Test und Verlässlichkeit Foliensatz 2: Problembeseitigung

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV_F2)

May 31, 2018

Inhalt Foliensatz TV_F2

Fehlerbeseitigung

Fehler sind nicht vollständig vermeidbar

Bei jedem Entstehungsschritt (Entwurf, Fertigung) und jedem Reparaturschritt können Fehler entstehen. Typische Fehlerentstehungsraten:

- 10 bis 100 Fehler auf 1000 NLOC,
- 20 neu entstehende Fehler bei 100 Reparaturversuchen,
- $Y = 70\%$ Schaltkreisausbeute bzw. 30% Ausschuß,
- ...

Je größer der Entstehungs- und Fehlerbeseitigungsaufwand, um so mehr Fehler entstehen, die entweder beseitigt werden oder im Einsatz FF verursachen.

Kenngrößen für Tests in Beseitigungsiterationen

Fehlerbeseitigung durch Reparatur:

- Fehleranzahl vor und nach Beseitigungsiteration: φ_{EP}, φ_{Rep}
- Fehlerüberdeckung: $FC = 1 - \frac{\varphi_{\mathrm{Rep}}}{\varphi_{\mathrm{EP}}}$
- Erkennungswahrsch. / zu erwartende Fehlerüberdeckung:

$$p_{\mathrm{E}} = E\left(FC\right) = 1 - \frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)}$$

Fehlerbeseitigung durch Ersatz kompletter Objekte:

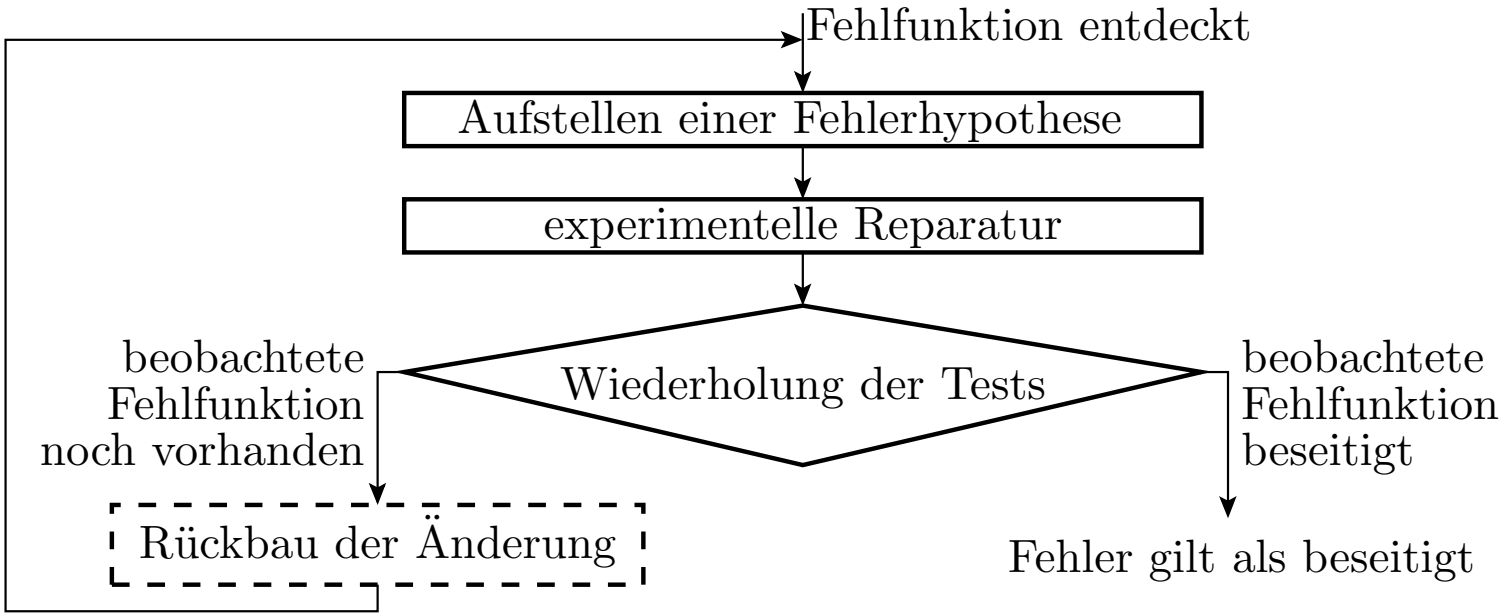
- Fehleranteil vor und nach Beseitigungsiteration: DL_{EP}, DL_{Ers}
- Fehlerüberdeckung: $FC = 1 - \frac{DL_{\mathrm{Ers}}}{DL_{\mathrm{EP}}}$
- Erkennungswahrsch. / zu erwartende Fehlerüberdeckung:

$$p_{\mathrm{E}} = E\left(FC\right) = 1 - \frac{E\left(DL_{\mathrm{Rep}}\right)}{E\left(DL_{\mathrm{EP}}\right)}$$

($\ldots_{\mathrm{EP}}$ – nach Entstehungsprozess; φ_{Rep} – nach Beseitigung durch Reparatur; DL_{Ers} – nach Beseitigung durch Ersatz).

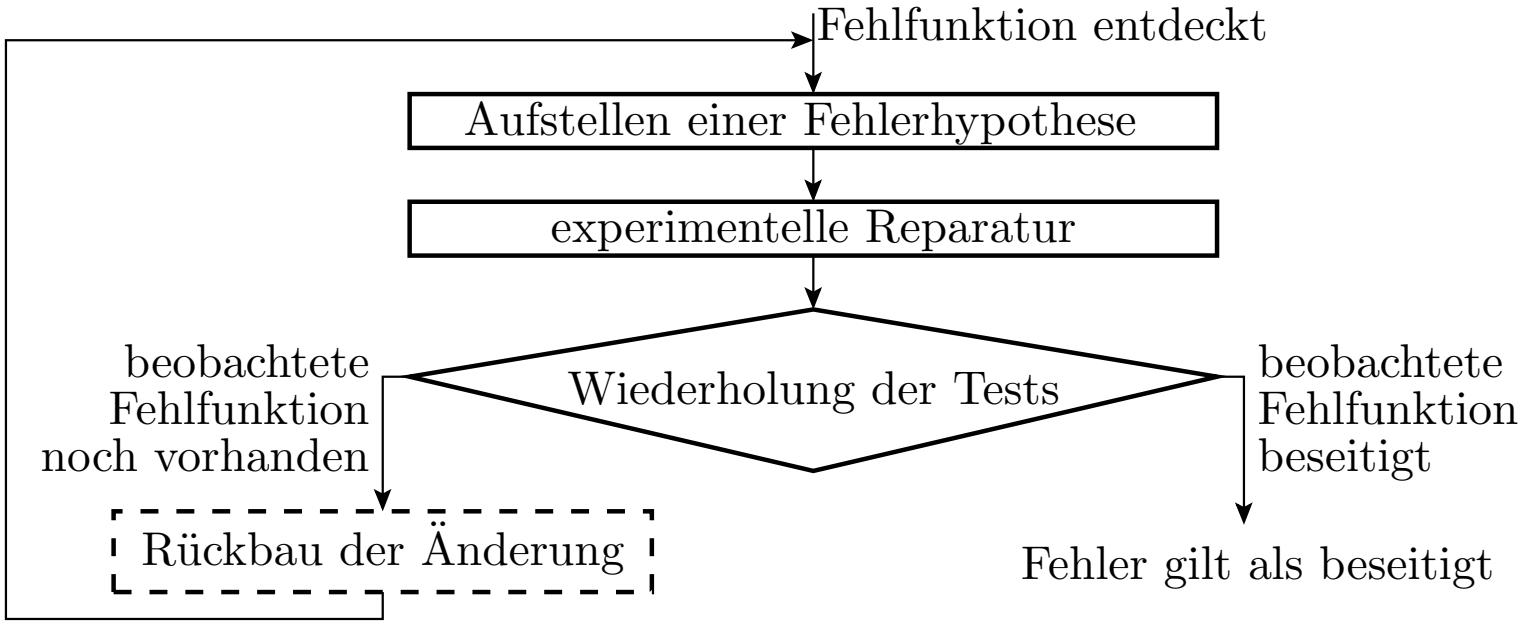
Experimentelle Reparatur

Experimentelle Reparatur



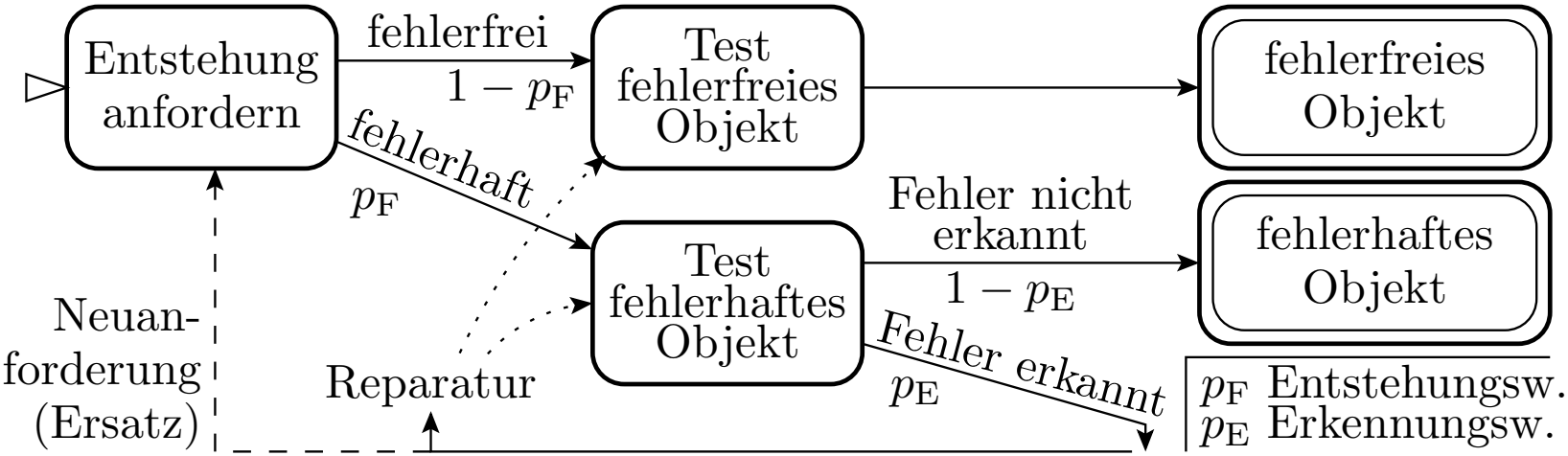
- Deterministische Sollfunktion.
- Der Test weist den Fehler bei jeder Testwiederholung nach.
- Beseitigung durch »intelligentes Probieren«
- Fehlerbeseitigungskontrolle durch Testwiederholung.

Diese Iteration beseitigt jeden erkennbaren Fehler.



- Nicht beseitigt werden nicht erkennbare Entwurf-, Fertigungs- und bei der Reparatur entstehende Fehler.
- Die Fehlerbeseitigungswahrscheinlichkeit hängt hauptsächlich von der Erkennungswahrscheinlichkeit der Tests ab.
- Die Erfolgsrate der Reparaturversuche hat nur mittelbar über die Anzahl der bei der Reparatur entstehenden Fehler Einfluss.
- »Rückbau« mindert die Fehlerentstehung bei der Reparatur.

Experimentelle Reparatur als Markov-Kette



Ein potentieller Fehler i

- entsteht mit einer Wahrscheinlichkeit p_F und
- wird mit einer Wahrscheinlichkeit p_E erkannt.

Beseitigungsversuche für entstandene nicht erkannte Fehler:

- Ersatz des Gesamtsystems (Wiederholung des Entstehungsprozesses) oder
- Reparatur, Lokalisierung und Tausch defekter Teilsysteme.

Ersatz

Ersatz vs. Reparatur

Beim Ersatz erkannter defekter Systeme vor dem Einsatz aus demselben Fertigungsprozess

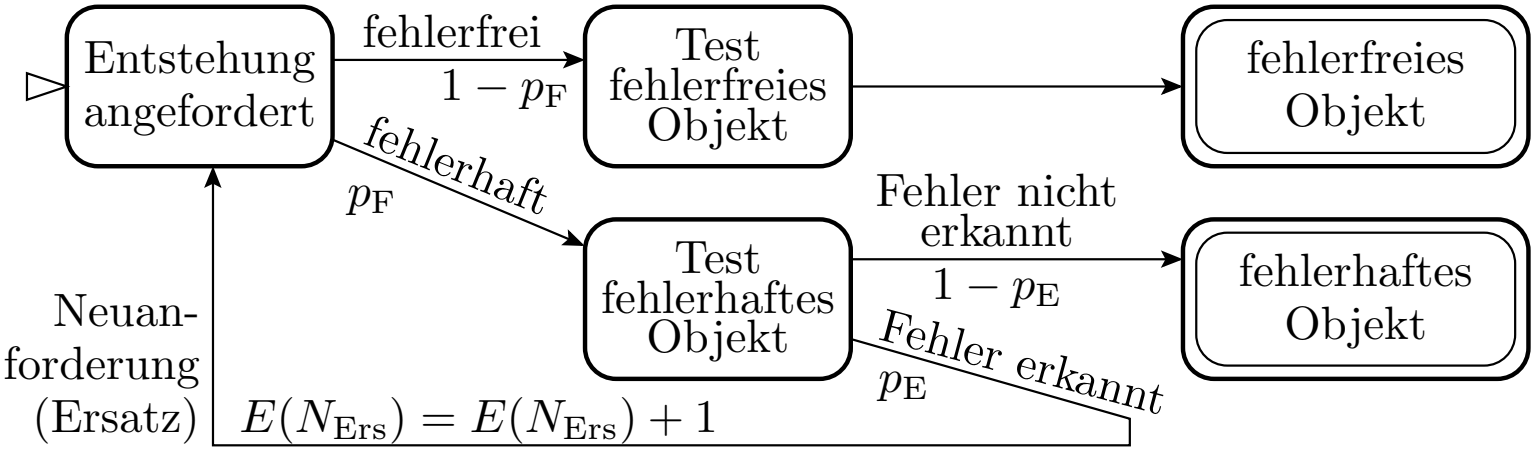
- haben Orginal- und Ersatzsystem dieselbe zu erwartende Ausbeute $E(Y)$,
- müssen im Mittel $\frac{1}{E(Y)}$ mal so viele Systeme gefertigt oder entworfen, wie am Ende eingesetzt werden.

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[1] und unbezahlbar für Ausbeuten $Y \ll 50\%$.

[1] Spart Aufwändungen für prüf- und reparaturgerechten Entwurf, Lokalisierung und Vorratshaltung von Reparaturkapazitäten.

Ersatz als Iteration

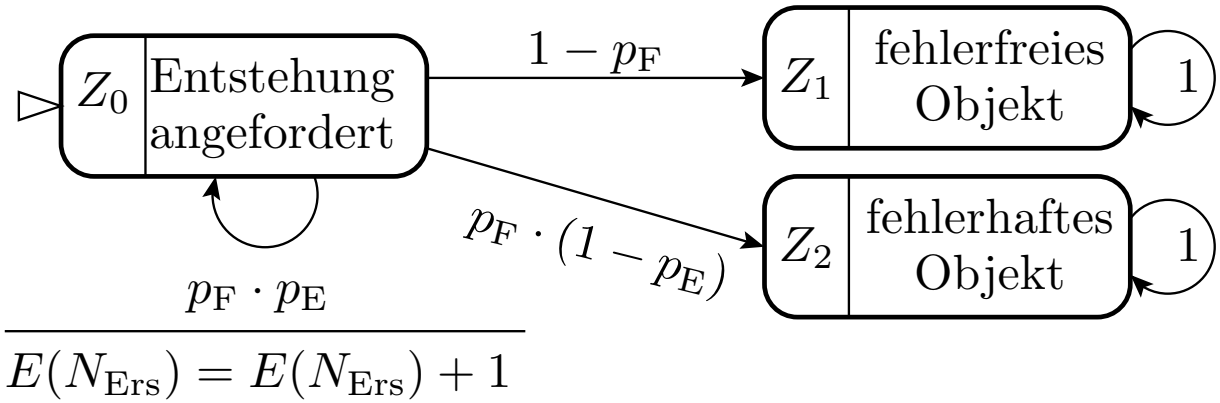


- Ersatzobjekte haben auch mit Wahrscheinlichkeit p_F Fehler.
- Diese entstehen unabhängig und sind unabhängig nachweisbar.

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.

Vereinfachte Markov-Kette



Nach Ersatz aller erkennbar defekten Objekte[2] :

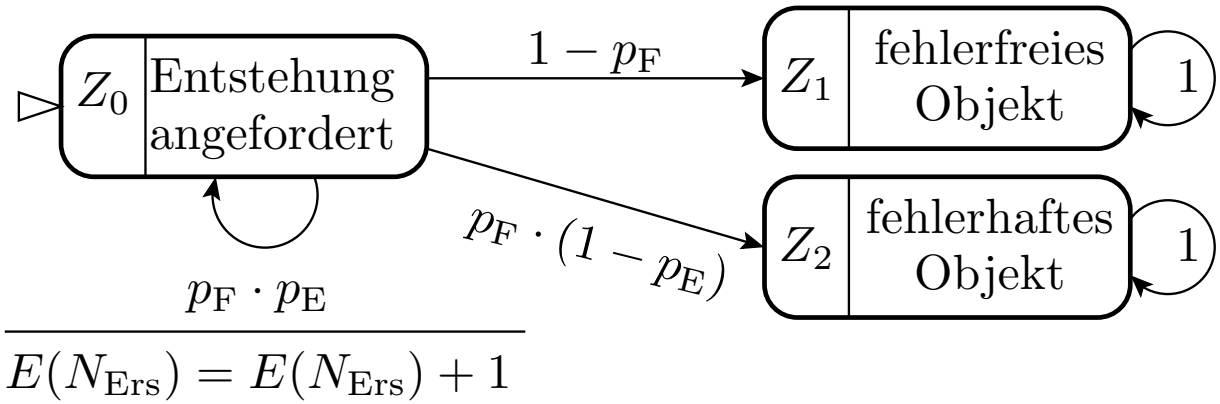
$$\lim_{n\to\infty}(p_{Z0}) = \lim_{n\to\infty}(p_F \cdot p_E)^n = 0$$

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

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

[2] Summenformel der geometrischen Reihe: $\sum_{n=0}^{\infty} a_0 \cdot q^n = \frac{a_0}{1-q}$

Abschätzbare Kenngrößen



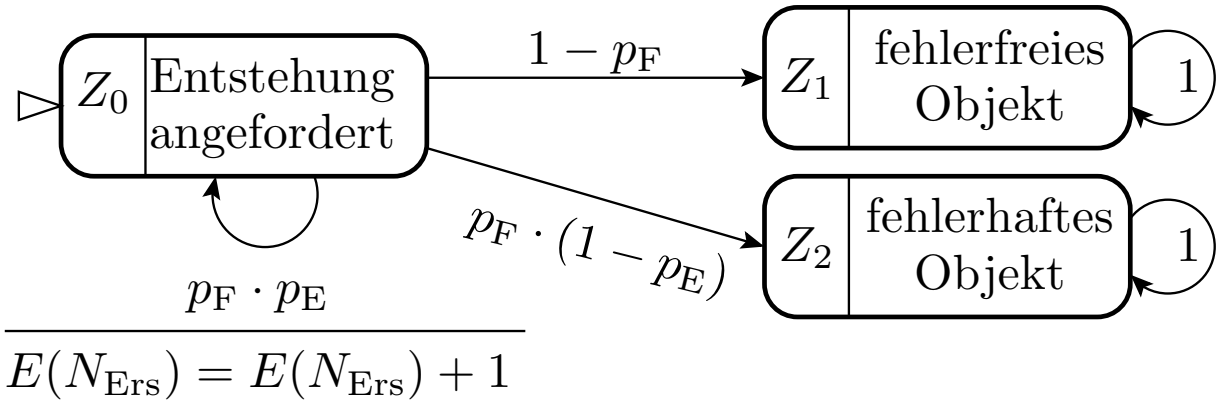
Zu erwartender Fehleranteil nach Ersatz defekter Objekte:

$$E\left(DL_{\mathrm{Ers}}\right) = \lim_{n\to\infty}\left(p_{Z2}\right) = \frac{p_F \cdot (1 - p_E)}{1 - p_F \cdot p_E} \tag{1}$$

Wahrscheinlichkeit, dass Fehler nicht beseitigt werden[3]:

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

[3] Verhältnis des zu erwartenden Fehleranteils DL_{Ers} nach dem Ersatz erkennbar defekten Objekte und DL_{EP} nach Entstehung (vor dem Ersatz).



Die zu erwartende Anzahl der Ersetzungen:

$$E(N_{\mathrm{Ers}}) = \sum_{n=1}^{\infty} (p_{\mathrm{F}} \cdot p_{\mathrm{E}})^n = \frac{p_{\mathrm{F}} \cdot p_{\mathrm{E}}}{1 - p_{\mathrm{F}} \cdot p_{\mathrm{E}}} \tag{2}$$

Zu erwartende Ausbeute[4]:

$$E(Y) = \frac{1}{E(N_{\mathrm{Ers}}) + 1} = 1 - p_{\mathrm{F}} \cdot p_{\mathrm{E}} \tag{3}$$

[4] 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 zu die erwartenden Schaltkreisausbeuten von $E(Y) = 10\%$, 30%, 50%, 80% und 90% und die zu erwartenden Fehlerüberdeckungen $E(FC) = p_{\mathrm{E}}$ von 90%, 99% und 99,9%

1. die zu erwartende Anzahl der aussortierten Schaltkreise je als gut befundener Schaltkreis $E(N_{\mathrm{Ers}})$ und
2. die Wahrscheinlichkeit p_{F}, dass ein Schaltkeis vor dem Aussortieren fehlerhaft ist?
3. Wie groß ist der zu erwartende Fehleranteil $E(DL_{\mathrm{Ers}})$ der als gut befundenen Schaltkreise für $p_{\mathrm{F}} = 100\%$, 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:

$$E\left(N_{\mathrm{Ers}}\right) = \frac{1}{E\left(Y\right)} - 1$$

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

2. Die Wahrscheinlichkeit p_{F}, dass ein Schaltkeis vor dem Aussortieren fehlerhaft ist, mit $p_{\mathrm{E}} = E\left(FC\right)$ ist nach Gl. 3:

$$p_{\mathrm{F}} = \frac{1 - E\left(Y\right)}{E\left(FC\right)}$$

$E\left(FC\right)$	$E\left(Y\right) = 10\%$	...=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 $E(DL_{\mathrm{Ers}})$ der als gut befundenen Schaltkreise nach Ersatz aller erkennbar fehlerhaften Objekte nach Gl. 1 beträgt mit $p_{\mathrm{E}} = E(FC)$:

$$E(DL_{\mathrm{Ers}}) = \frac{p_{\mathrm{F}} \cdot (1 - E(FC))}{1 - p_{\mathrm{F}} \cdot E(FC)}$$

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

Reparatur

Fehlerbeseitigung durch Reparatur

Bei Reparatur werden Teilsysteme, die möglicherweise defekt sind, getauscht[5]. Zu ersetzende Teilsysteme:

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

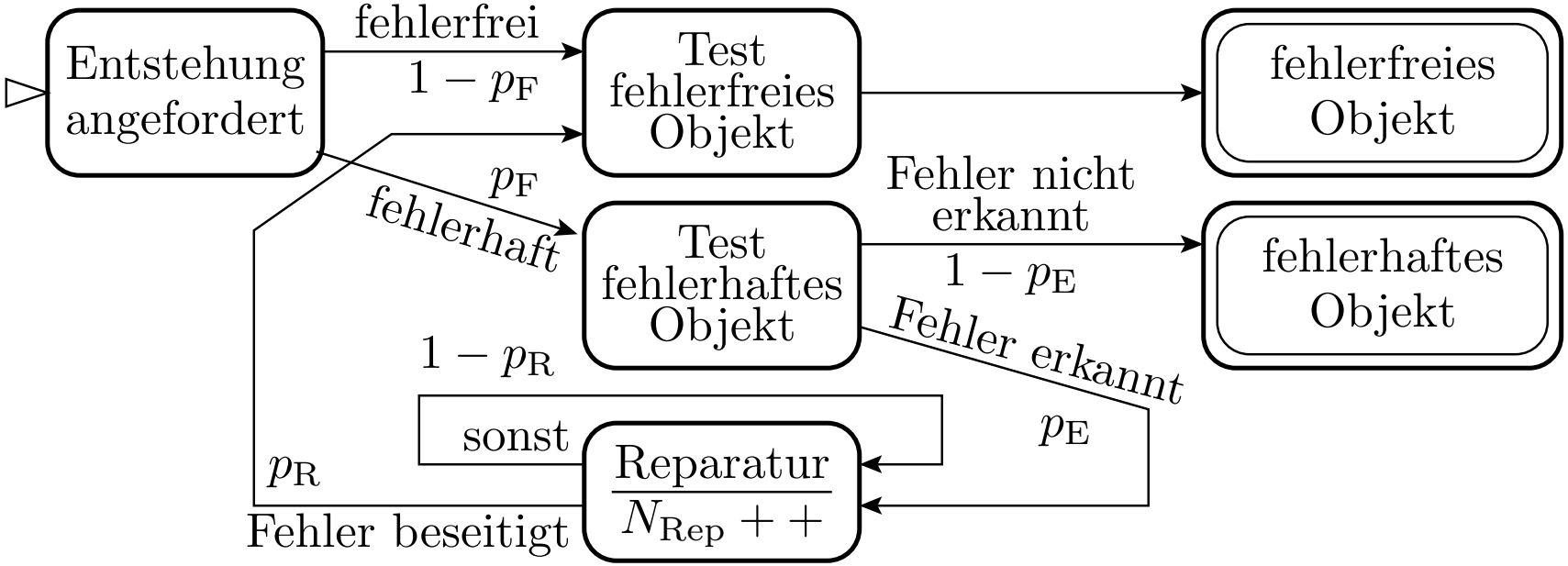
Dafür verlangt Reparatur Zusatzaufwändungen:

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

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

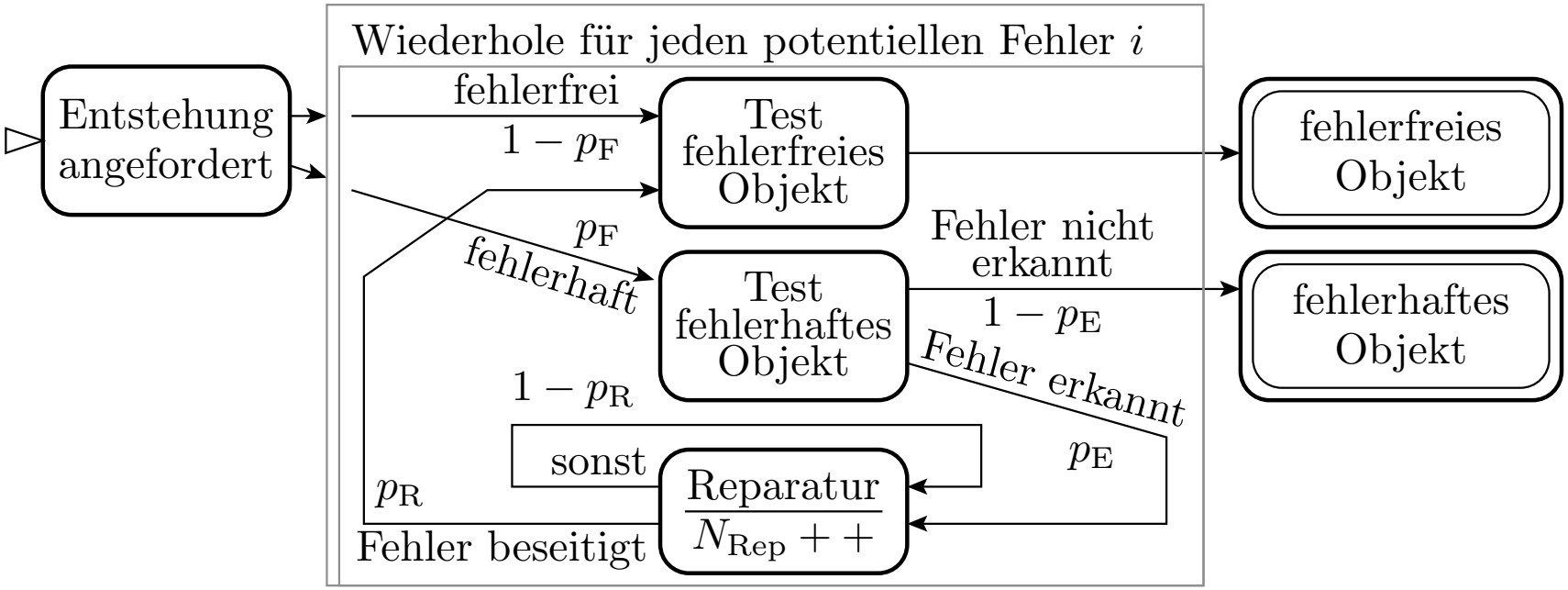
[5] Für Software »tauschen von Anweisungen«

Beseitigungsiteration für einen Fehler

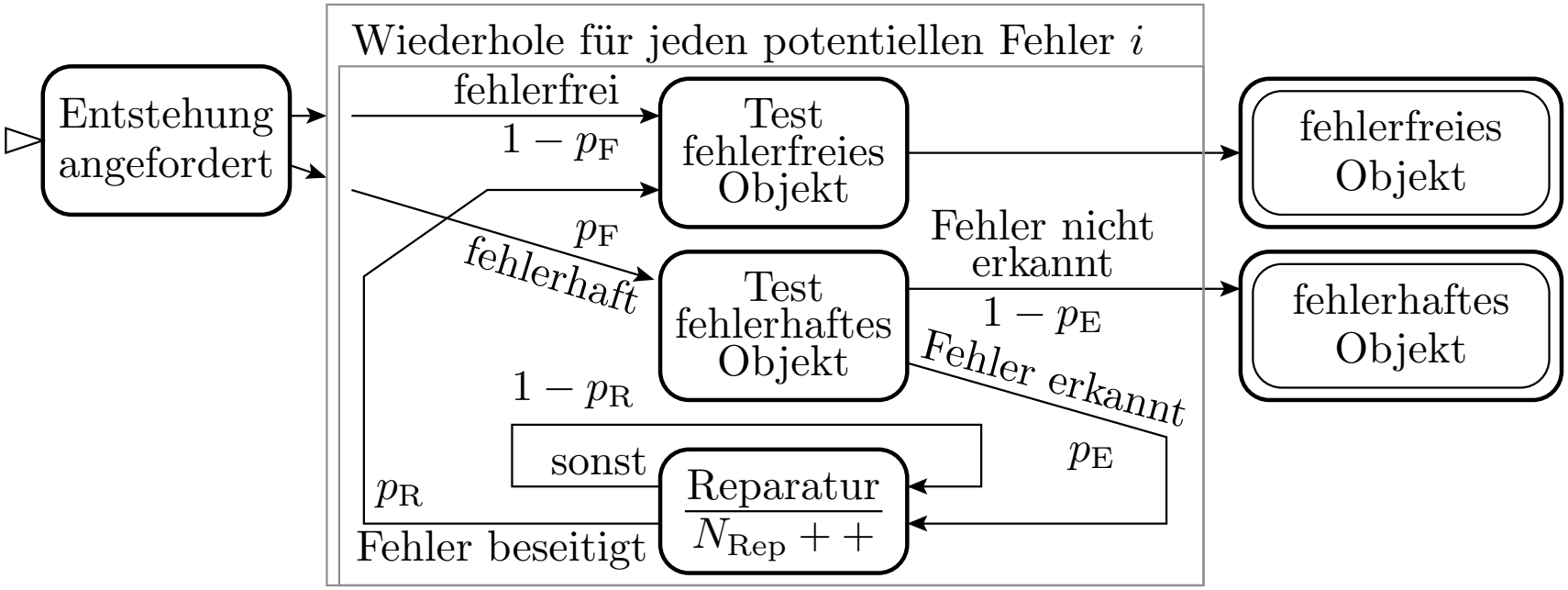


- Bei einem erkennbaren Fehler wird solange mit einer Erfolgswahrscheinlichkeit p_R repariert, bis das vom Test nachweisbare Fehlverhalten beseitigt ist.
- Es sei unterstellt, dass die zu erwartende Anzahl der durch Reparatur neu entstehenden Fehler proportional zur Anzahl N_{Rep} der Reparaturversuche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

Systeme mit mehr als einem zu erwartenden Fehler



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



Aus dem Modell abzuschätzbar:

- zu erwartende Anzahl der Reparaturen $E(N_{\mathrm{Rep}})$ und
- zu erwartende Fehleranzahl $E(\varphi_{\mathrm{Rep}})$ nach allen Reparaturen

in Abhängigkeit von der

- zu erwart. Fehleranz. aus dem Entstehungsprozess $E(\varphi_{\mathrm{EP}})$,
- der Fehlererkennungswahrscheinlichkeit p_{E} und
- der Erfolgswahrscheinlichkeit der Reparatur p_{R}.

Fehleranzahl nach der Beseitigungsiteration

Zu beseitigen sind

- φ_{EP} Fehler aus dem Entstehungsprozess und
- φ_{ERep} bei Reparaturversuchen entstehende Fehler:

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

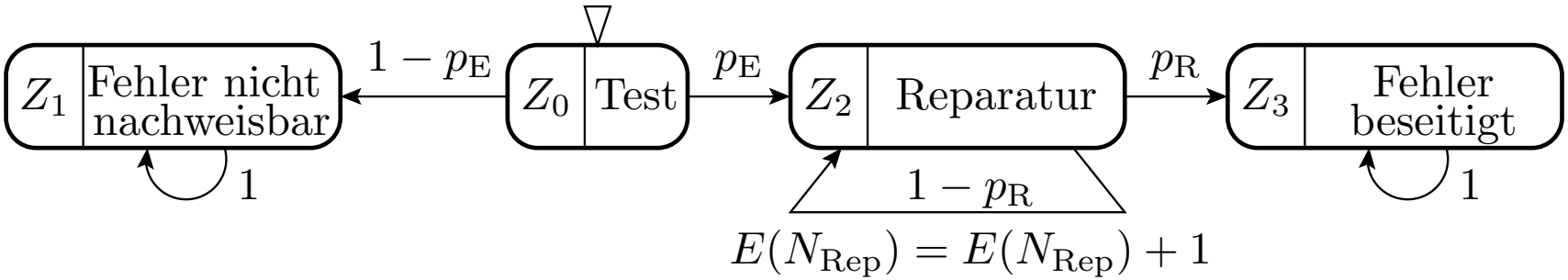
(N_{Rep} – Reparaturversuche je Fehler; φ_{EP} – Fehleranzahl Entstehungsprozess; η_{Rep} – Fehlerentstehungsrate, neu entstehende Fehlern je Reparaturversuch).

Zu erwartende Fehleranzahl nach der Beseitigungsiteration:

$$E\left(\varphi_{\mathrm{Rep}}\right) = \left(E\left(\varphi_{\mathrm{EP}}\right) + \underbrace{E\left(N_{\mathrm{Rep}}\right) \cdot E\left(\varphi_{\mathrm{EP}}\right) \cdot \eta_{\mathrm{Rep}}}_{E(\varphi_{\mathrm{ERep}})} \right) \cdot \left(1 - p_{\mathrm{B}}\right) \qquad (4)$$

(p_{B} – Beseitigungswahrscheinlichkeit, abschätzbar über eine Markov-Kette der Reparaturiteration für vorhandene Fehler).

Markov-Kette für einen vorhandenen Fehler



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

$$p_B = p_{Z3} = p_E \cdot p_R \cdot \sum_{n=0}^{\infty} (1 - p_R)^n = p_E$$

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

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

Fehleranzahl nach der Beseitigungsiteration

Gl. 4 mit $p_B = p_E$ und $E(N_{Rep}) = \frac{p_E \cdot (1-p_R)}{p_R}$:

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

Zusammenfassen von p_R und η_{Rep} zu einer Gütekenngröße:

$$Q_{Rep} = \frac{p_R}{(1-p_R) \cdot \eta_{Rep}}$$

* η_{Rep} in neue Fehler je beseitigter Fehler

η^*_{Rep}	$p_R = 10\%$	20%	50%	80%
2	0,056	0,125	0,5	2
1	0,25	0,5	1	4
0,5	0,5	1	2	8
0,1	2,5	5	10	40

Zu erwartende Fehleranzahl nach der Reperaturiteration:

$$E(\varphi_{Rep}) = E(\varphi_{EP}) \cdot \left(1 + \frac{p_E}{Q_{Rep}} \right) \cdot (1-p_E) \qquad (5)$$

Verringerrungsfaktor der Fehleranzahl:

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

Im Idealfall $Q_{\mathrm{Rep}} \to \infty$ verringert sich die zu erwartende Fehleranzahl um die Erkennungswahrscheinlichkeit p_{E}:

$$\frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)} = p_{\mathrm{E}}$$

Für große Reparaturgüte $Q_{\mathrm{Rep}} \gg p_{\mathrm{e}}$ unerhebliche schlechter als Idealfall:

$$\frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)} \approx p_{\mathrm{E}}$$

Bei geringer Reparaturgüte $Q_{\mathrm{Rep}} \ll p_{\mathrm{e}}$ kann sich die Fehleranzahl sogar erhöhen, z.B. $p_{\mathrm{R}} = 20\%$ und $\eta_{\mathrm{Rep}} = 2$ neue je beseitigter Fehler $\Rightarrow Q_{\mathrm{Rep}} = 0{,}125$, $p_{\mathrm{E}} = 50\%$:

$$\frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)} = \left(1 + \frac{p_{\mathrm{E}}}{Q_{\mathrm{Rep}}}\right) \cdot \left(1 - p_{\mathrm{E}}\right) = \left(1 + \frac{50\%}{0,125}\right) \cdot \left(1 - 50\%\right) = 2{,}5$$

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

```
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
```

Q_{Rep}	$p_{\mathrm{E}} = 50\%$	$p_{\mathrm{E}} = 80\%$	$p_{\mathrm{E}} = 90\%$	$p_{\mathrm{E}} = 99\%$	$p_{\mathrm{E}} = 99{,}9\%$
0,5	1	52,0%	28,0%	2,98%	0,30%
1,0	75,0%	36,0%	19,0%	1,99%	0,20%
2,0	62,5%	28,0%	14,5%	1,50%	0,15%
10,0	52,5%	21,6%	10,9%	1,10%	0,11%

Typische studentische Programmierarbeiten

$$\boxed{E\left(\varphi_{\mathrm{Rep}}\right) = \left(E\left(\varphi_{\mathrm{EP}}\right)\left(1 + \frac{p_{\mathrm{E}} \cdot (1 - p_{\mathrm{R}})}{p_{\mathrm{R}}} \cdot \eta_{\mathrm{Rep}}\right)\right) \cdot (1 - p_{\mathrm{E}})}$$

Fall A: wenige Testbeispiele, brauchbarer Reparaturprozess, z.B. $p_{\mathrm{E}} = 30\%$ erkennbare Fehler, $\eta_{\mathrm{Rep}} = 0{,}4$ neue je beseitigter Fehler, $p_{\mathrm{R}} = 50\%$:

$$\frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)} = \left(1 + \frac{30\% \cdot (1 - 50\%) \cdot 0{,}4}{50\%}\right) \cdot (1 - 30\%) = 82{,}6\%$$

- Reduktion der Fehleranzahl auf ≈83% (70% nicht erkannte ursprüngliche und 13% 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.

$$\boxed{E\left(\varphi_{\mathrm{Rep}}\right) = \left(E\left(\varphi_{\mathrm{EP}}\right)\left(1 + \frac{p_{\mathrm{E}} \cdot (1 - p_{\mathrm{R}})}{p_{\mathrm{R}}} \cdot \eta_{\mathrm{Rep}}\right)\right) \cdot (1 - p_{\mathrm{E}})}$$

Fall B: Entwurf wird beherrscht, aber Test und Reparaturtechniken nicht, z.B. $p_{\mathrm{E}} = 25\%$ erkennbare Fehler,$\eta_{\mathrm{Rep}} = 0{,}8$ neue je beseitigter Fehler, $p_{\mathrm{R}} = 30\%$:

$$\frac{E\left(\varphi_{\mathrm{Rep}}\right)}{E\left(\varphi_{\mathrm{EP}}\right)} = \left(1 + \frac{30\% \cdot (1 - 30\%) \cdot 0{,}8}{30\%}\right) \cdot (1 - 30\%) = 109{,}2\%$$

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

$$E\left(\varphi_{\mathrm{Rep}}\right) = \left(E\left(\varphi_{\mathrm{EP}}\right)\left(1 + \frac{p_{\mathrm{E}} \cdot (1 - p_{\mathrm{R}})}{p_{\mathrm{R}}} \cdot \eta_{\mathrm{Rep}}\right)\right) \cdot (1 - p_{\mathrm{E}})$$

Fall C: Studierender ist mit seiner Aufgabe überfordert

$$\eta_{\mathrm{Rep}} > 1$$

- Der Studierende »schleust die Fehler« mit sehr vielen Reparaturversuchen durch die Tests,
- erzeugt mehr neue Fehler als er beseitigt,
- lässt Tests, die auch nach vielen Fehlerbeseitigungsversuchen immer noch versagen, weg $p_{\mathrm{E}} \to 0$.

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

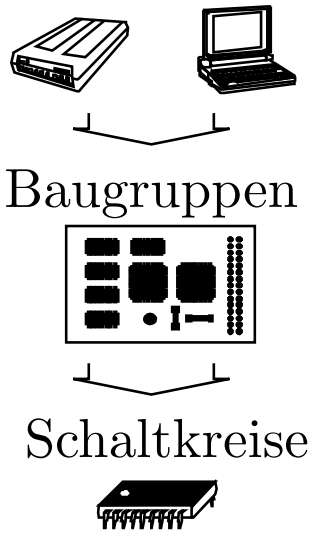
Wenige tauschbare Komponenten

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

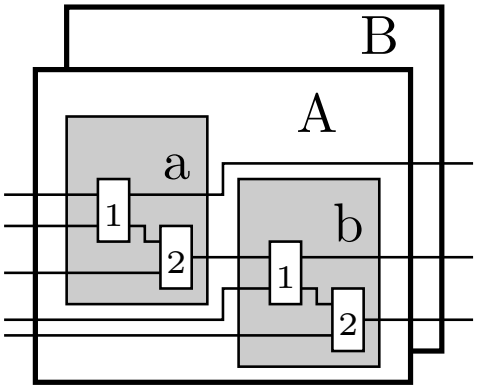
1. Ebene: Austauschbare Geräte.
2. Ebene: Austauschbare Baugruppen.
3. Ebene: Austauschbare Schaltkreise.

Fehlerlokalisierung durch systematisches Tauschen:

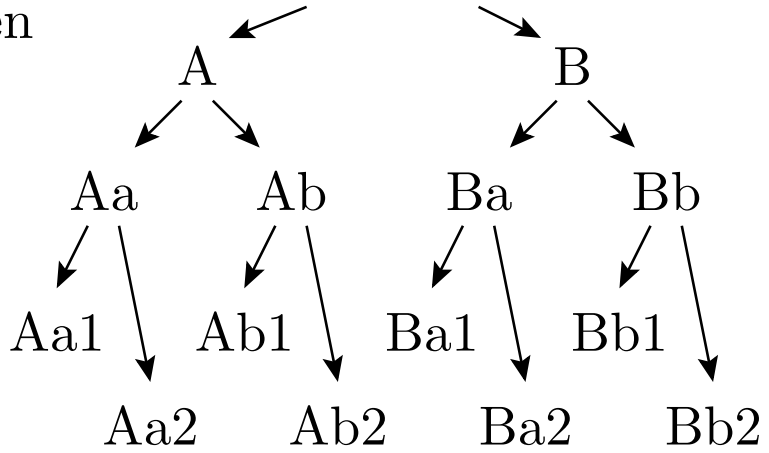
Hierarchie der Hardware



hierarchisches System mit tauschbaren Komponenten



Tauschbaum



Typisches Mechanikervorgehen:

- Grobabschätzung, welches Rechnerteil 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[7].

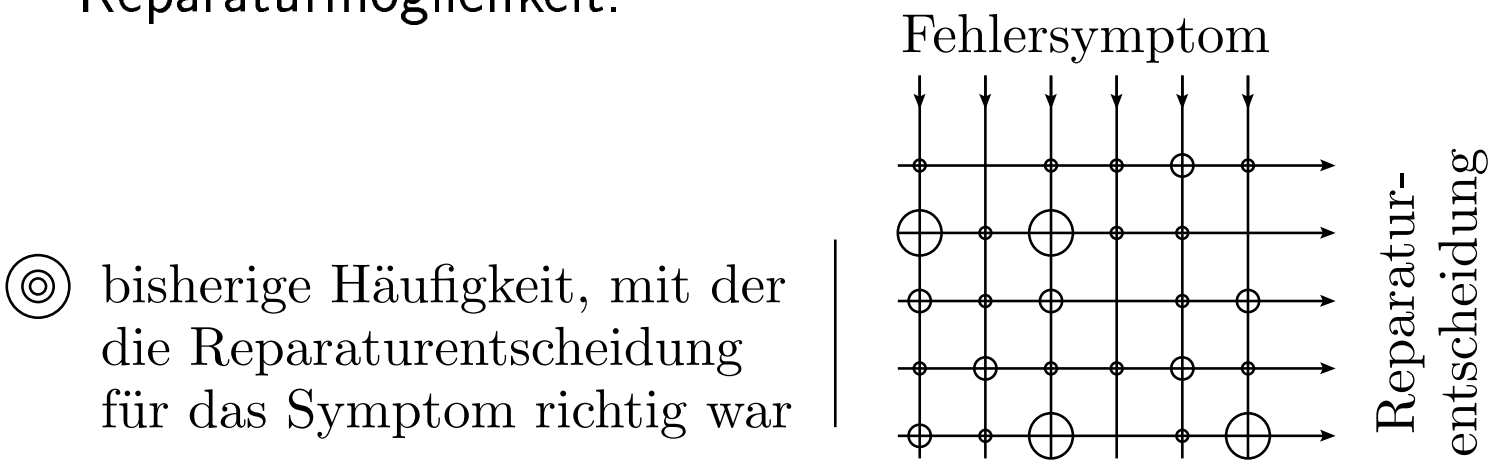
Wichtig ist der Rückbau nach jedem erfolglosen Reparaturversuch. Warum?

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

[7] Verständnis der Funktion des zu reparierenden Systems nicht zwingend.

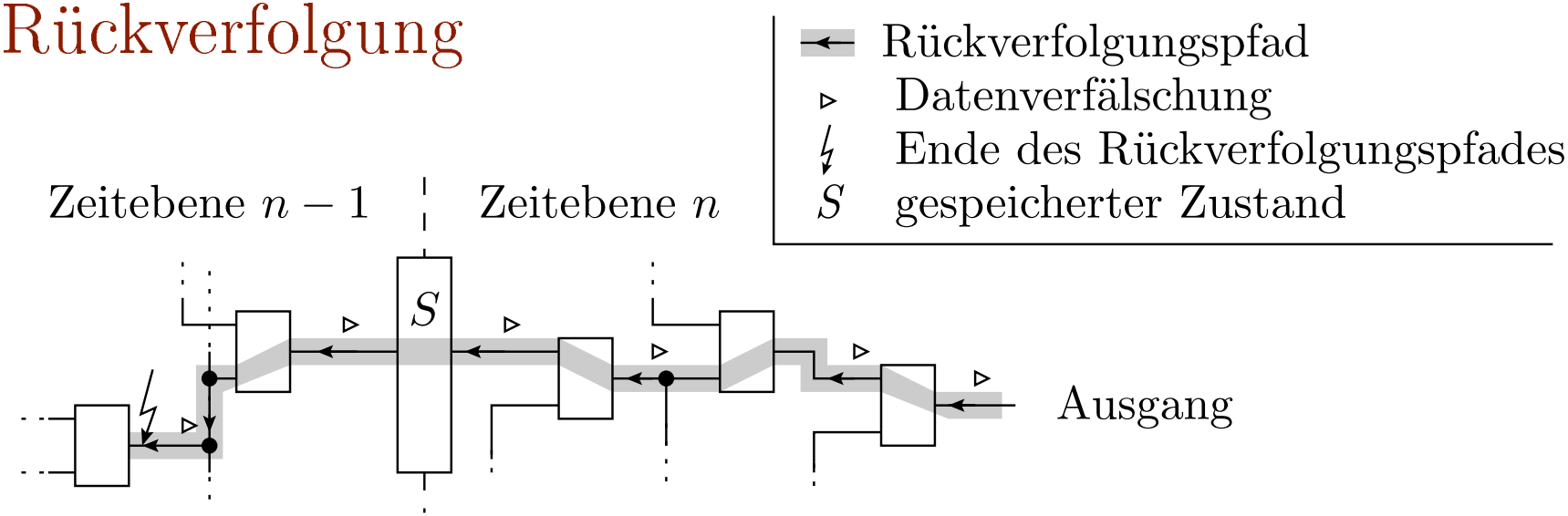
Pareto-Prinzip

- 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.



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

Reifeprozesse

Fehlerbeseitigung und Fehlerumgehung

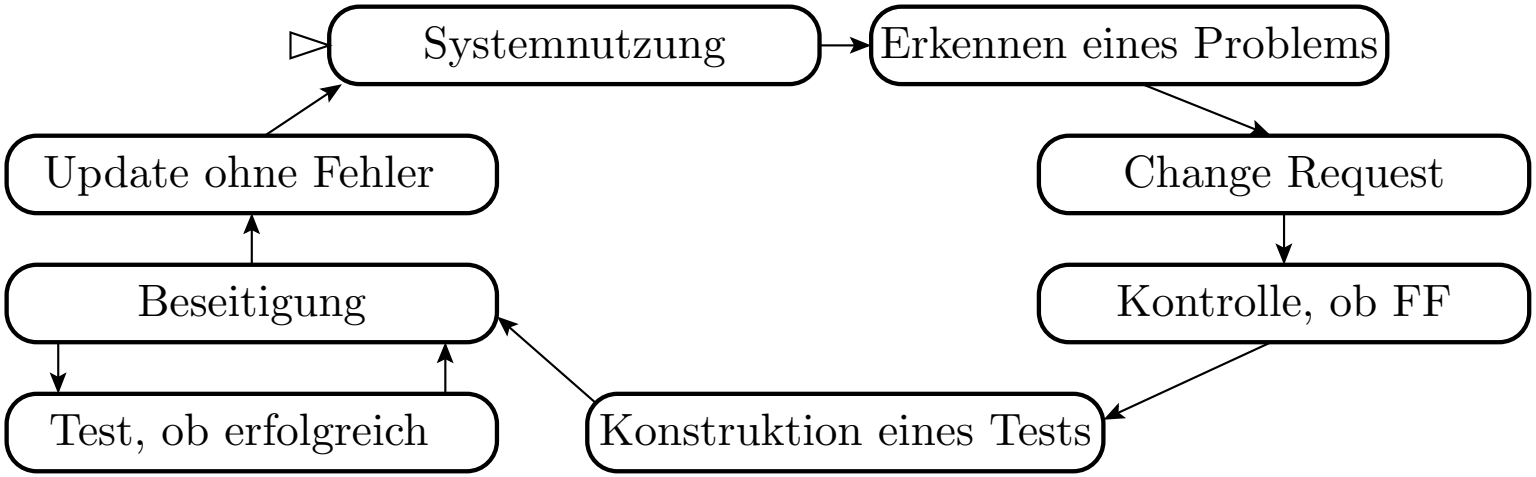
Die Anzahl der Entwurfsfehler in einem System nimmt mit der Systemgröße zu. Tests wirken wie Filter, die nur einen Anteil der Fehler erkennen. Die Zuverlässigkeitsabnahme durch die zunehmende Anzahl nicht beseitigter Fehler in größeren Systemen wird duch Reifeprozesse kompensiert.

Ein Reifeprozess beschreibt eine Zunahme der Zuverlässigkeit mit der Nutzungsdauer durch Einbeziehung der Nutzer als Tester:

- Fortsetzung der Iteration aus Test und Fehlerbeseitigung für Entwurfsfehler im Einsatz.
- Fehlerumgehung: Lernprozess der Nutzer, SL so zu nutzen, dass die Häufigkeit der die Arbeit beeinträchtigenden FF abnimmt.

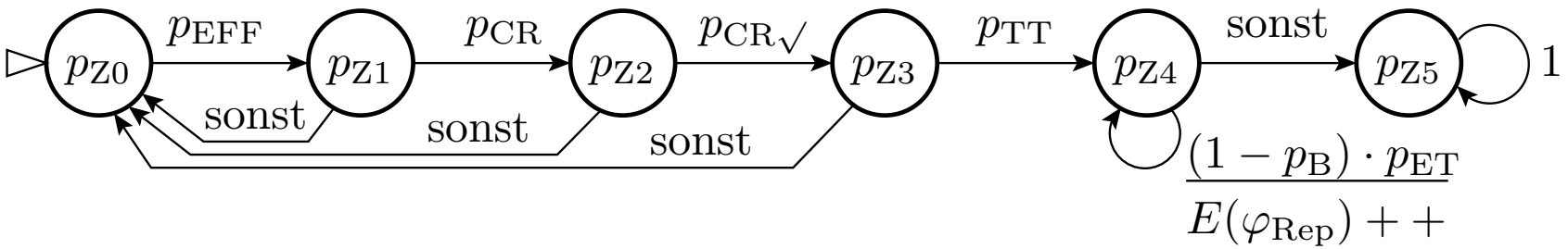
Fehlerbeseitigung

Fehlerbeseitigung in der Nutzungsphase



- Bei einer vermuteten Fehlfunktion stellt der Anwender einen Änderungsanforderung (Change Request).
- Der Hersteller prüft diese, selektiert daraus FFs und versucht, für jede FF reproduzierbare Testbeispiele zu finden.
- Fehlerbeseitigung beim Nutzer erfolgt über Einspielen von Updates, in seltenen Ausnahmen über eine Rückrufaktion für Hardware oder komplette Geräte.

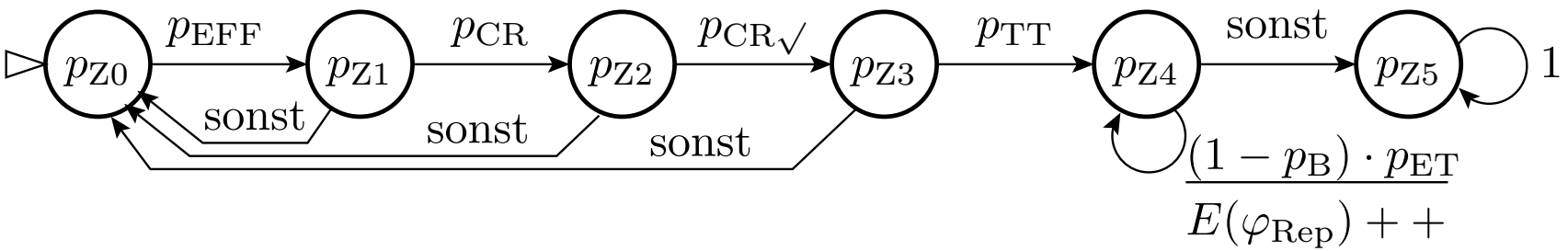
Reifeprozess für einen Fehler als Markov-Kette



p_{EFF}	Fehlererkennungswahrscheinlichkeit	Z_0	System mit Fehler i
p_{CR}	Wahrsch. Change Request gestellt	Z_1	Change Request gestellt
$p_{CR\surd}$	Wahrsch., dass als FF eingestuft	Z_2	Fehlfunktion bestätigt
p_{TT}	Wahrsch. Test konstruierbar	Z_3	Test gefunden
p_{ET}	Erkennungswahrscheinlichkeit des Tests	Z_4	Beseitigungsversuch
p_B	Beseitigungswahrscheinlichkeit	Z_5	Fehler i beseitigt
φ_{Rep}	Anzahl der bei einem Reperaturversuch entstehenden neuen Fehler		

Die Wahrscheinlichkeit, dass ein Fehler beseitigt wird, ist das Produkt der Wahrscheinlichkeiten, dass

- der Fehler bei irgend einem Anwender eine FF verursacht,
- ein Änderungsantrag (Change Request) gestellt, ...



- ...
- der Hersteller die FF bestätigt,
- einen Test für ihren Nachweis findet,
- die Beseitigungsiteraton erfolgreich ist.

- Bei den Fehlerbeseitigungsversuchen und anderen Verbesserungsversuchen entstehen neue Fehler.
- Wenn mehr Fehler beseitigt werden als neue entstehen, reift das System.
- Reifen ist erkennbar an einer mit der Nutzungsdauer abnehmenden Häufigkeit der beobachtbaren Fehlfunktionen (Bedienprobleme, Abstürze, falsche Ergebnisse, ...).

Beseitigung zufällig erkannter Fehler

Die Wahrscheinlichkeit, dass ein Fehler, der im Mittel aller x SL eine FF verursacht, bei einer zufällig ausgewählten SL erkannt wird:

$$p_{\mathrm{FFF}} = \frac{1}{x}$$

Erkannte Fehler werden mit $p_{\mathrm{Bes}} = f\left(p_{\mathrm{CR}}, p_{\mathrm{CR}_{\surd,\ldots}}\right)$ beseitigt.
Wahrscheinlichkeit der Nichtbeseitigung je SL für x SL je FF

$$p_{\mathrm{NBes}}(1) = 1 - \frac{p_{\mathrm{Bes}}}{x}$$

Wahrscheinlichkeit der Nichtbeseitigung nach n Testschritten:

$$p_{\mathrm{NBes}}(n) = \left(1 - \frac{p_{\mathrm{Bes}}}{x}\right)^n \text{ für } \frac{p_{\mathrm{Bes}}}{x} \ll 0{,}1 \quad p_{\mathrm{NBes}} = e^{-\frac{p_{\mathrm{Bes}} \cdot n}{x}}$$

QQ-Funktion für Testlängen $n \gg x_0$:

$$h(x, n, p_{\mathrm{Bes}}) = c \cdot x^{-(k+1)} \cdot e^{-\frac{p_{\mathrm{Bes}} \cdot n}{x}}$$

Zuverlässigkeitswachstum

Verringerung der Wahrscheinlichkeit einer FF je SL:

$$\begin{aligned} p_{\mathrm{FFF}}(n) &= \int_0^\infty c \cdot x^{-(k+2)} \cdot e^{-\frac{p_{\mathrm{Bes}} \cdot n}{x}} \cdot dx \\ &= \frac{c \cdot \Gamma(k+1)}{(p_{\mathrm{Bes}} \cdot n)^{k+1}} \\ &= p_{\mathrm{FFF}}(n_0) \cdot \left(\frac{n}{n_0}\right)^{-(k+1)} \end{aligned}$$

Zunahme der fehlerbezogen Teilzuverlässigkeit mit der Nutzung:

$$Z_{\mathrm{F}}(n) = \frac{(p_{\mathrm{Bes}} \cdot n)^{k+1}}{c \cdot \Gamma(k+1)} = Z_{\mathrm{F}}(n_0) \cdot \left(\frac{n}{n_0}\right)^{k+1}$$

(n – Anzahl der genutzten SL bei allen Nutzern; p_{Bes}– Wahrscheinlichkeit, dass bei einer erkannten FF ein Change-Request gestellt wird, ... und der die FF verursachende Fehler gefunden und beseitigt wird).

Anmerkungen

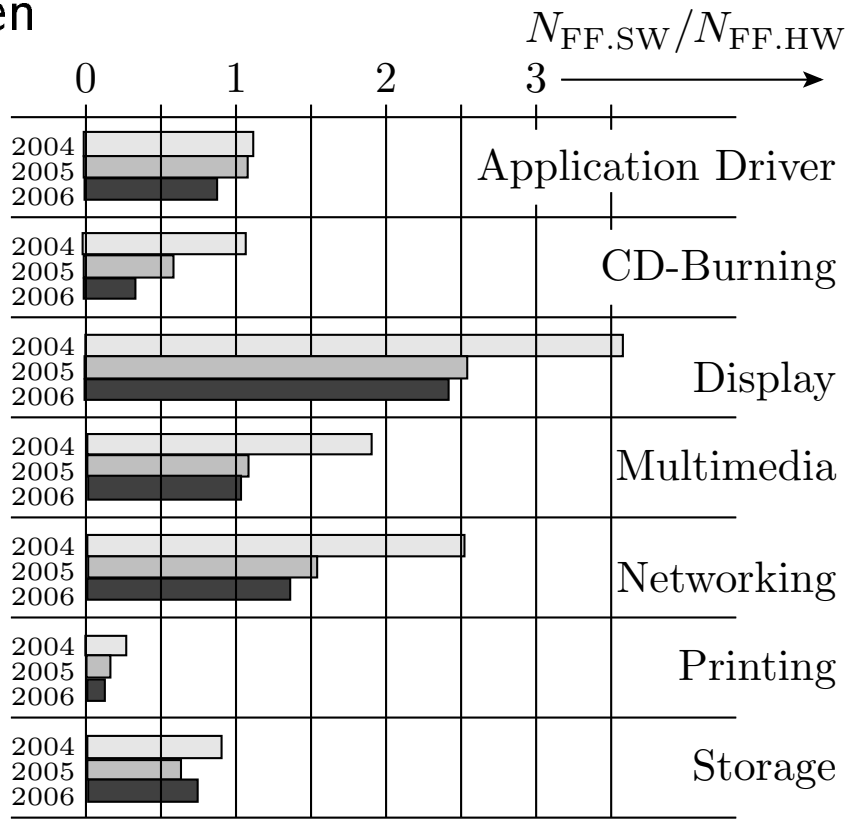
- Die fehlerbezogene Teilzuverlässigkeit nimmt überproportional mit der akkumulierten Nutzungsdauer zu.
- Die Reifegeschwindigkeit hängt von der Nutzungshäufigkeit und der Beseitigungswahrscheinlichkeit p_{Bes} ab. Einflüsse auf p_{Bes}: Informationsfluss über bemerkte FFs von den Anwendern zum Hersteller, ausreichend Bearbeitungskapazität des Herstellers für die Fehlerbeseitigung, ...
- Systeme, die lange gereift sind, haben hohe, auf anderem Wege schwer zu erreichende Zuverlässigkeiten. Schwer ersetzbar durch neue Systeme. (siehe Y2K[8] -Problem).
- Neue (innovative) Systeme sind in den ersten Nutzungsjahren vielfach unzuverlässer als die zuvor genutzten Systeme. Wenn das die Akzeptanz beeinträchtigt, reifen sie auch nicht ...

[8] Year 2000.

Zuverlässigkeitswachstum von Windows-Betriebssystemen[9]:

- Windows 98: $MTBF \approx 1$ Woche
- NT 4.0: $MTBF \approx 5{,}5$ Wochen
- Windows 2000 Professional: $MTBF \approx 4$ Monate.

Durch Treiber verursachte Abstürze unter Windows im Verhältnis zur Anzahl der durch Hardware-Fehler verursachen Abstürze[10].



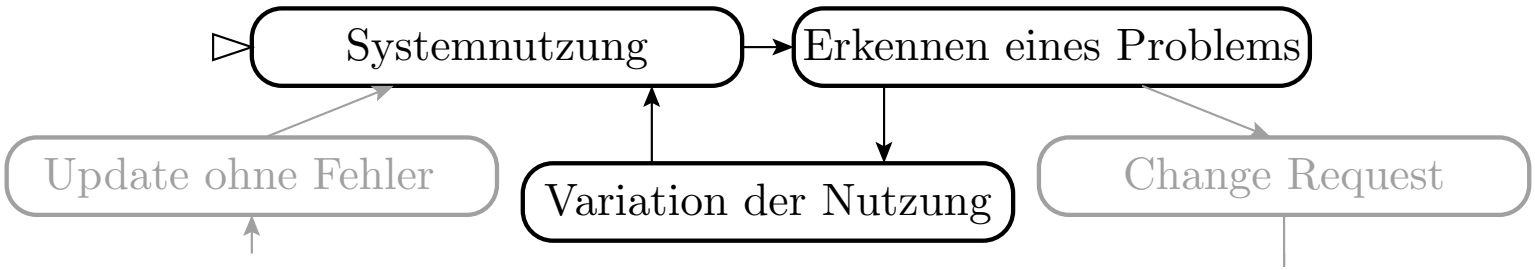
[9] NSTL Test Report, Microsoft Windows 2000 Professional – Comparison of the Reliability of Desktop Operating Systems. Die Quelle sagt nicht, für welche Arten von FF die angegebenen MTBFs gelten.

[10] Glerum, K., Debugging in the (Very) Large: Ten Years of Implementation and Experience (2009), S. 11-14, Fig. 15

Fehlerumgehung

Fehlerumgehung

Große Systeme haben in der Regel eine enorme funktionale Redundanz. Für die meisten Aufgaben gibt es unzählige Lösungswege. Statt über »Chance Request« lösen die Nutzer aufgetretene Probleme durch Variation der Service-Anfragen[11].



- Ähnliches Reifeverhalten wie bei Fehlerbeseitigung.
- Bei Erfahrungsweitergabe mit der Nutzungsdauer aller Nutzer[12].

[11] Jeder Nutzer lernt intuitiv mit der Zeit, welche Eingaben- und Bedienreihenfolgen funktionieren. Dadurch nimmt die beobachtete Zuverlässigkeit auch ohne Fehlerbeseitigung zu.

[12] FAQ-Seiten, Internet-Foren, ...

- Zuverlässigkeitseinbruch bei Personalwechsel[13].

Studentische Arbeiten im Arbeitsbereich, die nur mit schwer zu findenden Fehlerumgehungsmaßnahmen 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. ...
- Falsch platzierte Unterbrechungspunkte in Atmel-Studio, ...

Bei Problemen mit neuer Hard- oder Software am besten zuerst im Internet nach bekannten Fehlerumgehungen suchen, ...

[13] Neue Nutzer haben zu Beginn ähnliche Bedienprobleme, wie sie die erfahrenen Nutzer zu Beginn hatten. Ihr Bedienverhalten reift jedoch schneller, weil für viele Probleme schon Lösungen im Internet zu finden sind.

Ausfälle

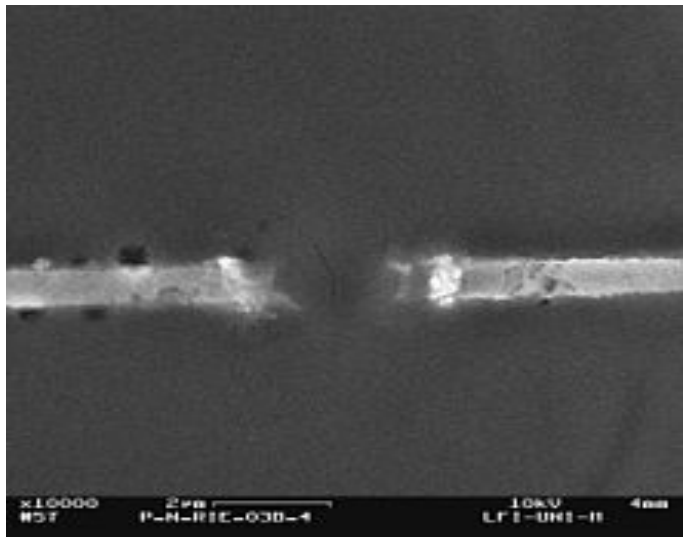
Ausfälle

- Hardware und Mechanik unterliegt einem Verschleiß, der zu Ausfällen führen kann. Bei einem Ausfall entsteht ein Fehler.
- Im Gegensatz zu den nicht nachgewiesenen Herstellungsfehlern verursachen neue Fehler durch Ausfälle im Mittel ähnliche oft FF wie die Fehler in ungetesteten Systemen.
- Komplette Funktionsunfähigkeit $(Z = 0)$ bis unwesentlicher Minderung von Z.
- Eine Sonderstellung haben Frühausfälle. Ihre Ursache sind Beinahefehler[14], die den Verschleiß beschleunigen. Für sie haftet der Hersteller in Form von Garantieleistungen.

[14] Materialrisse, kalte Lötstellen, ...

Ausfallmechanismen

Verschleiß elektronischer Bauteile



Langsam ablaufende physikalische Vorgänge:

- Korrosion (Stecker, Schalter, Isolationen, Leiterbahnen, ...).
- Elektromigation: 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«:

$$R\left(t\right) = P\left(t < t_\mathrm{L}\right)$$

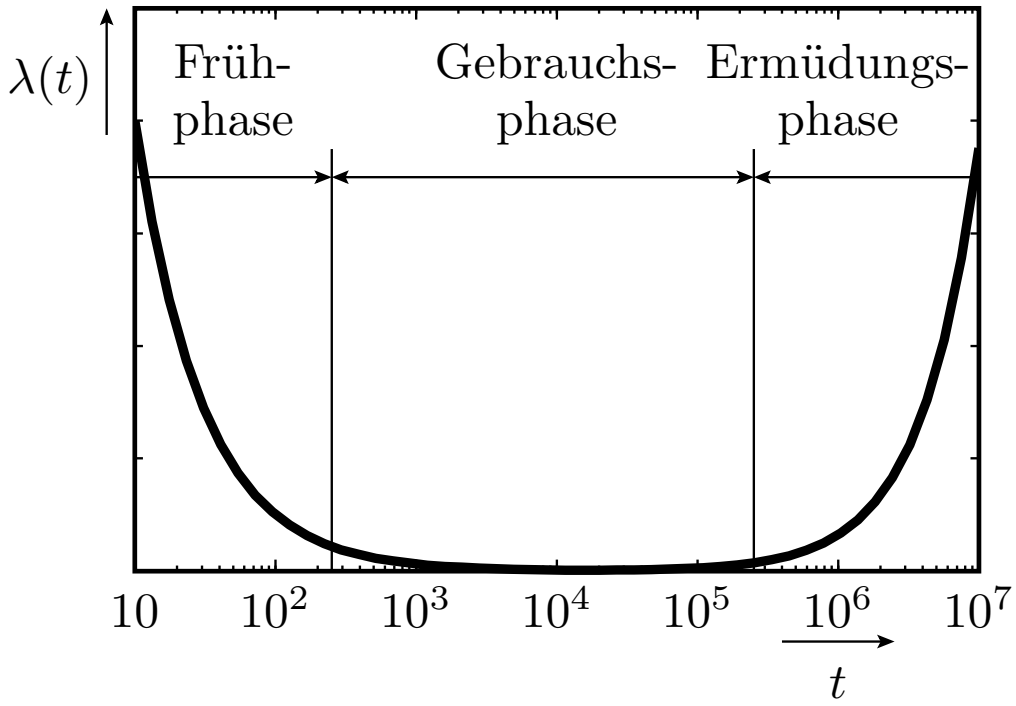
- Ausfallrate λ: Relative Abnahme der Überlebenswahrscheinlichkeit mit der Zeit:

$$\lambda\left(t\right) = -\frac{1}{R\left(t\right)} \cdot \frac{d\,R\left(t\right)}{dt}$$

- Mittlere Lebensdauer:

$$E\left(t_\mathrm{L}\right) = \int_0^\infty R\left(t\right) \cdot dt$$

Ausfallphasen



- Frühausfälle (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[15].

Maßeinheit der Ausfallrate: fit (failure in time)

$$1\,\text{fit} = 1\,\text{Ausfall in } 10^9\,\text{Stunden}$$

[15] Bei SW gibt es diese Phänomen nur als »geplante Obsoleszenz«.

Ausfallraten in der Hauptnutzungsphase nach[16]

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.

[16] Kärger, R.: Diagnose von Computern, Teubner 1996, S. 68

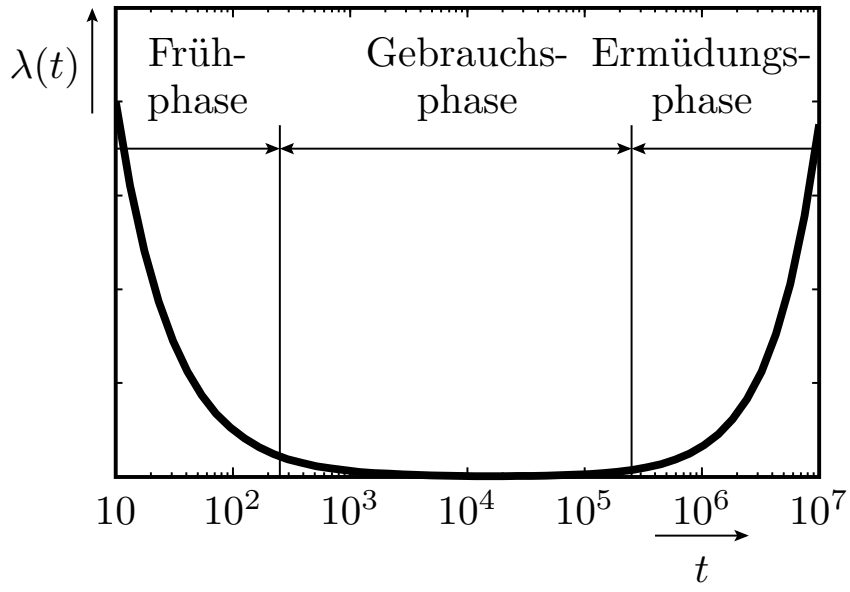
- Ausfallrate einer Baugruppe ist die Summe der Ausfallraten aller Komponenten:

Bauteiltyp	Anzahl n	Ausfallrate λ	$n \cdot \lambda$
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

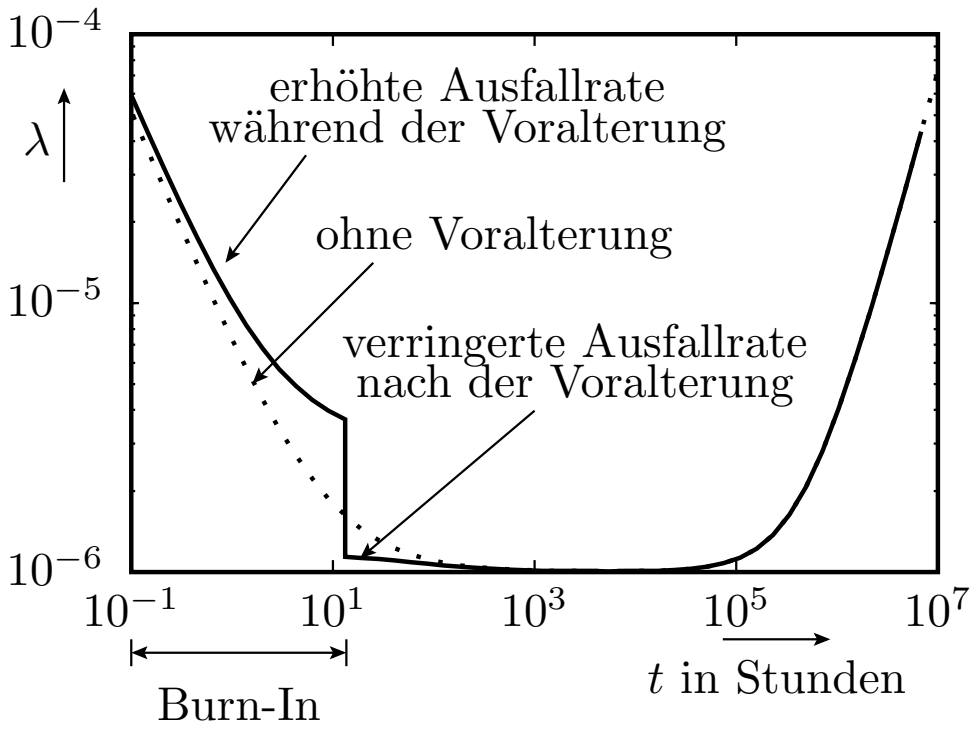
Frühausfälle



- Auf 100 richtige Fehler kommt etwa ein Beinahefehler, der zu einem Frühausfall führt[17].
- 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 und verursachen Kosten für Reparatur, Ersatz, Auftragsabwicklung, ... Was tun?

[17] Barnett, T. S., Singh, A. D.: Relating Yield Models to Burn-In Fall-Out in Time. ITC, 12/2003, S.77-84.

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:

- Periodische Wartungstests.
- Einschalttest.
- 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.

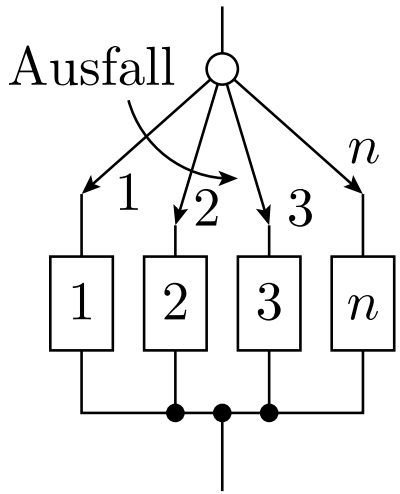
Arten der Reservekomponenten:

- 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	$E\left(t_{\mathrm{L}.1}\right)$
2	$E\left(t_{\mathrm{L}.2}\right)$
3	$E\left(t_{\mathrm{L}.3}\right)$
...	...
Summe:	$E\left(t_{\mathrm{L.ges}}\right) = \sum_{i=1}^{n} E\left(t_{\mathrm{L}.i}\right)$



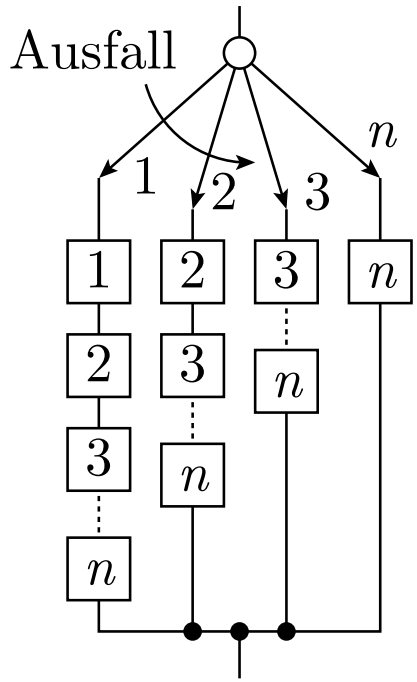
- Die Lebensdauern aller Komponenten addieren sich[18].

[18] Unter der Annahme, dass die Umschalter und die ungenutzten Reserveeinheiten Ausfallrate null haben.

Heiße Reserve

- Alle noch lebenden Komponenten können gleichermaßen ausfallen:

$$E\left(t_{\mathrm{L}.i}\right) = \frac{1}{\sum_{j=1}^{i} \lambda_j}$$

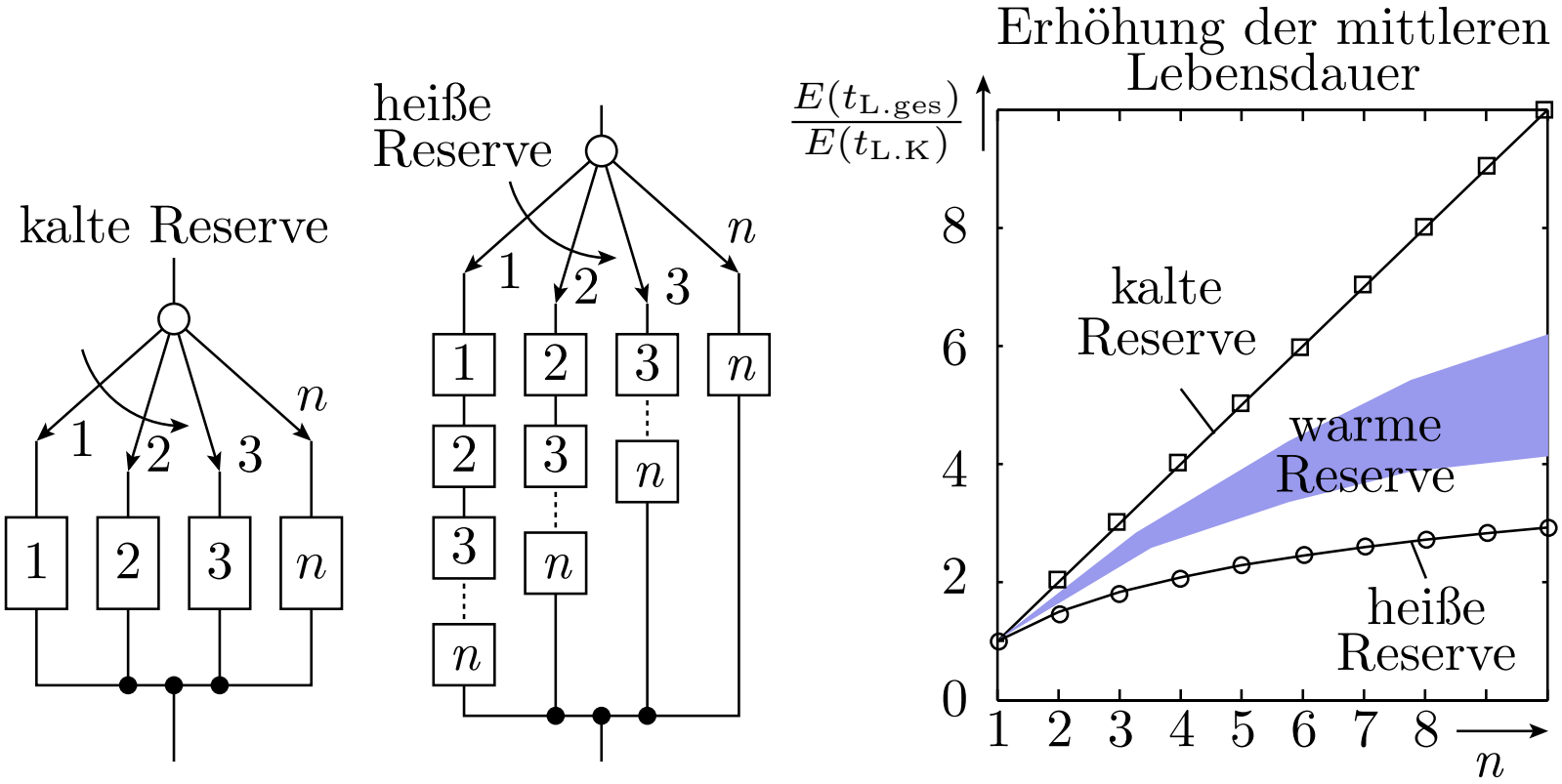


- Komponenten mit gleicher Ausfallrate λ_{K}:

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

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

Warme Reserve

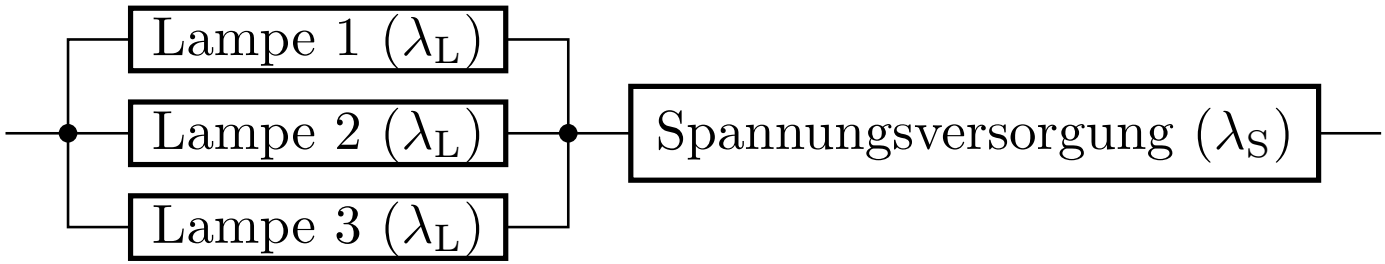


- Die Ausfallrate der »kalten« Ersatzkomponenten ist kleiner als im aktiven Zustand, aber größer null.
- »Warme« Reserveeinheiten verlängert die Lebensdauer mehr als »heiße« und weniger als »kalte«.

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

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

und halbieren die Ausfallrate der Lampeneinheit. Ausfallrate des Gesamtsystems:

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

Umgang mit FF

Programmkonstrukte zum Abfangen von FF

Bei erkannter Fehlfunktionen (Zugriff auf nicht vorhandene Daten, Division durch Null, Wertebereichsverletzung ...) ist eine Fehlerbehandlung erforderlich. Ein typisches Programmkonstrukt zum Abfangen von FF ist die Assert-Anweisung, z.B. in VHDL

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

- *Bedingung*: logische Kontrollausdruck, der bei korrekter Abarbeitung wahr ist.
- *Beschreibung*: Ausgabetext für den Nutzer bzw. das Log-File mit Angaben zur FF und ihrer Umgehung.
- *Severity*: Fehlerschwere. Klassifizierung der Chance, dass der aktuelle Service korrekt zu Ende geführt werden kann.
- Fehlerbehandlung mit SW-Interrupt im System-Kontext.

Eigentliche Fehlerbehandlung im Systemmodus (mit einem nicht kontaminierten Systemzustand, Fehlerisolation):

- Potentielle Gefährdungen beseitigen.
- Daten zur Wiederherstellung und Fehlerlokalisierung sichern.
- Bearbeitung abbrechen oder fortsetzen:
  - Fail Fast: Abbruch bei Risko einer FF. Max. Zuverlässigkeit, verringerte Verfügbarkeit,
  - Fail Slow: Abbruch erst, wenn keine sinnvolle Weiterarbeit mehr möglich. Max. Verfügbarkeit, verringerte Zuverlässigkeit.
- Korrektur der FF nach Abbruch durch Wiederholung[19]:
  - automatisch / mit Benutzerinteraktion.
  - In Eigenverantwortung des Benutzers u.U. mit anderen Eingaben, einem anderen System, ... (Fehlerumgehung).

[19] Wiederholung ist nur sinnvoll, wenn die Chance besteht, dass nicht wieder dieselbe Ursache dieselbe FF bewirkt. Das verlangt Diversität (Verschiedenartigkeit) in Bezug auf die FF-Entstehung.

Fehlerisolation

Fehlerisolation

Eine sinnvolle automatische Reaktion auf eine FF benötigt

- von der Enstehungsursache der FF unbeeinträchtigte Systemteile und
- von der FF nicht kontaminierte Daten.

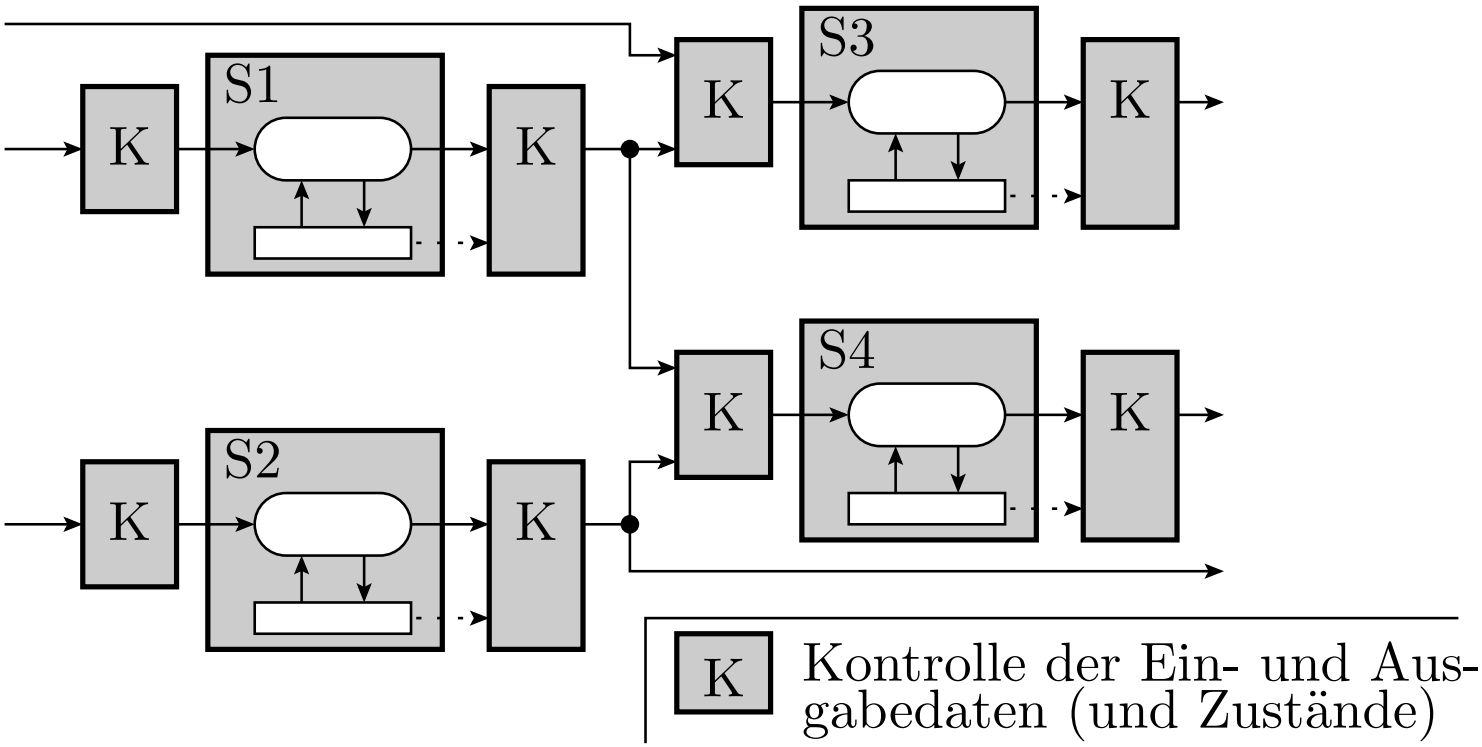
Definition Fehlerisolation

Isolation der Teilsystemen so, dass FF und deren Ursachen (Fehler, Ausfälle, Störungen, ...) in einem Teilsystem die Wahrscheinlichkeit von FF im den anderen Teilsystemen nicht / unerheblich erhöhen.

Techniken zur Fehlerisolation:

- Datenkontrolle und nur Weitergabe korrekter Daten.
- Beschränkung des Schreib- und Lesezugriffs auf fremde Daten.
- Physikatisch und räumlich getrennte Systeme (Risikominderung gleicher Fehler, zeitgleicher Ausfälle, ...).
- ...

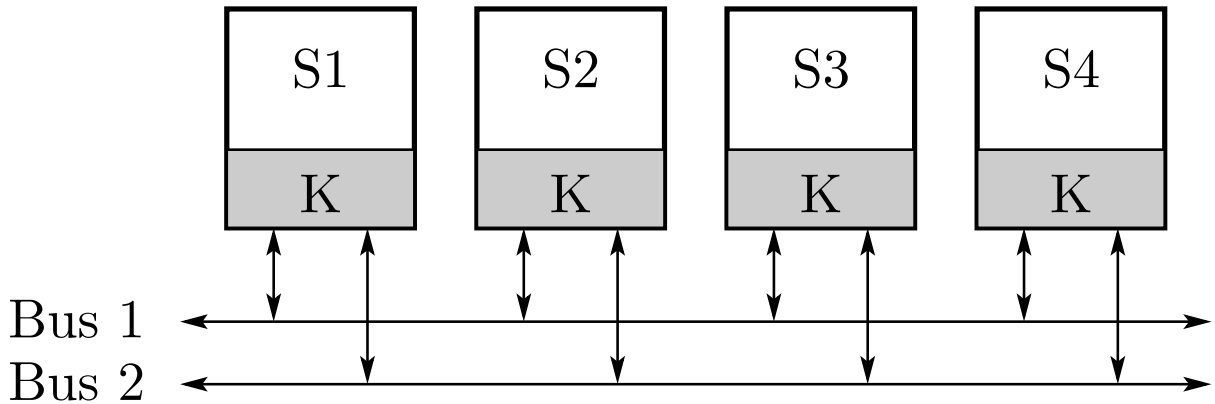
Datenkontrolle und nur Weitergabe korrekter Daten



- Kontrollen der Teil-SL an den Schnittstellen.
- Fehlerbehandlung bei FF.
- Nur Weiterverarbeitung korrekter SL.

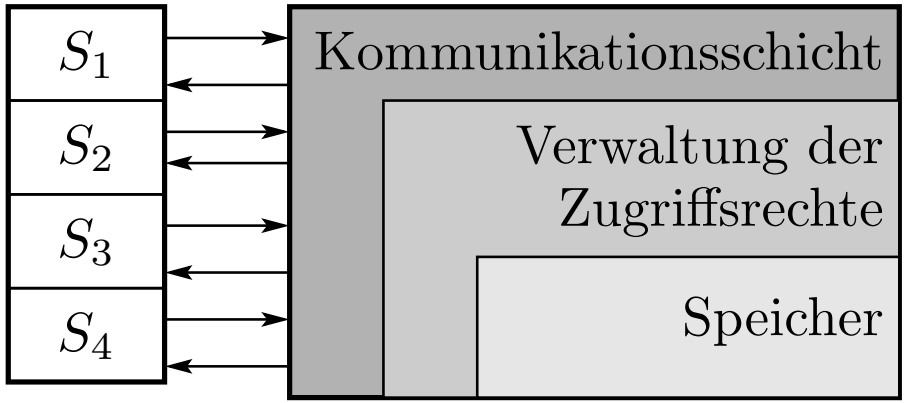
Fehlerisolation in Systemen mit Busstrukturen

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



- Bei Fahrzeugsteuergeräten überwacht jedes, ob die anderen Steuergeräte noch funktionieren.
- Bei diagnostiziertem Versagen, Notfallbehandlung und Protokollierung im Fehlerspeicher.
- Bus 2 als Reserve bei Ausfall der Kommunikation.

Fehlerisolation in Betriebssystemen



- 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.

Resourcen-Zuordnung (physikalischer Speicher, Ein- und Ausgabegeräte, Kommunikation zu anderen Prozessen, ...) nur über Systemrufe möglich. Nur der Betriebssystemkern hat Universalzugriff (und kann alle Daten kontaminieren).

Gefärdungsfreier Zustand

Herstellung eines gefährungsfreien Zustands

Maßnahme zur Erhöhung der Sicherheit (vergl. Abschn. 2.3):

- Betriebssicherheit (Personen- und Umweltschäden),
- Datensicherheit (security, Datendiebstahl),
- Sicherheit Datenerhalt (Datenverlust),
- ...

Der gefährdungsfreie Zustand hängt von der Art der Sicherheit ab.

Beispiele für Fail-Safe-Systeme zur Erhöhung der Betriebssicherheit:

- Zwangsbremsung, wenn Lokführer Haltesignal überfährt.
- Bei erkanntem Sensorausfall Maschinen oder Anlagen kontrolliert anhalten.
- Bei Erkennen eines freien Falls (Laptop fällt herunter), Festplattenköpfe in Parkposition bringen.
- Ruhestromprinzip (siehe nächste Folie).
- ...

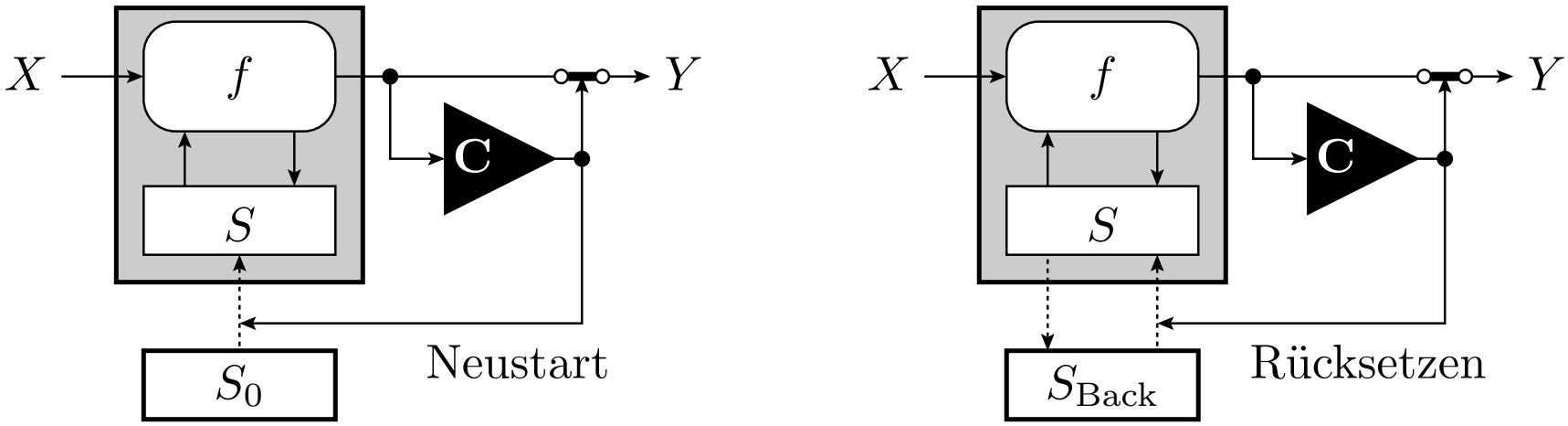
Ruhestromprinzip (gebräuchliche Fail-Safe Technik)

- Sicherheitsbremsen: Bei Stromausfall, Drahtbruch, geplatztem Bremsschlauch, ... setzt Bremswirkung ein.
- Bei Durchtrennen einer Signalleitung einer Alarmanlage oder eines Brandmelders wird Alarm auslösen.
- Bei Drahtseilriss für ein Streckenfreigabesignal bei der Eisenbahn Strecke sperren (Signalarm fällt runter).
- Bei fehlendem Ruhestrom auf einer Meldeleitung, Anzeige einer Störung.
- Bei Leitungsbruch im Notauskreis wird das System ausgeschaltet.
- ...

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.
- Betriebssystemaufruf zur Fehlerbehandlung und/oder 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[20].

[20] So kann der aktuelle Zustand aus dem Backup und den protokollierten Eingaben wiederhergestellt werden.

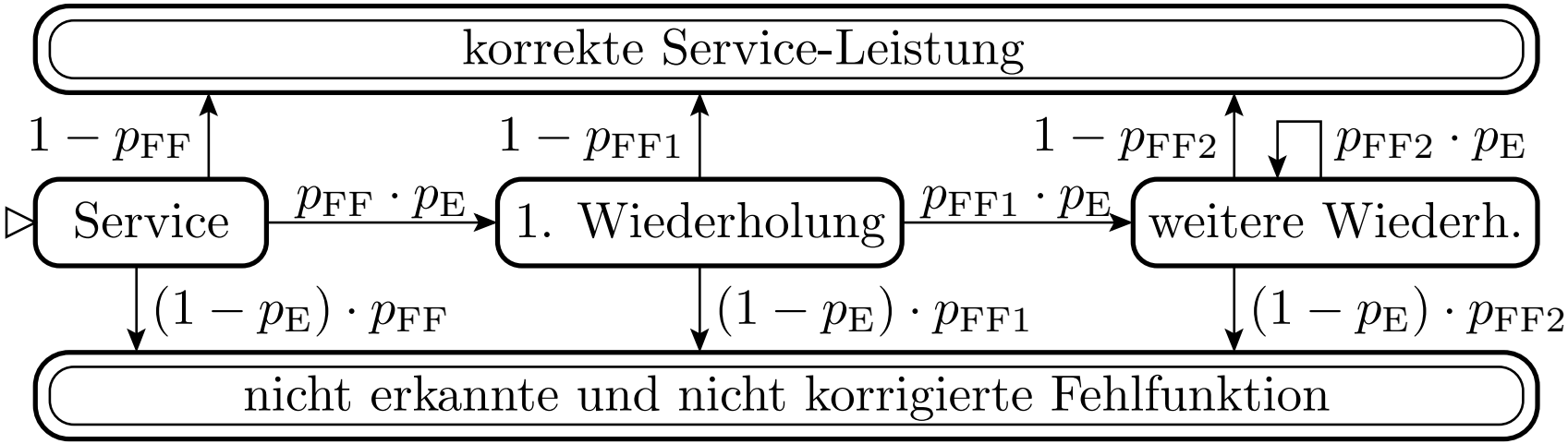
Wiederholung

Erkannte FF werden durch Wiederholung beseitigt. Voraussetzung ist eine Chance, dass die Ursache der FF bei der Wiederholung nicht in derselben Weise zur Wirkung kommt:

- Bei (zufälligen) Störungen oder nicht deterministischer Fehlerwirkung genügt oft einfache Wiederholung.
- Bei Fehlern als Ursachen genügt meist eine (zufällige) Änderung des Berechungswegs durch Änderung der Service-Anfrage.
- Bei Hardware-Fehlern und -Ausfällen hilft auch die Berechnung auf verschiedener /verschiedenartiger HW.
- Alternative zur »Verschiedenartigkeit« (Diversität): Beseitigung der Ursachen (Fehler, Ausfälle, ...) vor der Wiederholung.

Diversität

Fehlerbeseitigung als Markov-Kette



- p_{FF} – Wahrscheinlichkeit einer FF ohne vorherige FF.
- p_{FF1} – Wahrscheinlichkeit einer FF bei der ersten Wiederholung.
- p_{FF2} – Wahrsch. FF bei jeder weiteren Wiederholung.
- p_E – Erkennungswahrscheinlichkeit.

Gleiche Ursache SL und Wiederholung: $p_{FF1} \to 1$. (Wied. hilft nicht.)
Gleiche Ursache bei allen Wiederholungen: $p_{FF2} \to 1$. (Mehr als eine Wiedolung hilft nicht.)

Diversität[21]

Die Erfolgschancen einer Fehlerbeseitigung durch (einfache) Wiederholung hängt von der Verschiedenartigkeit ab bei der Abbildung der Eingaben und Fehler auf die Ausgaben bei der SL und der Wiederholung nach einer FF.

Definition Diversität:

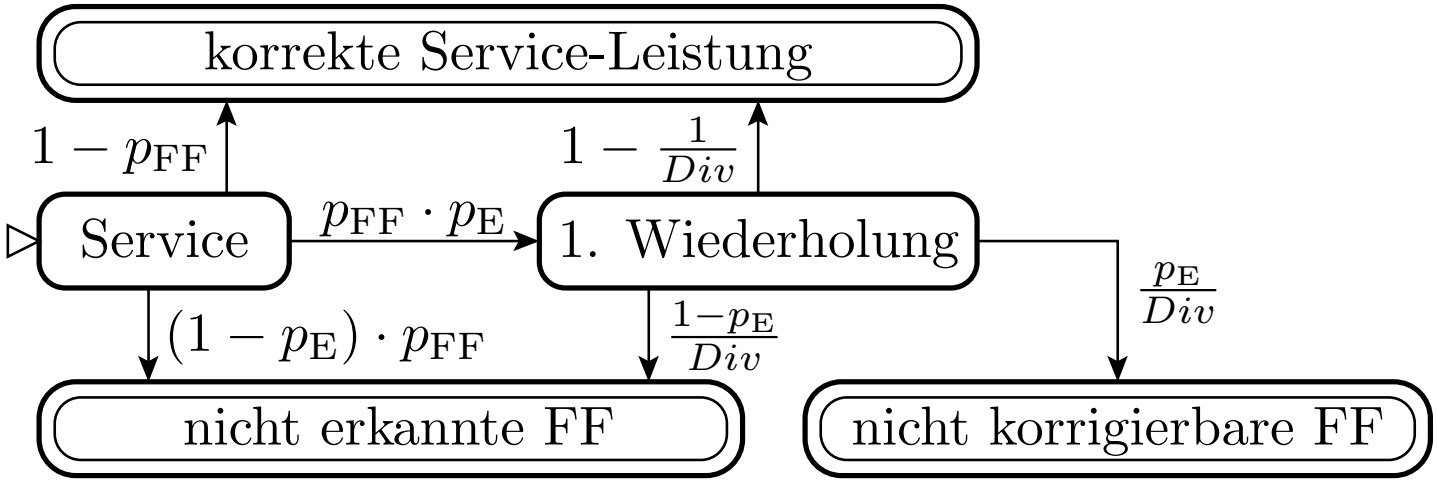
Kehrwert der Wahrscheinlichkeit einer nochmaligen FF bei Wiederholung:

$$Div = \frac{1}{p_{\mathrm{FF1}}} \quad \text{mit } 1 \leq Div \leq \frac{1}{p_{\mathrm{FF}}} = Z$$

Die Diversität ist mindestens 1 (keine Fehlerbeseitigung durch Wiederholung möglich) und maximal gleich der Zuverlässigkeit der SL (unabhängige FF bei Wiederholung).

[21] Von lateinisch »diversitas« (Vielfalt) abgeleitet.

Geringe Diversität, max. eine Wiederholung

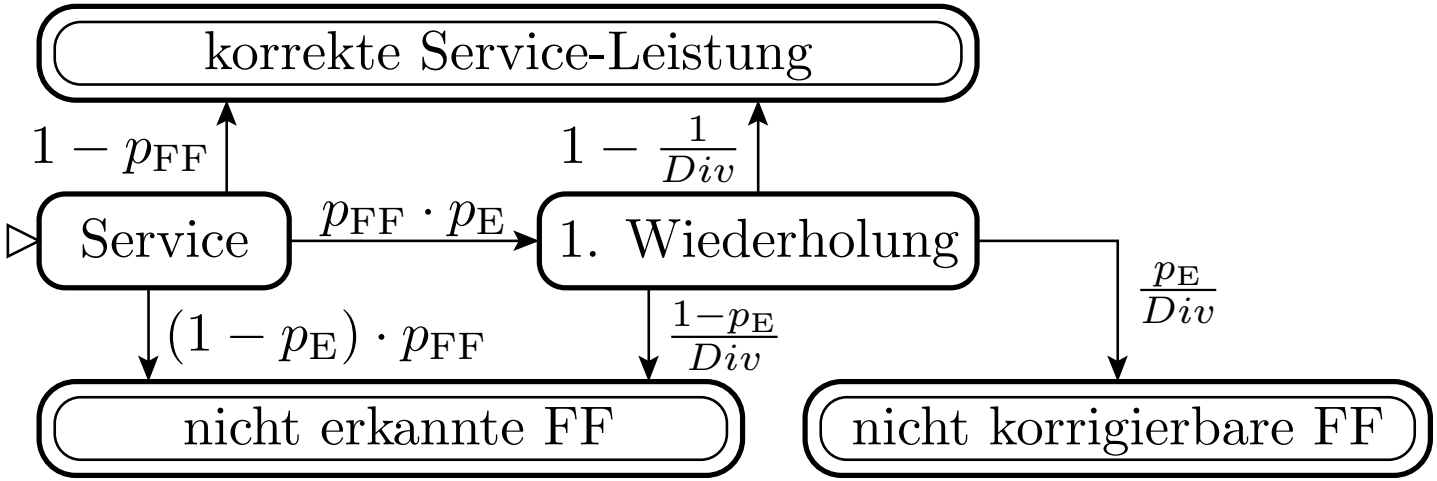


Wahrscheinlichkeit einer insgesamt korrekten SL:

$$p_{\mathrm{KSW}} = 1 - p_{\mathrm{FF}} + p_{\mathrm{FF}} \cdot p_{\mathrm{E}} \cdot \left(1 - \frac{1}{Div}\right)$$

Wahrscheinlichkeit einer erkennbaren nicht korrigierbaren FF:

$$p_{\mathrm{EFW}} = \frac{p_{\mathrm{FF}} \cdot p_{\mathrm{E}}^2}{Div}$$



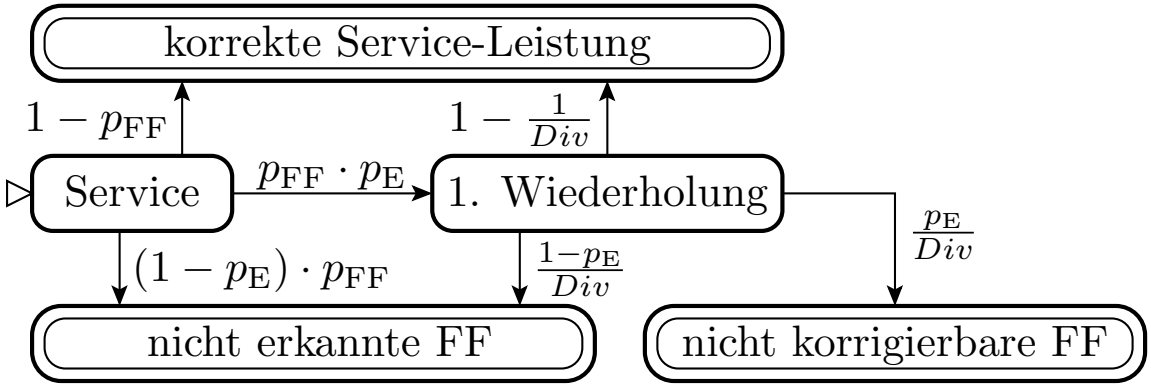
Verfügbarkeit V der SL ohne Wiederholung und Verfügbarkeit V_{W} mit Wiederholung:

$$V = 1 - p_{\mathrm{FF}} \cdot p_{\mathrm{E}};\ V_{\mathrm{W}} = 1 - \frac{p_{\mathrm{FF}} \cdot p_{\mathrm{E}}^2}{Div}$$

Verringerungsfaktor der Nichtverfügbarkeit:

$$\frac{1 - V_{\mathrm{W}}}{1 - V} = \frac{p_{\mathrm{E}}}{Div}$$

umgekehrt proportional zur Diversität.



Wahrscheinlichkeit p_{NF}/p_{NFW} einer nicht erkennbaren FF ohne/mit Wiederholung:

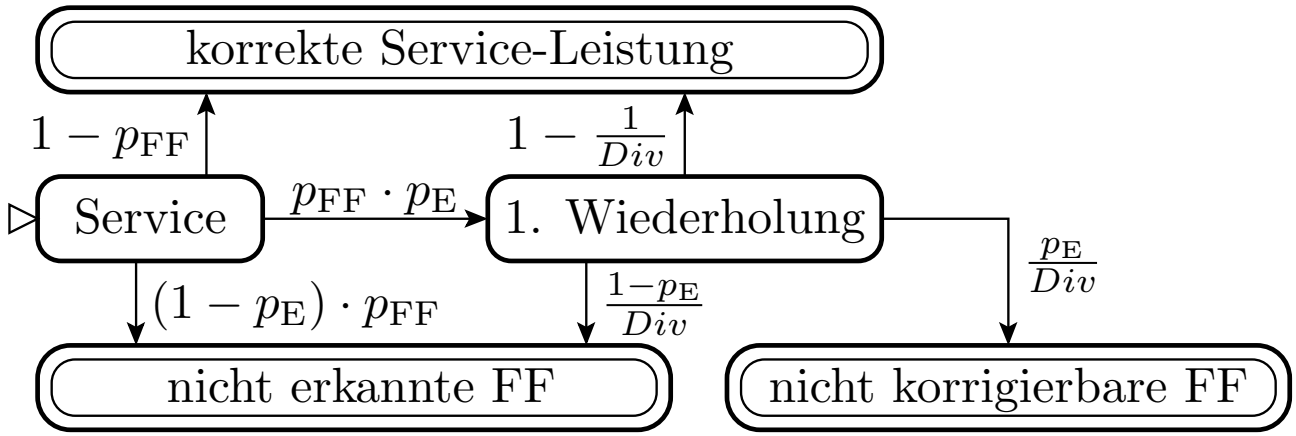
$$\begin{aligned} p_{NF} &= (1 - p_E) \cdot p_{FF} \\ p_{NFW} &= (1 - p_E) \cdot p_{FF} + \frac{p_{FF} \cdot p_E \cdot (1 - p_E)}{Div} \\ &= (1 - p_E) \cdot p_{FF} \cdot \left(1 + \frac{p_E}{Div}\right) \end{aligned}$$

Zuverlässigkeit als Kehrwert der Wahrsch. einer nicht erkannten FF:

$$Z = \frac{1}{(1 - p_E) \cdot p_{FF}};\ Z_W = \frac{1}{(1 - p_E) \cdot p_{FF} \cdot \left(1 + \frac{p_E}{Div}\right)}$$

Zuverlässigkeitsminderung?

... bei Korrektur durch max. eine Wiederholung



$$\frac{Z_W}{Z} = \frac{(1 - p_E) \cdot p_{FF}}{(1 - p_E) \cdot p_{FF} \cdot \left(1 + \frac{p_E}{Div}\right)} = \frac{1}{1 + \frac{p_E}{Div}}$$

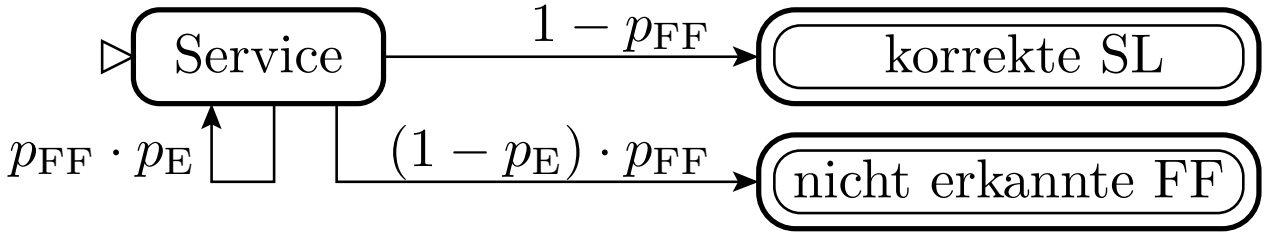
Mit der getroffenen Modelannahme, dass die FF der SL und der Wiederholung unabhängig voneinander erkannt werden, und $Div \geq 1$ $Z_W \geq \frac{Z}{2}$. Da aber Kontrollen meist gleiche FF wiedererkennen, keine Zuverlässigkeitsverringerung.

Ideale Diversität

Bei idealer (maximaler) Diversität

$$Div = \frac{1}{p_{\mathrm{FF}}} = Z \quad \text{und} \quad p_{\mathrm{FF1}} = p_{\mathrm{FF2}} = p_{\mathrm{FF}}$$

Iteration bis zur Korrektur aller erkennbaren FF[22]:



Wahrscheinlichkeit einer nicht erkennbaren FF:

$$p_{\mathrm{NF}} = (1 - p_{\mathrm{E}}) \cdot p_{\mathrm{FF}} \quad \text{ohne Wiederholung}$$

$$p_{\mathrm{NFW}} = (1 - p_{\mathrm{E}}) \cdot p_{\mathrm{FF}} \sum_{i=0}^{\infty} (p_{\mathrm{E}} \cdot p_{\mathrm{FF}})^i = \frac{(1 - p_{\mathrm{E}}) \cdot p_{\mathrm{FF}}}{1 - p_{\mathrm{E}} \cdot p_{\mathrm{FF}}}$$

[22] (Verfügbarkeit $V_{\mathrm{W}} = 1$)

Zuverlässigkeitsminderung?

Die Zuverlässigkeit ohne und mit Korrektur beträgt:

$$Z = \frac{1}{p_{\mathrm{NF}}} = \frac{1}{(1-p_{\mathrm{E}})\cdot p_{\mathrm{FF}}}$$

$$Z_{\mathrm{W}} = \frac{1}{p_{\mathrm{NFW}}} = \frac{1-p_{\mathrm{E}}\cdot p_{\mathrm{FF}}}{(1-p_{\mathrm{E}})\cdot p_{\mathrm{FF}}}$$

Die Verringerung der Zuverlässigkeit durch Wiederholung

$$\frac{Z_{\mathrm{W}}}{Z} = 1 - p_{\mathrm{E}}\cdot p_{\mathrm{FF}}$$

ist bei geringer Wahrscheinlichkeit einer FF je SL ($p_{\mathrm{FF}} \ll 1$) vernachlässigbar.

Natürliche und geplante Diversität

Jede Berechnung hat eine natürliche Diversität durch zufällig wirkende Einflüsse:

- Bitverfälschungen bei der Speicherung und Übertragung.
- FF durch Übertaktung, Überspannung, ...
- FF durch andere auf dem Rechner laufende Programme.

Bei natürlicher Diversität genügt zur Ergeniskorrektur Wiederholung auf dem selben System.

Geplante Diversität. Zu unterscheiden ist zwischen:

- diversitäre Realisierung: gleiche Sollfunktion / Sollwerte für ein Ergebniskontrolle durch Mehrfachberechnung und Vergleich.
- diversitäre Sollfunktion: Wahl eines verschiedenartigen Lösungswegs, auch mit abweichender Sollfunktion zur Lösung derselben Aufgabe, schließt Ergebniskontrolle durch Mehrfachberechnung und Vergleich aus.

Diversitäre Realisierung

Bei automatischer Korrektur durch Wiederholung erfolgt oft die Ergebniskontrolle durch Mehrfachberechnung und Vergleich. Das schränkt die Vielfalt der Berechnungsmöglichkeiten auf die mit übereinstimmenden Sollwerten ein. Unterschieden wird:

- Hardware-Diversität: Berechnung auf unterschiedlicher Hardware. Erkennbar sind zusätzlich FF durch Fertigungsfehler und Ausfälle der HW.
- Hardware-Entwurfsdiversität: Berechnung mit unabhängig voneinander entworfener HW. Zusätzlich FF durch HW-Entwurfsfehler.
- Syntaktische Diversität: Berechnung mit unterschiedlich übersetzter Software. Zusätzlich FF durch den Compiler.
- Software-Diversität: Berechnung mit unabhängig voneinander entworfener SW. Zusätzlich FF durch SW-Entwurfsfehler.

Anmerkungen zur Software-Diversität

Software-Fehler als Hauptquelle für FF verlangen SW-Diversität:

- Komplette Entwicklung mindestens zweimal.
- durch getrennte Teams, keine Kommunikation,
- aus einer nicht diversitären Spezifikation, ...

Ursprüngliche euphorische Meinung, so liesen sich alle FF durch Fehler außer denen in der Spezifikation korrigieren, nicht bestätigt. Die direkte oder indirekte Kommunikation der Entwicklungsteams über die Interpretation der Spezifikation, während des Test etc. trägt Gemeinsamkeiten in die Entwürfe. Neigung von Menschen, gewisse Fehler zu wiederholen, ... Erzielbare Divisität laut[23]

$$Div \leq 10$$

[23] U. Voges, Software-Diversität und ihre Modellierung - Software-Fehlertoleranz und ihre Bewertung durch Fehler- und Kostenmodelle, Springer (1989)

Diversitäre Sollfunktion

Variationen

- der Art der Eingabe (über Menüs, Fenster, Konsolen, ...)
- der Eingabereihenfolge und Werte,
- der genutzten Algorithmen, Programme, ...

führt auf unterschiedliche richtige Ergebnisse

- höhere Diversität,
- aber sich unterscheidende Sollwerte und Formate.

Diversitäre Lösungswege werden für die manuelle Fehlerumgehung genutzt (vergl. Abschn. 2.2, Folie 48). Für die automatische Korrektur in fehlertoleranten Systemen ungebräuchlich.

Fehlertoleranz

Redundanz und Fehlertoleranz

Fehlertoleranz: Korrektur eigener, selbst erkannter Fehlfunktionen unter Verwendung redundanter[24] Funktionseinheiten:

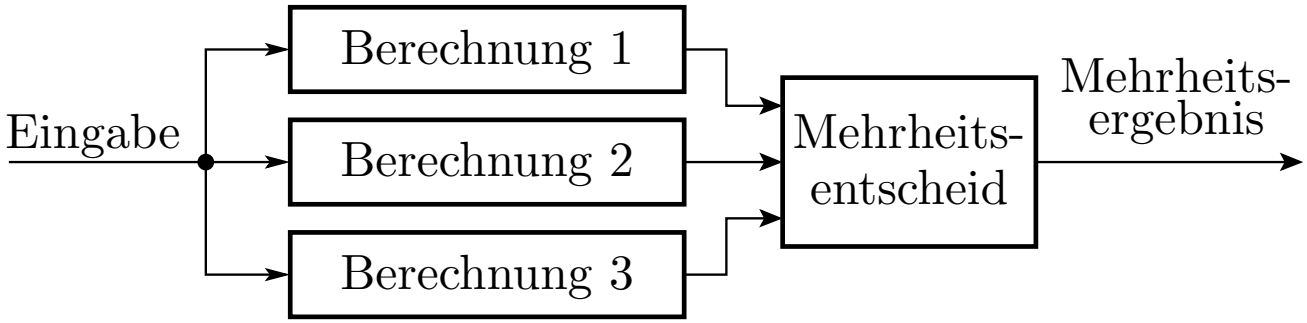
- Informationsredundanz: siehe fehlerkorrigierende Codes.
- homogene Redundanz (Redundanz mit gleichartigen Mitteln): Reserveeinheiten für Hardware-Ausfälle, z.B. zwei Glühbirnen im roten Teil einer Ampel (siehe heiße und kalte Reserve, Folie 64) und
- diversitäre Redundanz (Redundanz mit ungleichartigen Mitteln) zur Korrektur von FF durch Entwurfs- und Programmierfehler.

Im Bereich der diversitären Redundanz haben sich durchgesetzt:

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

[24] 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

- 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.

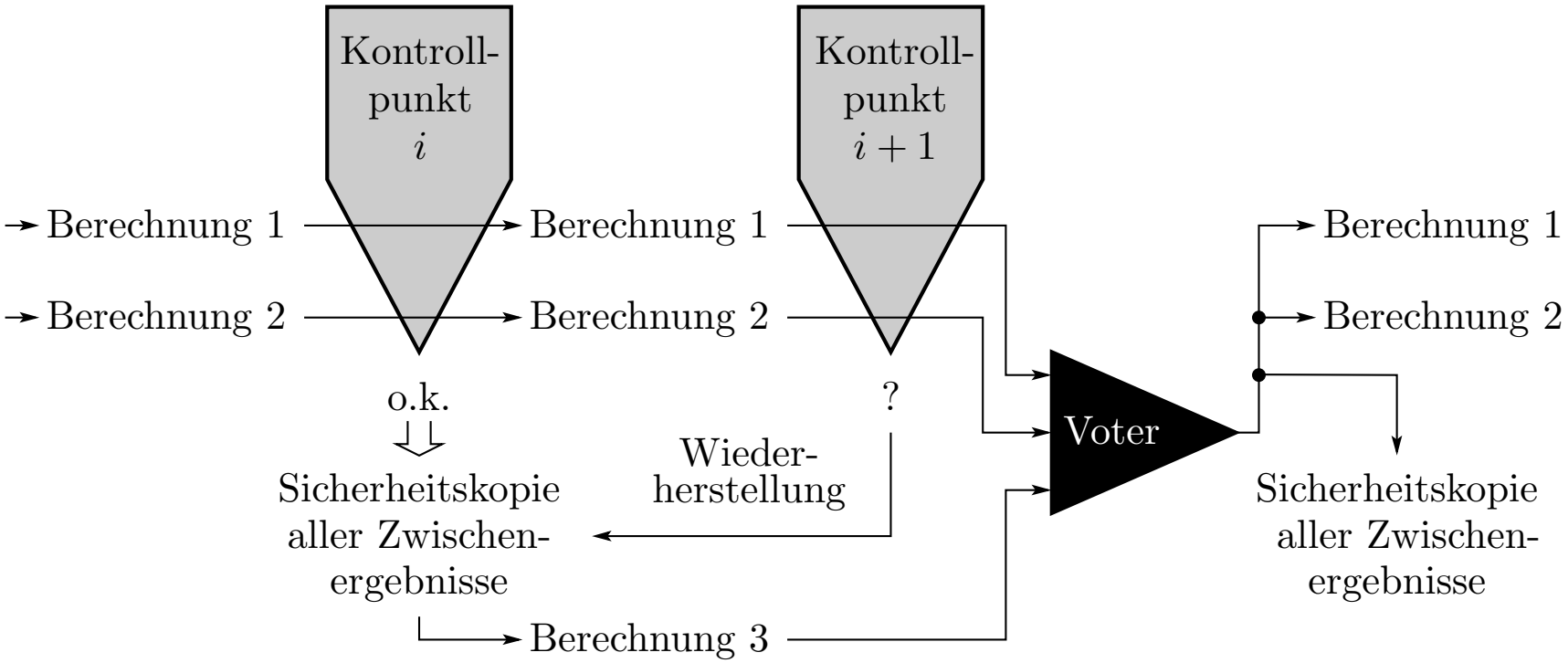
- Ein Mehrversionssystem in dieser Form toleriert keine übereinstimmenden Entwurfsfehler in den Systemen.

  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 eine Vielzahl weiterer Kontrollen und einer Vielzahl einprogrammierten Verhaltensweisen bei Fehlfunktionen.

Check-Point-Roll-Back-Recovery [3]



- Nur zwei parallel ausgeführte Berechnungen.
- An einprogrammierten Kontrollpunkten im Programm werden die Bearbeitungszustände[25] verglichen.

[25] Werte der Variablen, Register, ...

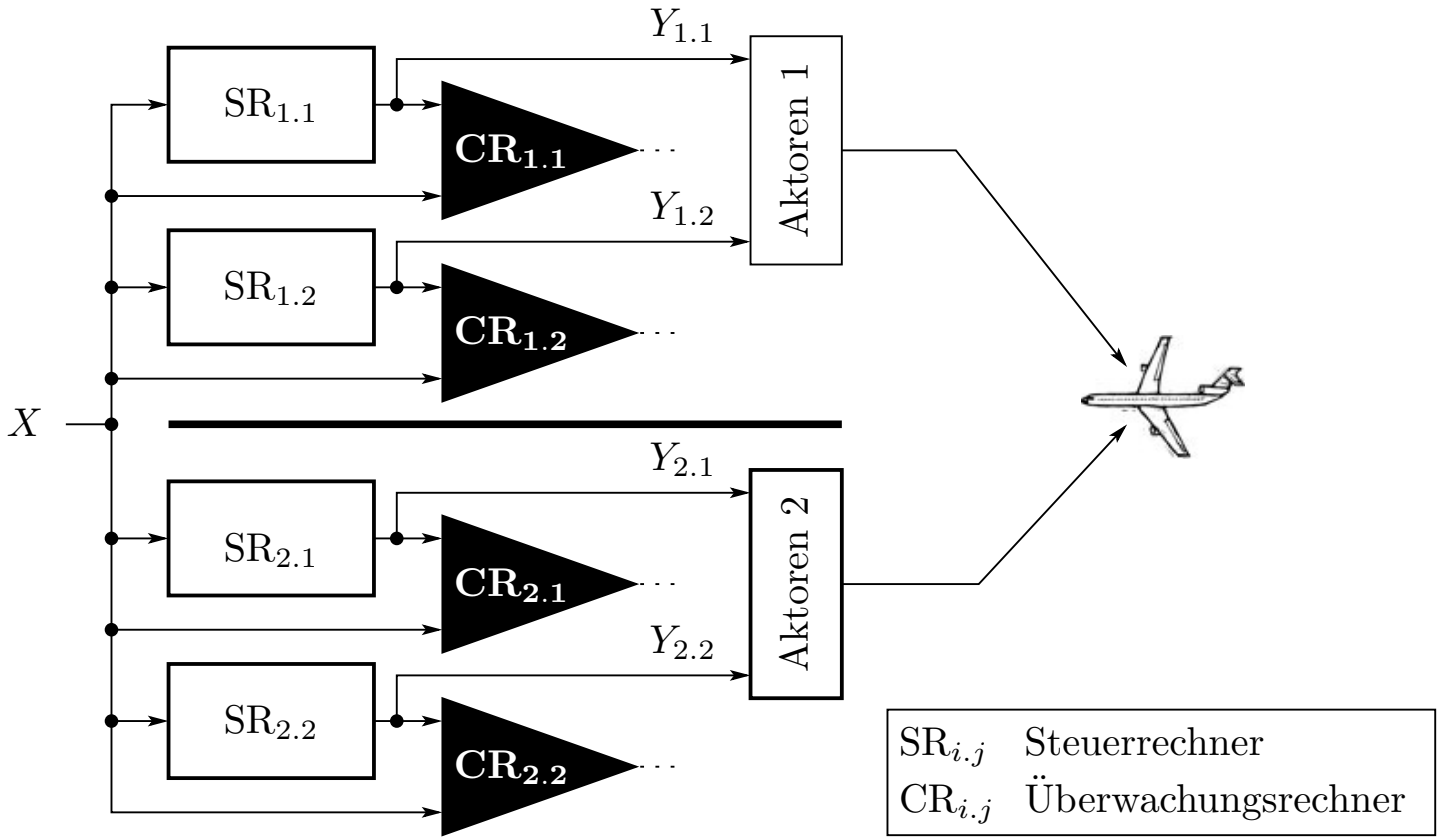
- Bei Übereinstimmung Speicherung des Bearbeitungszustands in einem geschützten Speicher.
- Bei Abweichung, Laden der letzten Sicherheitskopie und Berechnungswiederholung (Roll-Back Recovery).
- Nach Roll-Back Recovery am nächsten Kontrollpunkt wieder Mehrheitsentscheid.
- Wenn Mehrheitsergebnis, diesen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

Sequoia-System [2]:

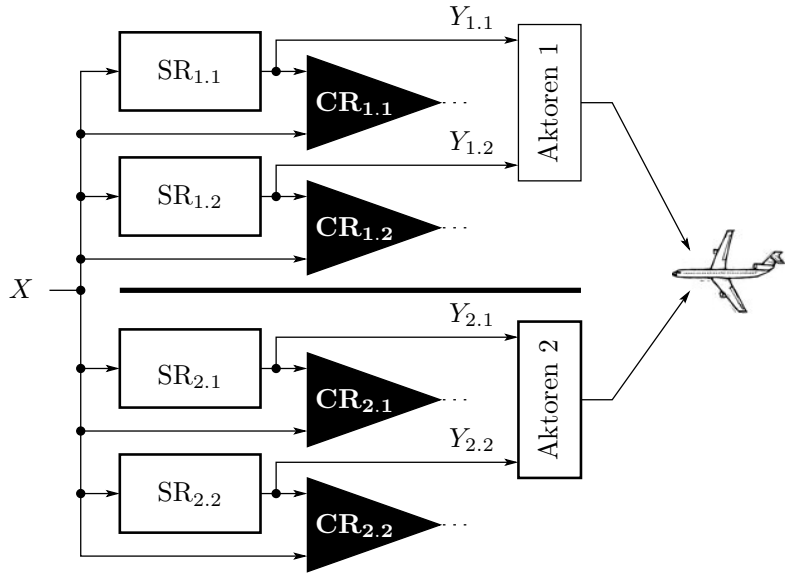
- Berechnung auf zwei Prozessoren mit eigenem Write-Back-Cache.
- Vergleich in jedem Takt.
- Zustands-Backup bei Ereignissen wie Stack-Überlauf und Prozesswechsel.
- Hauptspeicher hat die Funktion des stabilen Speichers.

Flugsteuersystem Airbus A3XX [4]

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 Komplett-, 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.

Fehlervermeidung

Fehlervermeidung

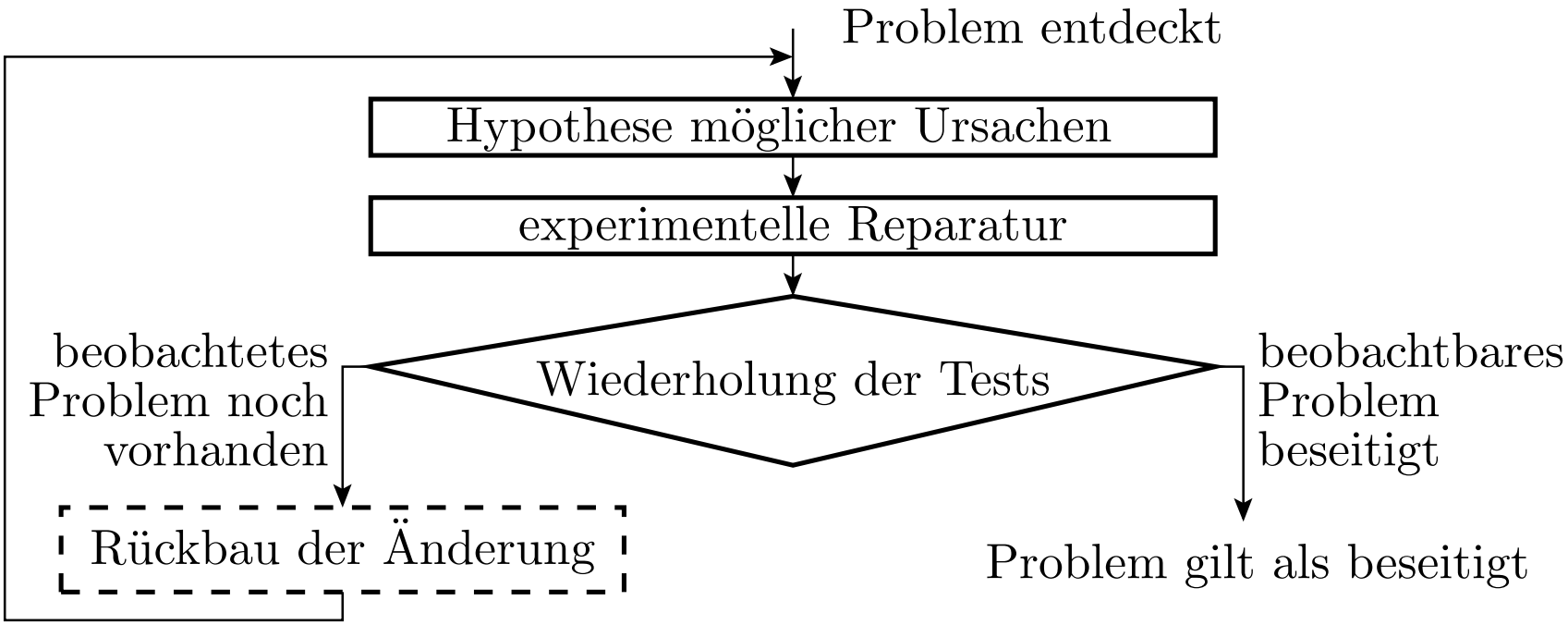
Nach TV-F1 sind Fehler die beseitigbaren Ursachen für die Fehlerentstehung. Vermeidung erfolgt durch Beseitigung der Entstehungsursachen. Zu unterscheiden sind

- Deterministische Prozesse: Erfolgskontrolle der Fehlerbeseitigung durch einmalige Wiederholung des Prozessablaufs, bei dem vorher der Fehler entstanden ist.
- Nicht deterministische Prozesse: Erfolgskontrolle durch statistische Tests. Erfordert viele Wiederholungen des Prozessablaufs, bei dem vorher der Fehler entstanden ist.
- Annähernd deterministische Prozesse: Erfolgskontrolle durch wenige Wiederholungen des Prozessablaufs.

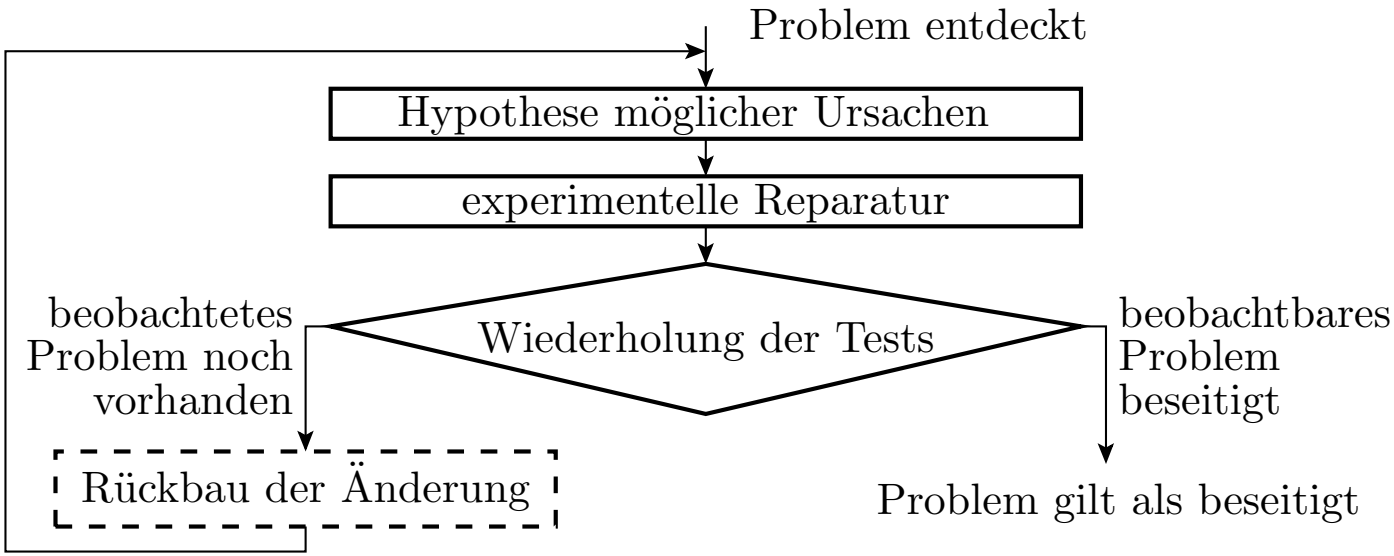
(Nicht-) Determinismus

Fehlervermeidung in deterministischen Prozessen

Fehlervermeidung in deterministischen Entstehungsprozesen 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 Prozessabarbeitung, bei der der Fehler nicht entsteht, den Reparaturerfolg 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, Projekttätigkeit, ... 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 Abschn. 1.1, Folie 45).
- 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 Fehlerbeseitungsprozess strebt deshalb der Fehleranteil durch deterministische Einflüsse mit einer kleinen Zeitkonstante (Wochen, Monate) und der Fehleranteil durch nichtdeterministische Einflüsse mit einer großen Zeitkonstante (Jahre) gegen null.

Der Technologigedanke

Technologie: Lehre von reproduzierbaren Abläufen zur Erzeugung von Produkten[26].

Technologiegedanke

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

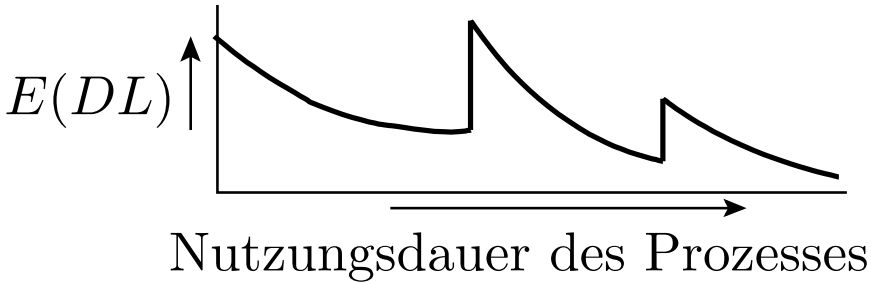
dient nicht nur zur Kostensenkungung, sondern ist auch wesentliche Grundlage für die Fehlervermeidung.

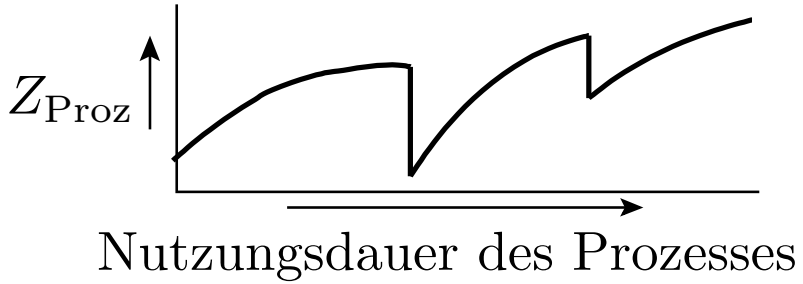
[26] Der Begriff »Technologie« wurde erstmalig von dem Göttinger Professor Johann Beckmann (1739-1811) in seinem Lehrbuch »Grundsätze der teutschen Landwirthschaft« verwendet. Heute interdisziplinäres Gebiet.

Sägezahnwachstum der Prozesszuverlässigkeit

Technologische Verbesserungen (neue Maschienen, Programme, Technologien, ...) erfolgen in größeren zeitlichen Schritte (Monate, Jahre) und haben das Potential, den Fehleranteil zu verringern.

- Bei jeder technologischen Umstellung entstehen Fehler, die den Fehleranteil sprunghaft erhöhen und in einem schnelleren Reifeprozess beseitigt werden.
- Der potentiell geringere Fehleranteil wird erst nach einer gewissen Nutzungsdauer erreicht.
- Sägzahnförmige Abnahme des zu erwartenden Fehleranteil.
- Die Prozesszuverlässigkeit als Kehrwert nimmt tendentiell zu.





Eine Schattenseite von Innovationen

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

- Verbesserte Wiederholgenauigkeit der Abläufe,
- verbesserte vorhersagbare Material- und Produkteigenschaften,
- weniger entstehende Fehler, Ausbeute ↑, Kosten ↓, ...

Schattenseite:

- Neuerungen führen fasts zwangsläufig zu »neuen Kinderkrankheiten«, die erst nach einer gewisssen Reifezeit beseitigt sind.
- Mehr entstehende Fehler bedeutet nicht nur schlechtere Ausbeute und mehr Kosten, sondern auch auch mehr Fehler in eingesetzten Systemen, mehr Frühausfälle, ...

Linux unterscheidet z.B. in seiner Versionsverwaltung:

- »Innovative« Beta-Versionen mit vielen Kinderkrankheiten, ...
- Einsatztaugliche (zuverlässige) Stable-Versionen.

Projekte, Vorgehensmodelle

Projektarbeit als Entstehungsprozess

Deterministische Prozesse reifen durch experimentelle Reparatur.
Massenfertigung reift durch technologische Weiterentwicklung, kontrollierbar durch statistische Tests (viele Kontrollen).

Wie verhält es sich mit Projekten:

- Manuelle kreative Teile der Entwurfsprozesse[27] und
- 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?

[27] Hier inbesondere der Software- und Hardware-Entwurf.

Vorgehensmodelle

Vereinheitlichung des Vorgehens für große Klassen von Projekten

- zur Aufwandsminimierung, besseren Vorhersagbarkeit und
- zur Fehlervermeidung durch »Lernen aus Fehlern«.

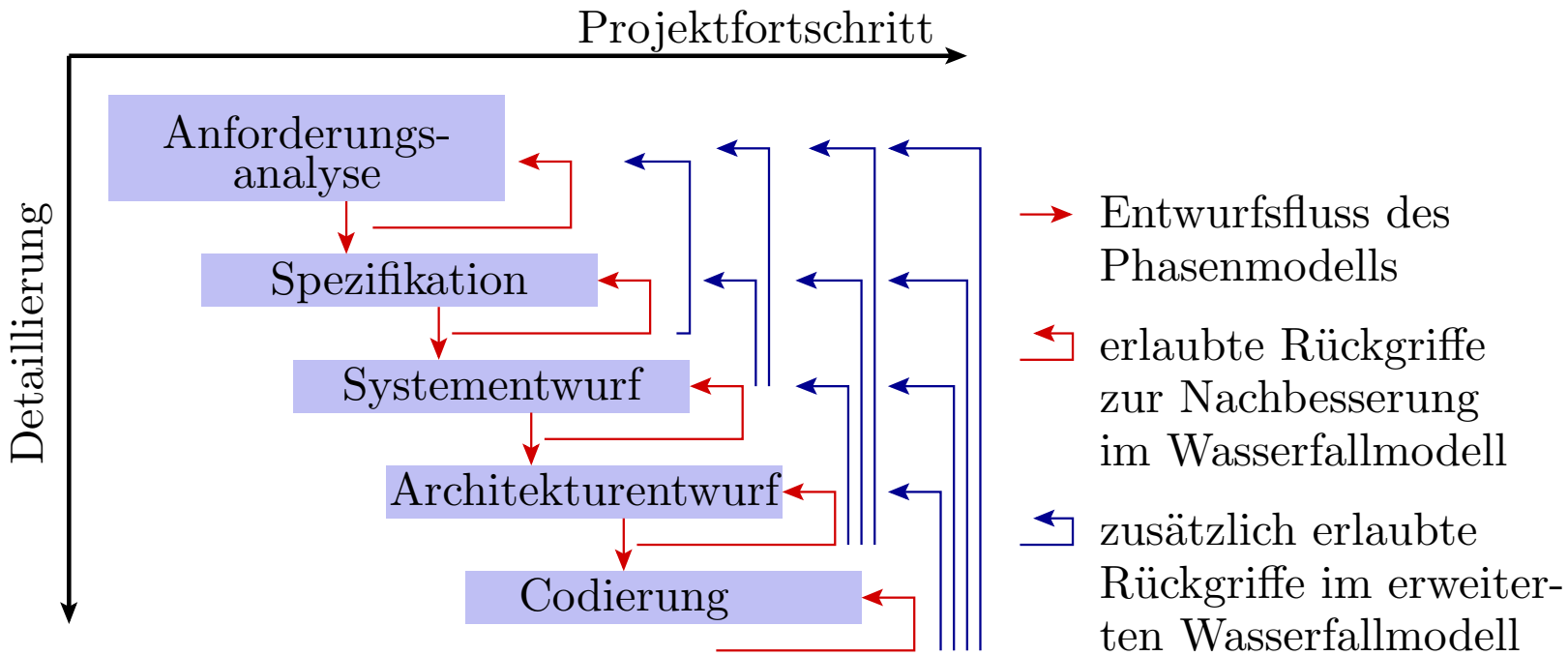
Typische Vorgehensmodelle für den Entwurf und die Fertigung von IT-Komponenten umfassen:

- 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:

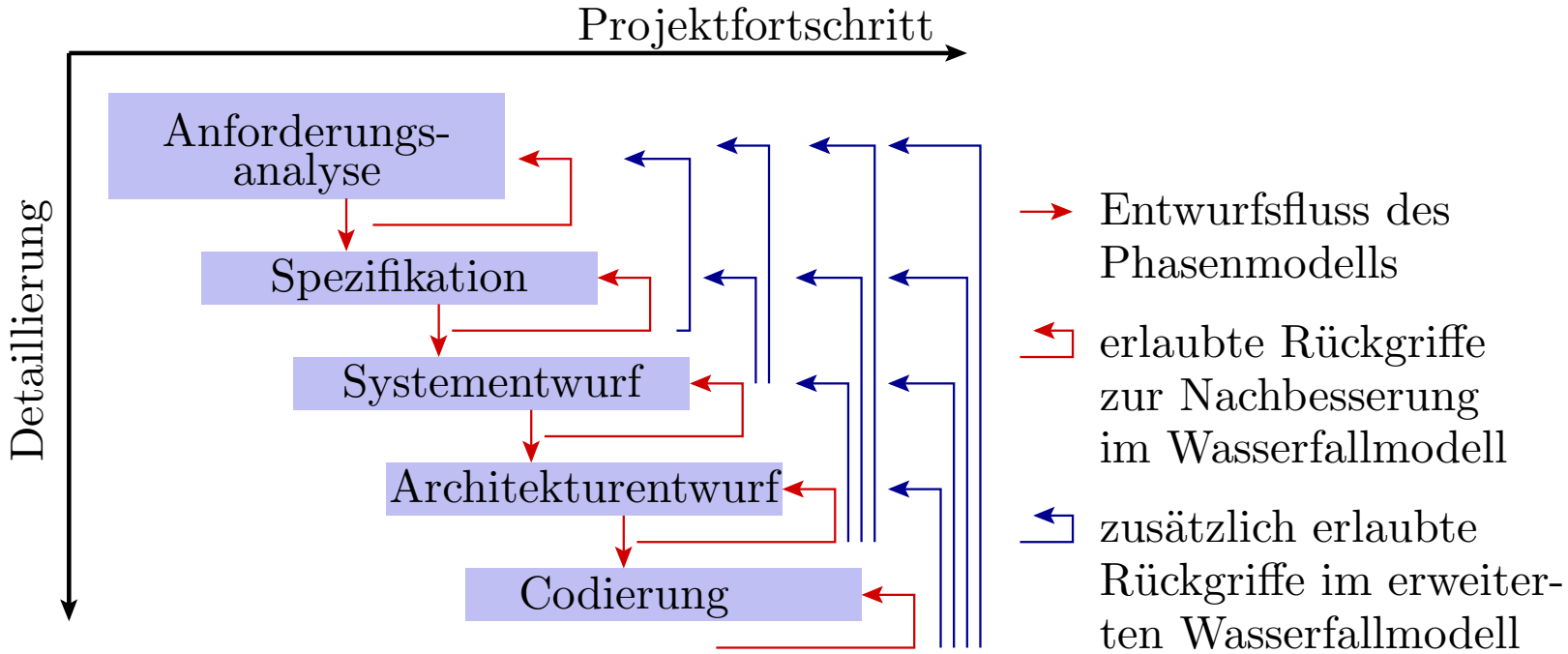
- Anforderungsanalyse,
- Spezifikation der Ziele,
- Architekturentwurf, Codierung, Test, ...

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:

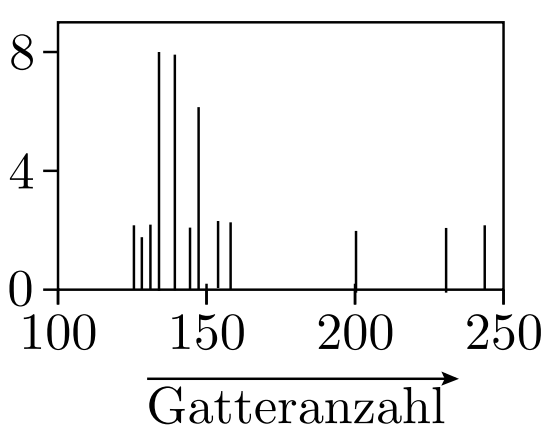
- Projektdauer, Projektkosten,
- Arbeitsschritte je entstehender Fehler, Umfrageergebnisse, ...

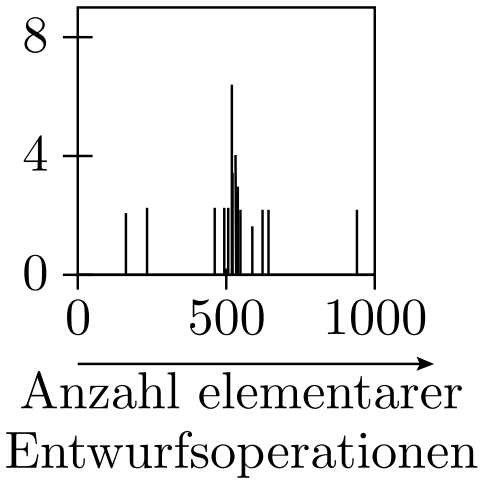
jeweils Erwartungswerte und Vorhersagbarkeit skalierbar auf die Projektgröße, Schwierigkeit, ...

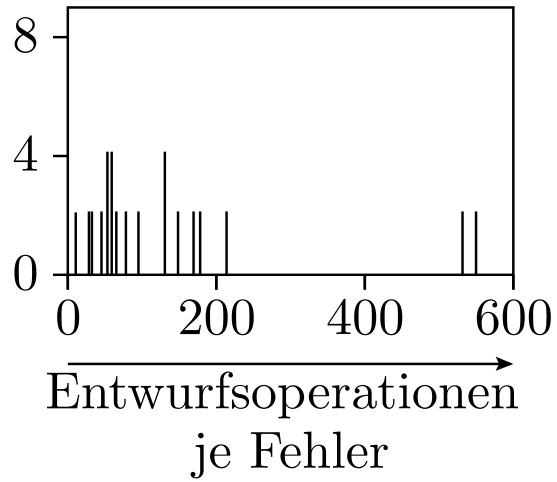
Signifikante Aussagen über Vorgehensmodelle verlangen die Beobachtung tausender Projekte mit vergleichbarem Vorgehen.

Ein Experiment [1]

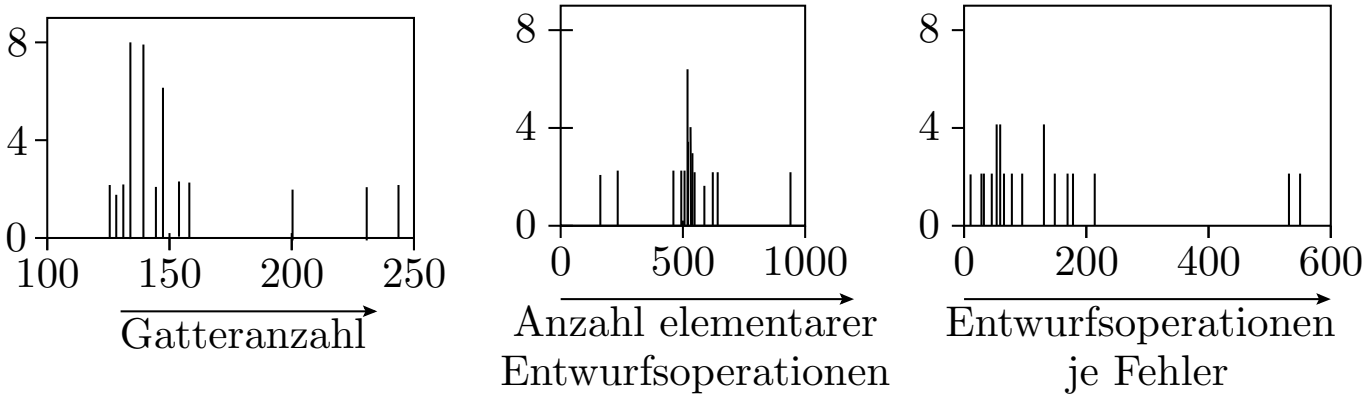
Eine Gruppe von 72 Studenten hatte die Aufgabe, aus der Beschreibung eines PLAs (**P**rogrammable **L**ogic **A**rrays) eine Gatterschaltung zu entwickeln und diese über die GUI eines CAD-Systems in den Rechner einzugeben. Für jeden Entwurf wurden die elementaren Entwurfsoperationen, die Gatteranzahl 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 Vergleichergenissen 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.

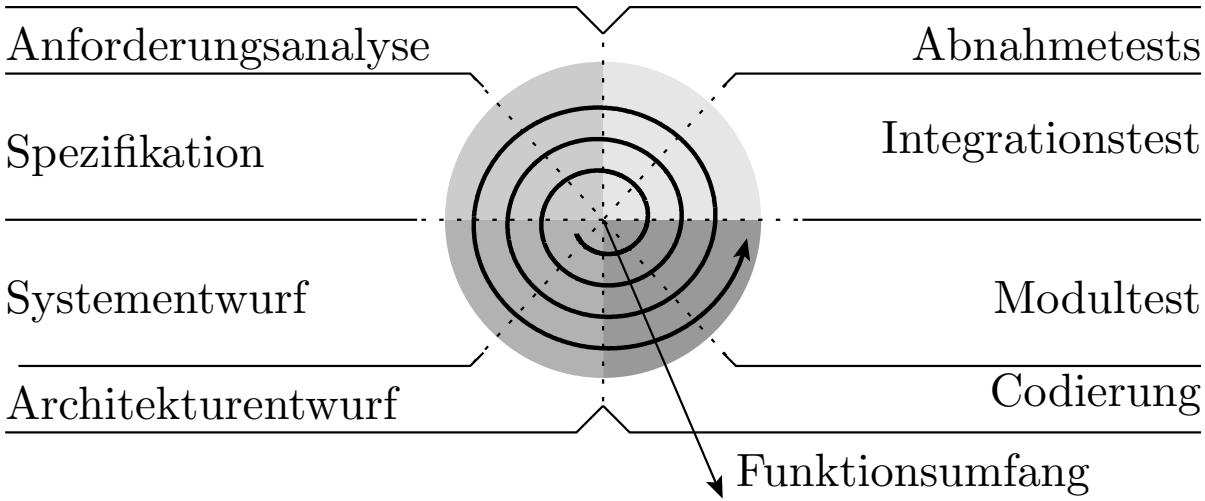
Schlussfolgerung

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

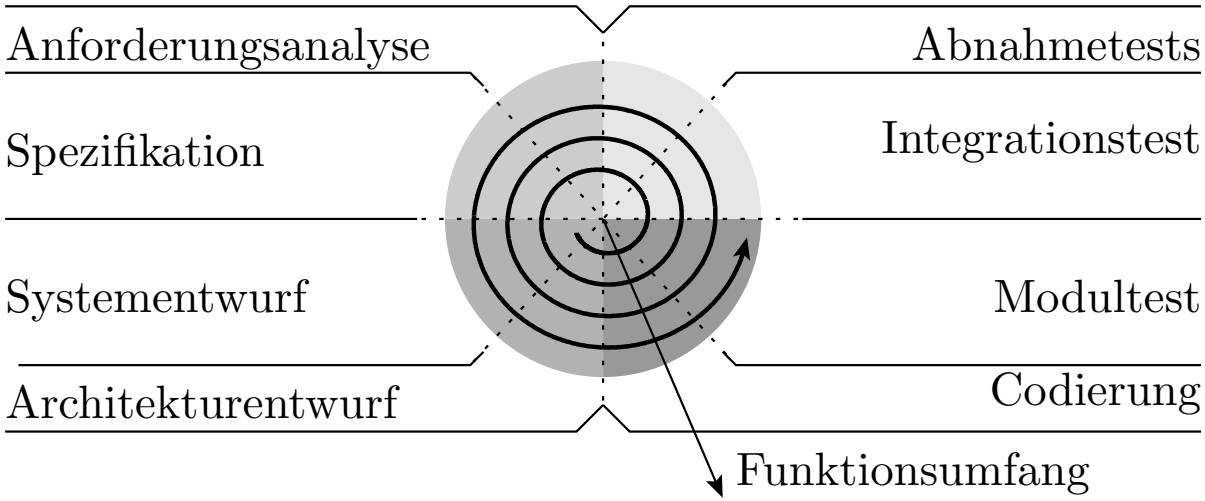
IT-Entwurf verlangt Qualität und Kreativität.

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 Lernprozesses unterliegen Vorgehensmodellen. ...
Der Bolonia-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 Lernrozessen

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[28], Praktika.
- Phase 3: Aus Zeitgründen erst in der Berufspraxis für den eigenen eingeschränkten Tätigkeitsbereich.

[28] 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.
- Deshalb rechnet es sich für die Industrie nicht, Hochschulen und Studenten in ihr Tagesgeschäft einzubinden.
- Industrielle Studenten-Projekte dienen der Ausbildung.
- Drittmittelforschung ist wertvoll für den Knowhow-Transfer, Literaturstudien, Demostratoren, ... aber im IT-Bereich ungeeignet für die Einbindung in die Produktentwicklung.

Literatur

6. 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] P.A. Bernstein.
Sequoia: a fault-tolerant tightly coupled multiprocessor for transaction processing.
Computer, 21(2):37–45, 1988.

[3] 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.

[4] Pascal Traverse.
Dependability of digital computers on board airplanes.
Dependable Computing for critical applications, 4:134–152, 1991.