Test und Verlässlichkeit (F6)
Foliensatz 6:
Problembeseitigung

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV-F6)

11. Juli 2016

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

Wartung
3.1 Ausfälle
3.2 Frühausfälle und Voralterung
3.3 Kalte, warme und heiße Reserve

Fehlerbehandlung
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 Problembeseitigung oder Vermeidung 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 Fehlerenstehung

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

Hilfreich ist Determinismus.

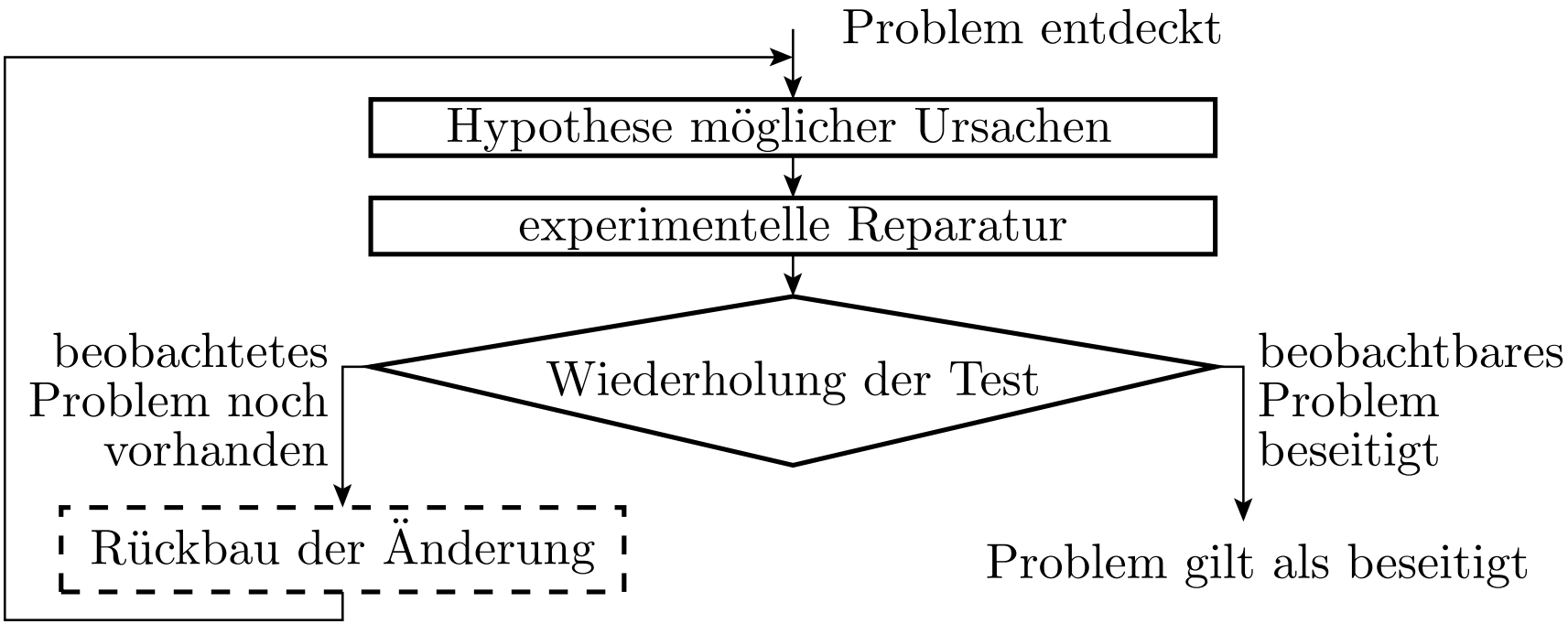
Fakt 1

Ein Service arbeitet deterministisch, wenn er immer gleich ausgeführt wird und für gleiche Eingaben (und gespeicherte Zustände) gleiche Ergebnisse liefert (F1, Abschn. 1.1).

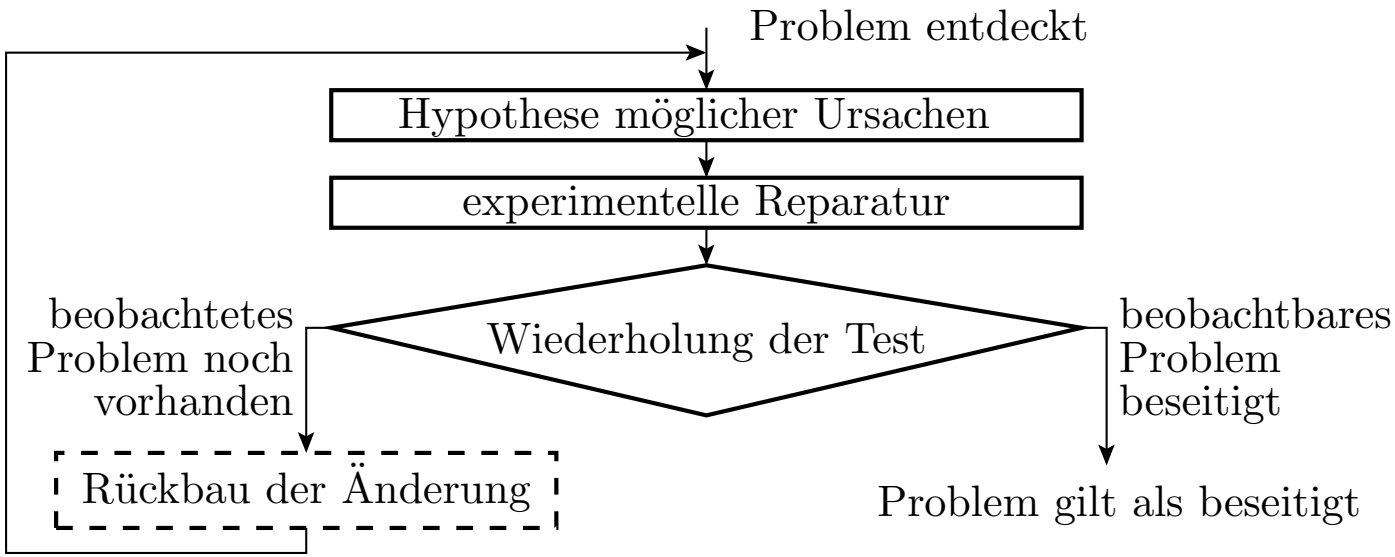
Deterministische Prozesse

Fehlervermeidung in deterministischen Prozessen

In deterministischen Entstehungsprozesen erfolgt die Fehlervermeidung 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 elektroni- sche 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 denselben Ergebnissen führen.
- Bei manueller kreativer Arbeit, Projekttätigkeit, ... unterscheiden sich die Ergebnisse bei gleicher Zielstellung erheblich. Nicht annähernd deterministische Prozesse.

Fehleranteil und Prozessnutzung

- Die Problembeseitigung in einem deterministischen Entstehungsprozess gehorcht ähnlichen statistischen Gesetzen wie die Fehlerbeseitigung in einem deterministischen Service (siehe später Abschn. 2).
- 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.
- Der Fehleranteil in einem annähernd deterministischen Entstehungsprozess strebt mit einer kleineren Zeitkonstante (Wochen, Monate) gegen den Anteil der durch zufällige Einflüsse verursachten Fehler und einer viel größeren Zeitkonstante (Jahre) gegen null.

Der Technologigedanke

Technologie: Lehre von reproduzierbaren Abläufen zur Erzeugung von Produkten[1].

Fakt 2

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.

[1] Der Begriff »Technologie« wurde erstmalig von dem Göttinger Professor Johann Beckmann (1739-1811) in seinem Lehrbuch »Grundsätze der teutschen Landwirthschaft« 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 den Ursachen für die abweichenden Erwartungswerte und Varianzen (Einflüsse Material, Personal, Prozessbedingungen, ...).
- Wahl der günstigsten Prozessbedingungen und Vereinheitlichung.
- Kontrolle der Beseitigung der Multimodalität und Varianzverringerung.

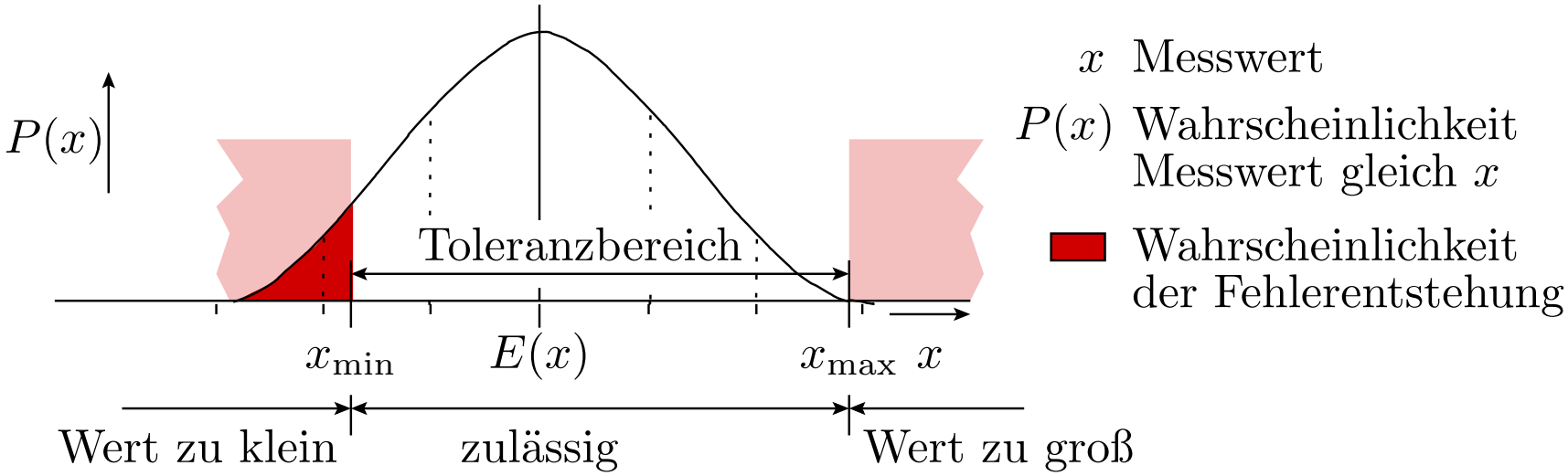
Aufwand für die Erfolgskontrolle:

- Erzeugung und Messung der betrachteten Eigenschaften für tausende Produkte[2].
- Prozessverbesserung verlangt sehr großen Stückzahlen.

[2] 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:

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

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

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

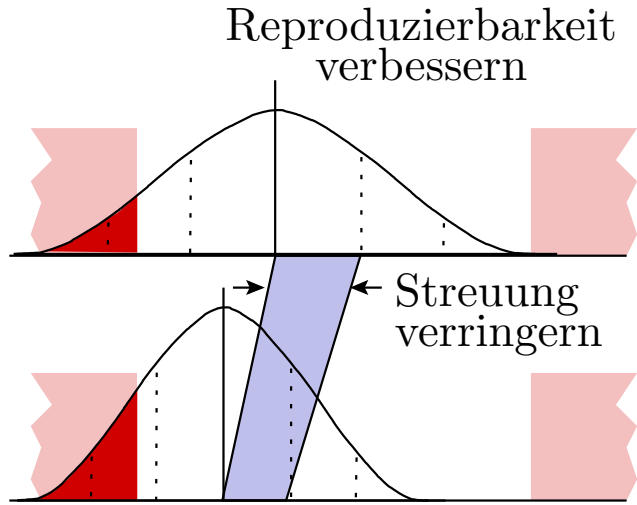
$$E(x) = \frac{x_{\min} + x_{\max}}{2}$$

Fakt 3

Prozesszentrierung bedeutet, die Erartungswerte messbarer Parameter durch geeignete Veränderung von Enflussgrößen in die Mitte der Gaus-Glocke zu schieben.

Der zu erwartende Widerstandswert eines Widerstands kann z.B. durch eine geringere Schichtdicke und die zu erwartende Schichtdicke durch kürzere Beschichtungszeiten vermindert werden.

Iteration aus Prozesszentrierung und -verbesserung

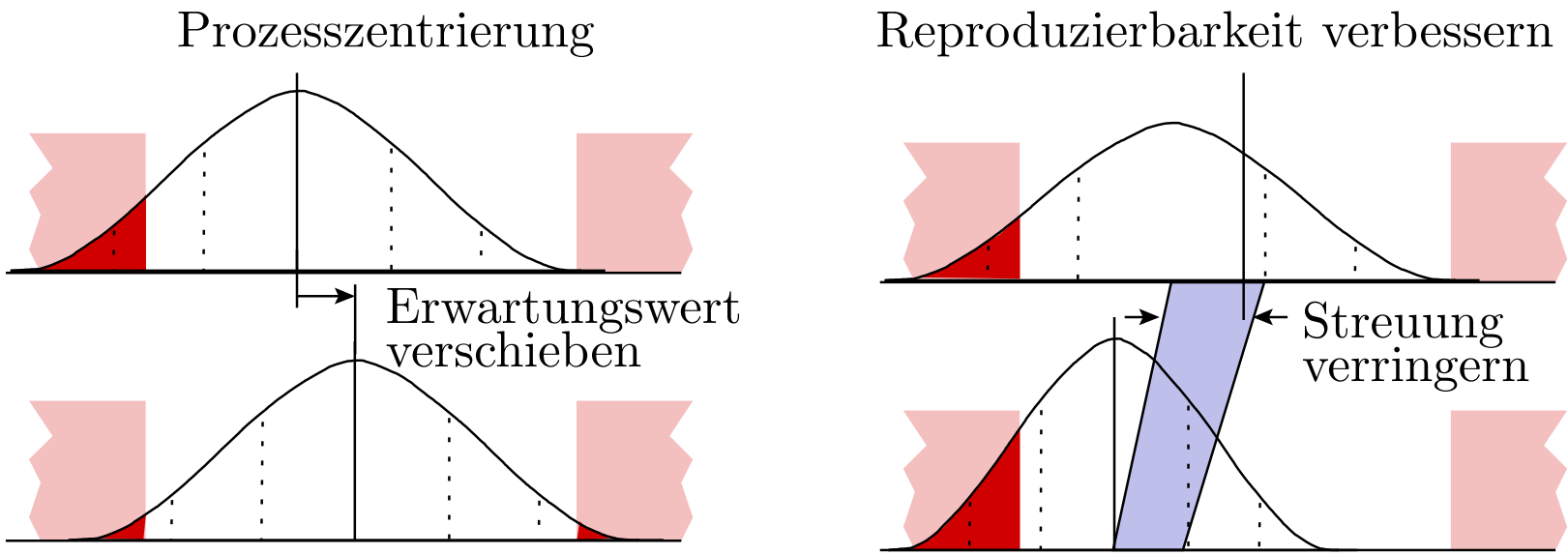


Bei einem zentrierten Prozess lässt sich der zu erwartende Fehleranteil nur noch durch eine Verringerung der Varianz absenken. Das erfordert größere Eingriffe in den Prozess:

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

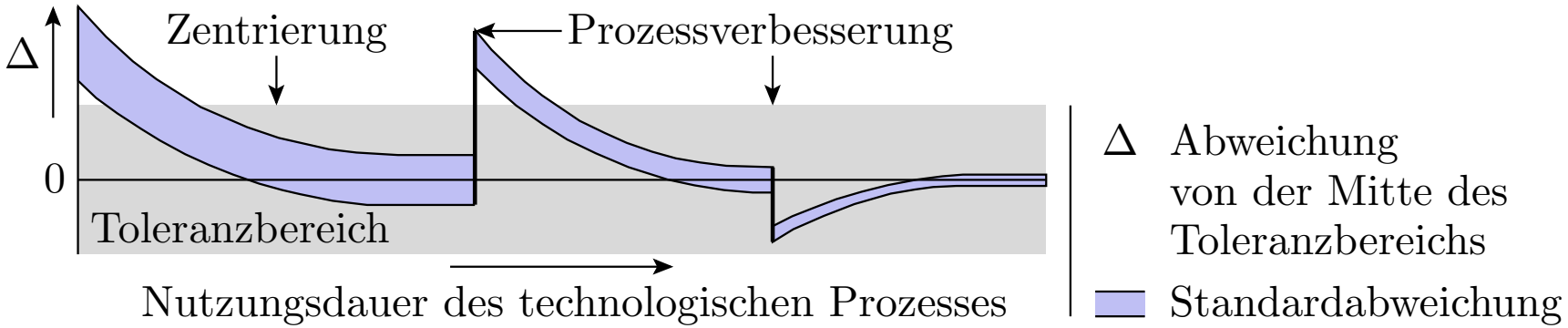
und erfolgt in größeren Zeitabständen.

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.

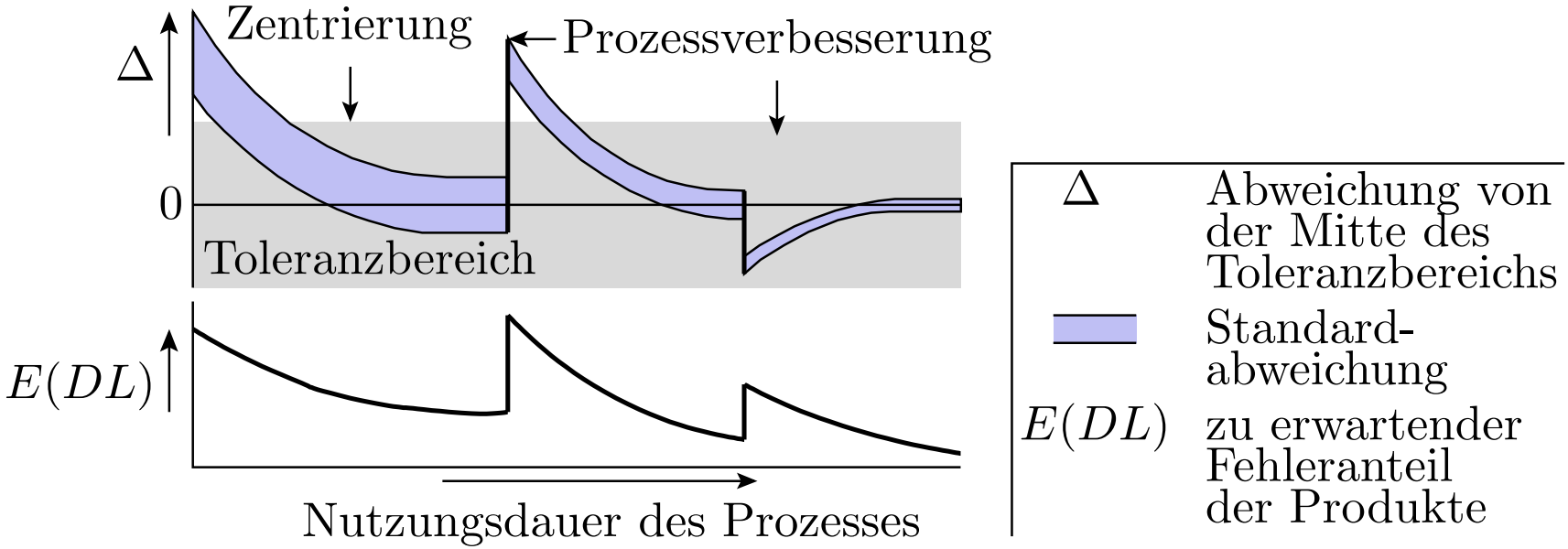


Technologien entwickeln sich in Phasen:

- 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.

Fakt 4

Innovationen verringern (temporär) die Zuverlässigkeit.

Projekte, Vorgehensmodelle

Projekarbeit als Entstehungsprozess

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

- den manuellen kreativen Teile der Entwurfsprozesse[3] und
- der Fertigung von Prototypen, Demonstratoren, ... ?

Fakt 5

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?

[3] Hier inbesondere 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, ...

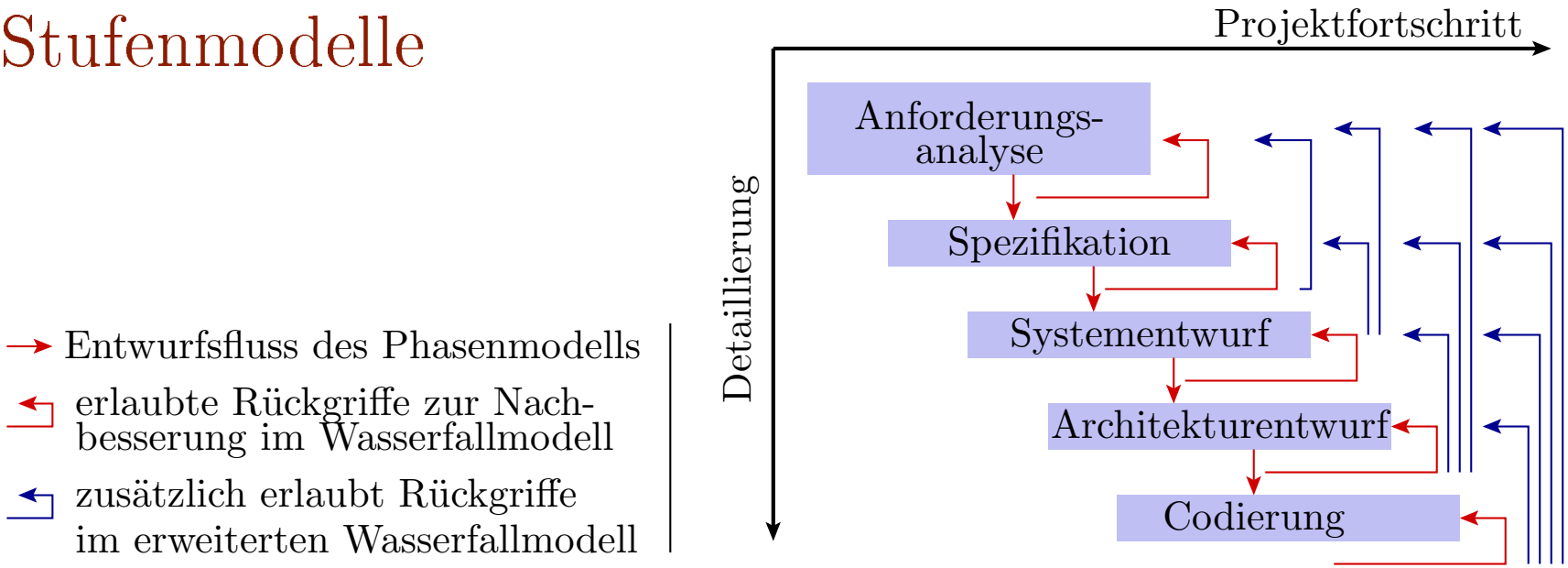
Typische Vorgehensmodelle für den Entwurf und die Fertigung von IT-Komponenten umfassen:

- 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, ...

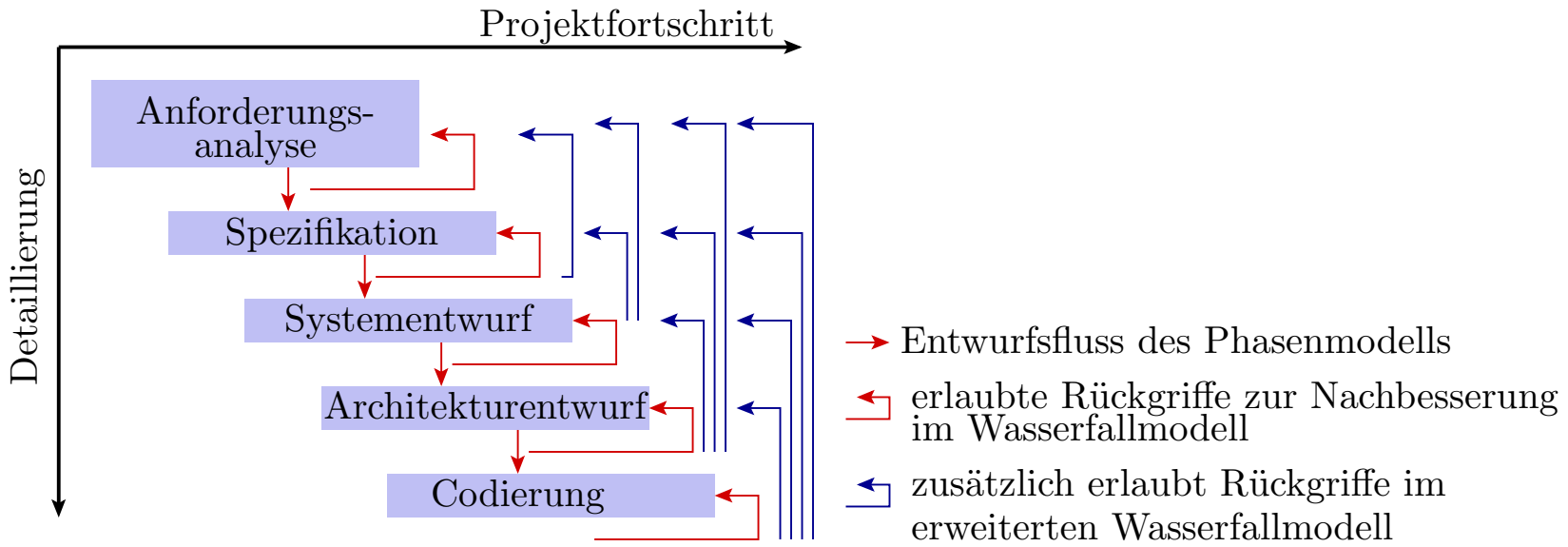
Stufenmodelle



Stufenmodelle variieren:

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

Stufenmodelle verbessern die Vorhersagbarkeit ..., ermöglichen Lernen aus Fehlern und beschränken die Kreativität.



Gestaltbare Parameter, die Kosten, Qualität, ... beeinflussen:

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

Fakt 6

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

Bewertung von Vorgehensmodellen

Jede Art der Fehlervermeidung benötigt eine Erfolgskontrolle:

Folgerung 7

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

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

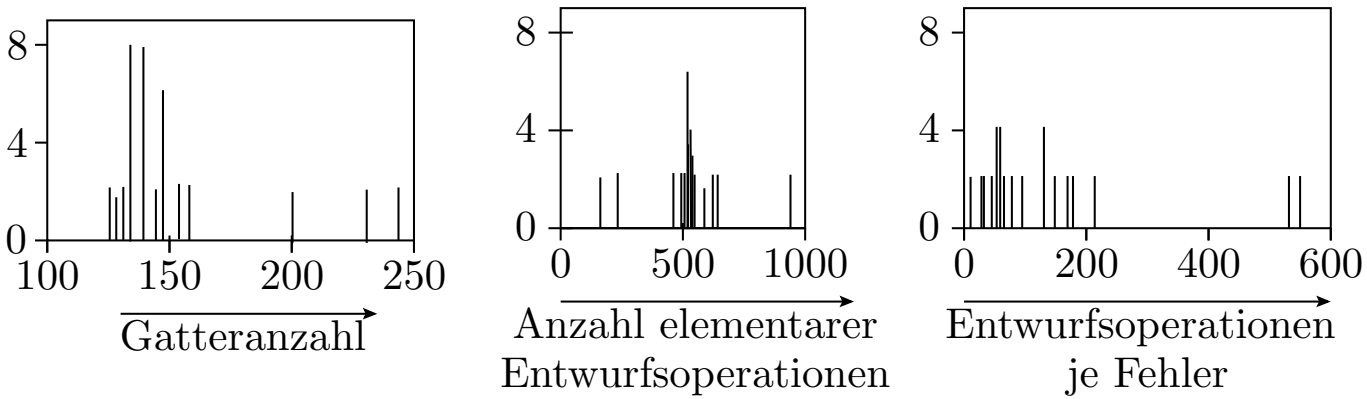
- Projektdauer, Projektkosten,
- Arbeitsschritte je entstehender Fehler, Umfrageergebnisse ...

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

Erwartungswerte, Vorhersagbarkeiten, ... verlangen statistisch signifikannte Stichproben. Vergleiche mit signifikannter Aussage verlangen die Beobachtung vieler Projekte über lange Zeiträume.

Ein Experiment [1]

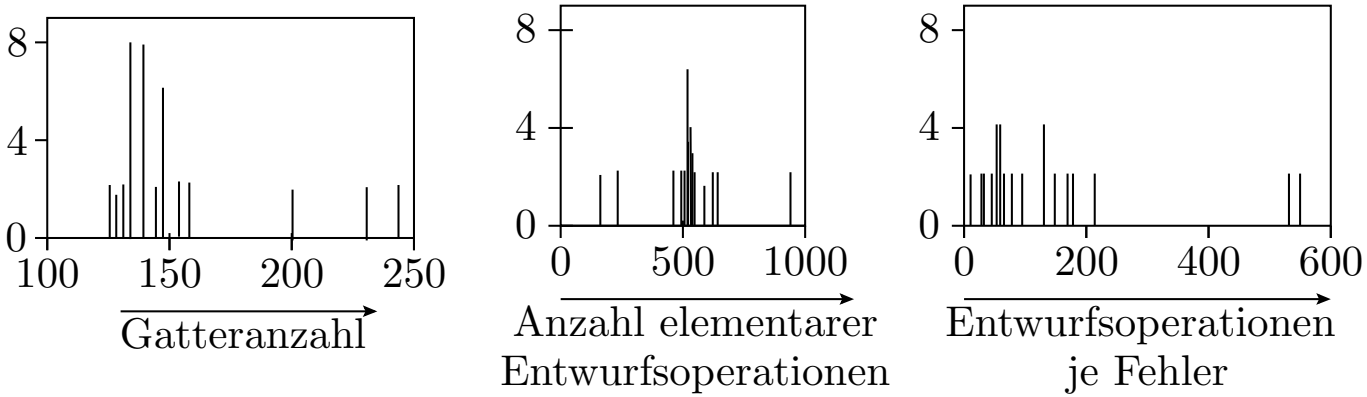
Eine Gruppe von 72 Studenten hatte die Aufgabe, aus der Beschreibung eines PLAs[4] eine Gatterschaltung zu entwickeln und diese über die grafische Benutzeroberfläche eines CAD-Systems in den Rechner einzugeben. Für jeden Entwurf wurden die elementaren Entwurfsoperationen[5], die Gatteranzahl und die Entwurfsfehler gezählt.



[4] PLA: Programmable Logic Array.

[5] 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 Vergleichergenisse ließe sich schlussfolgern, ob und an welcher Hochschule die Studierenden ein besserer Vorgehen für diese Aufgabe erlernt haben.

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«:

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

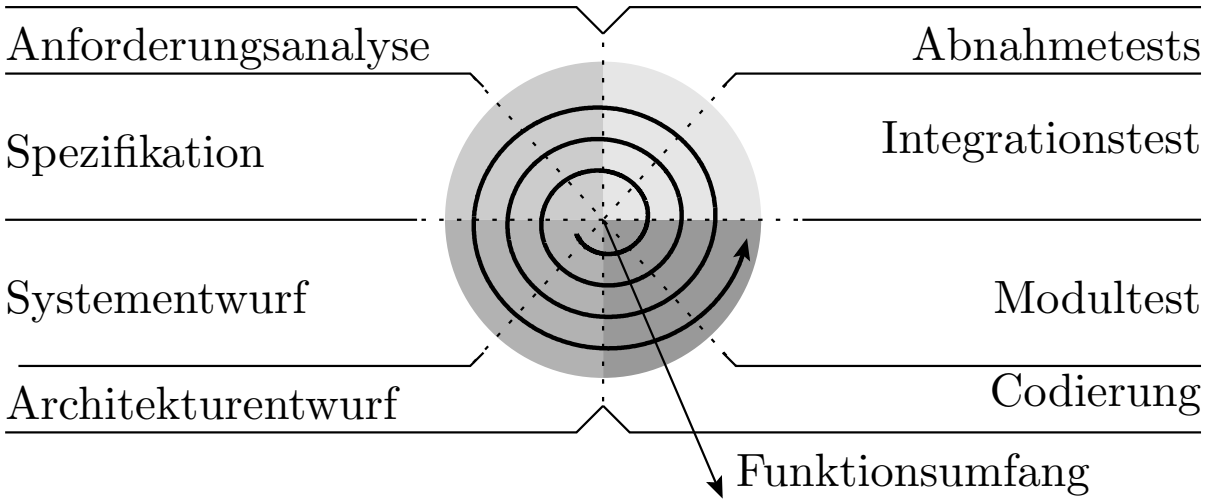
Folgerung 8

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

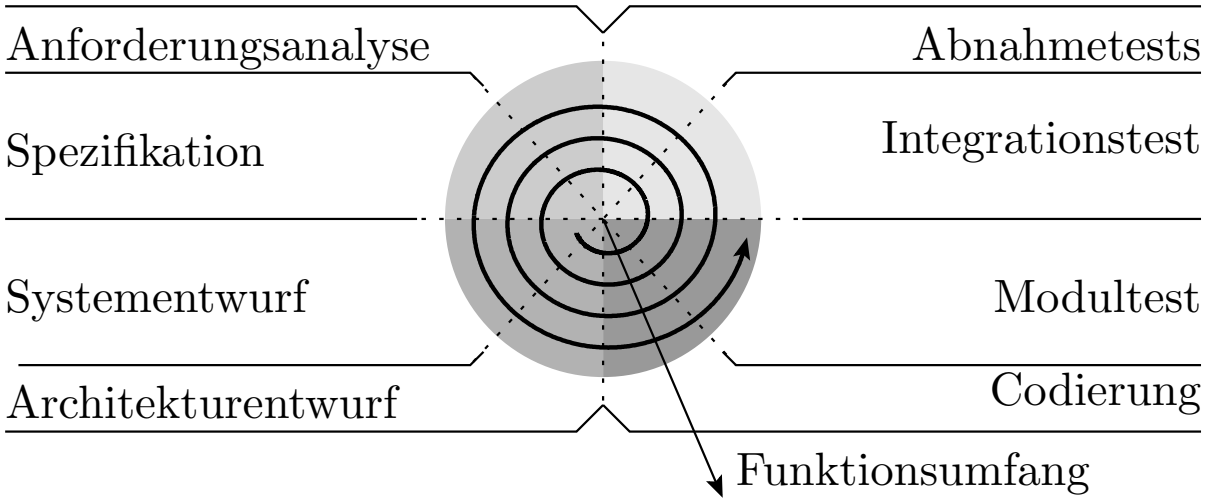
IT-Entwurf verlangt Qualität und Kreativität.

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

[6] Auch Bewertung in Arbeitsmenge pro Zeit und Fehler pro Arbeitsmenge.

Querverbindung Drittmittelprojekte

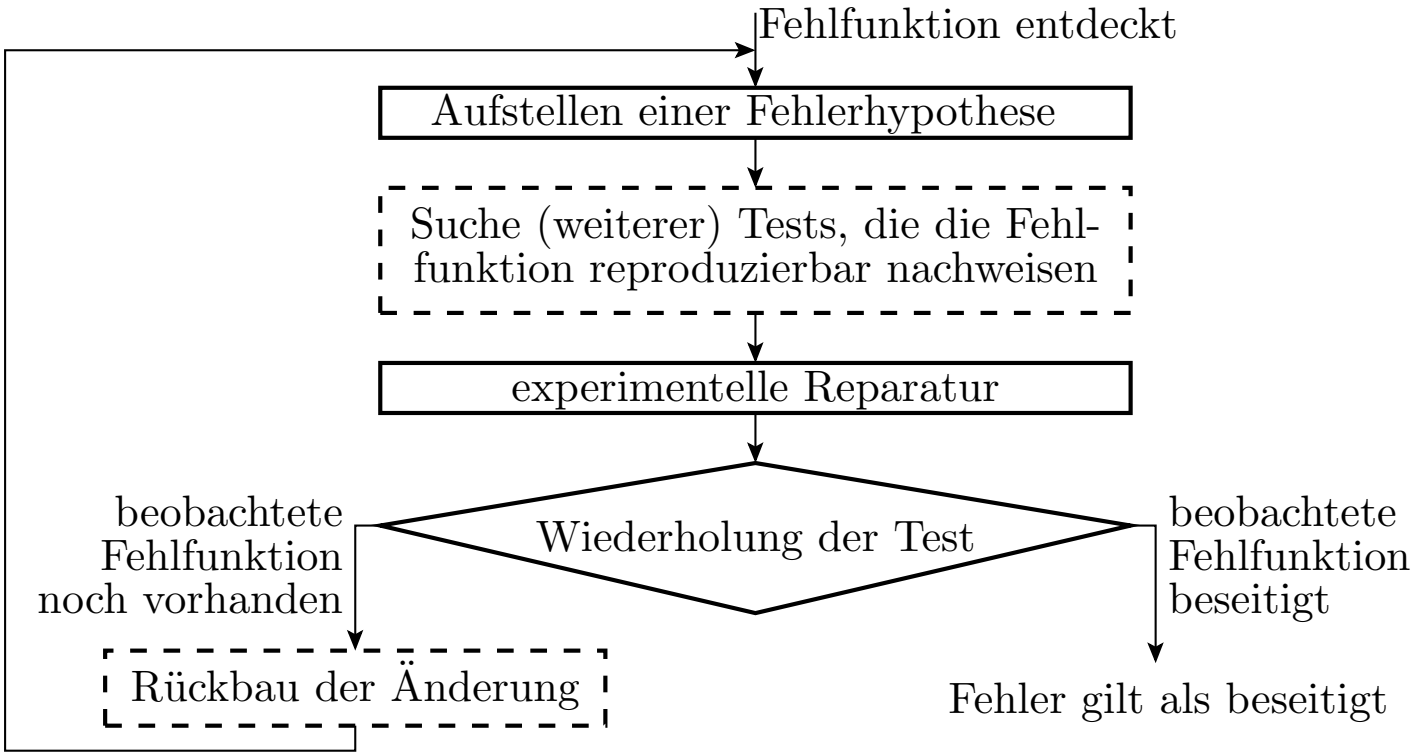
- Die Professionalisierungsphase durchlaufen erst die Absolventen in der Praxis.
- Akademiker und Studenten sind noch nicht für »fehlerarme Arbeitsabläufe« qualifiziert.
- In industriellen Software-Projekten entstehen durch Akademiker tendenziell mehr Fehler je Aufgabengröße.
- Die Kosten für die Fehlerbeseitigung 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, aber ineffektiv für den unmittelbaren Wertschöpfungsprozess.

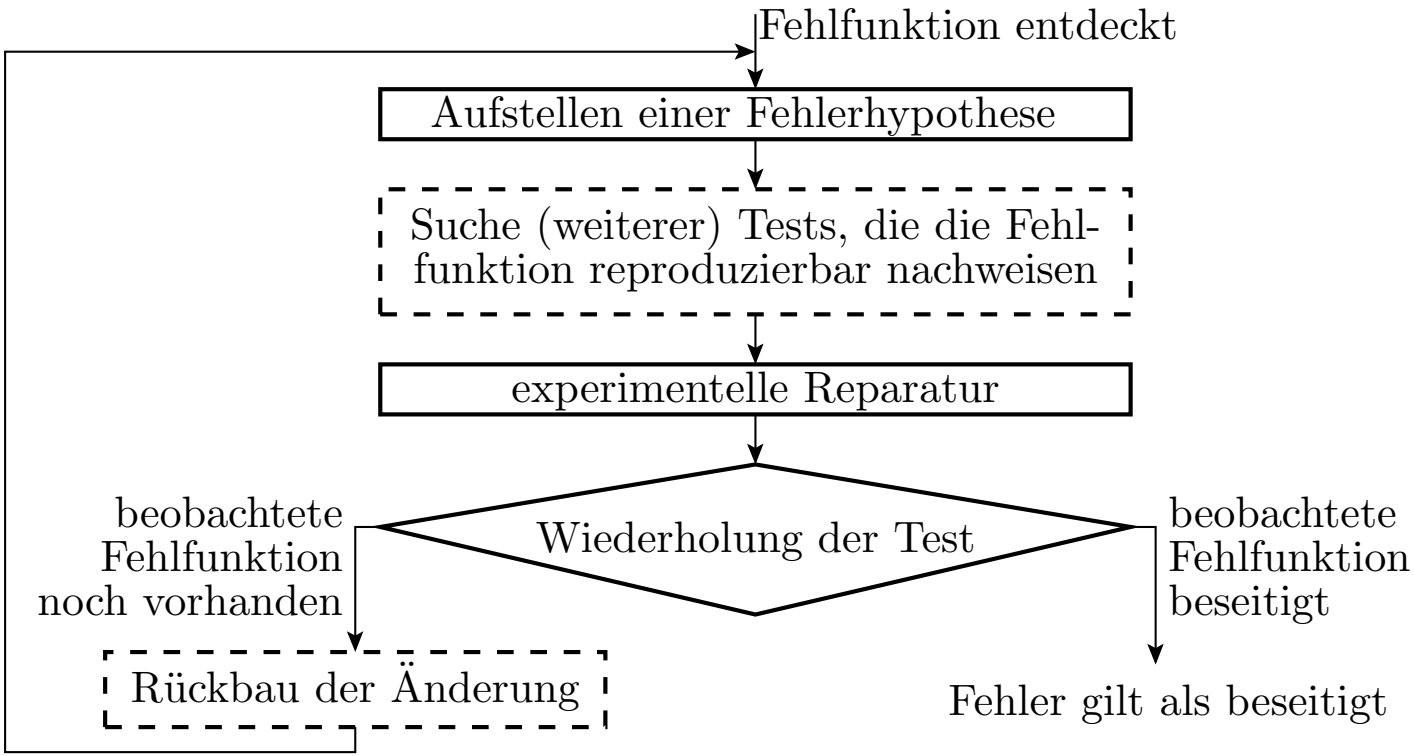
Fehlerbeseitigung

Experimentelle Reparatur

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

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

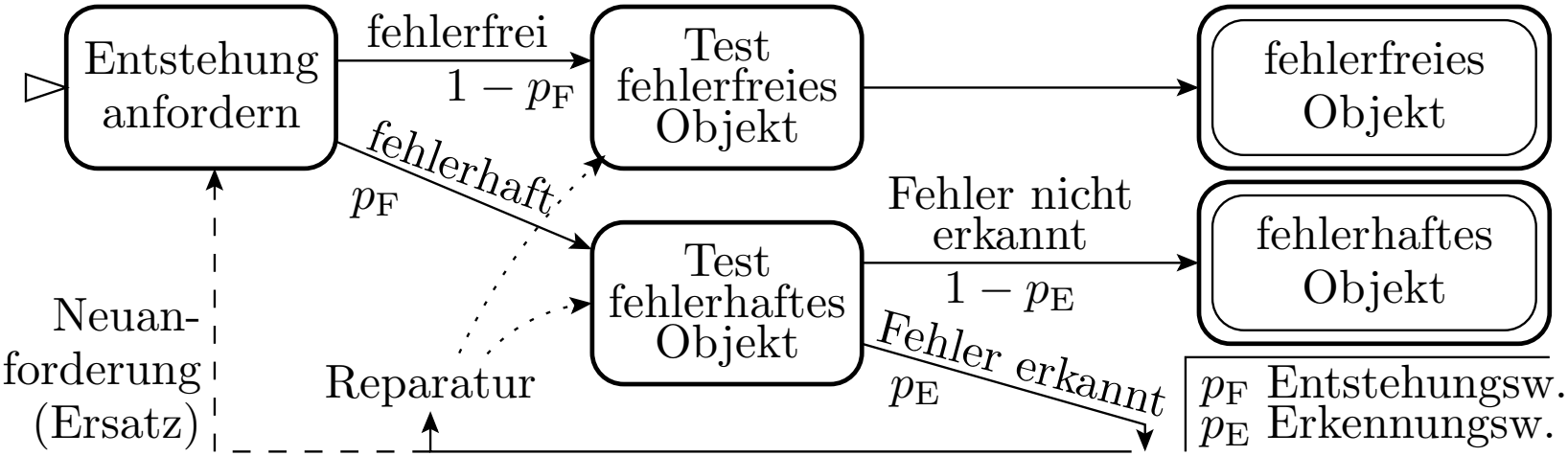




Die Erweiterung um »Suche (weiterer) Tests« ist erforderlich, wenn die bisherigen Tests das beobachtete Fehlverhalten nicht reproduzierbar nachweisen. Insbesondere bei Einbeziehung der Anwender in die Fehlerbeseitigungsiteration.

Rückbau dient zur Beseitigung der durch erfolglose Reparaturversuche entstehenden Fehler.

Experimentelle Reparatur als Markov-Prozess[7]



- 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.

[7] 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

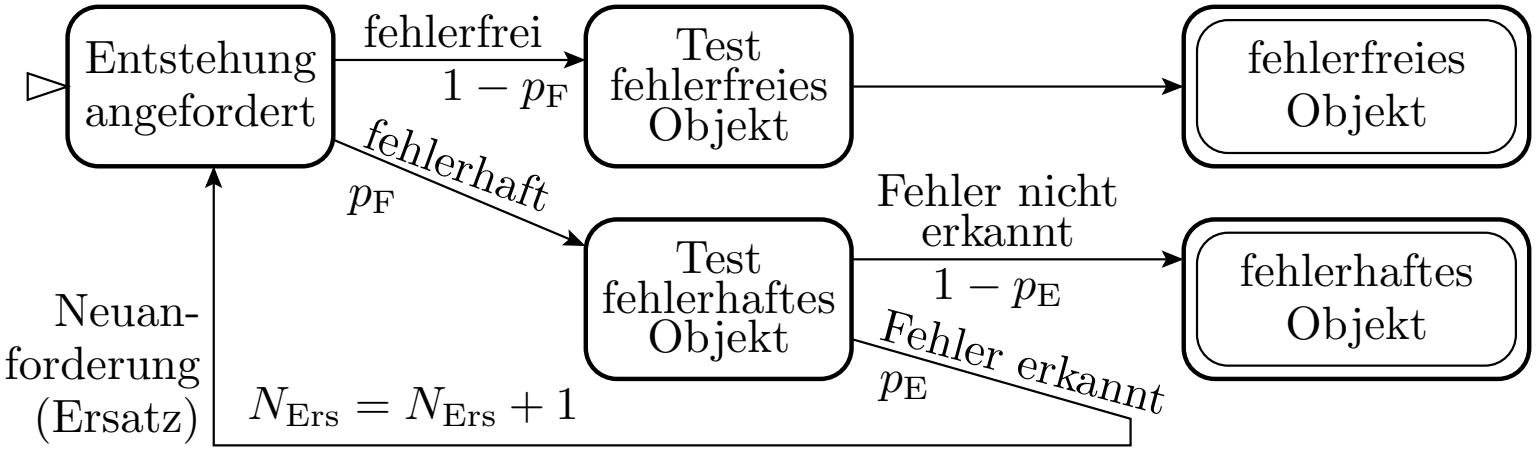
- haben Orginal- und Ersatzsystem dieselbe zu erwartende Ausbeute $E(Y)$,
- müssen im Mittel $\frac{1}{E(Y)}$ mal so viele Systeme gefertigt oder entworfen, wie am Ende eingesetzt werden.

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

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

Ersatz als Iteration



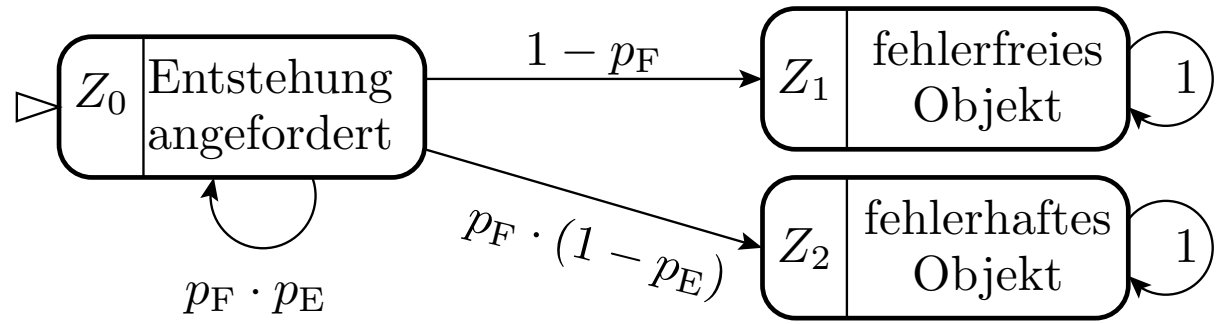
- Ersatzobjekte haben auch mit Wahrscheinlichkeit p_F Fehler.
- Diese entstehen unabhängig und sind unabhängig nachweisbar.

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[9].

[9] Phantomfehler (vergl. F1, Abschn. 3.5) sind hier vernachlässigt.

Modellierung als Markov-Prozess



Nach Ersatz aller erkennbar defekten Objekte[10] :

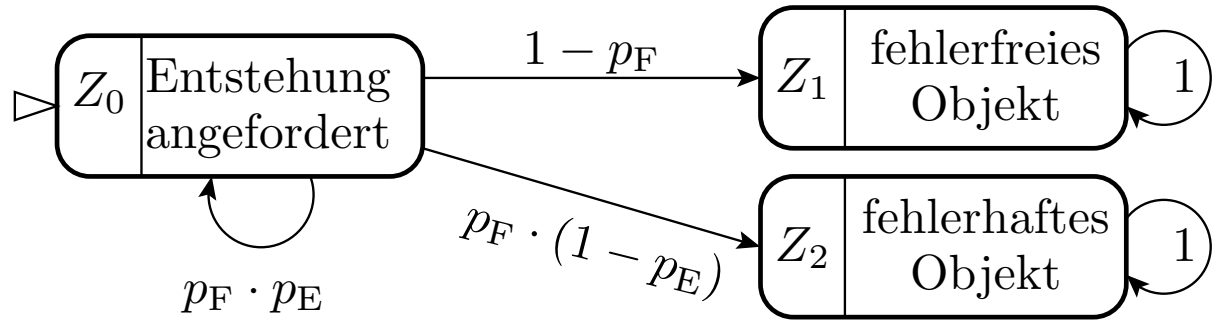
$$\lim_{n\to\infty}(p_{Z0}) = 0$$

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

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

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

Abschätzbare Kenngrößen



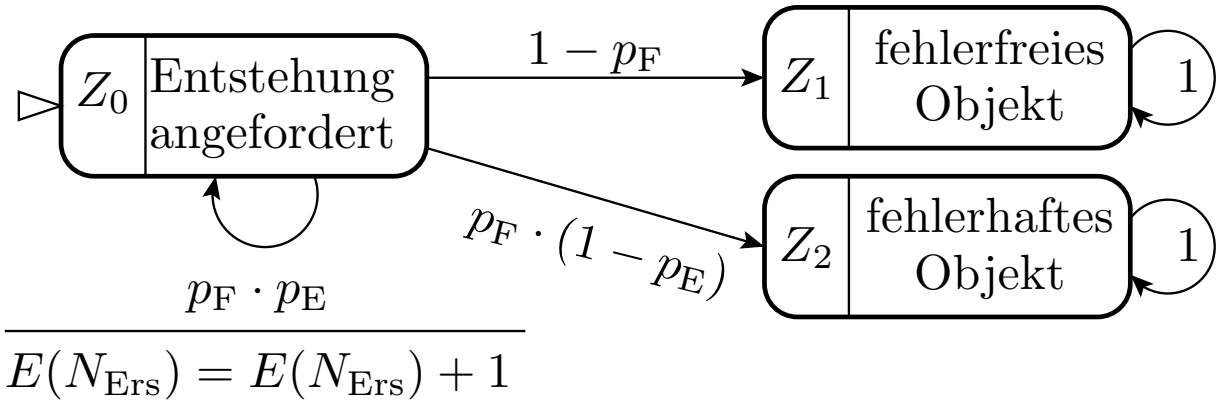
Zu erwartender Fehleranteil nach Ersatz defekter Objekte:

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

Wahrscheinlichkeit, dass Fehler nicht beseitigt werden[11]:

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

[11] Verhältnis der zu erwartenden Fehleranzahl ohne und mit Ersatz aller erkennbar defekten Objekte.



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[12]:

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

[12] 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 $E(FC) = p_{\mathrm{E}}$ 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

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

	$E(FC) = 90\%$	$E(FC) = 99\%$	$E(FC) = 99{,}9\%$
$p_{\mathrm{F}} = 100\%$	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

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

Reparatur

Fehlerbeseitigung durch Reparatur

Bei Reparatur werden nur vermutlich defekte Teil des Systems getauscht[13]. Zu ersetzende Teilsysteme:

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

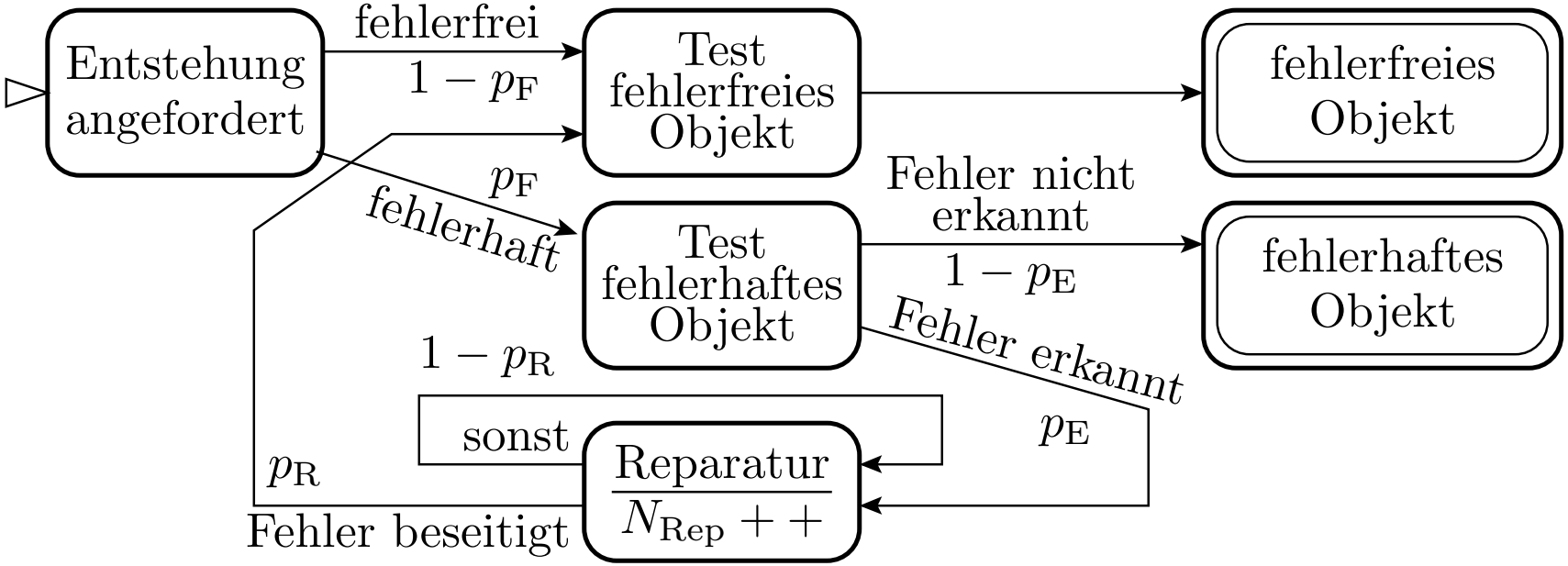
Dafür verlangt Reparatur Zusatzaufwändungen:

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

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

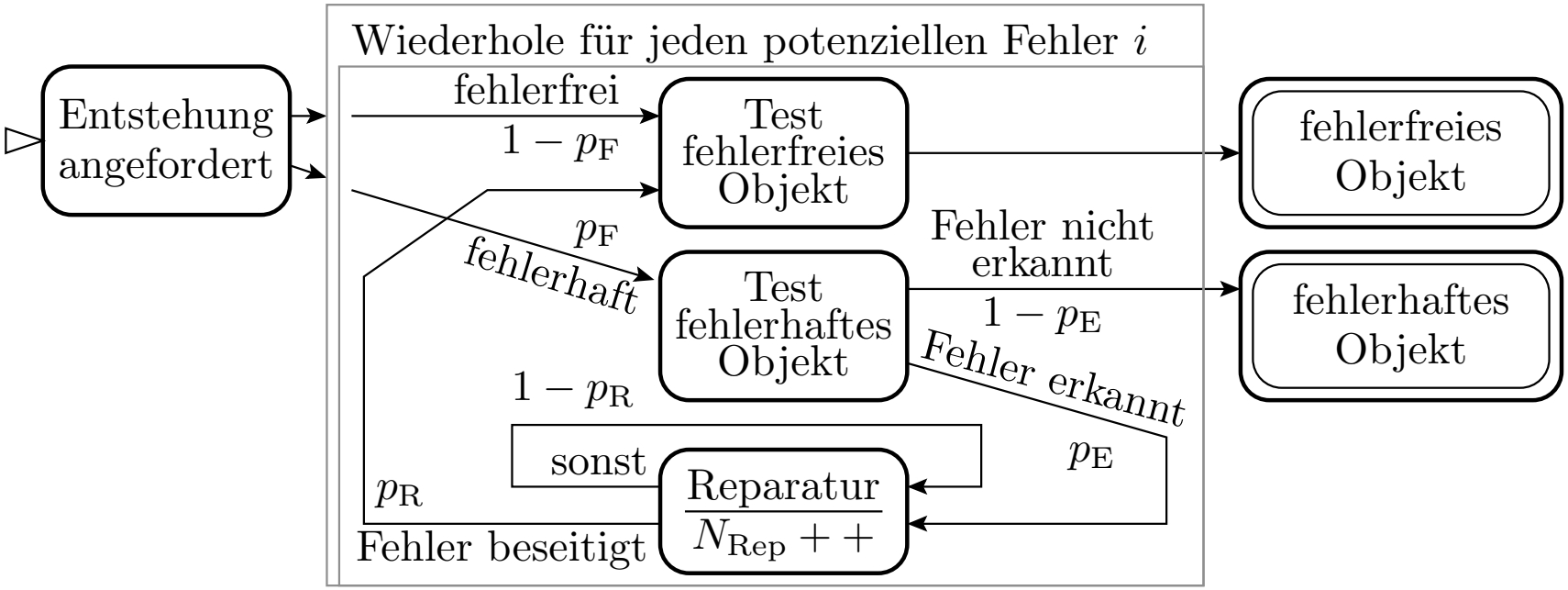
[13] Für Software »tauschen von Anweisungen«

Beseitigungsiteration für einen Fehler

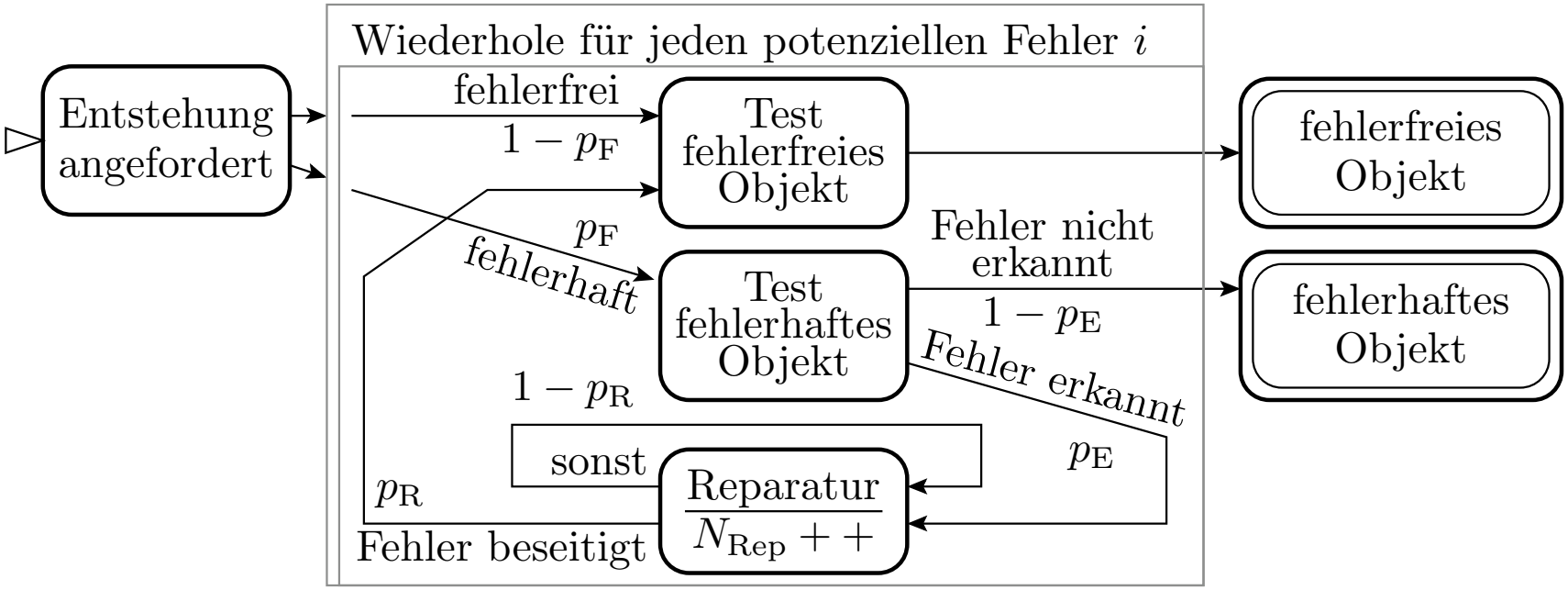


- Bei einem erkennbaren Fehler wird solange mit einer Erfolgswahrscheinlichkeit p_R repariert, bis das vom Test nachweisbare Fehlverhalten beseitigt ist.
- Es sei unterstellt, dass die zu erwartende Anzahl der durch Reparatur neu entstehenden Fehler proportional zur Anzahl N_{Rep} der Reparaturversuche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

Systeme mit mehr als einem zu erwartenden Fehler



- Je eine Markov-Kette für die Beseitigungsiteration eines potenziellen 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 N_{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:

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

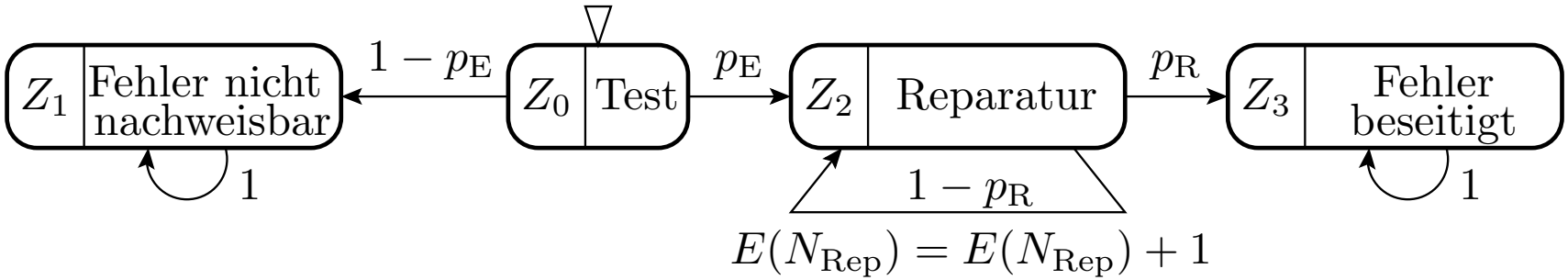
($E\left(N_{\mathrm{Rep}}\right)$ – zu erwartende Anzahl der Reparaturversuche je Fehler; φ_{EP} – Fehleranzahl aus dem Entstehungsprozess; η_{Rep} – Fehlerentstehungsrate in neu entstehenden Fehlern je Reparaturversuch).

Zu erwartende Fehleranzahl nach der Beseitigungsiteration:

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

(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:

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

Fehleranzahl nach der Beseitigungsiteration

Gl. 4 mit $p_{\mathrm{B}} = p_{\mathrm{E}}$ und $E\left(N_{\mathrm{Rep}}\right) = \frac{p_{\mathrm{E}}}{p_{\mathrm{R}}}$:

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

Das entscheidende Gütemaß einer Reparatur Q_{Rep} ist offenbar das Verhältnis aus der Erfolgswahrscheinlichkeit der Reparatur p_{R} und der zu erwartenden Anzahl der neu entstehenden Fehler je Reparaturversuch η_{Rep}:

$$Q_{\mathrm{Rep}} = \frac{p_{\mathrm{R}}}{\eta_{\mathrm{Rep}}} \tag{5}$$

Mit der so definierten Reparaturgüte:

$$E\left(\varphi_{\mathrm{Rep}}\right) = E\left(\varphi_{\mathrm{EP}}\right) \cdot \left(1 + \frac{p_{\mathrm{E}}}{Q_{\mathrm{Rep}}}\right) \cdot \left(1 - p_{\mathrm{E}}\right) \tag{6}$$

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

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

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

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

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

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

vernachlässigbar.

Beispielaufgabe

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

Lösung

Gesucht $p_{\mathrm{Bes}} = \left(1 + \frac{p_{\mathrm{E}}}{Q_{\mathrm{Rep}}}\right) \cdot (1 - p_{\mathrm{E}})$ für
$p_{\mathrm{E}} \in \{50\%, 80\%, 90\%, 99\%, 99{,}9\%\}$ und $Q_{\mathrm{Rep}} \in \{0{,}5, 1, 2, 10\}$?

Lösung mit einem 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 $p_{\mathrm{Bes}} = f\,(Q_{\mathrm{Rep}}, p_{\mathrm{E}})$:

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	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

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

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

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

- Reduktion der Fehleranzahl auf ≈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.

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

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

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

- Das System enthält nach der Test- und Fehlerbeseitigungsiteration mehr Fehler als zuvor.
- 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.

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

Fall C: Studierender ist mit seiner Aufgabe überfordert

$$Q_{\mathrm{R}} \to 0$$

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

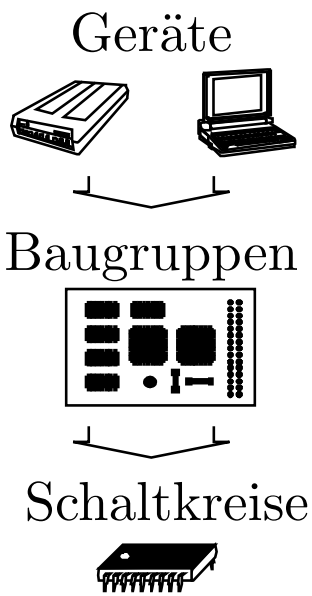
Wenige tauschbare Komponenten

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

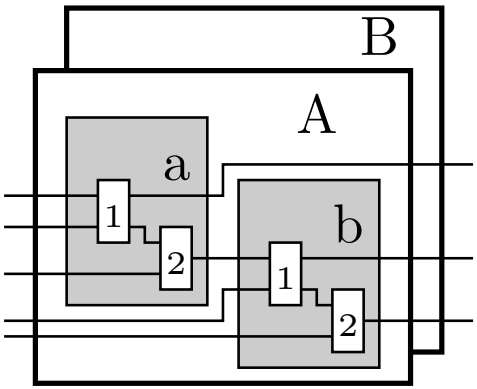
1. Ebene: Austauschbare Geräte.
2. Ebene: Austauschbare Baugruppen.
3. Ebene: Austauschbare Schaltkreise.

Fehlerlokalisierung durch systematisches Tauschen:

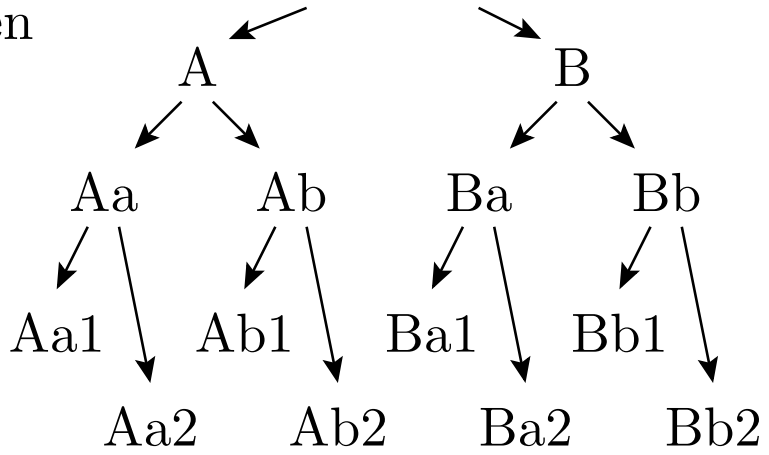
Hierarchie der Hardware



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[15].

Wichtig ist der Rückbau nach jedem erfolglosen Reparaturversuch. Warum?

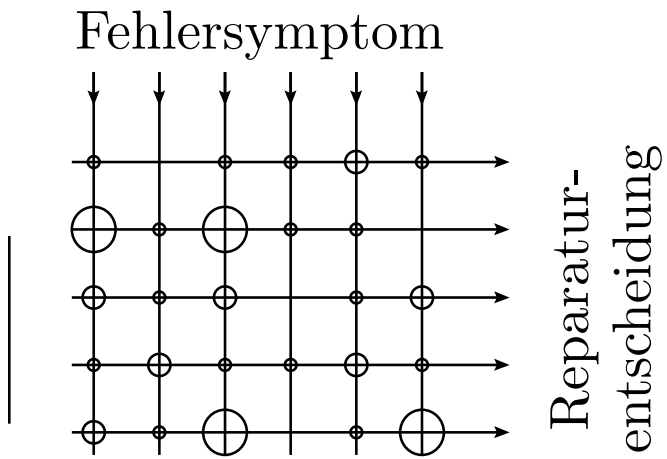
Günstig ist der Tausch der Hälfte, von der fehlerhaften Hälfte wieder der Hälfte, ... Warum?

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

Pareto-Prinzip

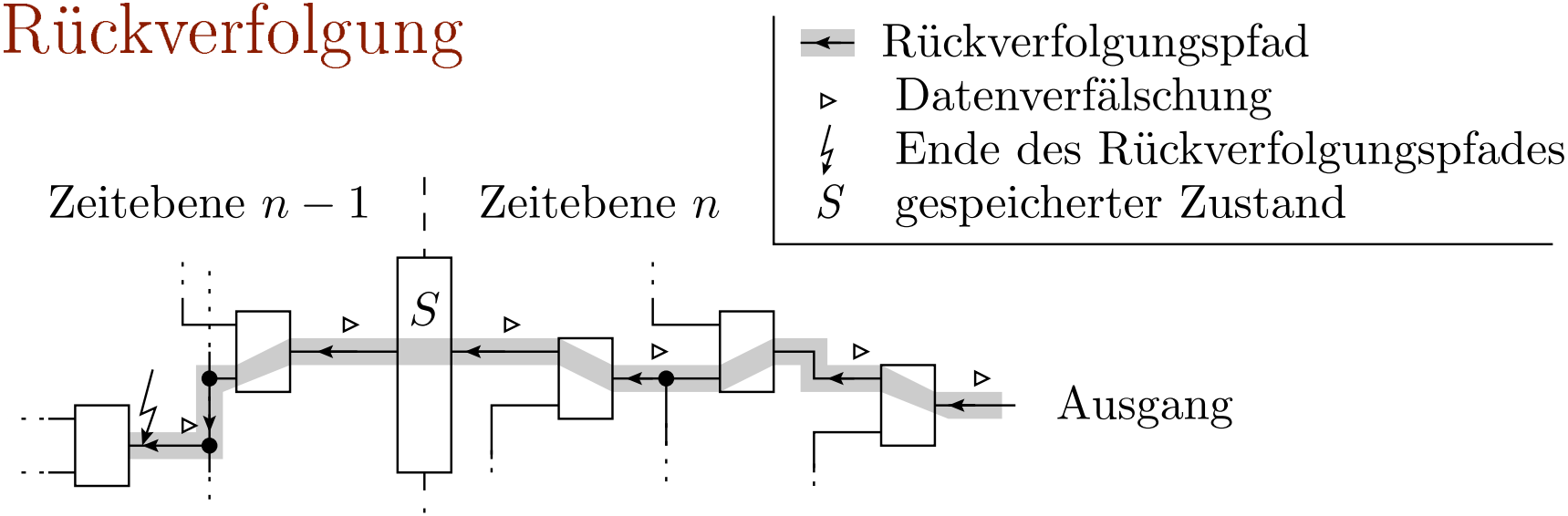
- Pareto-Prinzip: Produkte haben Schwachstellen. Richtwert: 80% der Probleme geht auf 20% der Ursachen zurück.
- Zählen der erfolgreichen und erfolglosen Reparaturversuche.
- Bei Alternativen, Beginn mit der erfolgsversprechendsten Reparaturmöglichkeit.

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



- 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 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 potenziell 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 unerkannt. Diese verursachen im Einsatz beim Nutzer Fehlfunktionen.

Die Nutzer reagieren auf Fehlfunktionen mit:

- der Suche von Workarounds[16] oder
- Änderungswünschen (Change Requests) an den Hersteller.
- Notlösung Ersatz[17].

[16] Im Idealfall Kooperation zwischen Nutzern und Hersteller, z.B. eine Knowhow-Weitergabe über gefundene Workarounds auf FAQ-Seiten.

[17] 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

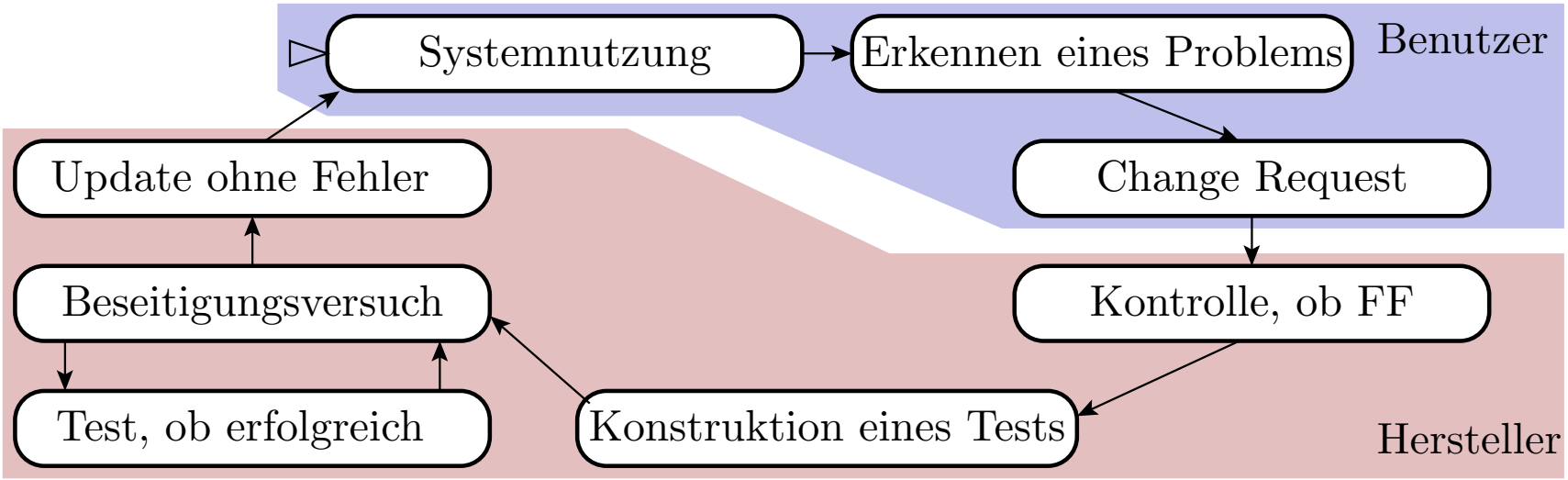
- werden in Summe um viele Zehnerpotenzen mehr Funktionsfälle als vom Hersteller ausprobiert.
- Werden um Größenordnungen mehr Fehler sichtbar.

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[18]).

[18] Jahr 2000-Problem.



Zusatzaufwand Benutzer bei Fehlverhalten:

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

Zusatzaufwand des Herstellers zur normalen Fehlerbeseitigung:

- Verwaltung der Infomationen 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[19]

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.

[19] 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:

- Nutzsystem sendet nur den Bucket-Label.
- System fordern nur bei Bedarf weitere Daten an[20].
- Abschließend sendet der WER-Service dem Nutzer eventuell bekannte Problemlösungen (Verweise auf Bug-Fixes).

[20] Progressive Datensammlung. Datenübenahme nur für eine Stichprobe von Reports je Bucket. Danach nur noch zählen.

Wartung

Wartung

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

- Laufende Kontrollen.
- 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.

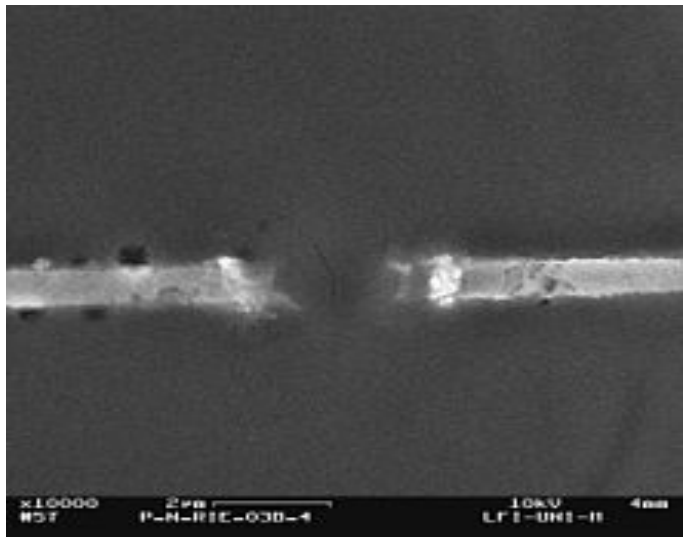
Ausfälle

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 FHNW-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[21], die den Verschleiß beschleunigen. Für sie haftet der Hersteller (Garantiepflicht, Produkthaftung).

[21] Materialrisse, kalte Lötstellen, ...

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«:

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

- Ausfallrate[22] λ: Relative Abnahme der Überlebenswahrscheinlichkeit mit der Zeit:

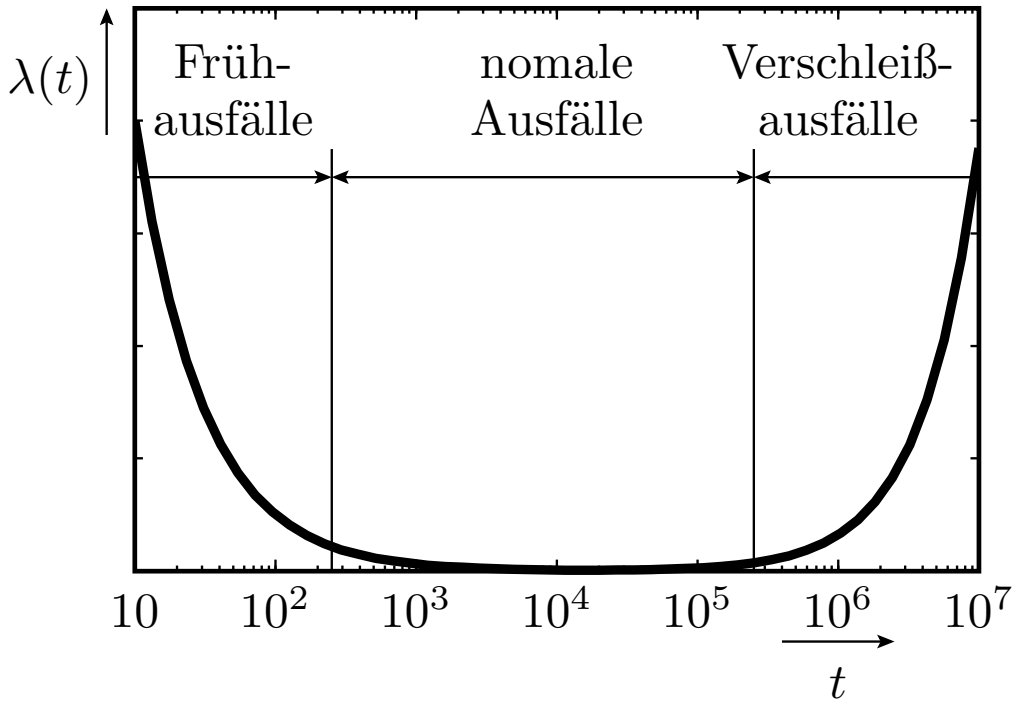
$$\lambda(t) = -\frac{1}{R(t)} \cdot \frac{dR(t)}{dt}$$

- Mittlere Lebensdauer:

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

[22] wichtigste Vergleichsgröße [4, S.68]

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[23].

Maßeinheit der Ausfallrate: fit (failure in time)

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

[23] 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.

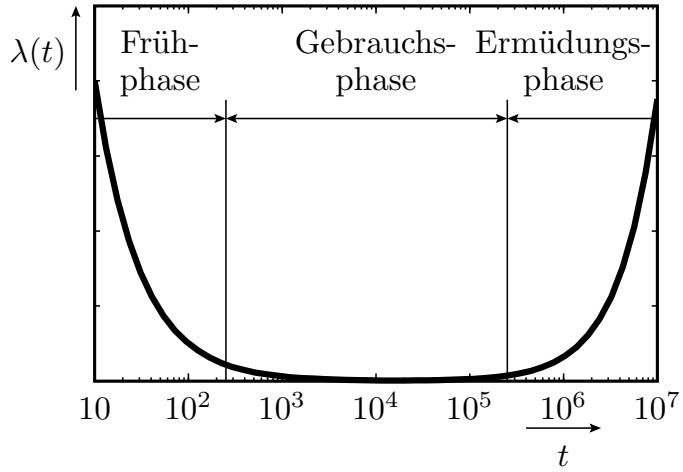
- Ausfallrate einer Baugruppe:

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

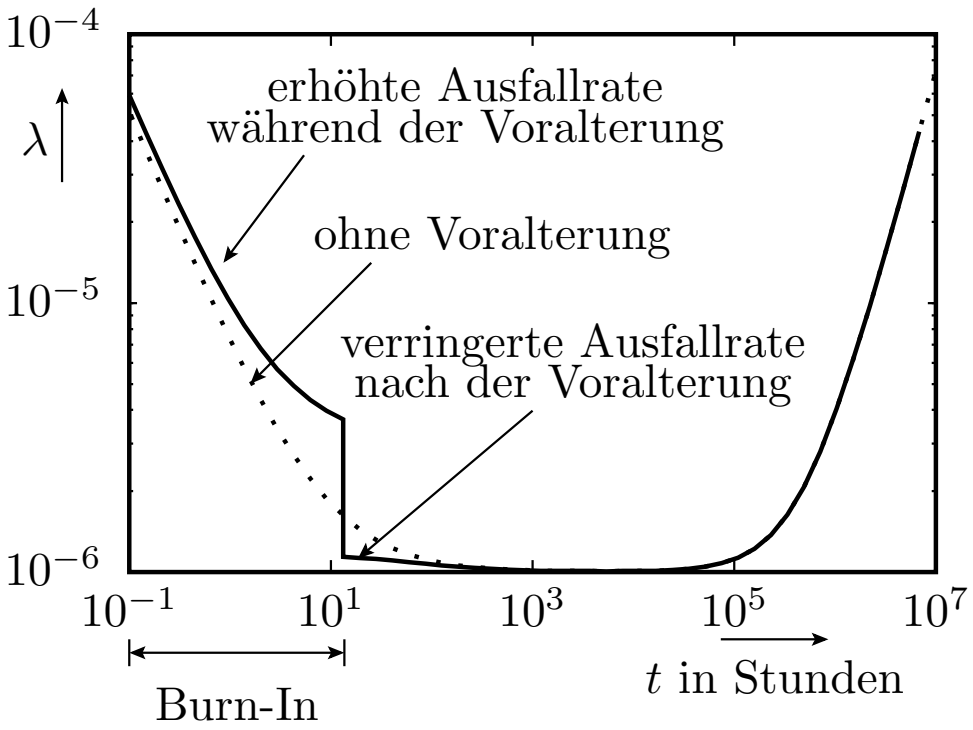
Frühausfälle



- 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.

Kalte, warme und heiße Reserve

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.

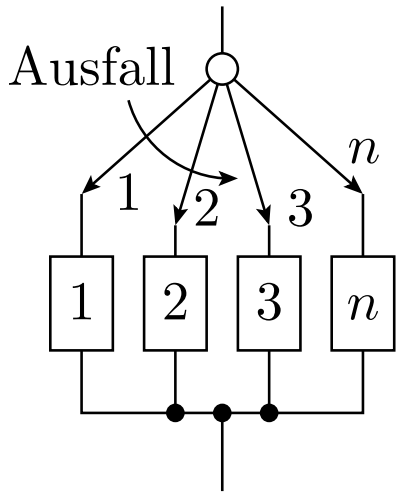
Arten der Reservekomponenten:

- 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, wenn sie nicht genutzt werden, nur weniger.

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[24].

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

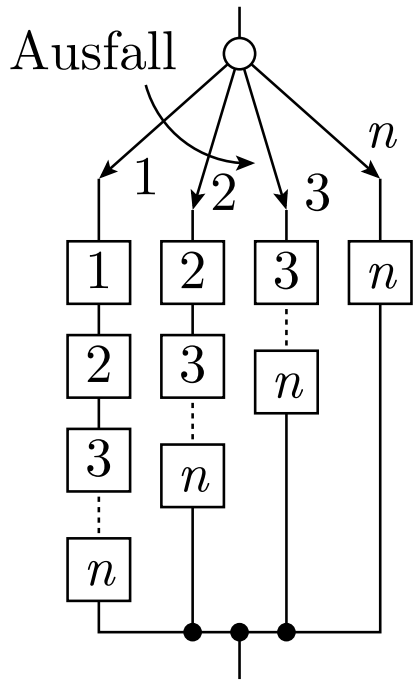
Heiße Reserve

- Alle noch lebenden Komponenten können gleichmaßen ausfallen:

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

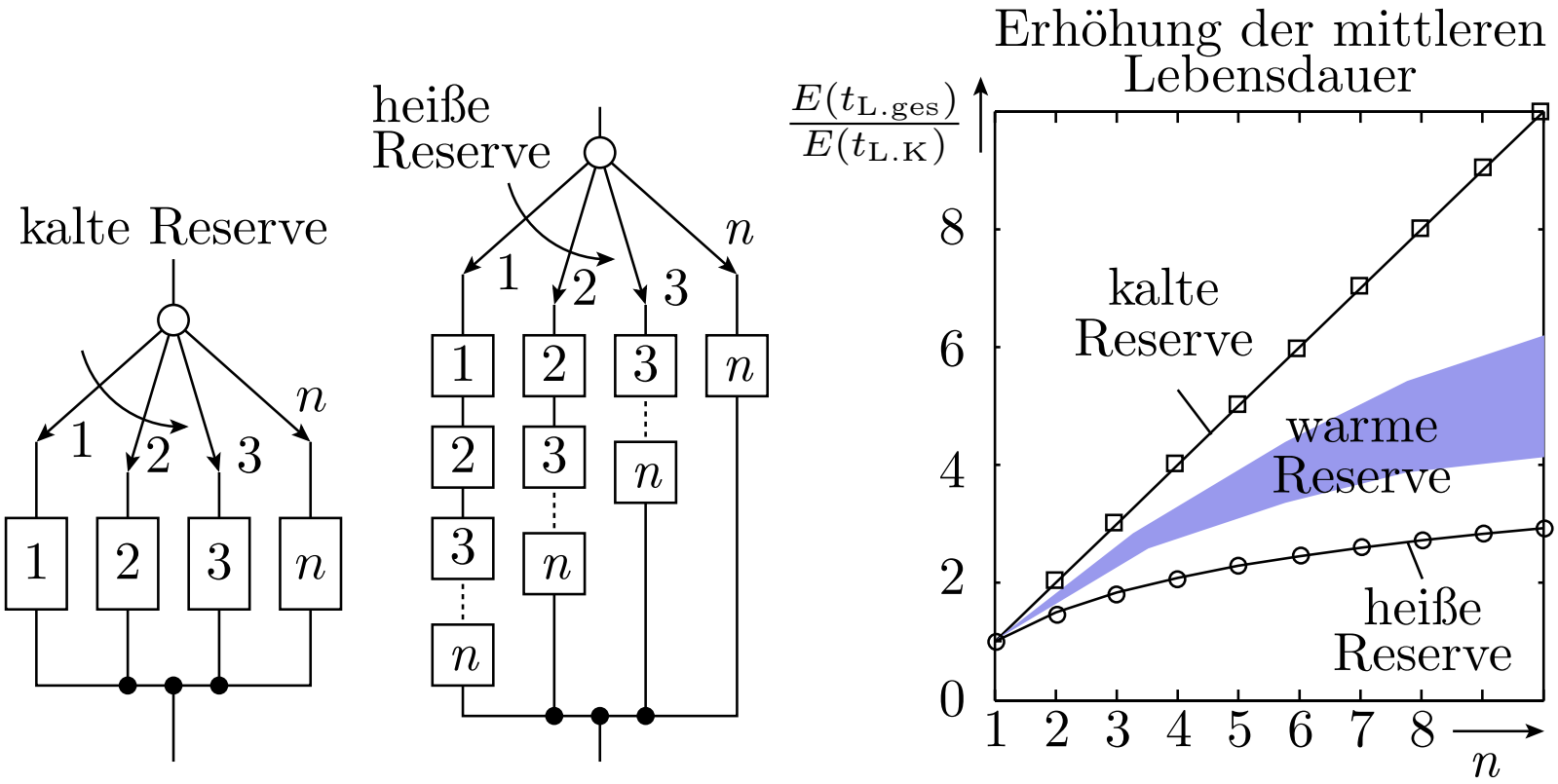
- Komponenten mit gleicher Ausfallrate λ_{K}:

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:	$E\left(t_{\mathrm{L.ges}}\right) = E\left(t_{\mathrm{L.K}}\right) \cdot \sum_{i=1}^{n} \frac{1}{i}$



- Erste Reservekomponente erhöht die mittlere Lebensdauer nur um die Hälfte, die zweite nur 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[25]

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

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

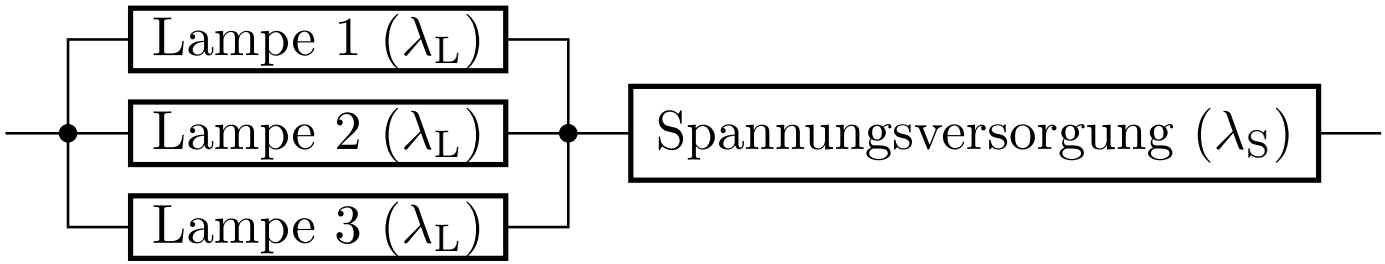
Bei heißer und warmer Reserve ist die Berechnung komplizierter, aber zumindest numerisch auch problemlos lösbar.

[25] 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:

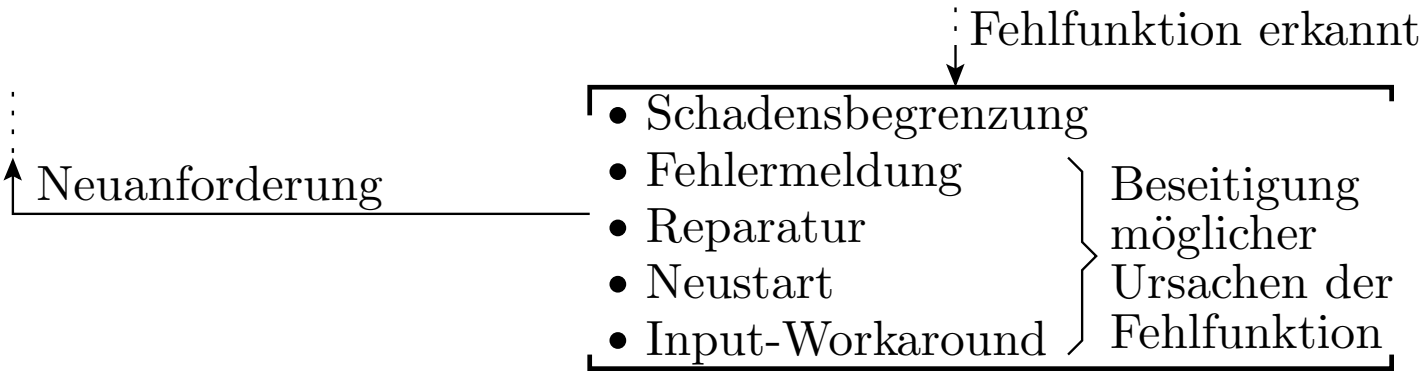
$$\lambda_{Lges} \approx \frac{\lambda_L}{\frac{1}{3}+\frac{1}{2}+1} \approx 0{,}5 \cdot \lambda_L$$

und halbieren die Ausfallrate der Lampeneinheit. Ausfallrate des Gesamtsystems:

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

Fehlerbehandlung

Reaktion auf 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 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[26] Funktionseinheiten.

[26] 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

- Zwangsbremsung, 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, geplatztem 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]

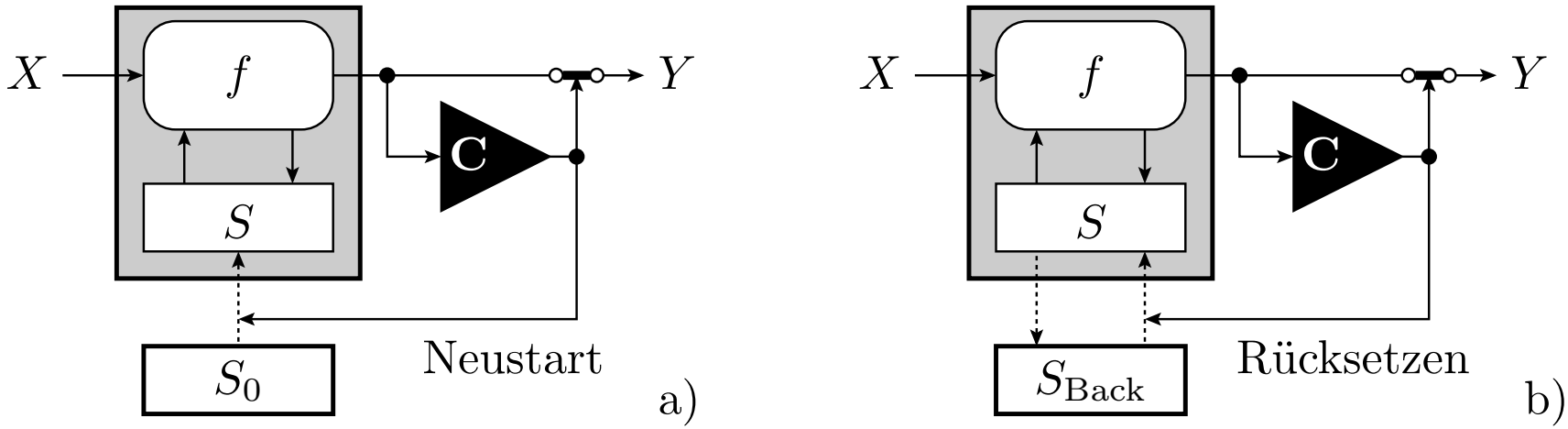
- Angabe der Fehlerschwere in den Assertion-Anweisung
- Umschaltung des Exeption-Handlers, dass nur bei schweren Fehlern ein Abbruch erfolgt.
- Protokollierung und Versendung aller Assertion-Ausgaben.[27]

[27] Höfliche Programme fragen bzw. informieren zumindest, bevor sie Mails nach Hause schicken.

Neustart, Wiederholung

Neuinitialisierung

Zur Fortsetzung der Arbeit nach einer Fehlfunktion ist der potenziell 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 kann nicht vom Programm, d.h. auch nicht durch Fehlfunktionen gestoppt werden, 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[28].

[28] 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[29].
In Ausnahmfällen 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.

[29] Nach F3, Abschn. 3.2 ist Diversität die Wahrscheinlichkeit, dass eine Fehlfunktion durch doppelte Berechnung und Vergleich erkannt wird. Das ist gleichzeitig die Wahrscheinlichkeit, dass bei Wiederholung nach einer Fehlfunktion ein abweichendes Ergebnis berechnet wird.

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 andere Weise:

- Änderung des Rechenwegs.
- Änderung des Berechnungsalgorithmus,
- Nutzung anderer Programme.
- Andere Programmeinstellungen.
- Workaround bei der Bedienung[30] (alternative Eingabemöglichkeiten (Menüs, Konsole, ...), alternative Eingabereihenfolge, ...)

[30] Jeder Nutzer eines größeren Programm lernt mit der Zeit intuitiv, welche Eingabe- und Bedienreihenfolgen funktionieren. Dadurch nimmt die 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-Kernals können die Hardware-Bugs abgewählt werden, für die einprogrammier- te 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

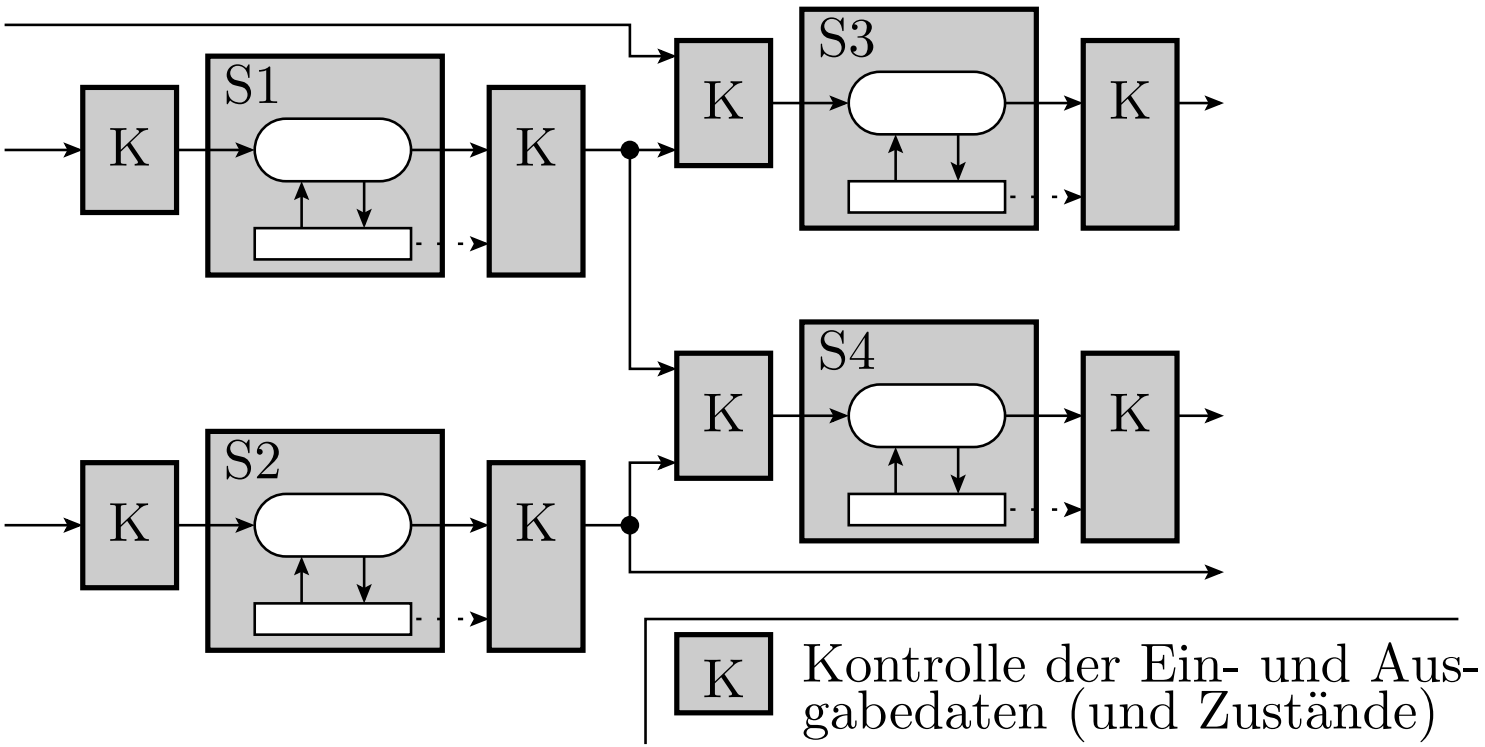
erfordern, dass der Zustand des Systems für die Fehlerbehand- lung, unverfälscht ist:

- 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.

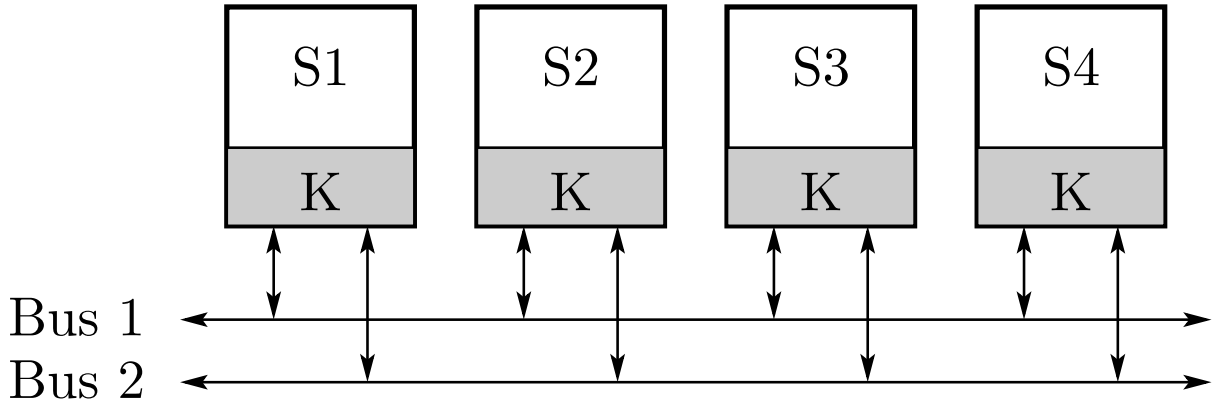
Prinzip der Fehlerisolation

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



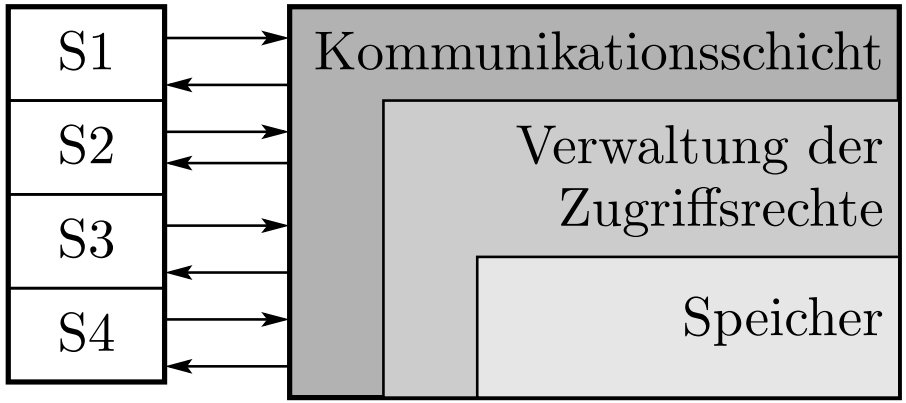
Fehlerisolation in Systemen mit Busstrukturen

- 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 Betriebssystems



- 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 Resourcen (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[31] Funktionseinheiten:

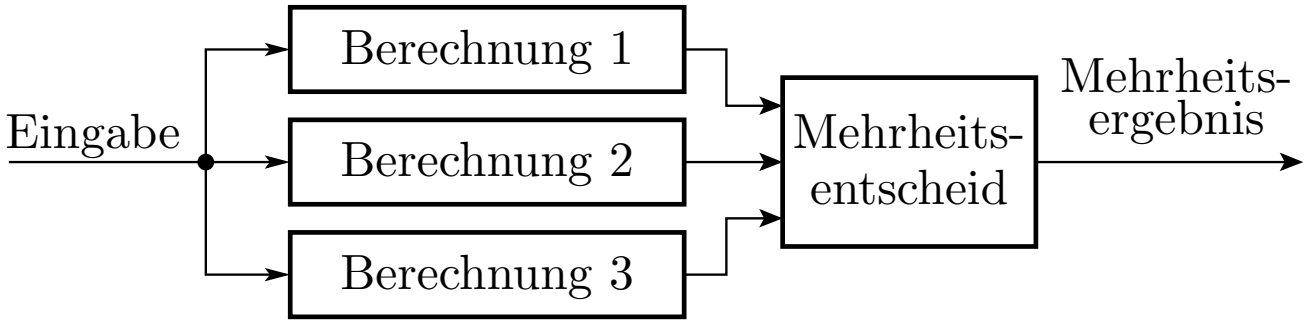
- Informationsredundanz: siehe fehlerkorrigierende Codes.
- homogene Redundanz (Redundanz mit gleichartigen Mitteln): Reserveeinheiten zur Vorbeugung gegenüber Hardware-Ausfällen z.B. zwei Glühbirnen im roten Teil einer Ampel (siehe heiße und kalte Reserve, Folie 92) und
- diversitäre Redundanz (Redundanz 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.

[31] 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.

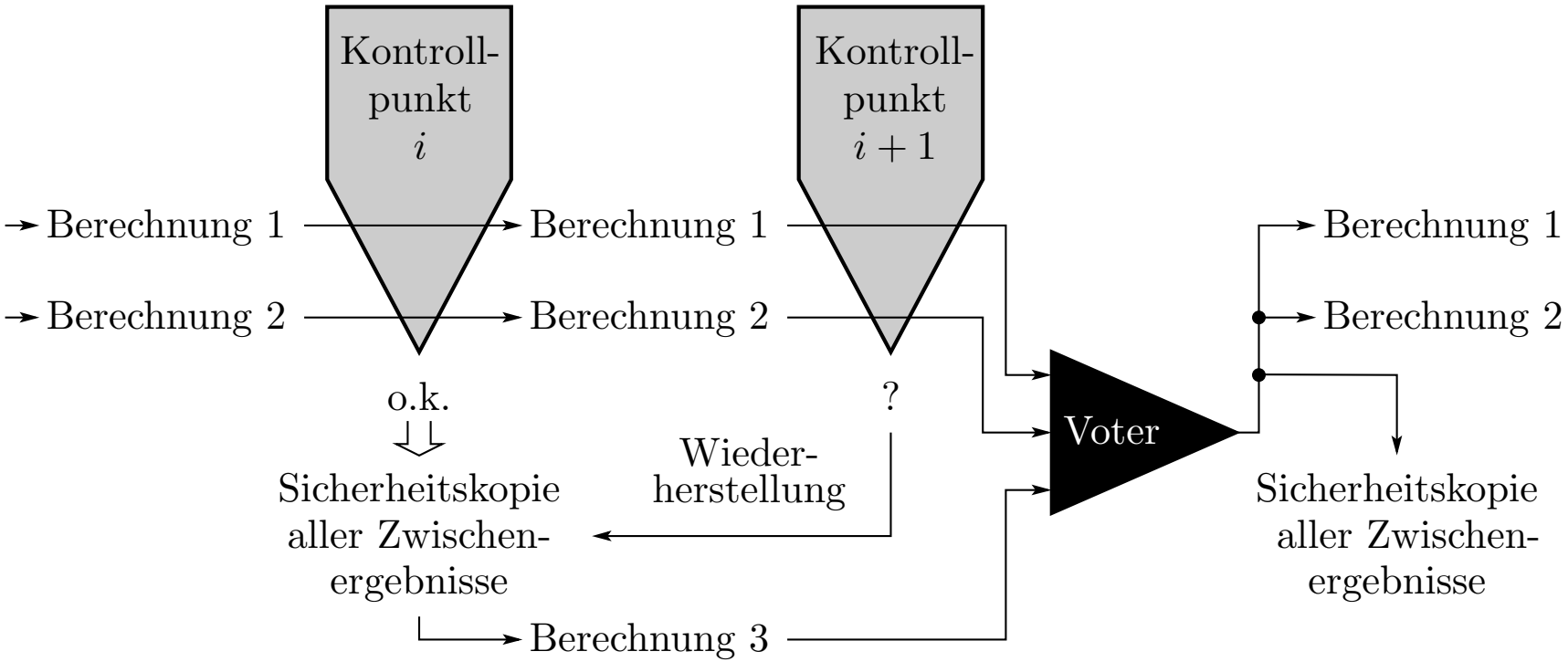
- Ein Mehrversionssystem in dieser Form toleriert keine übereinstimmenden Entwurfsfehler in den Systemen.

  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 eine 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[32] verglichen.

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

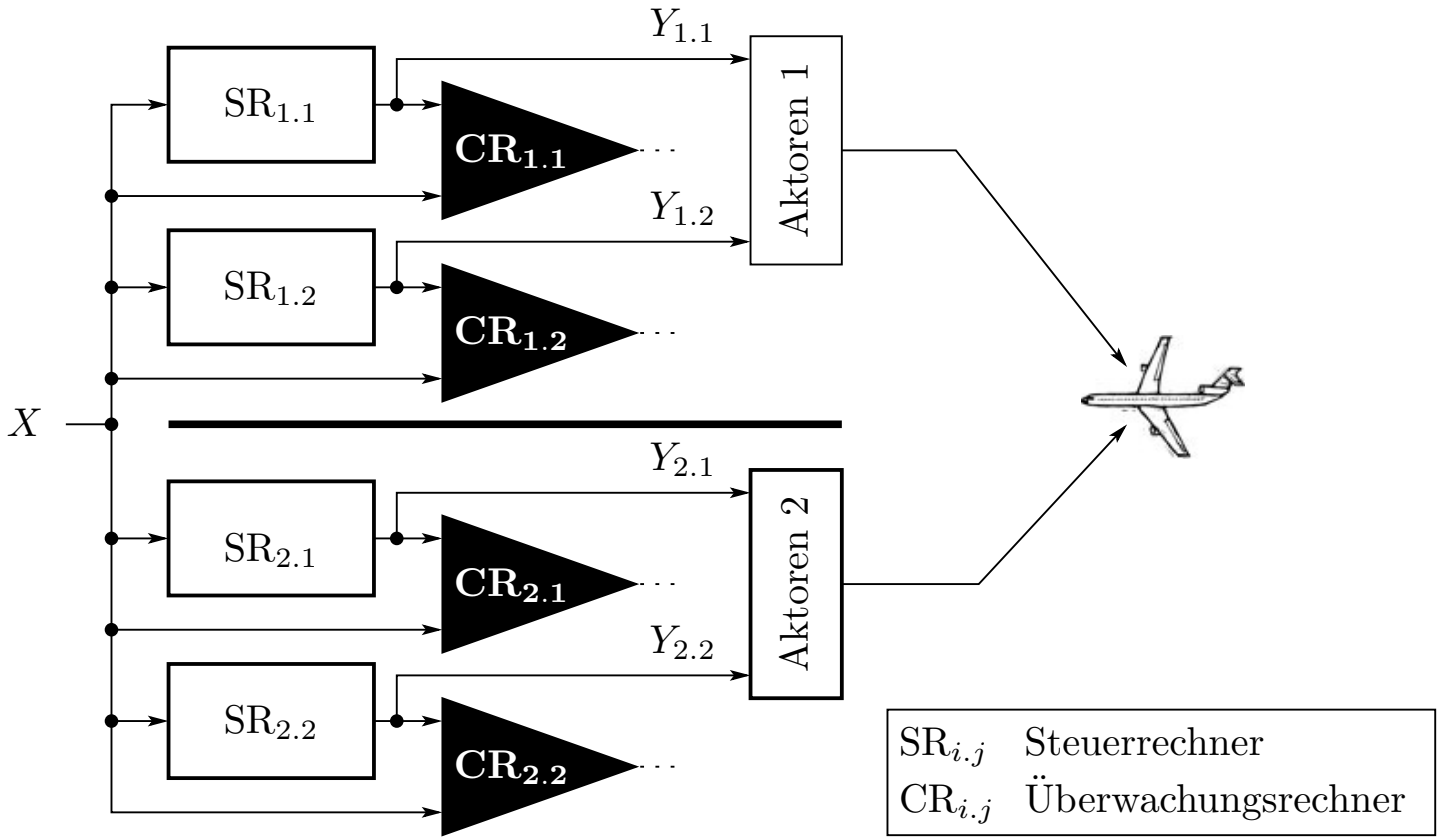
- Bei Übereinstimmung Speicherung des Bearbeitungszustands in einem geschützten Speicher.
- Bei Abweichung, Laden der letzten Sicherheitskopie und Berechnungswiederholung (Roll-Back Recovery).
- Nach Roll-Back Recovery am nächsten Kontrollpunkt wieder Mehrheitsentscheid.
- Wenn Mehrheitsergebnis, diesen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

Sequoia-System [3]:

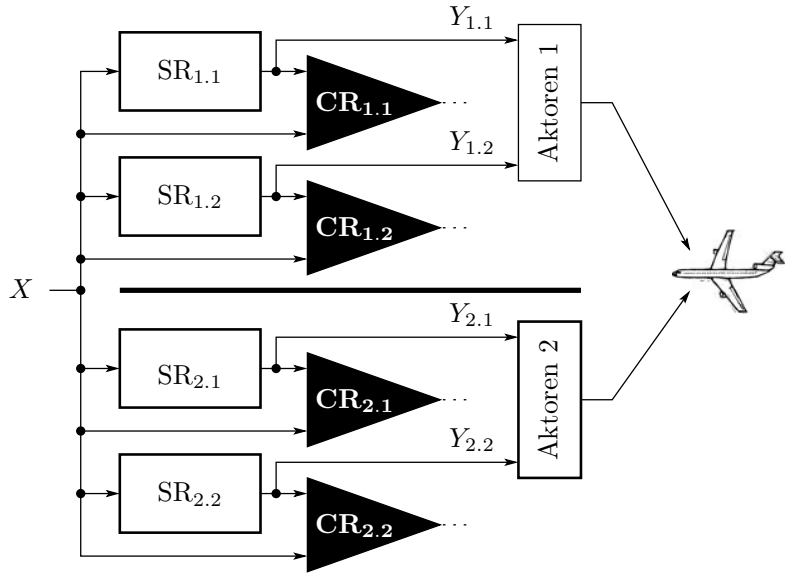
- Berechnung auf zwei Prozessoren mit eigenem Write-Back-Cache.
- Vergleich in jedem Takt.
- Zustands-Backup bei Ereignissen wie Stack-Überlauf und Prozesswechsel.
- Hauptspeicher hat die Funktion des stabilen Speichers.

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 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: Mehrversionssentwürfe, 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.