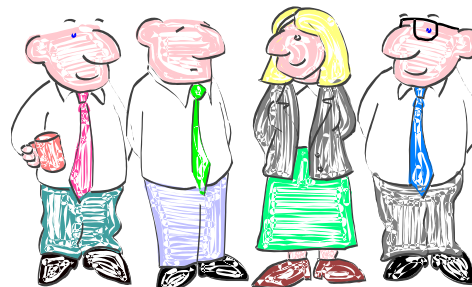


# Quality Point München



## Restrisikoabschätzungen beim Test



**Vielleicht steht es in den Sternen ...**

## Unsicherheiten des Managements

**Der Test ist zu Ende,  
weil Testfälle lt. Testplan erfolgreich bearbeitet wurden  
oder  
weil die geplante Zeit für den Test abgelaufen ist.**

**Wie gut ist die Software wirklich ?**

**Wie viele Fehler sind noch enthalten ?**

# Abschätzung der Fehleranzahl in einer Software

## Fehlerquelle Mensch

**Von 100 Zeilen, die der Mensch schreibt**

**bzw. 100 Strichen, die der Mensch zeichnet**

**werden 1,5 fehlerhaft sein.**

**Davon wird der Mensch 50 % selbst merken.**

**Es bleiben also 0,7 fehlerhafte Zeilen auf 100.**

**In einem Programm oder Dokument mit 1000 Zeilen sind das 7 Fehler.**

**In einem Programm oder Dokument mit 10000 Zeilen sind das 70 Fehler.**

(nach H.M. Sneed)

# Abschätzung der Fehleranzahl in einer Software

## Fehlerverteilung über die Softwareentwicklung (1)

### fachliche Spezifikationsfehler (ca. 1 Drittel)

**Unvollständige Erfassung des Problems**

**Lückenhafte Leistungsbeschreibung**

**Inkonsistente Leistungsbeschreibung**

**Ungenauere Festlegung der Leistungskriterien**

**Fehlerhafte Wiedergabe der Information**

**Veränderte Bedürfnisse der Anwender**

# Abschätzung der Fehleranzahl in einer Software

## Fehlerverteilung über die Softwareentwicklung (2)

### technische Entwurfsfehler (ca. 1 Drittel)

**Missverständnis der Leistungsbeschreibung**

**Unkenntnis des technischen Umfelds**

**Unkenntnis der Programmiermittel**

**Widersprüchliche Schnittstellendefinitionen**

**Unvollständige Datendefinitionen**

**Fehlerhafte Ablauflogik**

**Unvollständige Fehlerbehandlung**

**Unvorhergesehene Hardware- und Softwareprobleme**

# Abschätzung der Fehleranzahl in einer Software

## Fehlerverteilung über die Softwareentwicklung (3)

### Realisierungsfehler (ca. 1 Drittel)

**Falsche Bedingungsformulierung**

**Nicht initialisierte Daten**

**Fehlerhafte Parameterübergabe**

**Datenfeldüberlauf**

**Fehlerhafte Anweisungsreihung**

**Arithmetische Fehler**

**Flüchtigkeitsfehler**

# Abschätzung der Fehleranzahl in einer Software

## Erfahrungswerte lt. Untersuchungen

- **4-8 Fehler / 100 Befehle**
- **40% Codier-/Logik-Fehler und 60% Entwurfsfehler**
- **Probleme**
  - **o.g. Zahlen sind sehr alt und werden von Autor zu Autor weitergereicht**
  - **Bezugsgrößen und Voraussetzungen der Autoren sind unbekannt**
  - **eigene Untersuchungen existieren meist nicht**

## Abschätzung der Fehleraufdeckung durch Tests aufgedeckte Fehler beim funktionalen Test

- **Black-Box-Test (2-Personen-Gruppen):** 49%
- **Black-Box- und Entwurfs-Test:** 46%
- **Black-Box-Test (1-Personen-Gruppen):** 30%
- **Entwurfsgestützter Test:** 28%
- **Black-Box-Anforderungstest:** 24%

**Hypothese vom kompetenten Programmierer:  
realisierte Funktion weicht nur wenig von der Sollfunktion ab**

## Abschätzung der Fehleraufdeckung durch Tests aufgedeckte Fehler (prozentual)

Art/Ort Fehler	M1	M2	M3
Initialisierung	75%	46%	65%
Ablauflogik	67%	49%	43%
Berechnung	64%	59%	71%
Schnittstellen	31%	25%	47%
Daten	28%	27%	21%
Meldungstexte	08%	08%	17%

- **M1:** kombinierte Äquivalenzklassen- und Grenzwertmethode
- **M2:** strukturorientiertes Testen (Anweisungsüberdeckung)
- **M3:** manuelle statistische Analyse

# Abschätzung der Fehleraufdeckung durch Tests

## Vergleichende Wertung Blackbox –White Box

- mit spezifikationsorientiertem (**Black-Box**) Test werden 24% - 30% der Fehler zuverlässig gefunden, bei Kombination von Verfahren 46% - 49%
  - Dabei werden vor allem Auslassungsfehler (fehlende Fallunterscheidungen oder Teilfunktionen), Initialisierungsfehler, Ablauflogikfehler und fehlerhaft realisierte Teilfunktionen entdeckt .
- mit kontrollflussbezogenem (**White-Box**) Testen werden
  - durch Anweisungsüberdeckung 18% - 41 %
  - durch Zweigüberdeckung 16% - 92% -> 21%
  - durch Pfadüberdeckung 62% - 64%

der Fehler gefunden

# Abschätzung der Restfehleranzahl

## Fehlereinpflanzung (error seeding)

- zusätzliche Fehler von unabh. Personen oder Werkzeug in Programm einpflanzen
- Tests noch mal durchführen

Voraussetzung: echte und eingebaute Fehler werden mit der gleichen Wahrscheinlichkeit gefunden

$$\rightarrow GF/F = GE/E \quad \rightarrow F = E * GF/GE$$

(F= Anzahl der echten Fehler, E= Anzahl der eingepflanzten Fehler, GE= Anzahl der gefundenen eingepflanzten Fehler, GF= Anzahl der gefundenen echten Fehler)

**Gleiche Wahrscheinlichkeit erfordert folgende Bedingungen** (bei Softwareprodukten **meist nicht erfüllt**):

- Eine sehr große Anzahl von Fehlern im Programm
- Eine gleichmäßige Verteilung der Fehler im Programm
- Eine Verteilung der eingepflanzten Fehler, die der Verteilung der echten Fehler entspricht
- Eine gleiche Wahrscheinlichkeit, echte und eingepflanzte Fehler zu entdecken
- Keine Wechselwirkung zwischen echten und eingepflanzten Fehlern

# Abschätzung der Restfehleranzahl

## Test durch zwei unabhängige Gruppen

Gruppen G1, G2 entwickeln Testfallmengen T1, T2

$G_i$  entdeckt  $F_i$  Fehler,

$F_{1 \cap 2}$  = Fehler, die von G1 und G2 entdeckt wurden

$F$  = Anzahl der vorhandenen Fehler

$F_1/F = F_{1 \cap 2}/F_2$  Testeffizienz von Gruppe G1

$F_2/F = F_{1 \cap 2}/F_1$  Testeffizienz von Gruppe G2

$$F = F_1 * F_2 / (F_{1 \cap 2})$$

$NF = F - F_{1 \cup 2}$  (NF nicht gefundene Fehler)

( $F_{1 \cup 2}$  sind Fehler, die von G1 oder G2 entdeckt wurden)

**Voraussetzung:** beide Testgruppen haben bei allen Fehlern und beliebigen Teilmengen eine konstante Wahrscheinlichkeit des Aufdeckens von Fehlern

→ **sehr problematisch**

# Abschätzung der Restfehleranzahl

## Annahmen der statistischen Modelle

- Die Zahl der anfänglichen Fehler kann zuverlässig geschätzt werden (?)
  - Die Fehlerrate ist proportional zur Zahl der verbliebenen Fehler (**nein**: wenn die Fehlerrate z.B. auf die Hälfte sinkt, sind i.a. noch mehr als 50% der anfänglichen Fehler vorhanden. Die restlichen Fehler sind schwerer zu entdecken, da sie meist unter Bedingungen auftreten, die seltener vorkommen)
  - Korrektur eines Fehlers vermindert die Zahl der verbleibenden um 1 (**nein**: neue Fehler, nur teilweise Korrektur eines vorhandenen)
- > **Annahmen der Modelle zur Zuverlässigkeitsberechnung stimmen nicht oder sind nur schwer erfüllbar !!!**

# Was kann man pragmatisch tun ?

## Risikoklassen für Testobjekte definieren

### Ziele der Teststrategie:

Die wichtigsten Probleme werden gefunden.

Sie werden in einem frühen Stadium gefunden.

Aufwendige Probleme werden zuerst gefunden.

Ressourcen werden effizient eingesetzt.

-> so **früh** wie möglich die **wichtigsten** Fehler mit den **geringsten** Kosten finden

*Beispiel für Risiko-Klassen* (nach dem Rheinisch-Westfälischen TÜV):

- Klasse A: kein Risiko
- Klasse B: geringes Risiko, z.B. Imageverlust
- Klasse C: mittleres Risiko, z.B. finanzieller Schaden
- Klasse D: hohes Risiko, z.B. großer finanzieller Schaden möglich
- Klasse E: sehr hohes Risiko, z.B. Personenschaden möglich

## **Was kann man pragmatisch tun ?** **Fehlschlagwahrscheinlichkeit beurteilen**

**Fehlschlagwahrscheinlichkeit einer Funktion ist u.a. abhängig von**

**Einsatzfrequenz (ständig, stündlich,..., jährlich)**

**Durchauffrequenz**

**Fehlerwahrscheinlichkeit (hoch, wenn Funktion ...)**

**komplex ist**

**gänzlich neu ist**

**vielfach angepasst wurde**

**zum ersten Mal mit neuen Techniken entwickelt**

**von wechselnden Entwicklern realisiert**

**unter hohem Zeitdruck entwickelt**

**überdurchschnittlich optimiert**

**bereits früher fehlerhaft war**

**viele Schnittstellen hat**

# Was kann man pragmatisch tun ?

## Fehlschlagwahrscheinlichkeit beurteilen

### Menschliche Faktoren berücksichtigen

Erfahrung Entwickler

Beteiligung der Anwender

QS während der Entwicklung

Qualität der Modultests (Entwickler)

Beherrschung der Tools (neu/bekannt)

Größe des/der Teams

Kommunikation

## Was kann man pragmatisch tun ? Testendekriterien festlegen

- Testkonzept wurde sauber abgearbeitet.
- Test garantiert bestimmte vereinbarte Testabdeckungen.
- Keine abnahmehinderlichen Fehler mehr vorhanden.
- **Fehlerfindungsrate (z.B. gefundene Fehler/Woche) ist auf niedrigem Niveau stabilisiert** (abnahmehinderliche Fehler nahe Null).
- Festgelegte Anzahl der Fehler, die entdeckt werden sollen, wurde gefunden (Abschätzung).

**Ende gut - alles gut ?**



**Wenn es doch immer ein weißes Kaninchen wäre ...**