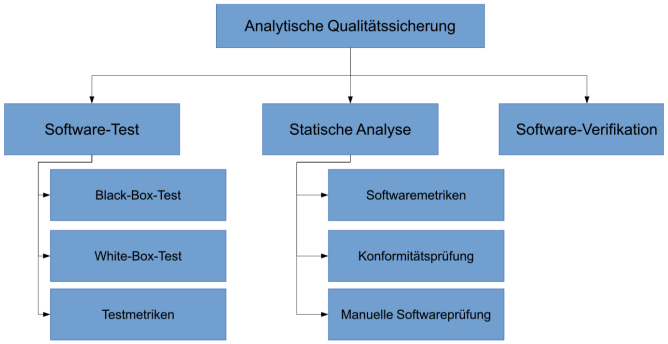


Vertragssichere Programmierung

modulo.c

```c
#include <assert.h>

int modulo(int a, int b)
{
    /* Berechnet den ganzzahligen Divisionsrest von a / b */

    assert(a >= 0);
    assert(b >= 0);
    assert(b != 0);

    int q,r;

    q = 0;
    r = a;

    while (r >= b) {
        assert(a == q * b + r);

        q++;
        r -= b;
    }

    assert(r >= 0 && r < b);
    return r;
}
```

Analytische Qualitätssicherung

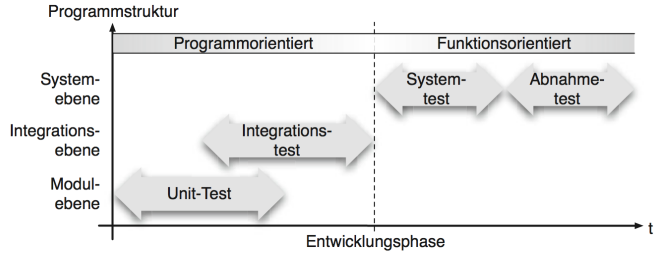


Definition Softwaretest nach IEEE

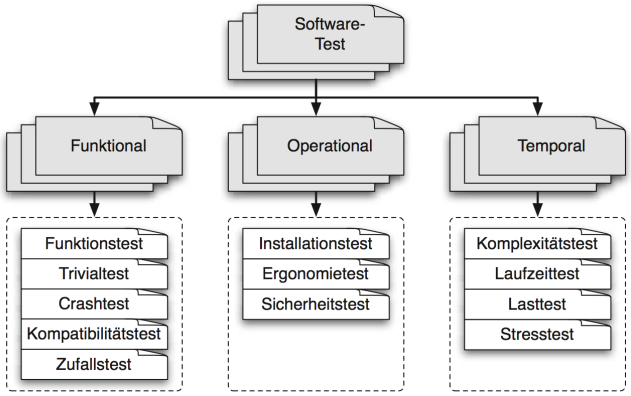
„An activity which a system or component is executed under specified conditions, the results are observed ord recorded, and an evalution is made of some aspect oft he system of component.“

Mit Tests wird die Anwesenheit von Fehlwirkungen nachgewiesen. Es kann durch Tests jedoch nicht gezeigt werden, dass keine Fehlerzustände im Testobjekt vorhanden sind.
Dafür muss bewiesen werden, dass die formalen Spezifikationen & die formale Implementation äquivalent sind.

Verschiedene Prüfebenen



Prüfkriterien



Prüftechniken im Vergleich

	Funktionstest	Trivialtest	Crashtest	Kompatibilitätstest	Zufallstest	Installationstest	Ergonomietest	Sicherheitstest	Komplexitätstest	Laufzeittest	Lasttest	Stresstest
Unit-Ebene	✘	✘			✘				✘			
Integrationsebene	✘	✘		✘	✘				✘			
Systemebene	✘	✘	✘	✘	✘	✘	✘	✘		✘	✘	✘
Abnahmeebene	✘		✘	✘		✘	✘	✘		✘	✘	✘
Black-Box-Technik	✘	✘	✘	✘	✘	✘	✘	✘		✘	✘	✘
White-Box-Technik	✘								✘			

Anweisungsüberdeckung C0

Alle Knoten des Kontrollflussgraphen werden durchlaufen

$$M_{C_0} = \frac{Anzahl\ der\ überdeckten\ Knoten}{Anzahl\ der\ Knoten} \times 100\ [\%]$$

Zweigüberdeckung C1

Alle Kanten des Kontrollflussgraphen werden durchlaufen

$$M_{C_1} = \frac{Anzahl\ der\ überdeckten\ Kanten}{Anzahl\ der\ Kanten} \times 100\ [\%]$$

Pfadüberdeckung

Alle Pfade des Kontrollflussgraphen werden durchlaufen

Qualitätskriterien

- Sicherheit
- Performance
- Skalierbarkeit
- Wartbarkeit (Erweiterbarkeit)
- Robustheit
- Bedienbarkeit
- Portabilität
- Verfügbarkeit
- Testbarkeit