



# Test und Verlässlichkeit Foliensatz 6: Problembeseitigung

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV\_F6)

18. Juli 2017



## Inhalt TV\_F6: Problembeseitigung

### Fehlervermeidung

- 1.1 Deterministische Prozesse
- 1.2 Nichtdeterministische Prozesse
- 1.3 Projekte, Vorgehensmodelle
- 1.4 Qualität und Kreativität

### Fehlerbeseitigung

- 2.1 Ersatz
- 2.2 Reparatur
- 2.3 Fehlerlokalisierung

### 2.4 Benutzer als Tester

### Ausfälle

- 3.1 Ausfallmechanismen
- 3.2 Frühausfälle und Voralterung
- 3.3 Wartung und Reserve

### FF im Betrieb

- 4.1 Fail-Safe/-Fast/-Slow
- 4.2 Neustart, Wiederholung
- 4.3 Fehlerisolation
- 4.4 Fehlertoleranz

### Literatur



## Problembeseitigungsiterationen

Die Problembeseitigung erfolgt auf allen drei Ebenen zur Sicherung der Verlässlichkeit

- Fehlervermeidung,
- Fehlerbeseitigung und
- Schadensbegrenzung incl. Korrektur von Fehlfunktionen,

in einer Iteration aus

- Problemerkennung,
- Suche/Auswahl möglicher Maßnahmen zur Beseitigung oder Vermeidung der Probleme und
- Erfolgskontrolle.

Quantitative Beschreibung durch Markov-Ketten



## Fehlervermeidung



## Fehlervermeidung

Die Fehler in IT-Systemen entstehen mit dem System. Ursachen:

- im einfachsten Fall Fehler im Entstehungsprozess,
- oft jedoch ungünstiges Zusammentreffen zufälliger Ereignisse.

Fehlervermeidung ist die Beseitigung der Ursachen für die Fehlerentstehung

- für Software und Entwürfe im Entwurfsprozess,
- für Hardware und Mechanik auch in den Fertigungsprozessen.

Hilfreich ist Determinismus.

### Determinismus

Ein Service arbeitet deterministisch, wenn er immer gleich ausgeführt wird und für gleiche Eingaben (und gespeicherte Zustände) gleiche Ergebnisse liefert (TV\_F1, Abschn. 1.1).

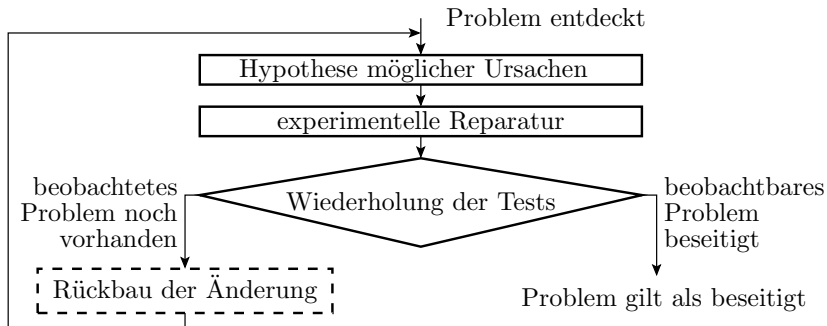


## Deterministische Prozesse

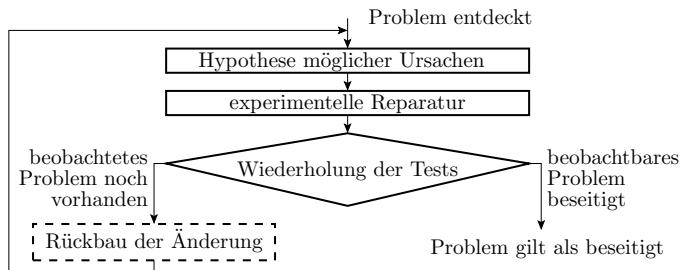


## Fehlervermeidung in deterministischen Prozessen

Fehlervermeidung in deterministischen Entstehungsprozessen erfolgt nach dem Prinzip der experimentellen Reparatur:



Der Reparaturversuch ist die Kontrolle, ob die aufgestellte Hypothese über die Ursache des Problems richtig war.



Wenn der Entstehungsprozess deterministisch ist

- entstehen bei gleichen Vorgaben (für die Fertigung oder den Entwurf) gleiche Systeme mit denselben Fehlern,
- weist eine wiederholte Prozessarbeit, bei der der Fehler nicht entsteht, den Reparatur Erfolg nach,
- lässt sich jedes beobachtbare Problem beseitigen.

Nach erfolglosen Reparaturversuchen ist ein Rückbau der vorgenommenen Änderungen zu empfehlen, um die dabei möglicherweise neu entstandenen Probleme zu beseitigen.





## Sind Entstehungsprozesse deterministisch?

- Rechnergestützte Entwurfsschritte, z.B. Compilieren eines Programms) arbeiten meist deterministisch.
- Bei automatisierten Fertigungsschritten (z.B. für elektronische Bauteile, Baugruppen, ...) wird ein deterministischer Entstehungsprozess angestrebt, der jedoch, wegen eingebetteter physikalischer Schritte störunganfälliger als eine Programmabarbeitung ist.
- Manuelle Routinearbeit, z.B. Codierung nach vorgegebenen Programmablaufplänen, tendiert dazu, dass gleiche Vorgaben zu ähnlichen Ergebnissen führen.
- Bei manueller kreativer Arbeit, Projektstätigkeit, ... unterscheiden sich die Ergebnisse bei Wiederholung mit derselben Zielstellung oft erheblich. Nicht annähernd deterministisch.



## Fehleranteil und Prozessnutzung

- Die Problembeseitigung in einem deterministischen Entstehungsprozess gehorcht ähnlichen statistischen Gesetzen wie die Fehlerbeseitigung in einem deterministischen Service (siehe später Folie 37).
- Nur nimmt statt der Häufigkeit der Fehlfunktionen der Fehleranteil der Produkte mit der Nutzungsdauer ab.
- Für die Beseitigung nichtdeterministischer Einflüsse auf die Fehlerentstehung (Prozessstörungen, Umwelteinflüsse, ...) ist die Erfolgskontrolle wesentlich unsicherer und aufwändiger.
- In einem Fehlerbeseitigungsprozess strebt deshalb der Fehleranteil durch deterministische Einflüsse mit viel kleineren Zeitkonstanten (Wochen, Monate) gegen null als der Fehleranteil durch nichtdeterministische Einflüsse (Jahre).



## Der Technologiedanke

Technologie: Lehre von reproduzierbaren Abläufen zur Erzeugung von Produkten<sup>1</sup>.

### Technologiedanke

Ein technologischer Prozess ist so zu gestalten, dass, wenn er unter gleichen Bedingungen wiederholt wird, gleiche Produkte mit (nahezu) gleichen Eigenschaften entstehen.

Die technologische Entwicklung hin zur

- automatisierten menschenfreien Fertigung und
- rechnergestützten / automatisierten Entwurfsprozessen

dient nicht nur zur Kostensenkung, sondern ist auch wesentliche Grundlage für die Fehlervermeidung.

<sup>1</sup>Der Begriff »Technologie« wurde erstmalig von dem Göttinger Professor Johann Beckmann (1739-1811) in seinem Lehrbuch »Grundsätze der teutschen Landwirtschaft« verwendet. Heute interdisziplinäres Gebiet.



## Nichtdeterministische Prozesse



## Modellierung von Zufallseinflüssen

Zufallseinflüsse werden durch Zufallsexperimente beschrieben und durch die Verteilung von Zufallsgrößen charakterisiert. Beispiel Fertigung von Widerständen. Zufallsgrößen: Widerstandswert, Abmessungen, ...

Guter technologischer Prozess:

- Alle Widerstände entstehen in einem Prozess, in dem sich Erwartungswerte und Streuungen nicht ändern.
- Messbare Prozessergebnisse (Widerstandswert, Abmessungen, ...) normalverteilt.

Schlechter technologischer Prozess:

- Mischung von Widerständen aus Prozessen, in denen Parameter unterschiedliche Erwartungswerte und Varianzen haben.
- Vergrößerung der Varianz bis multimodale Verteilung.



## Fehlervermeidung durch Prozessverbesserung:

- Suche nach Ursachen für Abweichungen der Erwartungswerte und Varianzen (Einflüsse Material, Personal, Prozessablauf, ...).
- Wahl der günstigsten Prozessbedingungen und Vereinheitlichung.
- Kontrolle der Beseitigung der Multimodalität und Varianzverringern.

Aufwand für die Erfolgskontrolle:

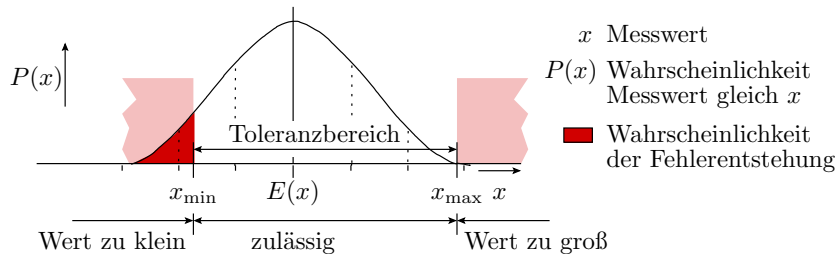
- Erzeugung und Messung der betrachteten Eigenschaften für tausende Produkte<sup>2</sup>.
- Prozessverbesserung verlangt sehr großen Stückzahlen.

<sup>2</sup>Zur Erfolgskontrolle für eine experimentelle Reparatur in einem deterministischen Prozess muß im Gegensatz hierzu nur ein Produkt erzeugt und getestet werden.



## Prozesszentrierung

Produkte mit nur einem normalverteilten Parameter, der im Toleranzbereich zwischen  $x_{\min}$  und  $x_{\max}$  liegen soll:



Der zu erwartende Fehleranteil ist die Wahrscheinlichkeit, dass der Parameter außerhalb des Toleranzbereichs liegt:

$$E(DL) = P(x < x_{\min}) + P(x > x_{\max})$$



$$E(DL) = \Phi\left(\frac{x_{\min} - E(x)}{\sqrt{D^2(x)}}\right) + 1 - \Phi\left(\frac{x_{\max} - E(x)}{\sqrt{D^2(x)}}\right)$$

Für gegebene  $x_{\min}$ ,  $x_{\max}$  und Standardabweichung  $\sqrt{D^2(x)}$  ist  $E(DL)$  am geringsten für;

$$E(x) = \frac{x_{\min} + x_{\max}}{2}$$

## Prozesszentrierung

Verschiebung der Erwartungswerte messbarer Parameter in die Mitte der Gaus-Glocke durch Veränderung von Einflussgrößen.

Der Erwartungswert eines Widerstandswerts kann z.B. durch Änderung der Schichtdicke und diese wiederum durch Variation der Beschichtungszeit angepasst werden.



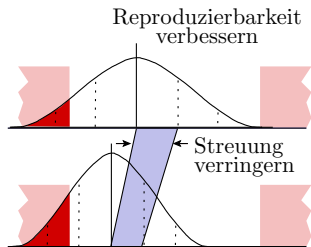
## Iteration aus Prozesszentrierung und -verbesserung

Bei einem zentrierten Prozess lässt sich der zu erwartende Fehleranteil nur noch durch eine Verringerung der Varianz absenken. Das erfordert größere Eingriffe in den Prozess:

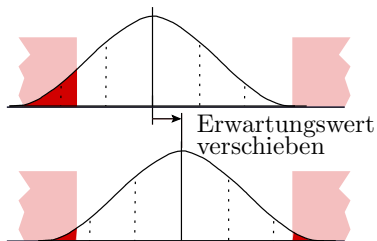
- neue Geräte, Anlagen, Materialien, Verfahren,
- neue Management-Strategien, ...

und erfolgt in größeren Zeitabständen.

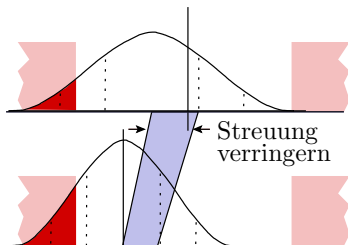
Bei größeren Prozesseingriffen geht in der Regel die Zentrierung verloren. Die Fehlerentstehungswahrscheinlichkeit nimmt sprunghaft zu. Danach folgt wieder eine Prozesszentrierung. Erst die erneute Zentrierung verringert den zu erwartenden Fehleranteil.



## Prozesszentrierung

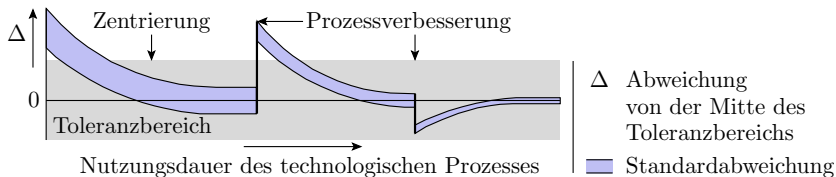


## Reproduzierbarkeit verbessern



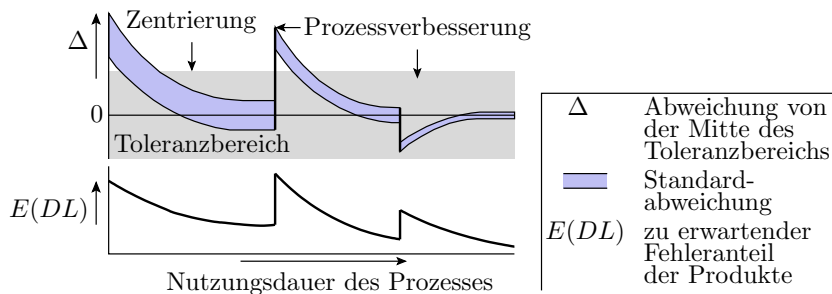
Technologien entwickeln sich in Phasen:

- Prozesszentrierung (kontinuierlich) und
- Prozessverbesserung (typisch aller Jahre).





## Reifen technologischer Prozesse



Für alle Produktparameter gilt tendenziell, dass sie nach jeder Prozessverbesserung weniger streuen, aber die Mitte ihrer Toleranzbereiche verlassen, beobachtbar an einer sprunghaften Zunahme des Fehleranteils. In der Zentrierungsphase nimmt der Fehleranteil ab, und zwar weiter als vor der Prozessverbesserung.



## Innovationen haben Sonnen- und Schattenseiten

Technologische Reifeprozesse sind heute bei jeder Art von Produkten und Service-Leistungen zu beobachten:

- Verbesserung der Wiederholgenauigkeit der Eigenschaften,
- bei gleicher Systemgröße
  - verringerte Entstehungskosten,
  - weniger Fehler (zuverlässiger), ...

Schattenseiten:

- Mehr entstehende Fehler je Systemgröße, bis der Prozess wieder »zentriert« ist.
- Mehr Funktionalität bei gleichen Kosten. Dadurch Zusatztendenz zu mehr Fehlern.
- Fehlerbeseitigung muss neu starten.

Innovationen verringern (temporär) die Zuverlässigkeit.



## Projekte, Vorgehensmodelle



## Projektarbeit als Entstehungsprozess

Deterministische Prozesse reifen durch experimentelle Reparatur.  
Massenfertigung reift durch Prozesszentrierung und  
Varianzverringern. Wie verhält es sich mit Projekten:

- den manuellen kreativen Teile der Entwurfsprozesse<sup>3</sup> und
- der Fertigung von Prototypen, Demonstratoren, ... ?

Ein Projekt ist ein zielgerichtetes, einmaliges Vorhaben, das aus einem Satz von abgestimmten, gelenkten Tätigkeiten besteht. ...

Projekten fehlt aus Sicht der Fehlervermeidung nicht nur die Reproduzierbarkeit sondern auch die häufige Wiederholung.

Schließt das eine Fehlervermeidung aus?

---

<sup>3</sup>Hier insbesondere der Software- und Hardware-Entwurf.



## Vorgehensmodelle

Vereinheitlichung des Vorgehens für eine große Klasse von Projekten.  
Eröffnet Möglichkeiten

- der Fehlervermeidung »Lernen aus Fehlern«,
- der Aufwandsminimierung und -vorhersagbarkeit, ...

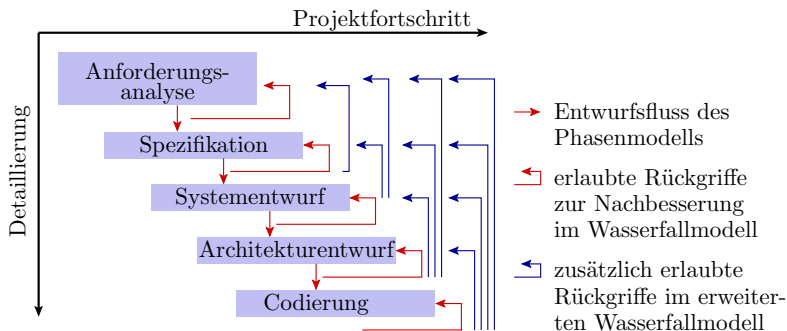
Typische Vorgehensmodelle für den Entwurf und die Fertigung von IT-Komponenten umfassen:

- Unterteilung in Schritte und Phasen,
- Referenzabläufe,
- Definition von Zwischen- und Endkontrollen, ...

Die klassischen Vorgehensmodelle für den Software-Entwurf sind Stufenmodelle. Sie unterteilen Entstehungsprozesse in Phasen:

- Anforderungsanalyse,
- Spezifikation der Ziele,
- Architekturentwurf, Codierung, Test, ...

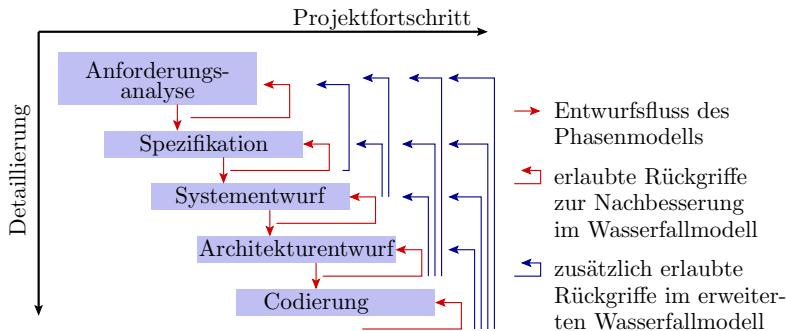
## Stufenmodelle



Stufenmodelle variieren:

- in den Abgrenzungen der Entwurfsphasen,
- Dokumentation und Kontrolle bei Phasenübergängen,
- dem Vorgehen bei Rückgriffen (rückwirkende Änderungen an den Ergebnissen bereits abgeschlossener Phasen). ...





Gestaltbare Einflussfaktoren auf Qualität und Kosten:

- Arbeitsorganisation der Phasen,
- geforderte Tests, Dokumentation, ... bei Phasenübergängen,
- Regeln für Rückgriffe zur Nachbesserung, ...

Fehlervermeidung bei Projektarbeit ist die empirische Suche nach einem guten Vorgehensmodell und seine Einhaltung.



## Bewertung von Vorgehensmodellen

Jede Art der Fehlervermeidung benötigt eine Erfolgskontrolle:

### Daraus resultierende Frage

An welchen mess- oder abschätzbaren Parametern ist eine Verbesserung eines Vorgehensmodells erkennbar?

Diese Parameter müssen zwischen unterschiedlichen konkreten Projekten eines Vorgehensmodells vergleichbar sein:

- Projektdauer, Projektkosten,
- Arbeitsschritte je entstehender Fehler, Umfrageergebnisse, ...

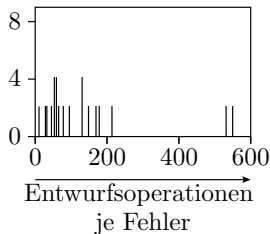
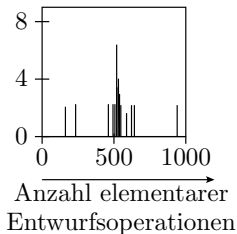
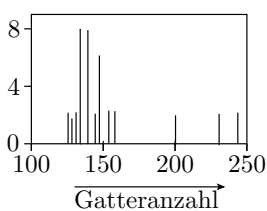
jeweils Erwartungswerte und Vorhersagbarkeit skalierbar auf die Projektgröße, Schwierigkeit, ...

Signifikante Aussagen über Vorgehensmodelle verlangen die Beobachtung tausender Projekte mit vergleichbarem Vorgehen.



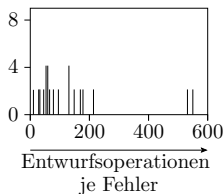
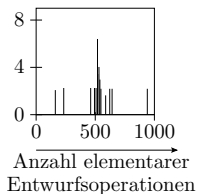
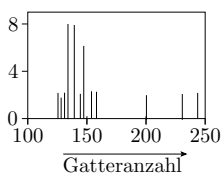
## Ein Experiment [1]

Eine Gruppe von 72 Studenten hatte die Aufgabe, aus der Beschreibung eines PLAs (**P**rogrammable **L**ogic **A**rrays) eine Gatterschaltung zu entwickeln und diese über die GUI eines CAD-Systems in den Rechner einzugeben. Für jeden Entwurf wurden die elementaren Entwurfsoperationen, die Gatteranzahl und die Entwurfsfehler gezählt. Als elementare Entwurfsoperationen galten das Anordnen eines Gatters auf dem Bildschirm, das Zeichnen einer Verbindung, ...





## Welche Rückschlüsse erlaubt das Experiment?



Angenommen, der Versuch wird genauso an anderen Hochschulen wiederholt:

- Auch hier dieselben Kenngrößen je Student bestimmen.
- Verteilung, Erwartungswert und Varianz vergleichen.
- Unterschiede statistisch relevant?

Aus den Vergleichsergebnissen ließe sich schlussfolgern, ob und an welcher Hochschule Studierende für diese Aufgabe besser ausgebildet werden. (Hat sicher noch niemand probiert.)



## Qualität und Kreativität



## Qualität und Kreativität

Qualität verlangt Fehlervermeidung. Fehlervermeidung verlangt Reproduzierbarkeit:

- eine hohe Wiederholrate gleicher oder ähnlicher Tätigkeiten,
- einzuhaltende Arbeitsabläufe,
- Protokollierung aller Unregelmäßigkeiten und Probleme, ...

Kreativität verlangt »Einzigartigkeit«:

- Einbringen neuer Konzepte,
- Ausprobieren neuer Lösungswege,
- flexible Anpassung an sich ändernde Anforderungen.

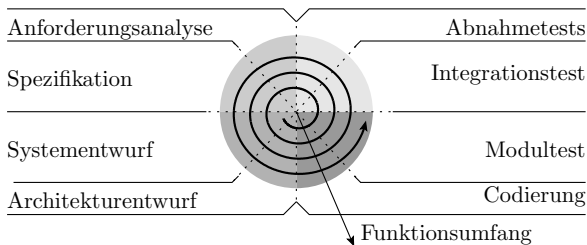
### Schlussfolgerung

Qualität und Kreativität haben entgegengesetzte Anforderungen an den Gestaltungsspielraum von Arbeitsabläufen.

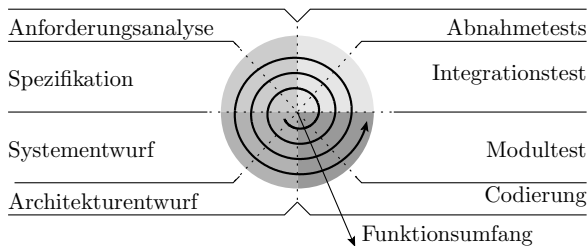
IT-Entwurf verlangt Qualität und Kreativität.

## Spiralmodell als Beispiel evolutionärer Modelle

Evolutionäre Vorgehensmodelle versuchen einen Rahmen für Projekte zu bieten, bei denen sich Kundenwünsche, Ziele, Vorgehen, ... mit dem Projekt weiterentwickeln. Weniger starre Abläufe. Mehr kreativer Gestaltungsspielraum. Beispiel Spiralmodell:



- Aufteilung einer Entwicklung auf ein mehrmaliges Durchlaufen eines Stufenmodells.



Aufteilung einer Entwicklung auf ein mehrmaliges Durchlaufen eines Stufenmodells.

- Durchlauf 1: Spezifikation von Grundanforderungen, Entwurf, Codierung, Test, ..., Abnahme und Einsatz.
- Durchlauf 2 bis  $n$ : Ideensammlung und Auswahl gewünschter Zusatzanforderungen und Änderungen. Entwurf bis Einsatz.

Innerhalb der Iteration ist der Ablauf festgeschrieben. Kreativer Freiraum in Form einer Ideensammlung für die nächste Version.





## Querverbindungen zum akademischen Alltag

Auch Lernprozesse unterliegen Vorgehensmodellen. ...

Der Bologna-Prozess (Bachelor-Master) strebt danach, Referenzabläufe zu etablieren.

Dahinter verbirgt sich die Hoffnung, dass sich mit dem Technologiegedanken im Bildungssystem ähnlich spektakuläre Fortschritte wie in Naturwissenschaft und Technik erzielen lassen:

- Vereinheitlichung der Abläufe.
- Verbesserung der Vorhersagbarkeit und Vergleichbarkeit der Bildungsergebnisse und Kosten.
- Übernahme der »Vorgehen« aus Bildungseinrichtungen mit besseren Ergebnissen von Bildungseinrichtungen mit schlechteren Ergebnissen.

Fehlervermeidung beschränkt die Kreativität. Sind Kreativitätsbeschränkungen für Universitäten akzeptabel?



## Ein Abstecher zu Lernprozessen

In der Schule und beim Erlernen praktischer Tätigkeiten werden zum erheblichen Teil Vorgehensmodelle vermittelt und trainiert:

- Rechnen, Schreiben, Handwerkern, Programmieren, ...
- Bewertung in Arbeitsmenge pro Fehler und pro Zeit.

Lernphasen:

- 1 Wissenvermittlung: anlesen, erklärt bekommen, ...
- 2 Training, bis Ergebnisse vorhersagbar.
- 3 Professionalisierung: Prozessüberwachung; Beseitigung von Vorgehensfehlern und -schwachstellen.

An Universitäten:

- Phase 1: Vorlesung, Seminare, Selbststudium, ...
- Phase 2: Übung, Klausurvorbereitung<sup>4</sup>, Praktika.
- Phase 3: Aus Zeitgründen erst in der Berufspraxis für den eigenen eingeschränkten Tätigkeitsbereich.

<sup>4</sup>Auch Bewertung in Arbeitsmenge pro Zeit und Fehler pro Arbeitsmenge.



## Querverbindung Drittmittelprojekte

- Die Professionalisierungsphase durchlaufen erst die Absolventen in der Praxis.
- Akademiker und Studenten sind selten für »fehlerarme Arbeitsabläufe« qualifiziert.
- In industriellen Software-Projekten entstehen durch Akademiker tendenziell mehr Fehler je Aufgabengröße.
- Die Kosten für die Fehlerbeseitigung trägt der Industriepartner.
- Dadurch lohnt es sich für die Industrie selten, Hochschulen und Studenten in ihr Tagesgeschäft einzubinden.
- Industrielle Studenten-Projekte dienen der Ausbildung.
- Drittmittelforschung ist wertvoll für den Knowhow-Transfer, Studien, ... aber im IT-Bereich ungeeignet für die Einbindung in die Produktentwicklung.



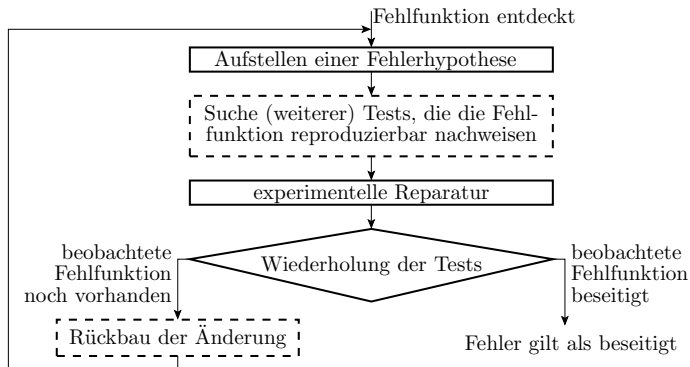
# Fehlerbeseitigung



### Experimentelle Reparatur

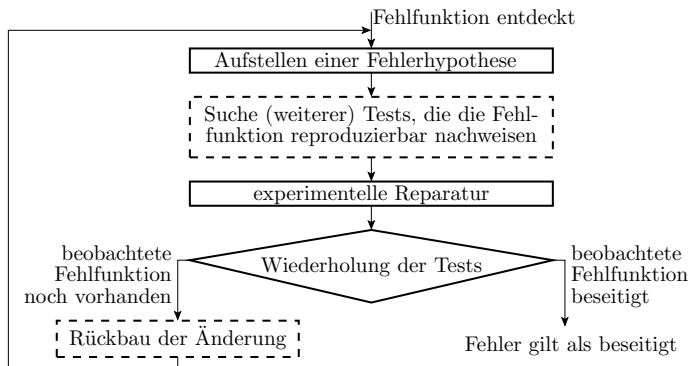
Die Sollfunktionen von IT-Systemen sind fast ausschließlich deterministisch. Gründe hierfür:

- Ergebniskontrolle durch Soll-/Ist-Vergleich,
- Fehlerbeseitigung durch »experimentelle Reparatur«, ...





## 2. Fehlerbeseitigung

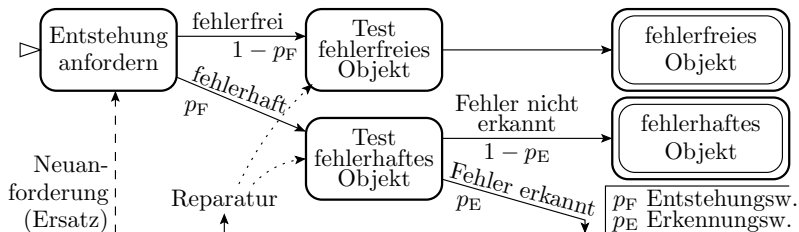


Die Erweiterung um »Suche (weiterer) Tests« ist erforderlich, wenn die bisherigen Tests das beobachtete Fehlverhalten nicht reproduzierbar nachweisen<sup>5</sup>.

Rückbau beseitigt die bei erfolglosen Reparaturversuchen entstehenden Fehler.

<sup>5</sup>Insbesondere für Fehlfunktionen, die Anwender gefunden haben.

### Experimentelle Reparatur als Markov-Prozess<sup>6</sup>



- Mit einer Wahrscheinlichkeit  $p_F$  entstehen Fehler.
- Diese werden mit einer Wahrscheinlichkeit  $p_E$  erkannt.
- Die Fehlerbeseitigung erfolgt über
  - Ersatz, modellierbar als Wiederholung des Entstehungsprozesses oder
  - Reparatur, Lokalisierung und Tausch defekter Teilsysteme.

<sup>6</sup>Ein Markov-Prozess kann nur Fehlerauftritts-, -erkennungswahrscheinlichkeiten für die Reparaturiteration von Einzelobjekten beschreiben.



Ersatz





### Ersatz vs. Reparatur

Beim Ersatz erkannter defekter Systeme vor dem Einsatz aus demselben Fertigungsprozess

- haben Original- und Ersatzsystem dieselbe zu erwartende Ausbeute  $E(Y)$ ,
- müssen im Mittel  $\frac{1}{E(Y)}$  mal so viele Systeme gefertigt oder entworfen, wie am Ende eingesetzt werden.

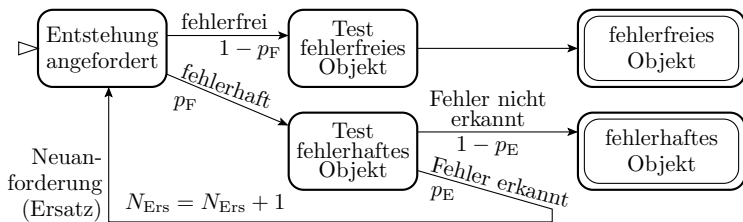
Aus diesem modellhaften Überschlag leitet sich ab:

- Die Fertigungskosten pro verkauftes System sind  $\approx \frac{1}{E(Y)}$  mal so hoch wie die Kosten für die Fertigung eines Systems.
- Ersatz ist die kostengünstigste Fehlerbeseitigung bei hoher Ausbeute<sup>7</sup> und unbezahlbar für Ausbeuten  $Y \ll 50\%$ .

---

<sup>7</sup>Spart Aufwändungen für prüf- und reparaturgerechten Entwurf, Lokalisierung und Vorratshaltung von Reparaturkapazitäten.

### Ersatz als Iteration



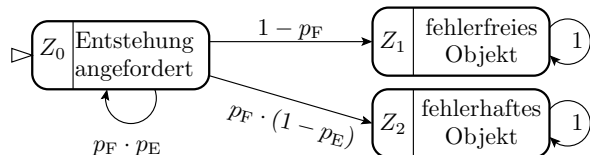
- Ersatzobjekte haben auch mit Wahrscheinlichkeit  $p_F$  Fehler.
- Diese entstehen unabhängig und sind unabhängig nachweisbar.

Insgesamt ist das Ergebnis jeder Entstehungsanforderung mit

- $1 - p_F$  ein fehlerfreies Objekt,
- $p_F \cdot (1 - p_E)$  ein nicht erkanntes fehlerhaftes Objekt,
- $p_F \cdot p_E$  eine Wiederhol- (Ersatz-) Anforderung<sup>8</sup>.

<sup>8</sup>Phantomfehler (vergl. TV\_F1, Abschn. 6.1 Überwachung) sind hier vernachlässigt.

### Modellierung als Markov-Prozess



$$E(N_{\text{Ers}}) = E(N_{\text{Ers}}) + 1$$

Nach Ersatz aller erkennbar defekten Objekte<sup>9</sup> :

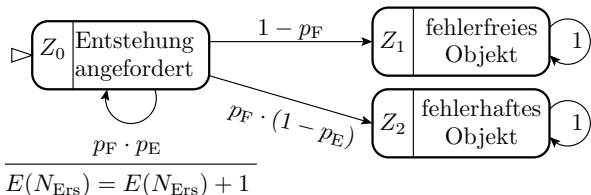
$$\lim_{n \rightarrow \infty} (p_{Z0}) = 0$$

$$\lim_{n \rightarrow \infty} (p_{Z1}) = (1 - p_F) \cdot \sum_{n=0}^{\infty} (p_F \cdot p_E)^n = \frac{1 - p_F}{1 - p_F \cdot p_E}$$

$$\lim_{n \rightarrow \infty} (p_{Z2}) = 1 - \lim_{n \rightarrow \infty} (p_{Z1}) = 1 - \frac{1 - p_F}{1 - p_F \cdot p_E} = \frac{p_F \cdot (1 - p_E)}{1 - p_F \cdot p_E}$$

<sup>9</sup>Summenformel der geometrischen Reihe:  $\sum_{n=0}^{\infty} a_0 \cdot q^n = \frac{a_0}{1-q}$

## Abschätzbare Kenngrößen



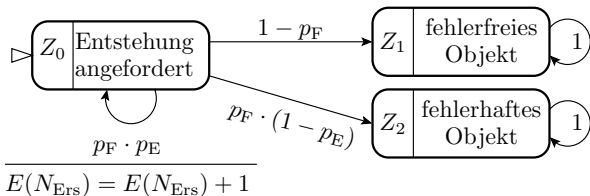
Zu erwartender Fehleranteil nach Ersatz defekter Objekte:

$$E(DL_{\text{Ers}}) = \lim_{n \rightarrow \infty} (p_{Z2}) = \frac{p_F \cdot (1 - p_E)}{1 - p_F \cdot p_E} \quad (1)$$

Wahrscheinlichkeit, dass Fehler nicht beseitigt werden<sup>10</sup>:

$$p_{\text{NBes}} = \frac{E(DL_{\text{Ers}})}{E(DL_{\text{EP}})} = \frac{\frac{p_F \cdot (1 - p_E)}{1 - p_F \cdot p_E}}{p_F} = \frac{1 - p_E}{1 - p_F \cdot p_E}$$

<sup>10</sup>Verhältnis der zu erwartenden Fehleranzahl ohne und mit Ersatz aller erkennbar defekten Objekte.



Die zu erwartende Anzahl der Ersetzungen:

$$E(N_{\text{Ers}}) = \sum_{n=1}^{\infty} (p_F \cdot p_E)^n = \frac{p_F \cdot p_E}{1 - p_F \cdot p_E} \quad (2)$$

Zu erwartende Ausbeute<sup>11</sup>:

$$E(Y) = \frac{1}{E(N_{\text{Ers}}) + 1} = 1 - p_F \cdot p_E \quad (3)$$

<sup>11</sup>Die zu erwartende Anzahl der pro funktionierendes System zu fertigenden Systeme ist um eins größer als zu erwartende Anzahl der Ersetzungen und gleich dem Kehrwert der zu erwartenden Ausbeute.

## Beispielaufgabe



Wie groß ist für eine zu erwartende Schaltkreisausbeute von  $E(Y) = 10\%$ ,  $30\%$ ,  $50\%$ ,  $80\%$  und  $90\%$  und eine zu erwartende Fehlerüberdeckung  $E(FC) = p_E$  von  $90\%$ ,  $99\%$  und  $99,9\%$

- 1 die zu erwartende Anzahl der aussortierten Schaltkreise je als gut befundener Schaltkreis  $E(N_{Ers})$  und
- 2 die Wahrscheinlichkeit  $p_F$ , dass ein Schaltkreis vor dem Aussortieren fehlerhaft ist?
- 3 Wie groß ist der zu erwartende Fehleranteil  $E(DL_{Ers})$  der als gut befundenen Schaltkreise für  $p_F = 100\%$ ,  $90\%$ ,  $70\%$ ,  $50\%$ ,  $20\%$  und  $10\%$  und die Werte der Erkennungswahrscheinlichkeit  $p_E$  oben?

## Lösung Aufgabenteile 1 und 2

- 1** Die zu erwartende Anzahl der Ersetzungen je guter Schaltkreis ist nach Gl. 3:

$$E(N_{\text{Ers}}) = \frac{1}{E(Y)} - 1$$

$Y$	10%	30%	50%	80%	90%
$E(N_{\text{Ers}})$	9	2,33	1	0,25	0,11

- 2** Die Wahrscheinlichkeit  $p_F$ , dass ein Schaltkreis vor dem Aus-sortieren fehlerhaft ist, mit  $p_E = E(FC)$  ist nach Gl. 3:

$$p_F = \frac{1 - E(Y)}{E(FC)}$$

$E(FC)$	$E(Y) = 10\%$	...=30%	...=50%	...=80%	...=90%
90%	100,0%	77,8%	55,6%	22,2%	11,1%
99%	90,9%	70,7%	50,50%	20,2%	10,1%
99,9%	90,1%	70,1%	50,1%	20,0%	10,0%



## Lösung Aufgabenteil 3

- 4 Der zu erwartende Fehleranteil  $E(DL_{\text{Ers}})$  der als gut befundenen Schaltkreise nach Ersatz aller erkennbar fehlerhaften Objekte nach Gl. 1 beträgt mit  $p_E = E(FC)$ :

$$E(DL_{\text{Ers}}) = \frac{p_F \cdot (1 - E(FC))}{1 - p_F \cdot E(FC)}$$

	$E(FC) = 90\%$	$E(FC) = 99\%$	$E(FC) = 99,9\%$
$p_F = 100\%$	100,0%	100,0%	100,0%
$p_F = 90\%$	47,4%	8,26%	8920 dpm
$p_F = 70\%$	18,9%	2,28%	2328 dpm
$p_F = 50\%$	9,09%	9901 dpm	999 dpm
$p_F = 20\%$	2,43%	2494 dpm	250 dpm
$p_F = 10\%$	1,10%	1110 dpm	111 dpm





Zu erwartender Fehleranteil  $E(DL_{\text{Ers}})$  nach Ersatz aller erkennbar fehlerhaften Objekte:

	$E(FC) = 90\%$	$E(FC) = 99\%$	$E(FC) = 99,9\%$
$p_F = 100\%$	100,0%	100,0%	100,0%
$p_F = 90\%$	47,4%	8,26%	8920 dpm
$p_F = 70\%$	18,9%	2,28%	2328 dpm
$p_F = 50\%$	9,09%	9901 dpm	999 dpm
$p_F = 20\%$	2,43%	2494 dpm	250 dpm
$p_F = 10\%$	1,10%	1110 dpm	111 dpm

- Für getestete Schaltkreise wird ein Fehleranteil  $DL_{\text{Ers}} \approx 10^{-4}$  angegeben. Bei einer typischen Ausbeute von  $Y = 50\%$  bis  $90\%$  verlangt das eine Fehlerüberdeckung von  $FC \approx 99,9\%$ .
- Typische Haftfehlerüberdeckungen für Schaltkreistests  $FC_{\text{sa}} = 95\%$  bis  $99\%$ . Offenbar gilt  $FC \gg FC_{\text{sa}}$ .



# Reparatur



### Fehlerbeseitigung durch Reparatur

Bei Reparatur werden nur vermutlich defekte Teil des Systems getauscht<sup>12</sup>. Zu ersetzende Teilsysteme:

- sind billiger als zu ersetzende Gesamtsysteme und
- haben einen kleinerer Fehleranteil (weniger Mehrfachersetzungen).

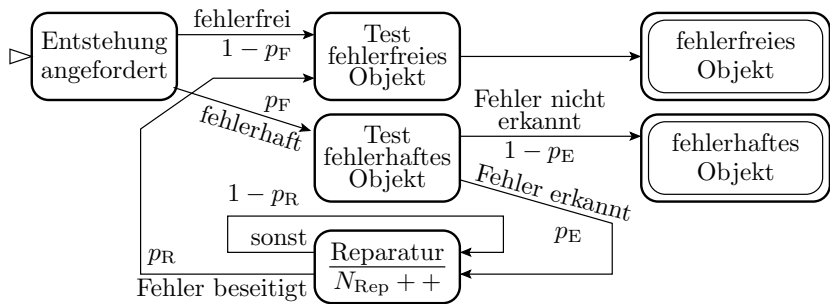
Dafür verlangt Reparatur Zusatzaufwendungen:

- Reparaturgerechter Entwurf (modulare Austauschbarkeit),
- Fehlerlokalisierung und
- Organisationseinheiten + Personalkapazität für Reparatur (bei Software für Wartung).

Für Systeme mit Ausbeute  $E(Y) \gg 50$  unrentabel.

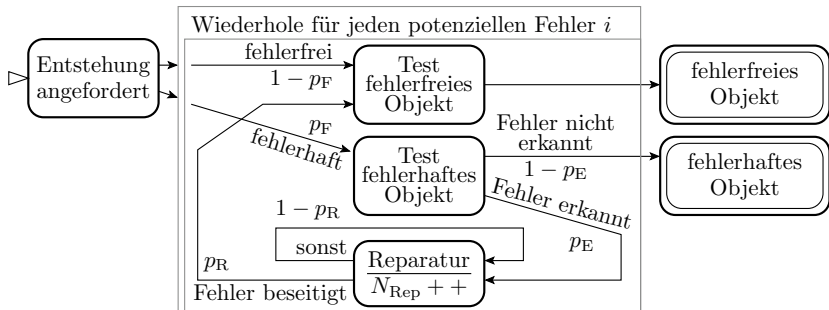
<sup>12</sup>Für Software »tauschen von Anweisungen«

### Beseitigungsiteration für einen Fehler

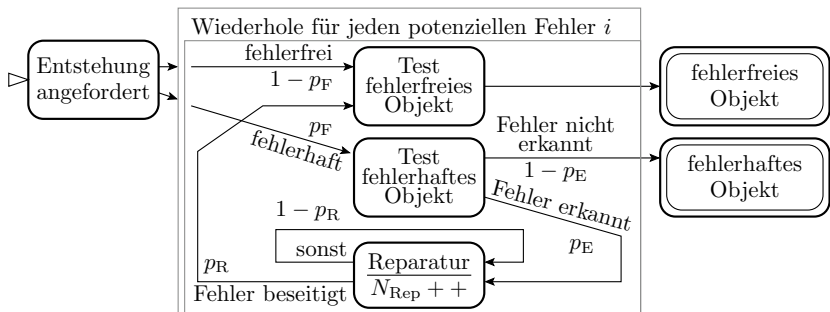


- Bei einem erkennbaren Fehler wird solange mit einer Erfolgswahrscheinlichkeit  $p_R$  repariert, bis das vom Test nachweisbare Fehlverhalten beseitigt ist.
- Es sei unterstellt, dass die zu erwartende Anzahl der durch Reparatur neu entstehenden Fehler proportional zur Anzahl  $N_{Rep}$  der Reparaturversuche zunimmt.

### Systeme mit mehr als einem zu erwartenden Fehler



- Je eine Markov-Kette für die Beseitigungsiteration eines potentiellen Fehlers  $i$  .
- Zusätzlich Zähler  $N_{\text{Rep},i}$  zur Bestimmung der zu erwartenden Anzahl der Reparaturversuche.



Aus dem Modell abzuschätzbar:

- zu erwartende Anzahl der Reparaturen  $E(N_{\text{Rep}})$  und
  - zu erwartende Fehleranzahl  $E(\varphi_{\text{Rep}})$  nach allen Reparaturen
- in Abhängigkeit von der
- zu erwart. Fehleranz. aus dem Entstehungsprozess  $E(\varphi_{\text{EP}})$ ,
  - der Fehlererkennungswahrscheinlichkeit  $p_E$  und
  - der Erfolgswahrscheinlichkeit der Reparatur  $p_R$ .

## Fehleranzahl nach der Beseitigungsiteration

Zu beseitigen sind

- $\varphi_{EP}$  Fehler aus dem Entstehungsprozess und
- $\varphi_{ERep}$  bei Reparaturversuchen entstehende Fehler:

$$E(\varphi_{ERep}) = E(N_{Rep}) \cdot E(\varphi_{EP}) \cdot \eta_{Rep}$$

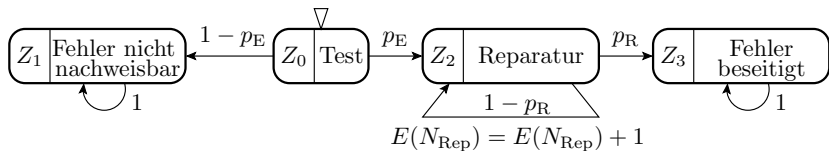
( $N_{Rep}$  – Reparaturversuche je Fehler;  $\varphi_{EP}$  – Fehleranzahl Entstehungsprozess;  $\eta_{Rep}$  – Fehlerentstehungsrate, neu entstehende Fehlern je Reparaturversuch).

Zu erwartende Fehleranzahl nach der Beseitigungsiteration:

$$E(\varphi_{Rep}) = \left( E(\varphi_{EP}) + \underbrace{E(N_{Rep}) \cdot E(\varphi_{EP}) \cdot \eta_{Rep}}_{E(\varphi_{ERep})} \right) \cdot (1 - p_B) \quad (4)$$

( $p_B$  – Beseitigungswahrscheinlichkeit, abschätzbar über eine Markov-Kette der Reparaturiteration für vorhandene Fehler).

## Markov-Prozess für einen vorhandenen Fehler



- Wahrscheinlichkeit der Beseitigung eines vorhandenen Fehlers ist gleich der Erkennungswahrscheinlichkeit:

$$p_B = p_{Z3} = p_E \cdot p_R \cdot \sum_{n=1}^{\infty} (1 - p_R)^n = p_E$$

- Zu erwartende Anzahl der Reparaturversuche je zu beseitigender Fehler ist das Verhältnis aus Erkennungs- und Erfolgswahrscheinlichkeit der Reparatur:

$$E(N_{\text{Rep}}) = p_E \cdot \sum_{n=1}^{\infty} (1 - p_R)^n = \frac{p_E}{p_R}$$





## Fehleranzahl nach der Beseitigungsiteration

Gl. 4 mit  $p_B = p_E$  und  $E(N_{\text{Rep}}) = \frac{p_E}{p_R}$ :

$$E(\varphi_{\text{Rep}}) = \left( E(\varphi_{\text{EP}}) + E(\varphi_{\text{EP}}) \cdot \frac{p_E}{p_R} \cdot \eta_{\text{Rep}} \right) \cdot (1 - p_E)$$

Das entscheidende Gütemaß einer Reparatur  $Q_{\text{Rep}}$  ist offenbar das Verhältnis aus der Erfolgswahrscheinlichkeit der Reparatur  $p_R$  und der zu erwartenden Anzahl der neu entstehenden Fehler je Reparaturversuch  $\eta_{\text{Rep}}$ :

$$Q_{\text{Rep}} = \frac{p_R}{\eta_{\text{Rep}}} \quad (5)$$

Mit der so definierten Reparaturgüte:

$$E(\varphi_{\text{Rep}}) = E(\varphi_{\text{EP}}) \cdot \left( 1 + \frac{p_E}{Q_{\text{Rep}}} \right) \cdot (1 - p_E) \quad (6)$$



Die Wahrscheinlichkeit, dass Fehler nicht beseitigt werden, ist abschätzungsweise das Verhältnis aus der Fehleranzahl in Objekten vor- und nach der Reparaturiteration:

$$p_{\text{NBes}} = \frac{E(\varphi_{\text{Rep}})}{E(\varphi_{\text{EP}})} = \left(1 + \frac{p_{\text{E}}}{Q_{\text{Rep}}}\right) \cdot (1 - p_{\text{E}})$$

Im Idealfall  $Q_{\text{Rep}} \rightarrow \infty$  ist  $p_{\text{NBes}}$ , dass ein Fehler nicht beseitigt wird, eins abzüglich der Erkennungswahrscheinlichkeit  $p_{\text{E}}$ :

$$p_{\text{NBes}} = 1 - p_{\text{E}}$$

Bei einer realen Reparatur ist die Fehleranzahl nach der Reparatur maximal um den Faktor  $1 + Q_{\text{Rep}}^{-1}$  größer. Vergrößerung ab Reparaturgüten  $Q_{\text{Rep}} > 2 \dots 10$  gegenüber

- zufälligen Abweichungen der tatsächlichen Fehleranzahl vom Erwartungswert und
- modellbedingten systematischen Abweichungen

vernachlässigbar.

## Beispielaufgabe



Wie groß ist die Beseitigungswahrscheinlichkeit  $p_{\text{Bes}} = 1 - p_{\text{NBes}}$  für Fehler durch Reparaturiterationen mit den Erkennungswahrscheinlichkeiten  $p_{\text{E}} \in \{50\%, 80\%, 90\%, 99\%, 99,9\%\}$  und den Reparaturgüten  $Q_{\text{Rep}} \in \{0,5, 1, 2, 10\}$  ?



## Lösung

Gesucht  $p_{\text{Bes}} = \left(1 + \frac{p_E}{Q_{\text{Rep}}}\right) \cdot (1 - p_E)$  für

$p_E \in \{50\%, 80\%, 90\%, 99\%, 99,9\%\}$  und  $Q_{\text{Rep}} \in \{0,5, 1, 2, 10\}$  ?

Octave- (Matlab-) Programm:

```
pE = [0.5 0.8 0.9 0.99 0.999];
QRep = [0.5 1 2 10];
for idx_Q=1:4
    for idx_p=1:5
        V(idx_p)=(1-(1+pE(idx_p)/QRep(idx_Q))...
            *(1-pE(idx_p)))*100;
    end
    printf(' %4.1f | %7.1f %%%6.1f %%%6.1f %%%6.2f %%%6.3f %%\n' , ...
        QRep(idx_Q), V);
end
```



Tabellierte Funktionsergebnisse für  $p_{\text{Bes}} = f(Q_{\text{Rep}}, p_E)$ :

$Q_{\text{Rep}}$	$p_E = 50\%$	$p_E = 80\%$	$p_E = 90\%$	$p_E = 99\%$	$p_E = 99,9\%$
0,5	0%	48,0%	72,0%	97,02%	99,70%
1,0	25,0%	64,0%	81,0%	98,01%	99,80%
2,0	37,5%	72,0%	85,5%	98,50%	98,85%
10,0	47,5%	78,40%	89,1%	98,9%	99,89%

Ein interessanter Grenzfall:

- Es ist möglich, dass das System nach der Beseitigungsiteration noch genauso viele oder mehr Fehler enthält.
- Keinen der verbleibenden Fehler weist der Test nach.
- Da Tests tendenziell Fehler, die oft Fehlfunktionen verursachen besser erkennen, verbessern auch Reparaturiterationen ohne Verringerung der Fehleranzahl die Zuverlässigkeit.



## Typische studentische Programmierarbeiten

$$E(\varphi_{\text{Rep}}) = E(\varphi_{\text{EP}}) \cdot \left(1 + \frac{p_{\text{E}}}{Q_{\text{Rep}}}\right) \cdot (1 - p_{\text{E}})$$

Fall A: wenige Testbeispiele, brauchbarer Reparaturprozess, z.B.  
 $p_{\text{E}} = 30\%$  erkennbare Fehler,  $Q_{\text{Rep}} = 2$  beseitigte je neuer Fehler:

$$\frac{E(\varphi_{\text{Rep}})}{E(\varphi_{\text{EP}})} = \left(1 + \frac{30\%}{2}\right) \cdot (1 - 30\%) = 80,5\%$$

- Reduktion der Fehleranzahl auf  $\approx 80\%$ . Davon sind 70% nicht erkannte ursprüngliche und 10% bei der Reparatur entstandene Fehler.
- Erkannt und beseitigt werden die am meisten störenden Fehler (siehe Zufallstest). Es bestehen Chancen, dass das System einen Abnahmetest mit 1 bis 2 neuen zufälligen Testbeispielen erfolgreich passiert.

$$E(\varphi_{\text{Rep}}) = E(\varphi_{\text{EP}}) \cdot \left(1 + \frac{p_{\text{E}}}{Q_{\text{Rep}}}\right) \cdot (1 - p_{\text{E}})$$

Fall B: Entwurf wird beherrscht, aber Test und Reparaturtechniken nicht, z.B.  $p_{\text{E}} = 25\%$  erkennbare Fehler,  $Q_{\text{Rep}} = 0,5$  beseitigte je neuer Fehler:

$$\frac{E(\varphi_{\text{Rep}})}{E(\varphi_{\text{EP}})} = \left(1 + \frac{25\%}{0,5}\right) \cdot (1 - 25\%) = 112,5\%$$

- Das System enthält nach Test und Beseitigung der erkannten Fehler mehr Fehler als davor.
- Statt sie zu beseitigen »versteckt« die Beseitigungsiteration die Fehler.
- Begleitsymptom: übermäßig lange Test- und Reparaturphase.
- Ein Abnahmetest mit den für die Kontrolle studentischer Leistungen üblichen 1 bis 2 neuen zufälligen Testbeispielen, versagt oft.

$$E(\varphi_{\text{Rep}}) = E(\varphi_{\text{EP}}) \cdot \left(1 + \frac{p_{\text{E}}}{Q_{\text{Rep}}}\right) \cdot (1 - p_{\text{E}})$$

Fall C: Studierender ist mit seiner Aufgabe überfordert

$$Q_{\text{R}} \rightarrow 0$$

- Der Studierende »schleust die Fehler« mit sehr vielen Reparaturversuchen durch die Tests und
- lässt Tests, die auch nach vielen Fehlerbeseitigungsversuchen immer noch versagen, weg  $p_{\text{E}} \rightarrow 0$ .
- Ein Abnahmetest mit 1 bis 2 neuen zufälligen Testbeispielen versagt immer.

Warum ist die Übertragung einer Entwicklungsaufgabe an Studierende, z.B. im Rahmen von Drittmittelprojekten für den Auftraggeber riskant?







## Fehlerlokalisierung



### Wenige tauschbare Komponenten

Ein reparaturgerechtes System hat eine hierarchische Struktur aus tauschbaren Komponenten, z.B.

- 1 Ebene: Austauschbare Geräte.
- 2 Ebene: Austauschbare Baugruppen.
- 3 Ebene: Austauschbare Schaltkreise.

Fehlerlokalisierung durch systematisches Tauschen:

Hierarchie der Hardware

Geräte



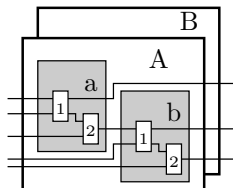
Baugruppen



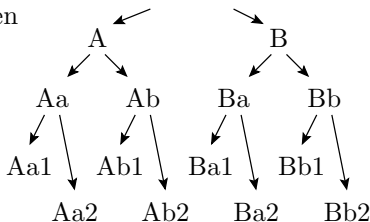
Schaltkreise



hierarchisches System mit tauschbaren Komponenten



Tauschbaum





Typisches Mechanikervorgehen:

- Grobabschätzung, welches Rechnerenteil defekt sein könnte aus den Fehlersymptomen.
  - Kontrolle der Steckverbinder auf Kontaktprobleme durch Abziehen, Reinigen, Zusammenstecken, Ausprobieren.
  - Tausch möglicherweise defekter Baugruppen gegen Ersatzbaugruppen, Ausprobieren, ...
- 

Voraussetzungen:

- Wiederholbare Tests, die den Fehler nachweisen.
- Ausreichend Ersatzteile. Allgemeine Mechnikerkenntnisse<sup>14</sup>.

Wichtig ist der Rückbau nach jedem erfolglosen Reparaturversuch.

Warum?

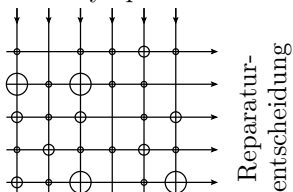
Günstig ist der Tausch der Hälfte, von der fehlerhaften Hälfte wieder der Hälfte, ... Warum?

<sup>14</sup>Verständnis der Funktion des zu reparierenden Systems nicht zwingend.

### Pareto-Prinzip

- Pareto-Prinzip: Produkte haben Schwachstellen. Richtwert: 80% der Probleme geht auf 20% der Ursachen zurück.
- Zählen der erfolgreichen und erfolglosen Reparaturversuche.
- Bei Alternativen, Beginn mit der erfolgsversprechendsten Reparaturmöglichkeit.

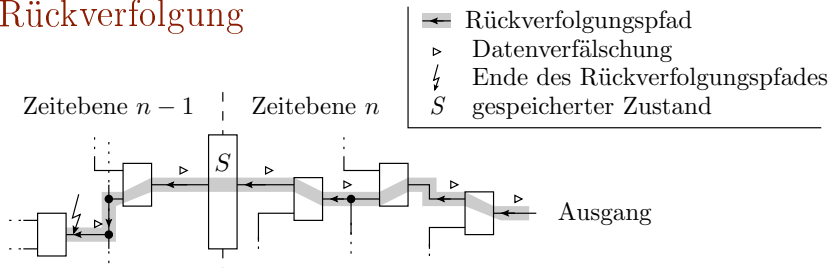
Fehlersymptom



◎ bisherige Häufigkeit, mit der die Reparaturrentscheidung für das Symptom richtig war

- Nach erfolglosen Reparaturversuchen Rückbau, z.B. vor der Änderung Bearbeitungsstatus speichern und nach erfolgloser Änderung Bearbeitungsstatus zurücksetzen.

## Rückverfolgung



- Ausgehend von einer erkannten falschen Ausgabe Rückwärtsuche nach dem Entstehungsort, gegebenenfalls über Zeitebenen.
- Suche endet am Teil-Service, der aus richtigen Eingaben falsche Ergebnisse erzeugt.
- Tausch oder weiter hierarchisch absteigende Suche.
- Verfälschungsursache kann außer dem erzeugenden Service auch ein anderer, z.B. mit fehlgeleitetem Schreibzugriff, sein.



Tauschstrategie am Entstehungsort der Verfälschung:

- systematisch, binärer Tauschbaum,
- beginnend mit am häufigsten zu beobachtenden Problemquellen.

Rückverfolgung über Zeitebenen:

- Neuinitialisierung und Testwiederholung bis zum Zeitschritt der Zwischenergebnisberechnung.
- Aufzeichnung der Zeitverläufe potentiell verfälschter Variablen und Signale (Logikanalyse, Trace-Aufzeichnung).

Bei einem nichtdeterministischen Fehlverhalten ist eine Rückverfolgung nur anhand aufgezeichneter Daten möglich.



## Benutzer als Tester



### Fehler und Fehlfunktionen im Einsatz

Bei großen IT-Systemen bleiben nach den

- statischen Tests,
- fehlerorientiert berechneten Tests und
- einer bezahlbaren Anzahl von Zufallstests

eine erhebliche Anzahl von Fehler unerkant. Diese verursachen im Einsatz beim Nutzer Fehlfunktionen.

Die Nutzer reagieren auf Fehlfunktionen mit:

- der Suche nach Workarounds<sup>15</sup> oder
- Änderungswünschen (Change Requests) an den Hersteller.
- Notlösung Ersatz<sup>16</sup>.

---

<sup>15</sup>Im Idealfall Kooperation zwischen Nutzern und Hersteller, z.B. eine Knowhow-Weitergabe über gefundene Workarounds auf FAQ-Seiten.

<sup>16</sup>Da Ersatzsysteme meist ähnlich viele Fehler enthalten, in der Regel nicht die erfolgsversprechendste Lösung.





### Benutzer als Tester

Bei IT-Systemen mit sehr vielen Nutzern über Jahre

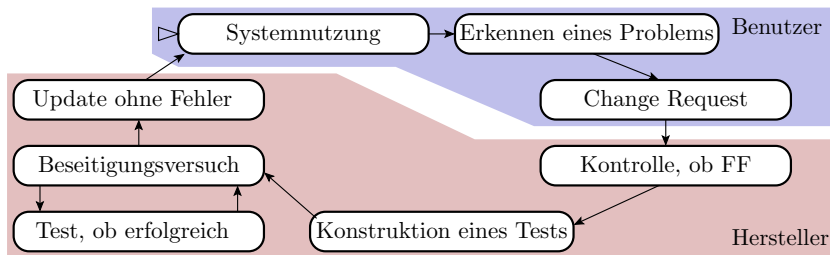
- werden in Summe um viele Zehnerpotenzen mehr Funktionsfälle als vom Hersteller ausprobiert.
- Werden um Größenordnungen mehr Fehler sichtbar.

Nutzer als Tester:

- Geziele (automatisierte) Erfassung der Workarounds und Änderungswünsche.
- Suche von Tests für die beschriebenen Fehlverhalten.
- Experimentelle Reparatur zur Fehlerbeseitigung.

In dieser Iteration können sehr fehlerarme Systeme entstehen, die so gut und zuverlässig funktionieren, dass sie im Flusse der technischen Weiterentwicklung nur noch sehr zögerlich ersetzt oder um neue Features erweitert werden (siehe Y2K-Problem<sup>17</sup>).

<sup>17</sup>Jahr 2000-Problem.



Zusatzaufwand Benutzer bei Fehlverhalten:

- Beschreibung des Fehlverhaltens,
- Beschreibung aller Randbedingungen, unter denen es aufgetreten ist.

Zusatzaufwand des Herstellers zur normalen Fehlerbeseitigung:

- Verwaltung der Informationen über Fehlverhalten.
- Kontrolle, ob es sich wirklich um Fehler handelt und Priorisierung, ob und wie dringend zu beseitigen.
- Für zu beseitigende Fehler, Suche von Tests.



### Automatische Fehlerreports

Nutzer und Hersteller sind in der Regel mit der manuellen Erstellung, Übergabe und Abarbeitung der Änderungswünsche überfordert. Funktionserweiterung des Anwendersystems um eine automatische Generierung von Fehlerreports:

- Kontrollen mit Fehlerbehandlung,
- Vorklassifizierung von Ort und Ursache der FF,
- Sammeln einer auf die FFs abgestimmten Datenmenge:  
Programm-, Betriebssystem-, Treiber- und Hardware- Versionen, Register- und Variablenwerte, Aufruf-Stack, ...
- Datenübermittlung an den Hersteller (nach System-Crash oder Offline-Betrieb beim nächsten Netzzugang).

Große Menge Zusatzfunktionalität, die natürlich auch Fehler enthält und dadurch die Zuverlässigkeit beeinträchtigt.



### Abarbeitung der Fehlerreports beim Hersteller

Bei großer Anwenderzahl und Wochen bis zur Fehlerbeseitigung verursacht jeder Fehler im Mittel sehr viel Fehlerreports.

- Zusammenfassen zu »Schubladen« (Buckets) die vermutlich denselben Fehler zur Ursache haben.
- Beschränkung der Datenerfassung auf eine ausreichende Stichprobe von Fehlerreports.
- Priorisierung und Auswahl der abzuarbeitenden Buckets.
- Suche von Tests, die das beobachtete Fehlverhalten nachweisen.
- Vereinfachung der Tests durch Ausschluss aller Nicht-Einflüsse auf das Fehlverhalten (Eingaben, Programm-, Betriebssystem-, Treiberversion, ...).
- Experimentelle Reparatur.
- Bereitstellung fehlerbereinigter Programmversionen.

Zusammenfassen von Fehlerreports zu Buckets<sup>18</sup>

Heuristic	Impact	Description
program_name	Expanding	Include program name in bucket label.
program_version	Expanding	Include program version.
program_timestamp	Expanding	Include program binary timestamp.
module_name	Expanding	Include faulting module name in label.
module_version	Expanding	Include faulting module version.
module_offset	Expanding	Include offset of crashing instruction in fault module.
exception_code	Expanding	Cause of unhandled exception.
bugcheck_code	Expanding	Cause of system crash (kernel only)
hang_wait_chain	Expanding	On hang, walk chain of threads waiting on synchronization objects to find root.
pc_on_stack	Condensing	Code was running on stack, remove module offset.
unloaded_module	Condensing	Call or return into memory where a previously loaded module has unloaded.
custom_parameters	Expanding	Additional parameters generated by application-specific client code.
assert_tags*	Condensing	Replace module information with unique in-code assert ID.

<sup>18</sup>WER- (Windows Error Report) Service. Teils in die Anwendung einprogrammiert, teils Vorklassifizierung auf dem Erfassungs-Server.



### Fehlerreport-Daten (WER-Service)

Ein Windows-System berechnet bei einem Versagen automatisch ein Bucket-Label und sammelt zu übergebende Daten in einem CAB-File. Die Datenauswahl ist bucket-weise anpassbar:

- Minidump (Register, Stack, ausgewählte Daten- und Programmspeicherbereiche).
- mit Treiberdaten, Hardware-Konfiguration, ...
- Dateien, weitere Speicherbereiche, ...

Interaktion des Nutzersystems bei einem Versagen:

- Nutzsysteem sendet nur den Bucket-Label.
- System fordern nur bei Bedarf weitere Daten an<sup>19</sup>.
- Abschließend sendet der WER-Service dem Nutzer eventuell bekannte Problemlösungen (Verweise auf Bug-Fixes).

<sup>19</sup>Progressive Datensammlung. Datenübernahme nur für eine Stichprobe von Reports je Bucket. Danach nur noch zählen.



## Reifen als Zufallsprozess

Die kontinuierliche Fehlerbeseitigung im Einsatz verhält sich ähnlich wie ein Zufallstest. Ein Fehler muss im Mittel  $a$ -mal nachgewiesen werden, bevor er beseitigt wird:

$$p_{i.\text{Beseit}}(n) = 1 - e^{-\frac{n \cdot p_{i.\text{nachw}}}{a}}$$

Für die aus dem Pareto-Prinzip abgeleitete FHSF-Funktion:

$$H(x) = \begin{cases} 0 & \text{für } x < x_0 \\ c \cdot x^{-(k+1)} & \text{für } x \geq 0 \end{cases} \quad \text{mit } k > 0$$

und Testlängen  $n \gg a \cdot x_0$  ergibt sich:

$$H(x, n) = c \cdot x^{-(k+1)} \cdot e^{-\frac{n}{a \cdot x}}$$

wobei die Testsatzlänge  $n$  viel größer als bei Hersteller möglich ist.



## Fehleranzahl und Zuverlässigkeit

Die zu erwartende Fehleranzahl ist das Integral über die FHSF-Funktion:

$$E(\varphi(n)) = \int_0^{\infty} c \cdot x^{-(k+1)} \cdot e^{-\frac{n}{a \cdot x}} \cdot dx = \frac{c \cdot \Gamma(k)}{\left(\frac{n}{a}\right)^k}$$

Sie nimmt mit der  $k$ -ten Potenz der Testdauer ab,  $a$  kürzt sich raus:

$$E(\varphi(n)) = E(\varphi(n_0)) \cdot \left(\frac{n_0}{n}\right)^k$$

Die Wahrscheinlichkeit für durch Fehler verursachte FF nimmt mit der  $k + 1$ -ten Potenz ab:

$$p_{\text{FFF}}(n) = \int_0^{\infty} \frac{H(x)}{x} \cdot e^{-\frac{n}{a \cdot x}} \cdot dx = \frac{a \cdot k \cdot E(\varphi(n))}{n} \sim n^{-(k+1)}$$

Die fehlerbezogene Zuverlässigkeit als Kehrwert nimmt mit der  $k + 1$ -ten Potenz zu:

$$Z_{\text{F}}(n) = Z_{\text{F}}(n_0) \cdot \left(\frac{n}{n_0}\right)^{k+1}$$



## Beispielaufgabe



Ein System wurde vor dem Einsatz mit  $n_0 = 10^5$  Zufallswerten mit vergleichbarem Operationsprofil wie im Einsatz getestet und erkannte Fehler beseitigt.

Zum Nutzungsbeginn betrug die fehlerbezogene Teilzuverlässigkeit  $Z_F(n_0) = 10^2$  korrekte Service-Leistungen je Fehlfunktion. Das System wird bei  $N_{\text{Nutzer}} = 100$  Nutzern eingesetzt, die je im Mittel pro Tag  $N_{\text{SL/Tag}} = 10^3$  Service-Leistungen abarbeiten und im Mittel bei jeder 10-ten Fehlfunktion ein »Change-Request« an den Hersteller schicken.

Der Hersteller beseitigt im Mittel 0,2 Fehler je Change-Request.

Wie lange dauert es abschätzungsweise, bis sich die fehlerbezogene Zuverlässigkeit des Systems ver Hundertfacht hat? Der Exponent der FHSF-Funktion sei mindestens  $k \geq 0,3$ .



## Lösung

Während der  $n_0$  Herstellertest werden die Fehler bei der ersten und bei den Anwender-Service-Leistungen im Mittel erst nach der  $a = 50$ -sten verursachten FF beseitigt. Ins Verhältnis zu  $n_0$  setzbare Service-Anzahl:

$$n = n_0 + \frac{N_{\text{Nutzer}} \cdot n_{\text{Tage}} \cdot N_{\text{SL/Tag}}}{a}$$

Für die hundertfache Zuverlässigkeit:

$$\frac{Z_F(n)}{Z_F(n_0)} = 100 = \left(\frac{n}{n_0}\right)^{k+1}$$

$$n = n_0 \cdot 100^{\frac{1}{1+k}} = n_0 + \frac{N_{\text{Nutzer}} \cdot n_{\text{Tage}} \cdot N_{\text{SL/Tag}}}{a}$$

$$n_{\text{Tage}} = \frac{a \cdot n_0 \cdot \left(100^{\frac{1}{1+k}} - 1\right)}{N_{\text{SL/Tag}} \cdot N_{\text{Nutzer}}} = \frac{50 \cdot 10^5 \cdot \left(100^{\frac{1}{1,3}} - 1\right)}{10^3 \cdot 100} = 1,68 \cdot 10^3$$



# Ausfälle



### Ausfälle

- Hardware und Mechanik unterliegt einem Verschleiß, der zu Ausfällen führen kann. Bei einem Ausfall entsteht ein Fehler.
- Im Gegensatz zu den nicht nachgewiesenen Herstellungsfehlern haben neue Fehler durch Ausfälle eine ähnliche FHSF-Funktion wie ungetestete Systeme, d.h. sie verursachen im Mittel weit häufiger Fehlfunktionen, aber nicht immer komplette Funktionsunfähigkeit.
- Eine Sonderstellung haben Frühausfälle. Ihre Ursache sind Beinahefehler<sup>20</sup>, die den Verschleiß beschleunigen. Für sie haftet der Hersteller (Garantiepflicht, Produkthaftung).

---

<sup>20</sup>Materialrisse, kalte Lötstellen, ...



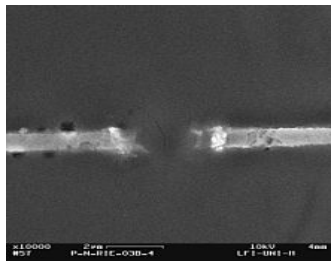
# Ausfallmechanismen

## Verschleiß elektronischer Bauteile

Langsam ablaufende physikalische Vorgänge:

- Korrosion (Stecker, Schalter, Isolationen, Leiterbahnen, ...).
- Elektromigration: strombedingte Wanderung von Metalatomen bei hohen Stromdichten.
- Gateoxiddurchschlag: Hochschaukelnde Tunnelströme, Ladungseinlagerung bis zum lokalen Schmelzen des Oxids. Bildung von Kurzschlüssen. Phänomen: Zunahme des Stromverbrauchs über Monate bis zum Ausfall.
- Parameterdrift: Widerstandswerte, Kapazitäten, Schwellspannungen etc.

Verbesserung Fertigung, Material etc.  $\Rightarrow$  weniger Ausfälle





## Kenngrößen des Ausfallverhaltens

- Lebensdauer  $t_L$ : Zeit vom Beanspruchungsbeginn bis zum Ausfall. Zufallsgröße.
- Überlebenswahrscheinlichkeit: Wahrscheinlichkeit, dass ein System zu einem Zeitpunkt  $t$  noch »lebt«:

$$R(t) = P(t < t_L)$$

- Ausfallrate<sup>21</sup>  $\lambda$ : Relative Abnahme der Überlebenswahrscheinlichkeit mit der Zeit:

$$\lambda(t) = -\frac{1}{R(t)} \cdot \frac{dR(t)}{dt}$$

- Mittlere Lebensdauer:

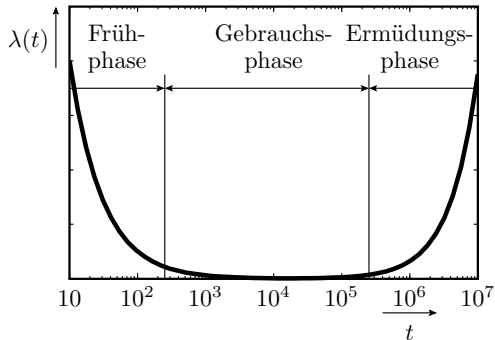
$$E(t_L) = \int_0^{\infty} R(t) \cdot dt$$

---

<sup>21</sup>wichtigste Vergleichsgröße [4, S.68]

## Ausfallphasen

- Fröhausfälle (infant mortalities): Erhöhte Ausfallrate durch Schwachstellen (Materialrisse, lokal stark überhöhte Feldstärke oder Stromdichte, ...).
- Normale Ausfälle: Näherungsweise konstante Ausfallrate in der Hauptnutzungsphase.
- Verschleißphase: Ausfall durch Materialermüdung<sup>22</sup>.



Maßeinheit der Ausfallrate: fit (failure in time)

1 fit = 1 Ausfall in  $10^9$  Stunden

<sup>22</sup>Bei SW gibt es diese Phänomen nur als »geplante Obsoleszenz«.





## Ausfallraten in der Hauptnutzungsphase nach [4]

Bauteil	Ausfallrate in fit	Bauteil	Ausfallrate in fit
diskrete HBT	1 bis 100	Widerstände	1 bis 20
digitale IC	50 bis 200	Kondensatoren	1 bis 20
ROM	100 bis 300	Steckverbinder	1 bis 100
RAM	bis 500	Lötstellen	0,1 bis 1
analoge IC	20 bis 300		

(HBT – Halbleiterbauteile; IC – Schaltkreise)

- Ausfallrate = Ausfallanzahl / Bauteilanzahl
- Bei mehreren Bauteilen und konstanten Ausfallraten addieren sich die Ausfallraten.



- Ausfallrate einer Baugruppe ist die Summe der Ausfallraten aller Komponenten:

Bauteiltyp	Anzahl $n$	Ausfallrate $\lambda$	$n \cdot \lambda$
Schaltkreise	20	150 fit	3000 fit
diskrete BT	15	30 fit	450 fit
Kondensatoren	15	10 fit	250 fit
Widerstände	30	10 fit	300 fit
Lötstellen	2000	0,5 fit	1000 fit
Baugruppe			5000 fit

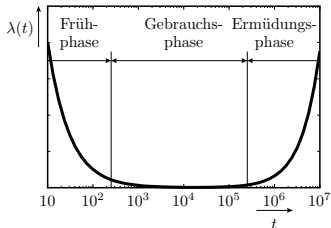
- Im Mittel 1 Ausfall in  $2 \cdot 10^5$  Stunden ( $\approx 23$  Jahre) Betriebsdauer.
- Von den heutigen PCs, Handys, ... fallen pro Jahr und hundert Stück nur wenige aus. Nach 2 ... 5 Jahren Ermüdungsausfälle, z.B. durch eingetrocknete Elkos.



# Frühausfälle und Voralterung

## Frühausfälle

- Nach [2] kommen auf 100 richtige Fehler etwa ein Beinahefehler, der zu einem Frühausfall führt.
- Bei 50% fehlerfreien und 50% aussortierten Schaltkreisen
- 50%/100 = 0,5% Beinahefehler.
  - Die Hälfte wird mit dem Ausschuss aussortiert.
  - $\approx 0,25\%$  (jeder 400ste) Schaltkreis verursacht ein Frühausfall.
  - Bei 20 Schaltkreisen pro Gerät jedes zwanzigste Gerät.
  - Bei großen Systemen fast jedes System.
- Frühausfälle sind Garantiefälle.
- Garantieleistungen sind teuer (Reparatur, Ersatz, Auftragsabwicklung, Image-Verlust).

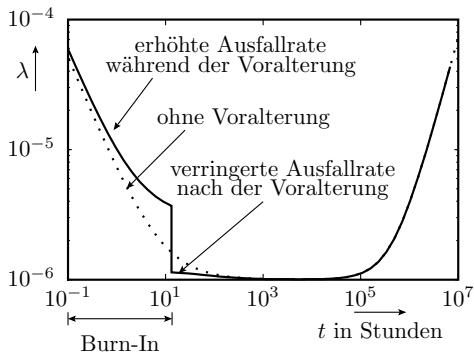


Was tun?



### Voralterung (Burn-In)

- Beschleunigung der Alterung vor dem Einsatz durch »harte« Umgebungsbedingungen
  - überhöhte Spannung,
  - überhöhte Temperatur,
  - Stress.
- Einsatz erst nach der Frühphase (wenn die kränklichen Bauteile gestorben und ausgetauscht sind).



Künstliche Voralterung ist auch in anderen Bereichen, z.B. Maschinenbau gebräuchlich.



# Wartung und Reserve



### Wartung

Maßnahmen zur Minderung von Ausfallzeiten und Zuverlässigkeitseinbrüchen durch Verschleiß elektronischer und mechanischer Komponenten:

- Überwachung.
- Periodische Wartungstests.
- Ergänzen und Ersatz von Betriebsstoffen und Verbrauchsmitteln (Schmierstoffen, bei Druckern Papier und Toner).
- Planmäßiger Austausch von Verschleißteilen, z.B. Festplatten in Servern.
- Fehlertoleranz.



## Kalte, warme und heiße Reserve

Systeme ohne Reparaturmöglichkeit, die lange verfügbar sein müssen (z.B. in einem Satelliten)

- erhalten Ersatzkomponenten und
- Funktionen zur automatischen Rekonfiguration (Komponententausch) nach einem Ausfall.

Arten der Reservekomponenten:

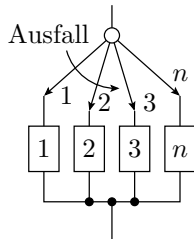
- Heiße Reserve: Reservekomponenten arbeiten parallel (z.B. Mehrversionssystem) und fallen mit derselben Wahrscheinlichkeit wie das aktive System aus.
- Kalte Reserve: Reservekomponenten werden geschont und funktionieren idealerweise noch alle zum Ausfallzeitpunkt der aktiven Komponente.
- Warme Reserve: Reserveeinheiten (z.B. das Reserverad im Auto) altern auch bei Nichtnutzung, nur langsamer.



## Kalte Reserve

Für jede Komponente beginnt die Belastung erst nach Ausfall der vorherigen Komponente.

Phase	mittlere Dauer
1	$E(t_{L.1})$
2	$E(t_{L.2})$
3	$E(t_{L.3})$
...	...
Summe:	$E(t_{L.ges}) = \sum_{i=1}^n E(t_{L.i})$



- Die Lebensdauern aller Komponenten addieren sich<sup>23</sup>.

<sup>23</sup>Unter der Annahme, dass die Umschalter und die ungenutzten Reserveeinheiten Ausfallrate null haben.

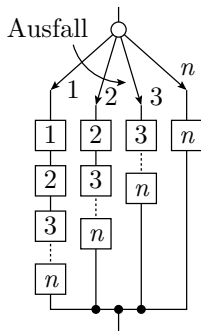
## Heiße Reserve

- Alle noch lebenden Komponenten können gleichmaßen ausfallen:

$$E(t_{L,i}) = \frac{1}{\sum_{j=1}^i \lambda_j}$$

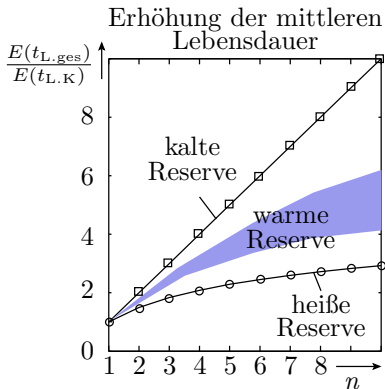
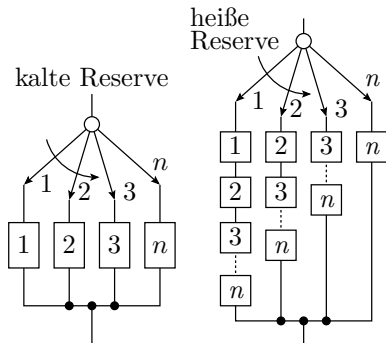
- Komponenten mit gleicher Ausfallrate  $\lambda_K$ :

Phase	mittlere Dauer
1	$\frac{1}{n \cdot \lambda_K} = \frac{E(t_{L,K})}{n}$
2	$\frac{1}{(n-1) \cdot \lambda_K} = \frac{E(t_{L,K})}{n-1}$
...	...
Summe:	$E(t_{L,ges}) = E(t_{L,K}) \cdot \sum_{i=1}^n \frac{1}{i}$



- Die erste Reservekomponente erhöht die mittlere Lebensdauer um die Hälfte, die zweite um ein Drittel etc.

## Warme Reserve



- Die Ausfallrate der »kalten« Ersatzkomponenten ist kleiner als im aktiven Zustand, aber größer null.
- »Warme« Reserveeinheiten verlängert die Lebensdauer mehr als »heiße« und weniger als »kalte«.



## Verfügbarkeitsplan<sup>24</sup>

Ausfälle beeinträchtigen die Verfügbarkeit. Im Verfügbarkeitsplan bilden Elemente, die funktionieren müssen, damit das Gesamtsystem funktioniert, Reihenschaltungen, und redundante Elemente, die ausgefallene Elemente ersetzen können, Parallelschaltungen. Abschätzung der Gesamtausfallrate aus den Ausfallraten der Komponenten:

- Reihenschaltung: Addition der Ausfallraten.
- Parallelschaltung (kalte Reserve): Addition der mittleren Lebensdauern.

Bei heißer und warmer Reserve ist die Berechnung komplizierter, aber zumindest numerisch auch problemlos lösbar.

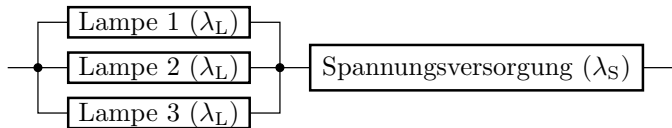
<sup>24</sup>In der Literatur »Zuverlässigkeitsstrukturen« [4, S.70]



## Beispiel Flurbeleuchtung

Die Flurbeleuchtung sei verfügbar, wenn mindestens eine von drei Lampen und die Spannungsversorgung funktioniert.

Verfügbarkeitsplan:



Die beiden nicht unbedingt erforderlichen Lampen bilden eine heiße Reserve:

$$\lambda_{L_{\text{ges}}} \approx \frac{\lambda_L}{\frac{1}{3} + \frac{1}{2} + 1} \approx 0,5 \cdot \lambda_L$$

und halbieren die Ausfallrate der Lampeneinheit. Ausfallrate des Gesamtsystems:

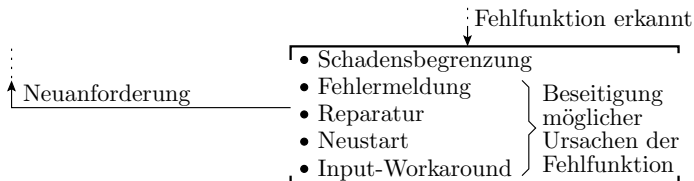
$$\lambda_{\text{ges}} \approx \lambda_S + 0,5 \cdot \lambda_L$$



# FF im Betrieb



### Umgang mit Fehlfunktionen im Betrieb



Schaden begrenzen:

- Bearbeitungsabbruch, Daten sichern, ...
- CP-Systeme: Aktoren in sicheren Zustand, Signale auf rot, ...

Daten zur Fehlerlokalisierung erfassen:

- Fehlermeldung, Core-Dump, Cap-Datei (Windows) erzeugen.

Zur Vermeidung desselben Versagens:

- Fehlerbeseitigung: Hardware-Tausch, Updates einspielen, ...
- Neuinitialisierung.
- Diversitäre Service-Anforderung / Berechnung: Geänderte Service-Reihung, Eingaben, Berechnungsfluß, ...



### Fehlerisolation und Fehlertoleranz

Automatische Reaktionen verlangen Fehlerisolation, d.h. dass Fehlfunktion erkannt und zurückgehalten werden, bevor sie Daten in anderen Systemteilen verfälschen können.

Die maximale Ausbaustufe ist Fehlertoleranz. Das ist die automatische Korrektur eigener, selbst erkannter Fehlfunktionen unter Verwendung redundanter<sup>25</sup> Funktionseinheiten.

---

<sup>25</sup>Redundanz ist eine System-Resource, die für die eigentliche Funktion des Systems nicht erforderlich ist.





# Fail-Safe/-Fast/-Slow



## Strategien zum Umgang mit Fehlfunktionen

- **Fail-Save:** Übergang in einen gefahrlosen Zustand.
- **Fail-Fast:** Sofortiger Bearbeitungsabbruch und Fehlersuche.
- **Fail-Slow:** Fortsetzung der Arbeit im Fehlerfall, solange noch eine sinnvolle Möglichkeit besteht.

### Beispiele für Fail-Safe-Systeme

- Zwangsbremmung, wenn Lokführer Haltesignal überfährt.
- Bei funkgesteuerten Modellen Motor aus und Bremsen ein bei schwachem Funkempfang oder schwacher Batterie.
- Bei erkanntem Sensorausfall Maschinen oder Anlagen kontrolliert anhalten.
- Bei Erkennen eines freien Falls (Laptop fällt herunter), Festplattenköpfe in Parkposition bringen.
- Ruhestromprinzip (siehe nächste Folie).



### Ruhestromprinzip (gebräuchliche Fail-Safe Technik)

- Sicherheitsbremsen: Bei Stromausfall, Drahtbruch, geplatzttem Bremsschlauch, ... setzt Bremswirkung ein.
- Alarmanlagen: Bei Durchtrennen einer Signalleitung Alarm auslösen.
- Eisenbahnsignal:
  - Bei Drahtseilriss für ein Streckenfreigabesignal Strecke sperren (Signalarm fällt runter).
  - Bei fehlendem Ruhestrom auf einer Meldeleitung, Anzeige einer Störung.
- Brandmeldeanlage: Alarm bei Durchtrennen der Signalleitungen eines Brandmelders.
- Notaus: Bei Leitungsbruch im Notaus-Kreis System ausschalten.



## Fail-Fast

Bei erkannter Fehlfunktionen (Zugriff auf nicht vorhandene Daten, Division durch Null, Wertebereichsverletzung ...) Abarbeitung stoppen, typ. mit Assert-Anweisung, z.B. in VHDL

```
assert <Bedingung> report <Beschreibung>  
    severity note|warning|fault|error;
```

- Nutzung Systemrufe (Software-Interrupts).
- Sinnvoll während des Tests.
- Ausgabe aller für die Lokalisierung der Ursache dienlichen Daten einprogrammieren!



## Fail-Slow

Fail-Slow: Das System soll bei einer erkannten Fehlfunktion solange wie möglich weiterarbeiten, z.B. beim Zugriff auf nicht vorhandene Daten, Weiterarbeit mit Standardwerten.

- weniger sichtbare Fehlfunktionen, höhere Zuverlässigkeit
- sinnvoll im Einsatz.
- Fehlerinformationen aber möglichst an die Hersteller zur Fehlersuche weiterleiten.

---

### Umschaltung zwischen Fail-Fast und Fail-Slow [6]

- Angabe der Fehlerschwere in den Assertion-Anweisung.
- Umschaltung des Exception-Handlers, dass nur bei schweren Fehlern ein Abbruch erfolgt.
- Protokollierung und Versendung aller Assertion-Ausgaben.<sup>26</sup>

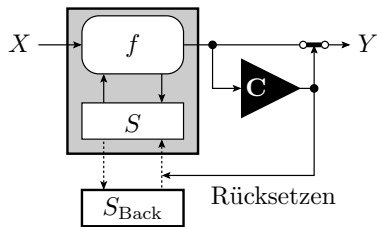
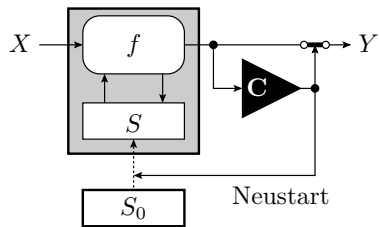
<sup>26</sup>Höfliche Programme fragen bzw. informieren, bevor sie Nachrichten nach Hause schicken.



## Neustart, Wiederholung

## Neuinitialisierung

Zur Fortsetzung der Arbeit nach einer Fehlfunktion ist der potentiell verfälschte Zustand auf einen zulässigen Zustand rückzusetzen:



- Statische Neuinitialisierung: fester Anfangszustand,
- Dynamische Neuinitialisierung: Laden des letzten gesicherten Zustands.



## Watchdog

Traditionell hat ein Rechner eine Möglichkeit zum Zurücksetzen:  
Taster, Tastenkombination, ...

Rechner ohne Bediener (SPS, Motorsteuergeräte, Regler, ...) haben einen Watchdog:

- Zeitüberwachung mit einem Zeitzähler, der vom Programm, solange es korrekt arbeitet, innerhalb definierter Zeitintervalle gelöscht wird.
- Rechnerneustart bei Zähler- (Zeit-) Überlauf.
- Der Watchdog wird vom Programm eingeschaltet und ist idealerweise nicht vom Programm, d.h. auch nicht durch Fehlfunktionen, ausschaltbar, sondern nur durch Neustart.





## Dynamische Neuinitialisierung

Erfordert regelmäßige Backups in einen gesicherten Speicher, z.B.

- für einen PC auf eine externe Festplatte oder einen Stick,
- für unsere Laborrechner mit dem Backup-System des Rechenzentrums oder
- für Mikrocontroller in den EEPROM.

Oft werden nur Daten gesichert, die sich nicht problemlos neu berechnen lassen:

- Eingabedaten,
- Ergebnisse langwieriger Berechnungen, ...

Editoren, Logistiksysteme, Datenbanken, ... speichern z.B. oft alle Eingaben und Aufträge seit dem letzten kompletten Backup<sup>27</sup>.

---

<sup>27</sup>So kann der aktuelle Zustand aus dem Backup und den protokollierten Eingaben wiederhergestellt werden.



## Wiederholung

Außer bei fehlerkorrigierenden Codes erfolgt die Korrektur von Fehlfunktionen durch Wiederholung. Voraussetzung Diversität<sup>28</sup>. In Ausnahmefällen genügt einfache Wiederholung:

- störungsbedingte Fehlfunktionen,
- Ausfall der Spannungsversorgung,
- Eingabefehler, Übertragungsfehler,
- Lesefehler von Festplatte, ...

Daten werden in der Regel mit Prüfsummen, CRC, ... gesichert. Erkennungssicherheit nahe 1. Auch bei vielen Wiederholungen kaum nicht erkennbare Verfälschungen.

<sup>28</sup>Nach TV\_F3, Abschn. 3.2 ist Diversität die mittlere Anzahl der erkennbaren Fehlfunktionen zwischen zwei Maskierungen bei Verdopplung und Vergleich. Das ist der Kehrwert der Wahrscheinlichkeit, dass bei Wiederholung nach einer Fehlfunktion noch einmal dieselbe Fehlfunktion auftritt.



Standardreaktion bei Prüfkennzeichenfehler:

- Wiederholung bis Erfolg oder Zeitüberschreitung.
- Bei Zeitüberschreitung Fehlermeldung und Abbruch oder Weiterarbeit ohne angeforderte Daten.

Reaktion auf Speichermangel:

- Platz schaffen und wiederholen.



## Schaffung von Diversität

Lösung derselben Aufgabe auf unterschiedliche Weise:

- Änderung Rechenweg (Algorithmus),
- Nutzung anderer Programme.
- Andere Programmeinstellungen.
- Bedien-Workaround<sup>29</sup>: alternative Eingabemöglichkeiten (Menüs, Konsole, ...), alternative Eingabereihenfolge, ...

---

<sup>29</sup>Jeder Benutzer eines Rechners lernt mit der Zeit intuitiv, welche Eingabe- und Bedienreihenfolgen funktionieren. Dadurch nimmt die beobachtbare Zuverlässigkeit auch ohne Fehlerbeseitigung mit der Nutzungsdauer zu. Bei einem neuen Nutzer treten sprunghaft wieder deutlich häufiger FFs auf.



Beispiele studentischer Arbeiten im Arbeitsbereich, die nur mit schwer zu findenden Workarounds lösbar waren:

- CAN-Busansteuerung c167: SFR-Register mussten in einer nicht in der Doku stehenden Reihenfolge initialisiert werden.
- Sharp-Abstandssensoren: Im Datenblatt steht nicht, dass sich mehrere Sensoren im Raum gegenseitig stören, ...
- Power-Cube (Gelenke des großen Laborroboters): Bei Nachrichtenkollision auf dem CAN-Bus keine Übertragungswiederholung und andere Bugs. ...

Anderes Beispiel:

- Vor der Compilierung des Linux-Kerns können die Hardware-Bugs abgewählt werden, für die einprogrammierte Workarounds überflüssig sind (z.B. Float/Div-Bug).

Bei Problemen mit neuer Hard- oder Software am besten Internet durchsuchen, ob schon Workarounds dafür zu finden sind.



## Fehlerisolation



## Fehlerisolation

Eine sinnvolle automatische Reaktion auf Fehlfunktionen:

- Fail-Safe,
- dynamische Neuinitialisierung und Wiederholung,
- andere anwendungsspezifische Reaktionen

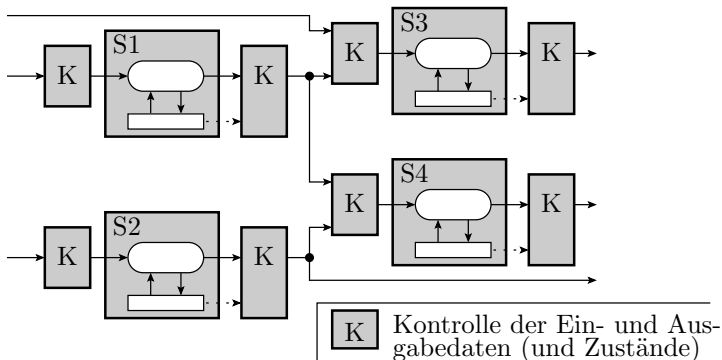
erfordern, dass der Zustand der Teilsysteme, die die Fehlerbehandlung durchführen, unverfälscht ist:

- Ein HW-Reset zur statischen Neuinitialisierung hat keine kontaminierbaren Zustände.
- Der Back-Up-Speicher zur dynamischen Neuinitialisierung benötigt eine Zugriffsschutz gegen Verfälschungen.
- Fail-Save-Funktionen, Notbetrieb, ... benötigen einen funktionierenden Rechner und geschützte Daten.

Fehlerisolation: Maßnahmen zur Verhinderung, dass sich Datenverfälschungen und andere Fehlfunktionen über Teilsystemgrenzen ausbreiten.

## Prinzip der Fehlerisolation

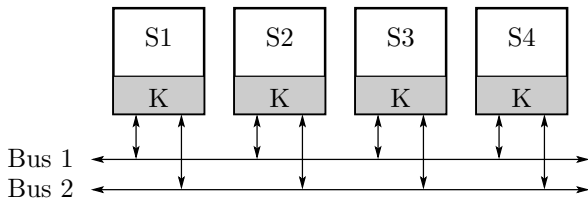
- Modulares System mit Kontrollen an den Schnittstellen.
- Keine Weitergabe kontaminierter Daten zwischen Teilsystemen.





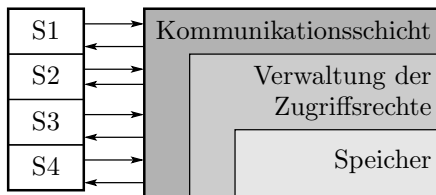
## Fehlerisolation in Systemen mit Busstrukturen

- Bus: zentrale Informationsschnittstelle.
- Jedes Teilsystem erhält Kontrollfunktionen für alle über den Bus empfangen und versendeten Ein- und Ausgabedaten.



- Fehlerbehandlung für Eingabefehler: Keine Weiterverarbeitung, Protokollierung der Fehlfunktion, Diagnose des Quellsystems, ...
- Fehlerbehandlung für Ausgabefehler: Keine Ausgabe, Neustart, Wiederholung, ...

## Fehlerisolation mit einem Betriebssystem



- Die zu isolierenden Teilsysteme sind die Prozesse  $S_i$ .
- Jeder Prozess sieht nur seinen eigenen virtuellen Speicher,
- die ihm zugeordneten Ein- und Ausgabegeräte und
- bekommt den Prozessor zeitscheibenweise zugeteilt.

Die Zuordnung von Ressourcen (physikalischer Speicher, Ein- und Ausgabegeräte, Kommunikation zu anderen Prozessen...) sind nur über Systemrufe möglich, bei denen das Betriebssystem Kontrolle über alle Daten hat.



# Fehlertoleranz



## Redundanz und Fehlertoleranz

Fehlertoleranz: Korrektur eigener, selbst erkannter Fehlfunktionen unter Verwendung redundanter<sup>30</sup> Funktionseinheiten:

- Informationsredundanz: siehe fehlerkorrigierende Codes.
- homogene Redundanz (Redundanz mit gleichartigen Mitteln):  
Reserveeinheiten zur Vorbeugung gegenüber Hardware-Ausfällen z.B. zwei Glühbirnen im roten Teil einer Ampel (siehe heiße und kalte Reserve, Folie 96) und
- diversitäre Redundanz (Redundanz mit ungleichartigen Mitteln) zur Korrektur von falschen Ergebnissen.

Im Bereich der diversitären Redundanz haben sich durchgesetzt:

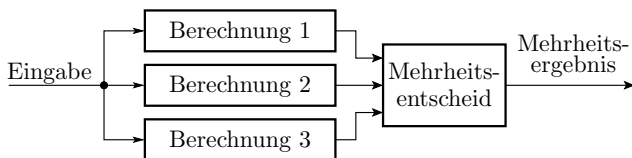
- N-Versionstechnik mit Mehrheitsentscheid und
- und Systeme mit Rücksetzblöcken.

---

<sup>30</sup>Redundanz ist eine System-Resource, die für die eigentliche Funktion des Systems nicht erforderlich ist.



## Mehrfachberechnung mit Mehrheitsentscheid



Klassische Architektur für ein fehlertolerantes System, bereits 1956 von »von Neumann« vorgeschlagen:

- Jede Berechnung erfolgt  $n \geq 3$  mal.
- In jedem Schritt Mehrheitsauswahl.
- Ohne Mehrheitsergebnis nur Fehlererkennung.

Die originale Idee sah die zeitgleiche Berechnung auf unterschiedlichen Rechnern vor.



## Probleme und Verbesserungsansätze

- Hoher Aufwand: Das originale n-Versionssystem verlangt mindestens drei statt nur einen Rechner.

Abhilfe schafft, die Berechnungen nacheinander auf demselben oder nur zwei Systemen auszuführen. Erspart Hardware und erhöht den Rechenaufwand.

- Ein Mehrversionssystem in dieser Form toleriert keine übereinstimmenden Entwurfsfehler in den Systemen.

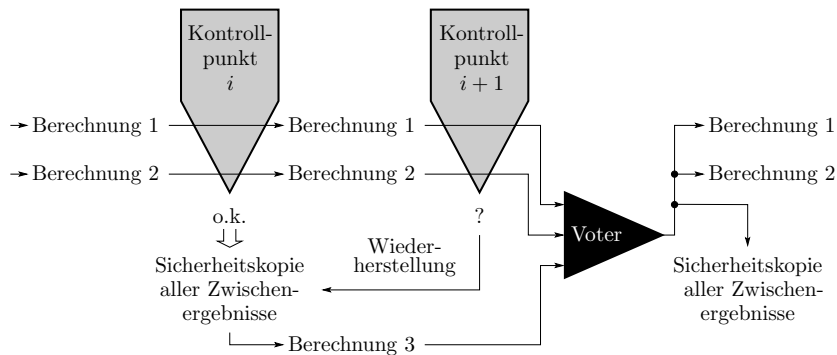
Das Original war für die Tolerierung der häufigen Ausfälle von Röhrenrechnern konzipiert. Heute sind Entwurfsfehler die häufigste Ursache für ein Versagen. Für deren Tolerierung müssen die Berechnungen verschiedenartig (diversitär) sein (unterschiedliche Hardware, verschiedene Software-Entwürfe, ...).



- Eine Diversität, die alle Ursachen für übereinstimmende Fehlverhalten ausschließt (fehlerhaft oder unvollständig Anforderungen, gleiche Denkfehler, ...), ist für Systeme mit angestrebtem identischen Verhalten im fehlerfreien Fall unerreichbar.
- Bei diversitären Service-Leistungen können sich auch die richtigen Ergebnisse unterscheiden. Ein Vergleich diagnostiziert das als Fehlfunktion (Phantomfehler).

Praktisch eingesetzte fehlertolerante Systeme nutzen verfeinerte Techniken mit einer Vielzahl weiterer Kontrollen und einer Vielzahl einprogrammierten Verhaltensweisen bei Fehlfunktionen.

## Check-Point-Roll-Back-Recovery [5]



- Nur zwei parallel ausgeführte Berechnungen.
- An einprogrammierten Kontrollpunkten im Programm werden die Bearbeitungszustände<sup>31</sup> verglichen.

<sup>31</sup>Werte der Variablen, Register, ...





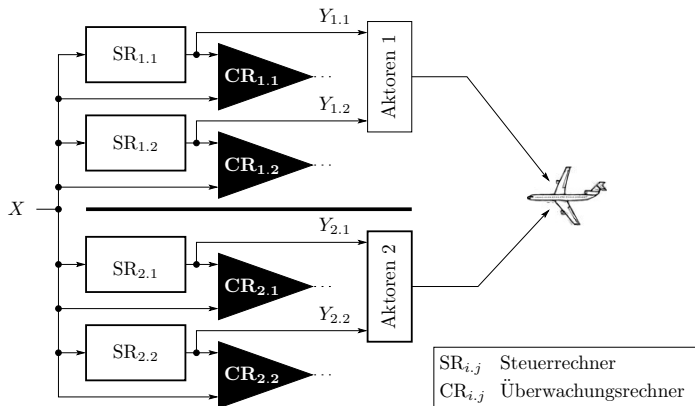
- Bei Übereinstimmung Speicherung des Bearbeitungszustands in einem geschützten Speicher.
- Bei Abweichung, Laden der letzten Sicherheitskopie und Berechnungswiederholung (Roll-Back Recovery).
- Nach Roll-Back Recovery am nächsten Kontrollpunkt wieder Mehrheitsentscheid.
- Wenn Mehrheitsergebnis, diesen als gesicherten Zustand speichern, sonst Abbruch.

Sequoia-System [3]:

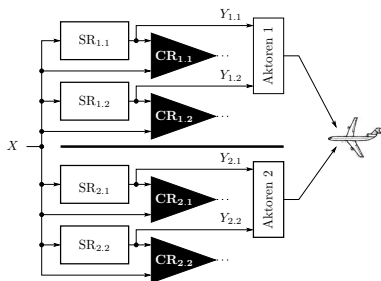
- Berechnung auf zwei Prozessoren mit eigenem Write-Back-Cache.
- Vergleich in jedem Takt.
- Zustands-Backup bei Ereignissen wie Stack-Überlauf und Prozesswechsel.
- Hauptspeicher hat die Funktion des stabilen Speichers.

## Flugsteuersystem Airbus A3XX [7]

Hochsicherheitskritische Anwendungen müssen möglichst alle Fehlfunktionen, auch solche durch nicht erkannte Entwurfsfehler, nicht erkannte Fertigungsfehler und Ausfälle tolerieren.



- Zwei identische Systeme mit allen Sensoren, Aktoren und zwei Rechnerpaaren.
- Jedes Rechnerpaar besteht aus einem Steuerrechner  $SR_{i,j}$ , der die Aktoren ansteuert, und einem Überwachungsrechner  $CR_{i,j}$ .
- Normalzustand Rechner  $SR_{1,1}$  steuert und  $CR_{1,1}$  überwacht. Zweites Rechnerpaar Stand-By. System 2 abgeschaltet.
- Bei Ausfall übernimmt Rechnerpaar 1 von Rechnerpaar 2. Bei Komplet-, Sensor- oder Aktorausfällen übernimmt System 2 von System 1.



Diversität: Rechner unterschiedlicher Hersteller, getrennte Software-Entwicklung nach Spezifikationen, die unabhängig von einer gemeinsamen Basisspezifikation abgeleitet wurden.



## Zusammenfassung

- Mehrfache Berechnung kostet mindestens die doppelte Rechenzeit oder den doppelten Hardware-Aufwand.
- Für nicht reproduzierbare Fehlfunktionen (verursacht z.B. durch Störungen, Eingabe- und Initialisierungsfehler) genügt eine Berechnung auf demselben oder einem gleichen System.
  - Standardfehlerbehandlung für nicht deterministische Fehlverhalten, verursacht z.B. durch Eingabe-, Übertragungs-, Initialisierungs- und Festplattenlesefehler)
- Reproduzierbare Fehlfunktionen verlangen eine Mehrfachberechnung mit diversitären Systemen:
  - Unter Testbedingungen: Regressionstest (Funktionsvergleich aufeinanderfolgender Software-Versionen).
  - Unter Betriebsbedingungen: Mehrversionsentwürfe, Verdopplung des Entwurfsaufwands.



# Literatur



## 5. Literatur

- [1] J. E. Aas and I. Sundsbo.  
Harnessing the human factor for design quality.  
*IEEE Circuits and Devices Magazine*, 11(3):24–28, 1995.
- [2] Thomas S. Barnett and Adit D. Singh.  
Relating yield models to burn-in fall-out in time.  
pages 77–84, 2003.
- [3] P.A. Bernstein.  
Sequoia: a fault-tolerant tightly coupled multiprocessor for transaction processing.  
*Computer*, 21(2):37–45, 1988.
- [4] R. Kärger.  
*Diagnose von Computern*.  
Teubner, 1996.
- [5] D. K. Pradhan, D. D. Sharma, and N. H Vaidya.  
Roll-forward checkpointing schemes.  
In *Lecture Notes in Computer Science* 744, pages 93–116. Springer Verlag, 1994.
- [6] Jim Shore.  
Fail fast.  
In *IEEE Software*, pages 21–25, 2004.
- [7] Pascal Traverse.  
Dependability of digital computers on board airplanes.  
*Dependable Computing for critical applications*, 4:134–152, 1991.