Qualitätssicherung von Software

Prof. Dr. Holger Schlingloff

Humboldt-Universität zu Berlin
und
Fraunhofer FIRST

Wo stehen wir?

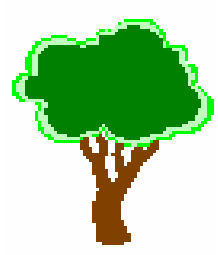
1. Einleitung, Begriffe, Software-Qualitätskriterien
2. manuelle und automatisierte Testverfahren
3. Verifikation und Validierung, Modellprüfung
4. statische und dynamische Analysetechniken
5. Softwarebewertung, Softwaremetriken
6. Codereview- und andere Inspektionsverfahren
7. Zuverlässigkeitstheorie
   - FMEA, FMECA
   - Fehlerbaumanalyse
   - stochastische Softwareanalyse
8. Qualitätsstandards, Qualitätsmanagement, organisatorische Maßnahmen

Zuverlässigkeitstheorie

- Quantitative Ermittlung von Ausfallwahrscheinlichkeiten
- Ursprung: Bewertung von Hardware
  - Alterung, Umwelteinflüsse, Materialfehler, ...
- Auch für Software einsetzbar?
  - Zertifizierungsproblematik, z.B. Cenelec
  - vorausschauende Hinweise auf Schwachstellen
- FMEA, FMECA

Fehlerbaumanalyse

(Fault tree analysis, FTA; DIN 25424*)*



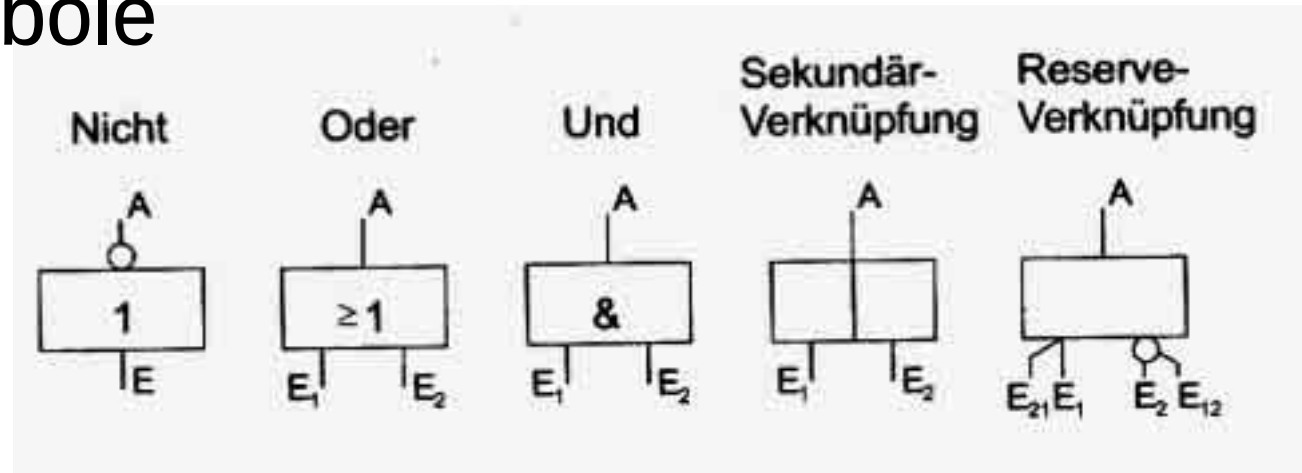
- entwickelt 1961 von H.A. Watson, Bell Labs
  - Bewertung eines Raketen-Abschuss-Systems
  - für SW und eingebettete Systeme erweitert

- Systematische Suche nach Fehlerursachen
- Elimination von singulären Schwachstellen

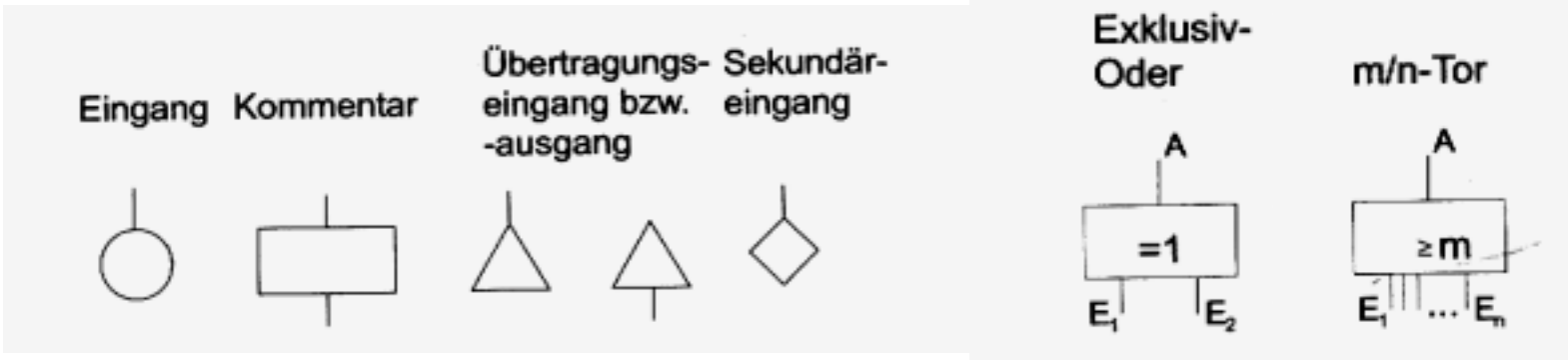
- Top-down, von der Wurzel zu den Blättern
  - Jede Ebene im Baum zeigt den selben Sachverhalt, jedoch mit verschiedenen Detaillierungsgraden
  - Wurzel repräsentiert Bedrohung, Blätter repräsentieren atomare Fehler (Ereignisse)
  - Innere Knoten sind Abstraktionen von Ereignismengen

Symbole

- Grundsymbole



- Zusatzsymbole



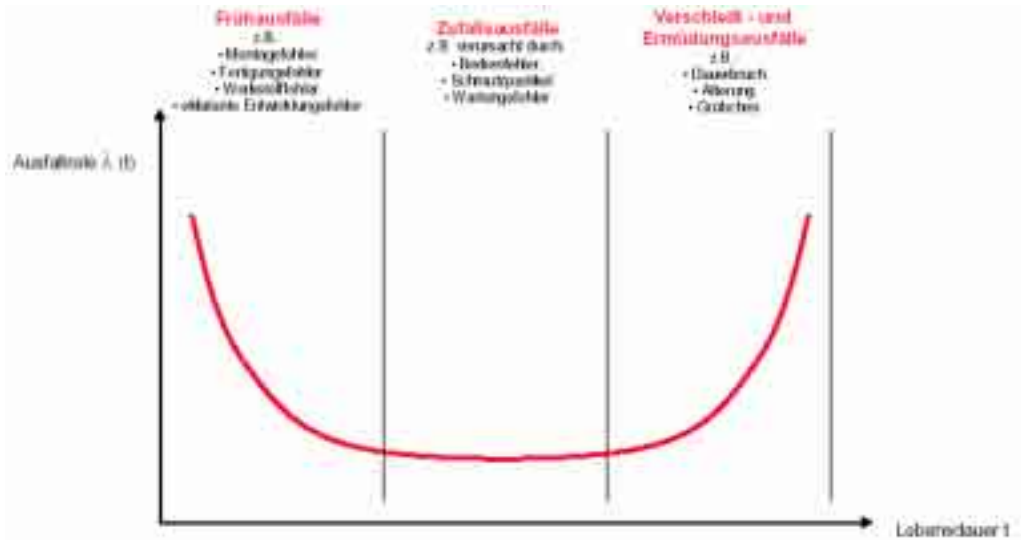
aus: Liggesmeyer, p.435

Vorgehensweise

- Top-Down-Analyse
  - Verfeinerung der Wirkzusammenhänge
  - beginnend mit unerwünschtem Ereignis

- DIN: Nichtverfügbarkeit einer Einheit = Wahrscheinlichkeit des Ausfalls zu einem Zeitpunkt (Verteilungsfunktion F(t))

- Analyse von
  - Systemfunktionen
  - Umgebungsbedingungen
  - Hilfsquellen
  - Komponenten
  - Organisation



Beispiele

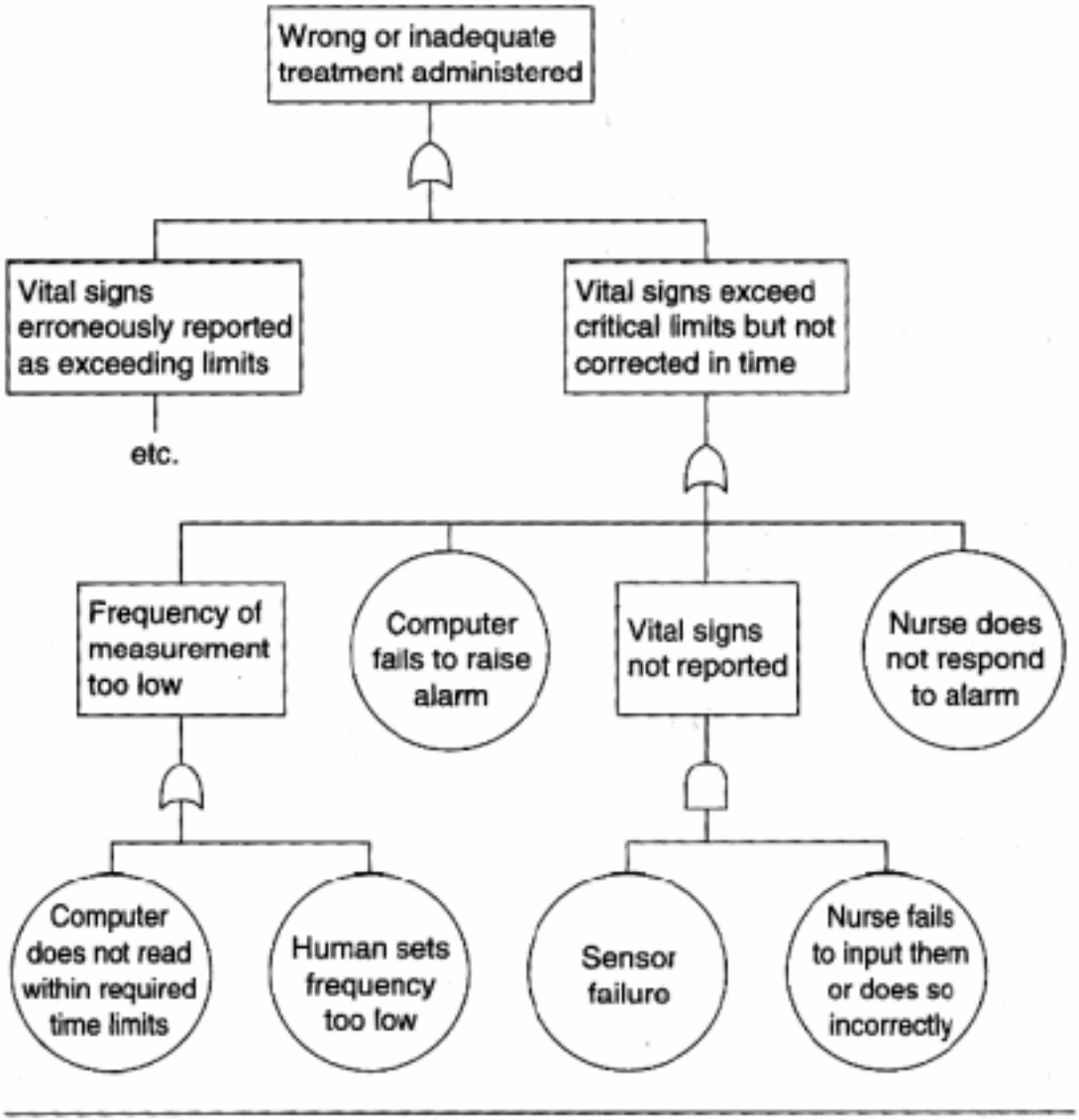
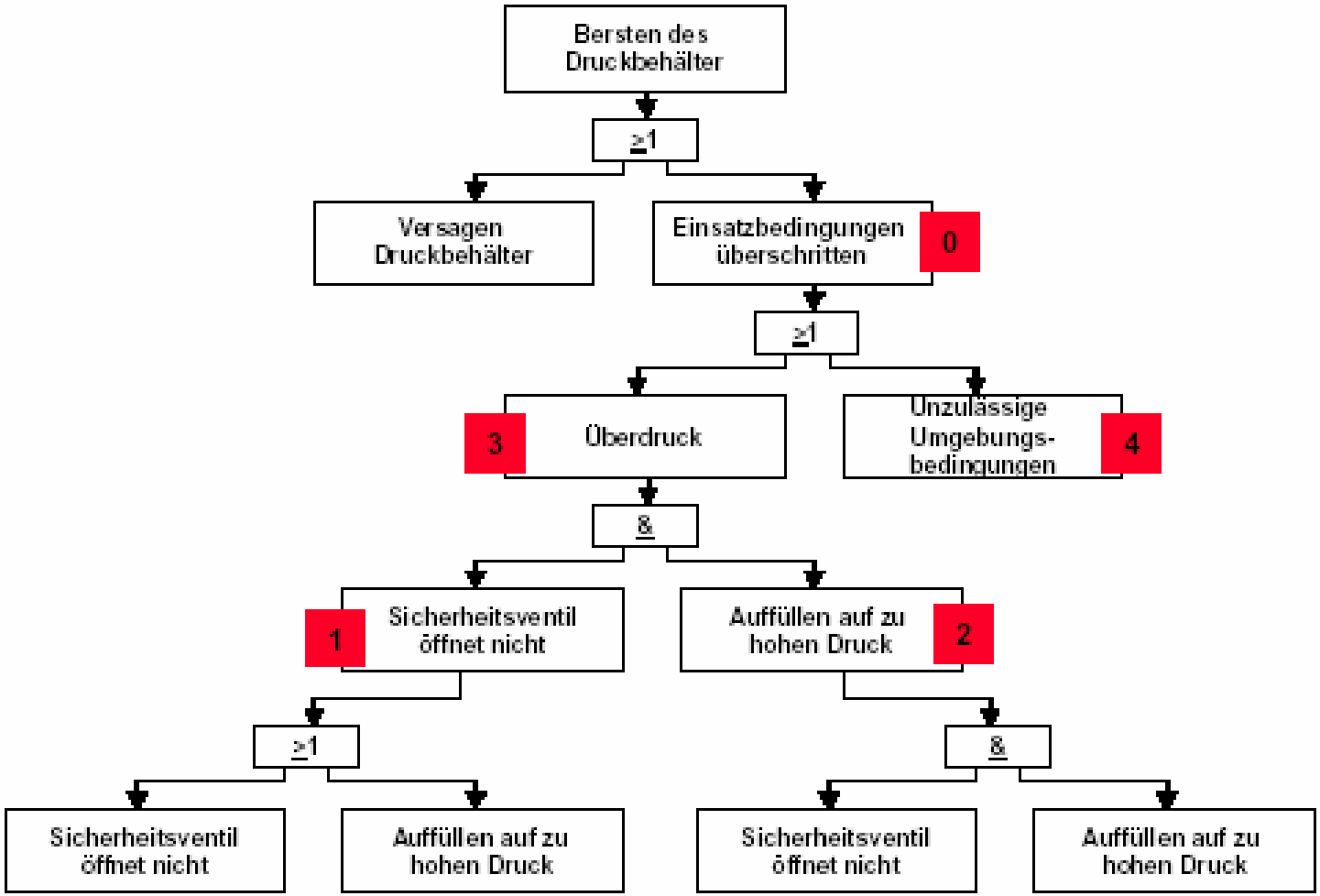


FIGURE 14.3
Portion of a fault tree for a patient monitoring system.

aus: N. Leveson, Safeware



Ausfallwahrscheinlichkeiten:

F1 = 0,002 = 0,2 %
F2 = 0,01 = 1 %
F4 = 0,01 = 1 %

$F_3 = F_1 * F_2 = 0{,}01 * 0{,}002 = 0{,}00002 = 0{,}002\ \%$

$F_0 = 1 - (1 - 0{,}00002) * (1 - 0{,}01) = 0{,}0100198 = 1{,}001\ \%$

Berechnung

- Für boolesche Verknüpfungen
  - Und: $F(t)=F_1(t) * F_2(t)$
  - Oder: $F(t)=F_1(t) + F_2(t) - F_1(t)*F_2(t)$
  - Nicht: $F(t)=1 - F´(t)$

➔Unabhängigkeit von Ereignissen beachten!

- Jedem Blatt im Fehlerbaum wird Risiko und Wahrscheinlichkeit zugeordnet
  - Risiko bedeutet finanzieller/materieller Verlust
  - Wahrscheinlichkeit ggf. als Funktion der Zeit

➔ Bottom-up-Berechnung nach vorstehenden Regeln ergibt Prioritätenliste der Risiken

Schnitte (Cut sets)

- Ein Schnitt ist eine Menge von Basisereignissen, so dass gilt:
  Falls irgendein Ereignis des Schnittes nicht eintritt, wird auch das Ereignis der Wurzel nicht eintreten
- Repräsentation des Fehlerbaumes durch Schnitte: Und-Oder-Baum mit zwei Stufen
- Informationen zur Bewertung von Systemschwächen (Relevanz von Ereignissen für den Schadensfall)

Pause!


ICH HABE IHNEN GLEICH GESAGT, ES IST EIN AUSLAUFENDES MODELL...

stochastische Zuverlässigkeitsmodelle

- Idee: Quantifizierung der Zuverlässigkeit R (reliability) eines technischen (HW/SW-) Systems (vgl. SW-Metriken)
- Alterung: R=R(t) abhängig von der Zeit (Überlebenswahrscheinlichkeit)
  - z.B. mittlere Betriebsdauer zwischen Ausfällen bei Ausführungen
- Software: abhängig vom „Reifegrad“

- HW: Erhöhung der Zuverlässigkeit durch Redundanz, Fehlertoleranzkonzepte, ...
- SW: Übertragung der Konzepte? (ANSI/AIAAR013-1992, Recommended Practice for Software Reliability)

Definition Zuverlässigkeit

- Zuverlässigkeit (reliability)
  - Grad der Fähigkeit einer Betrachtungseinheit, die geforderte Leistung während einer vorgegebenen Zeitspanne zu erbringen
  - Wahrscheinlichkeit der Abwesenheit von Ausfällen über dem Beobachtungszeitraum
  - R(t)=P[kein Ausfall im Intervall 0..t]

- vergleichbar
  - *Verfügbarkeit (availability):* Maß für die Wahrscheinlichkeit, dass die geforderte Leistung zu einem Zeitpunkt erbracht werden kann
  - *Verlässlichkeit (dependability):* Maß für das gerechtfertigte Vertrauen in die Leistung eines Systems

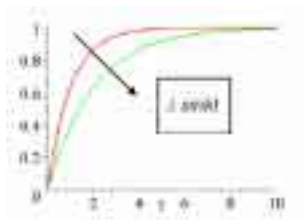
Ausfallwahrscheinlichkeit

- F(t) = Wahrscheinlichkeit (mindestens) eines Ausfalls im Beobachtungszeitraum

$$F(t) = 1 - R(t)$$

- Oft sind F(t) und R(t) exponentiell verteilt ($e^{-\lambda t}$)
- Software: Anteil der fehlerhaften an allen betrachteten Programmausführungen

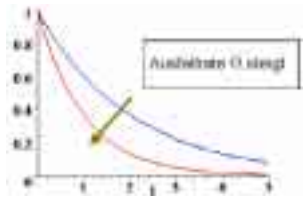
$$F(t) = \lim_{n \to \infty} (n_f / n)$$



- erste Ableitung ist *Ausfalldichte*:

$$f(t) = dF(t) / dt$$

Tendenz der Ausfallwahrscheinlichkeit



Ausfallrate

- Ausfallrate: Wahrscheinlichkeit, dass sich ein Ausfall pro Zeiteinheit in einem Intervall $[t,t+\delta]$ ereignet, gegeben Ausfallfreiheit bis zu t

  $$F(t+\delta)-F(t) \;/\; \delta * R(t)$$

  Häufigkeit, mit der sich Ausfälle in einem Intervall ereignen
- Hazard-Rate: bedingte Ausfalldichte

  $$z(t) = f(t) \;/\; R(t)$$

  Grenzwert der Ausfallrate für kleine Intervalle

- *p*: Wahrscheinlichkeit des Versagens in einem Programmlauf
- *(1-p)*: Wahrscheinlichkeit des Nichtversagens
- $(1-p)^n$: Wahrscheinlichkeit des Nichtversagens in *n* paarweise unabhängigen Läufen

Wahrscheinlichkeit mindestens eines Versagens in *n* Läufen: $1-(1-p)^n$

Aussagesicherheit

- Unter der Annahme, dass ein System bei allen Beobachtungen niemals versagt, ist p natürlich nicht exakt bestimmbar; berechenbar ist eine obere Schranke für p:

  $p \leq \mu$, wobei μ vorgegeben ist

- Mit Simulationsläufen ist die Wahrscheinlichkeit dieser Aussage bestimmbar: $\beta = P(p \leq \mu)$

  β heißt *Aussagesicherheit* der Hypothese

Zusammenhang von μ, β und *n*

- Wahrscheinlichkeit, dass in *n* Läufen höchstens *x* Versagensfälle auftreten:

$$p(n,x) = \sum_{m=0}^{x} \binom{n}{m} p^m (1-p)^{n-m}$$

Ausdruck nimmt mit wachsendem *p* monoton ab

- Wähle μ(x) so, dass $\sum_{m=0}^{X} \binom{n}{m} \mu(X)^m [1-\mu(X)]^{n-m} = 1-\beta$

- Dann gilt: $P[p \le \mu(X)] = P\left[\sum_{m=0}^{X} \binom{n}{m} p^m [1-p]^{n-m} \ge 1-\beta\right]$
- Daraus folgt (X=0):

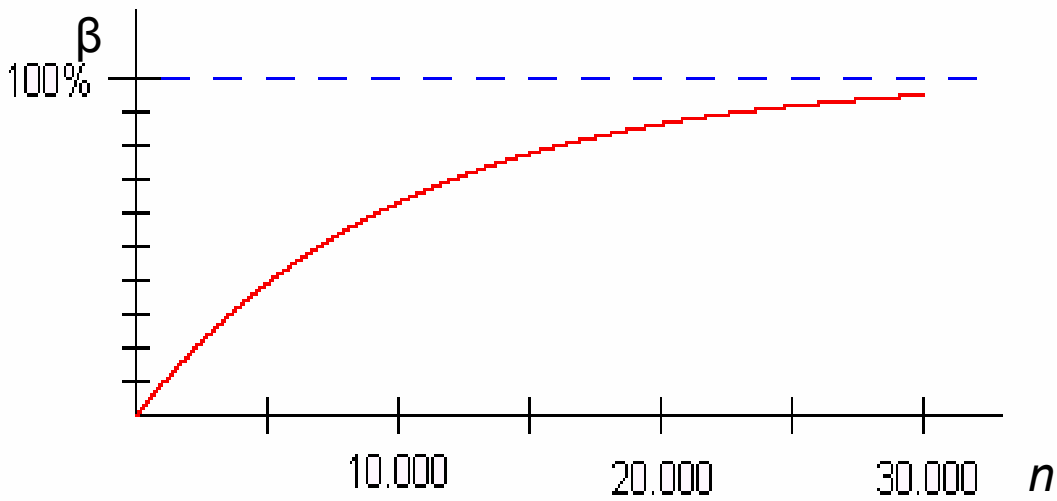
$$\sum_{m=0}^{X} \binom{n}{m} \mu(X)^m [1-\mu(X)]^{n-m} = 1 \cdot 1 \cdot [1-\mu(0)]^n = 1-\beta$$

$$\beta = 1 - \left[1 - \mu\right]^n$$

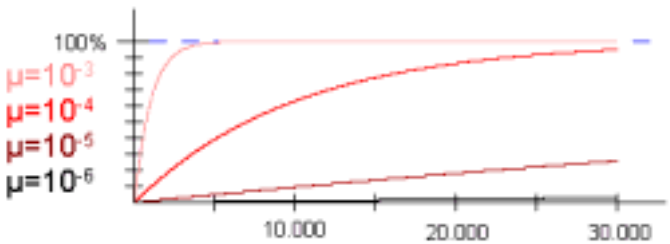
- Poisson-Verteilung statt binomischer Verteilung der Erwartungswerte ergibt eine ähnliche Ausfalldichtefunktion
- Formel reflektiert die erreichbare Aussagesicherheit bei gegebener Oberschranke für die Versagenswahrscheinlichkeit in Abhängigkeit von der Anzahl der versagensfreien Programmläufe

Beispielwerte für $\mu=10^{-4}$

$$\beta = 1 - (1 - \mu)^n = 1 - 0{,}9999^n$$



n	β
5000	0,3934845
10000	0,63213895
15000	0,77688658
20000	0,86467825
25000	0,91792526
30000	0,9502204
35000	0,9698079
40000	0,98168802
45000	0,9888935
50000	0,99326374



Zuverlässigkeitsmodelle

- Makromodelle
  - Außenverhalten des Systems, Black-Box-Sicht
  - Ausfallrate proportional zur Zahl der enthaltenen Fehler
- Mikromodelle
  - innere Struktur, White-Box-Sicht
  - Ausfallrate abhängig von der Aufrufstruktur

Beispiel: Jelinski-Moranda-Makromodell

- Hazard-Rate proportional zur Anzahl enthaltener Fehler
  - feste anfängliche Gesamtfehlerzahl
  - jeder Fehler verursacht mit gleicher Wahrscheinlichkeit und Rate Fehler
  - aufgetretene Fehler werden sofort beseitigt (konstante Änderung bei jeder Programmversion)
- Abschätzung der Zahl der ursprünglichen Fehler durch Beobachtung von Ausfällen über einen Zeitraum
- Berechnung der Anzahl der Restfehler und des erforderlichen QS-Aufwandes

Beispiel: Shooman-Mikromodell

- Berücksichtigung der Programmstruktur: Aufteilung in Segmente (Module, Klassen, Funktionen,...)
- Schätzung der relativen Ausführungshäufigkeit und –zeit sowie der Fehlerwahrscheinlichkeit je Segment (Instrumentierung des Quelltextes)
  - Fehler sind nicht gleichverteilt
  - große Module nicht zwangsweise fehlerbehaftet
  - keine strenge Korrelation Testfehler-Produktfehler
- Berechnung der Fehlerrate aus dem Quotienten von fehlerhaften Ausführungen und zugehörigen Ausführungszeiten
- Trendanalyse für verbleibende Fehleranzahl

Bewertung der Modelle

- über 40 Varianten in der Literatur
- kein ideales Modell (viel Spielraum)
- Schätzungen, keine Vorhersagen!
- langfristige Kalibrierung nötig
- Genauigkeit „bis zu +-5%“ erreichbar
- Werkzeuge verfügbar

Literaturempfehlungen

- Wolfgang Ehrenberger: Software-Verifikation, Verfahren für den Zuverlässigkeitsnachweis von Software;
  Hanser Verlag

- Peter Liggesmeyer: Formale und stochastische Methoden zur Qualitätssicherung technischer Software
  http://pi.informatik.uni-siegen.de/stt/20_3/20_3_Liggesmeyer.pdf

- F.Belli, M.Grochtmann, O. Jack: Erprobte Modelle zur Quantifizierung der Software-Zuverlässigkeit;
  Informatik Spektrum 21: 131–140 (1998)

- Jelinski, Z., Moranda, P.B., Software reliability research;
  in: *Statistical Computer Performance Evaluation*, W. Freiberger, Ed., New York, Academic, 465-484, 1972
- Shooman, M., Operational testing and software reliability during program development;
  in Rec. 1973 *IEEE Symp. Comput. Software Rel.*, New York, 51-57, 1973.