

**Eigenschaften mobiler und
eingebetteter Systeme:**

Verlässlichkeit

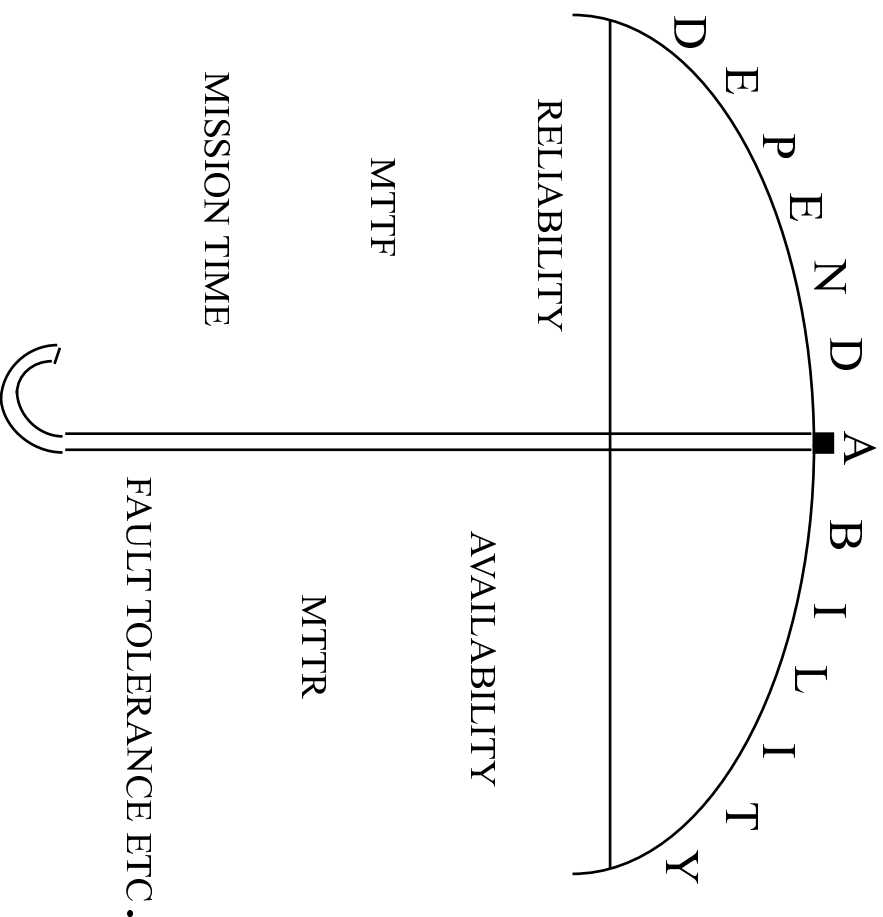
EMES

Dr.-Ing. Matthias Werner
Dipl.-Inf. Jan Richling
Wintersemester 2001/2002

Literatur

- T. Anderson, P.A. Lee: Fault Tolerance – Principles and Practice, Prentice Hall, 1982
- D.K. Pradhan (Hrsg.): Fault Tolerant Computer Systems, Prentice Hall, 1996
- D.P. Siewiorek, R.S. Swarz: The Theory and Practice of Reliable Systems Design, Digital Press, 1995
- F. Cristian: Understanding Fault-Tolerant Distributed Systems, Communications of the ACM, Vol 34(1991)2, Semesterapparat

Verlässlichkeit



- **Begriff der Verlässlichkeit**
Verlässlichkeit (dependability) ist ein Oberbegriff, der eine Vielzahl von Konzepten und Maßen abdeckt.
- **Allgemeine Frage:**
„Wie umgehen mit Fehlern?“

Fehlerintoleranz

„Wie umgehen mit Fehlern?“ ⇒ vermeiden

- Eliminierung der Ursachen von Unzuverlässigkeit durch:
 - Fehlervermeidung
 - Fehlerbeseitigung
- Keine Redundanzen
- Fehlerintoleranz bringt Verlässlichkeit durch:
 - Einsatz sehr zuverlässiger Komponenten
 - Ausgefeilte Entwurfstechniken
 - Ausgefeilte Produktionstechniken
 - Abschirmung
 - Eingehende Tests

Fehlertoleranz

„Wie umgehen mit Fehlern?“ ⇒ tolerieren.

- Fehlertoleranz
 - akzeptiert, daß ein fertiges System nicht fehlerfrei ist.
 - Fehlertoleranz wird durch Redundanz in Raum oder Zeit erreicht.
 - Automatische Behebung von Fehlerzuständen
 - Kombination von Redundanz und Fehlerintoleranz
- Vorteile:
 - Höhere Zuverlässigkeit
 - Evtl. Niedrigere Gesamtkosten
 - Vertrauen der Nutzer (Psychologische Unterstützung)
- Nachteile:
 - Kosten der Redundanz
 - Evtl. Höhere Komplexität

Fehler

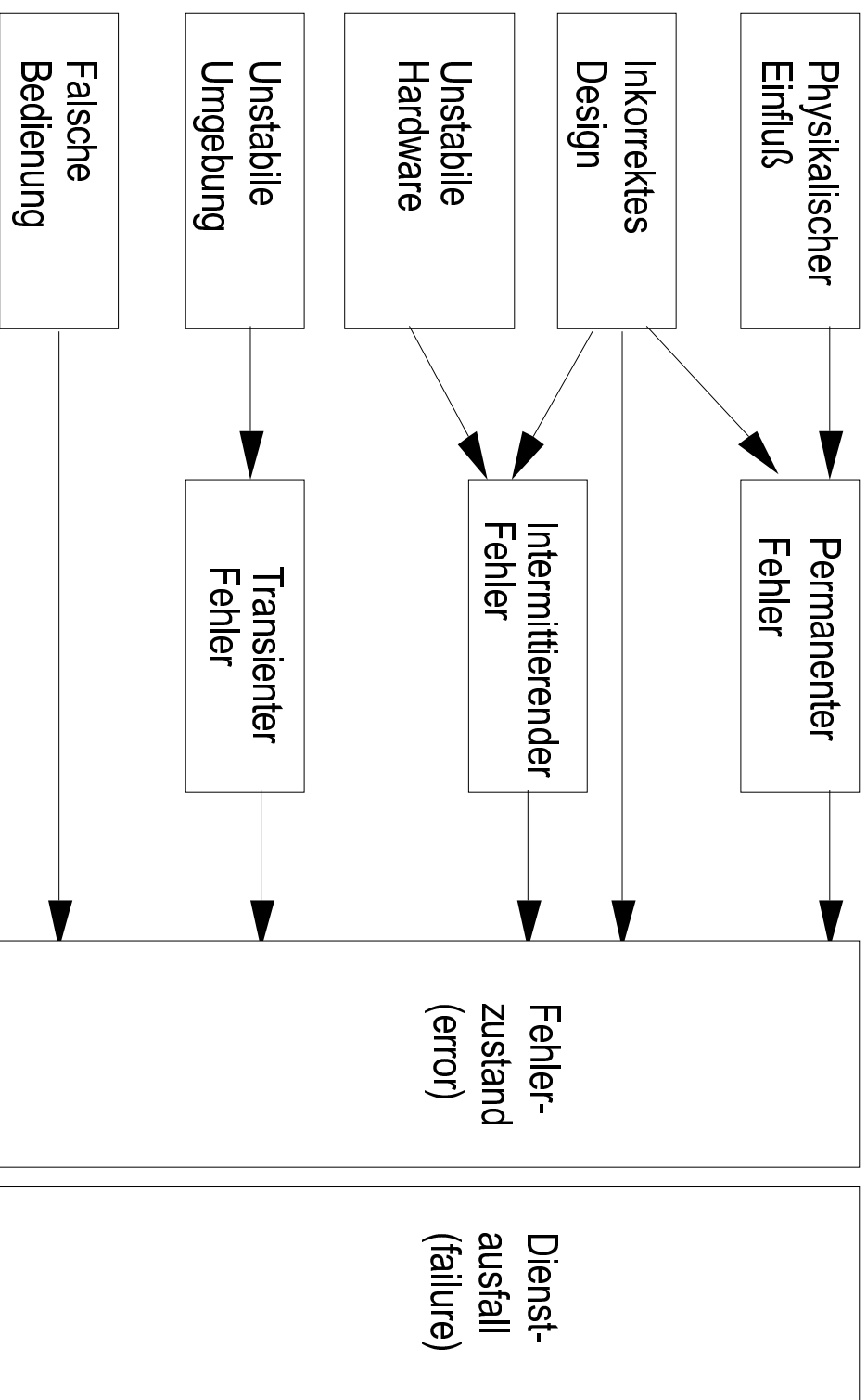
Fehler (allgemein): Abweichung von der Spezifikation (Erwartung).

- **Ausfall (failure)** Entsteht, wenn der erbrachte Dienst vom erwarteten abweicht. Ausfälle werden durch Fehler verursacht.
- **Fehlerursache (fault)** Inkorrektter Zustand der Hard- oder Software
- **Fehler(zustand) (error)** Manifestation einer Fehlerursache innerhalb eines Programmes oder in Daten, so daß eine Abweichung von erwarteten Berechnungsergebnis entsteht (falsches Ergebnis).

System und Fehlermodell

- **Fehlertolerantes Systemdesign:** Eigentlich Widerspruch in sich:
 - Design benötigt Spezifikation (Was wird erwartet?)
 - Fehler sind Abweichung von Spezifikation
- Lösung: Spezifikation für fehlerfreien Fall + zusätzliche Fehlerspezifikation
- **Achtung:** Wechselwirkung zwischen Systemmodell und Fehlerspezifikation

Kausalitäten bei Ausfällen



(nach Stewiörek/Swarz)

Beschreibung von Fehlern

- Ursache
 - Spezifikation
 - Design
 - Implementation
 - Komponenten
 - äußere Ursache
- Charakter
 - Hardware
 - Software
 - Analoges Verhalten
 - Digitales Verhalten
- Dauer
 - permanent
 - temporär
 - transient
 - intermittierend
 - latent
- Ausdehnung
 - lokal
 - verteilt
- Fehlerwert
 - deterministisch
 - nicht deterministisch

Fehlermodelle

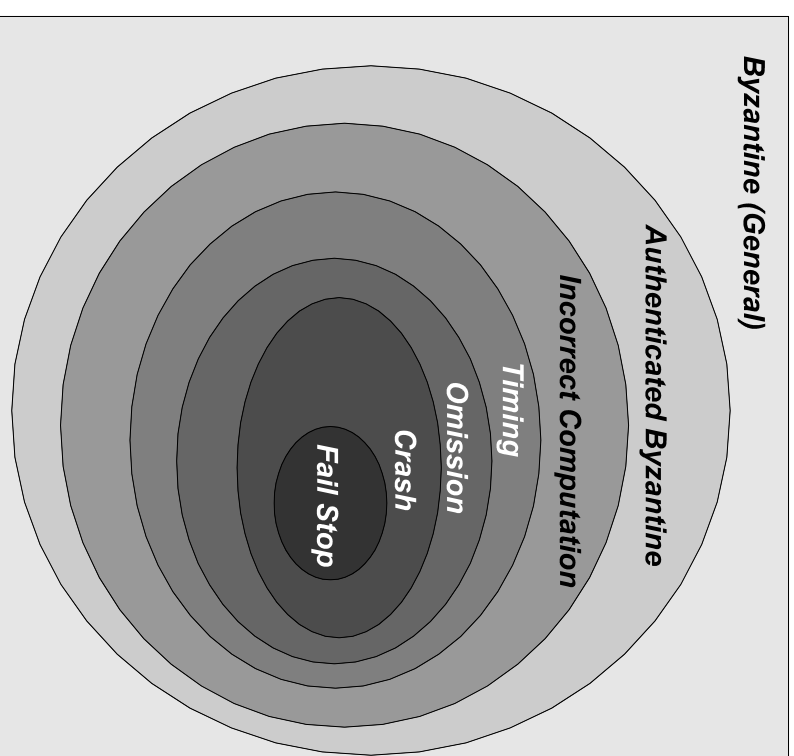
Fehlermodelle können auf verschiedenen Abstraktionsebenen beschrieben werden.

- (Physik [relativ ungebräuchlich])
- Schaltungsebene (*circuit level*)
- Schalterebene (*switching [circuit] level*)
- Registerebene (*register transfer level*)
- PMS-Ebene (*processor-memory-switch*)
- Systemebene (*system level*)

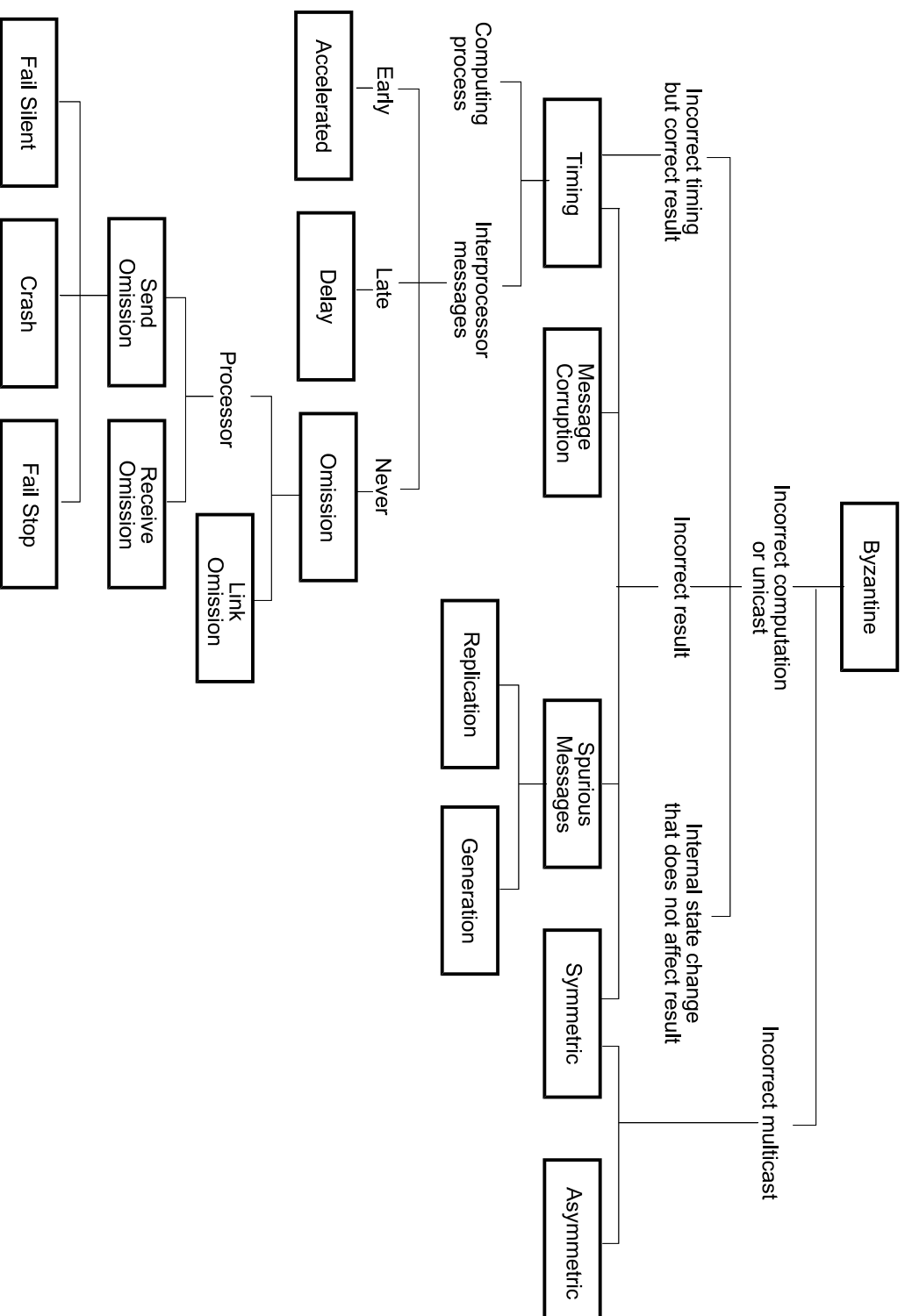
Fehlermodelle für verteilte Systeme

- Ausfallfehler (crash)
- Auslassungsfehler (omission)
- Zeitfehler (timing)
- Wertefehler (incorrect computation / communication)
- Beliebiger Fehler (arbitrary / Byzantine)

Link-Fehler werden auf Komponentenfehler (des Senders oder des Empfängers) abgebildet.

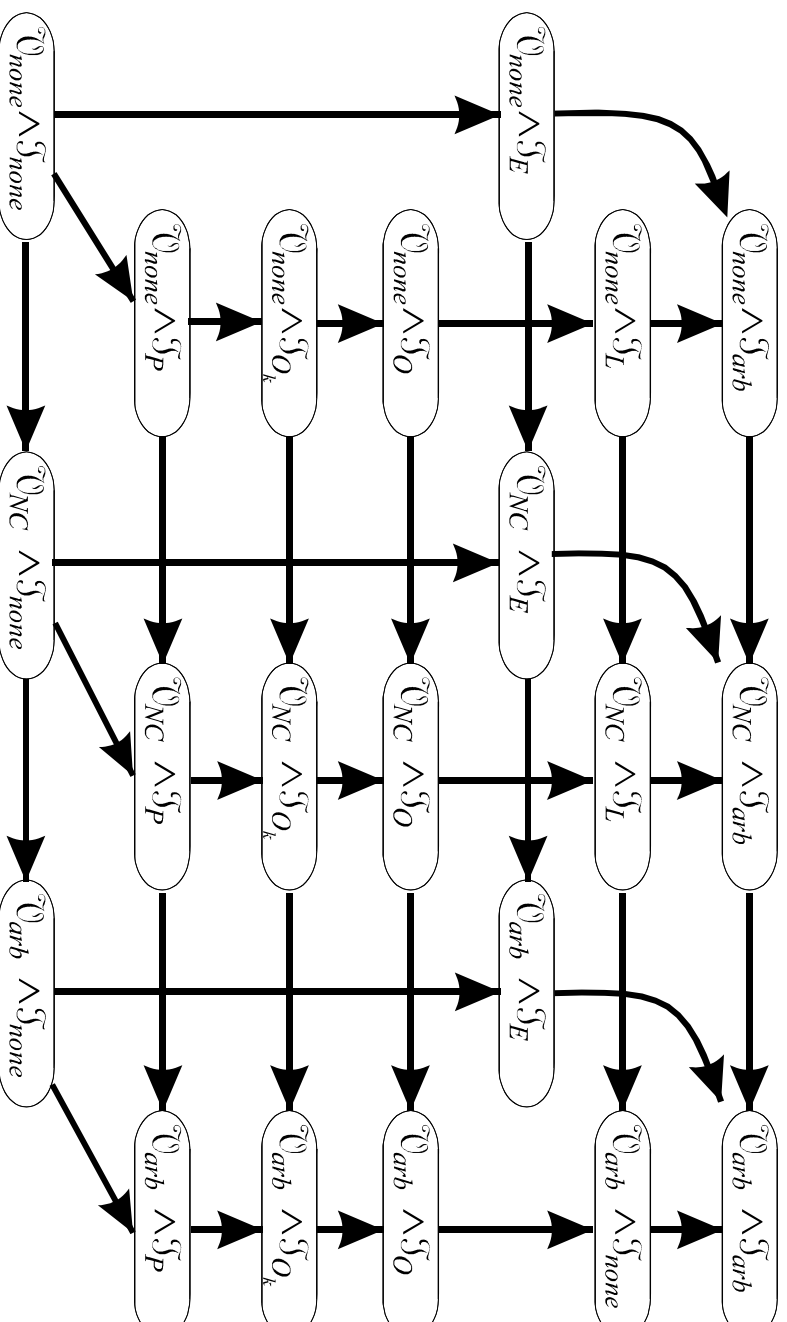


Fehlerhierarchie in verteilten Systemen



Halbordnung der Systemfehler

- Pfeile deuten abnehmende Striktheit der Annahmen an
- \mathcal{V} : Wertebereich, \mathcal{T} : Zeitbereich



Grundlegende Ansätze zur Erlangung von Fehlertoleranz

Fehlertoleranz benötigt stets Redundanz.

- Redundanz im Raum
 - ⇒ Zusätzliche Hardware, Speicherbedarf, ...
- Redundanz in der Zeit
 - ⇒ Zusätzliche Rechenzeit, Zeit für Fehlererkennung und -behebung, etc.
- (Funktionale Redundanz) ⇒ Zusätzliche Hardware, Speicherbedarf, ...

Achtung! Fehlertolerante Systeme sind **nicht automatisch** verlässlicher als Systeme ohne Fehlertoleranz.

Bewertungsmaße

- Um die Beeinträchtigung eines Systems oder den Erfolg einer Gegenmaßnahme zu bewerten, sind Bewertungsmaße notwendig.
- Unterschiedliche Anwendungen haben unterschiedliche Zielstellungen ⇒ unterschiedliche Bewertungsmaße sind notwendig.
- Beispiele:
 - **Telefonanlage:** Muß immer funktionieren, aber Fehler im Einzelfall (z.B. falsche Verbindung) sind hinnehmbar.
 - **Marssonde:** Muß während eines klar gesteckten Zeithorizontes in Funktion bleiben.
 - **Freund-Feind-Kennung:** Genaue Abwägung zwischen möglichen „false friend“ und „false enemy“ Fehlern.

Definitionen von Verlässlichkeit

Verlässlichkeit, häufig auch Zuverlässigkeit (*dependability*)

- **Verlässlichkeit** ist ein *quantitatives Maß* (wie z. B. Überlebenswahrscheinlichkeit oder Verfügbarkeit), daß durch den Nutzer beobachtet werden kann.
- **Verlässlichkeit** ist die *Eigenschaft* eines angebotenen Dienstes, daß man sich auf ihn begründet verlassen kann.
- **Verlässlichkeit** ist die *Fähigkeit* eines Systems, einen benötigten Dienst unter definierten Bedingungen für eine festgelegte Zeitspanne auszuführen

Zuverlässigkeit

Die Zuverlässigkeit (auch: Überlebenswahrscheinlichkeit, *reliability*) $R(t)$ eines Systems ist die Wahrscheinlichkeit, daß das System innerhalb einer Zeitspanne $[0, t]$ zufriedenstellend funktioniert, unter der Voraussetzung, daß das System zu $t_0 = 0$ funktioniert hat.

- Hardware
 - Exponentialverteilung, $R(t) = e^{-\lambda \cdot t}$,
 λ : Fehlerrate
 - Weibull-Verteilung, $R(t) = e^{-(\lambda \cdot t)^\alpha}$,
 α : Formparameter
- Software: Bisher keine einfachen (und stimmigen) Modell

Die Fehlerrate wird häufig als konstant angenommen, kann aber auch eine Funktion der Zeit sein.

Warum Exponentialfunktion?

Annahme: Die Anzahl der ausfallenden Geräte/Einheiten einer Charge pro Zeiteinheit ist konstant.

$$\frac{d}{dt}x(t) = -\lambda \cdot x(t), x(0) = x_0$$

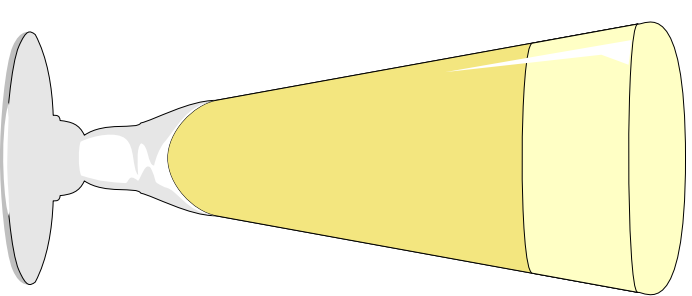
Wie bei allen nur von der Menge abhängenden Zerfallsprozessen (z.B. Bierschaum), gibt es die allgemeine Lösung

$$x(t) = x_0 \cdot e^{-\lambda \cdot t}$$

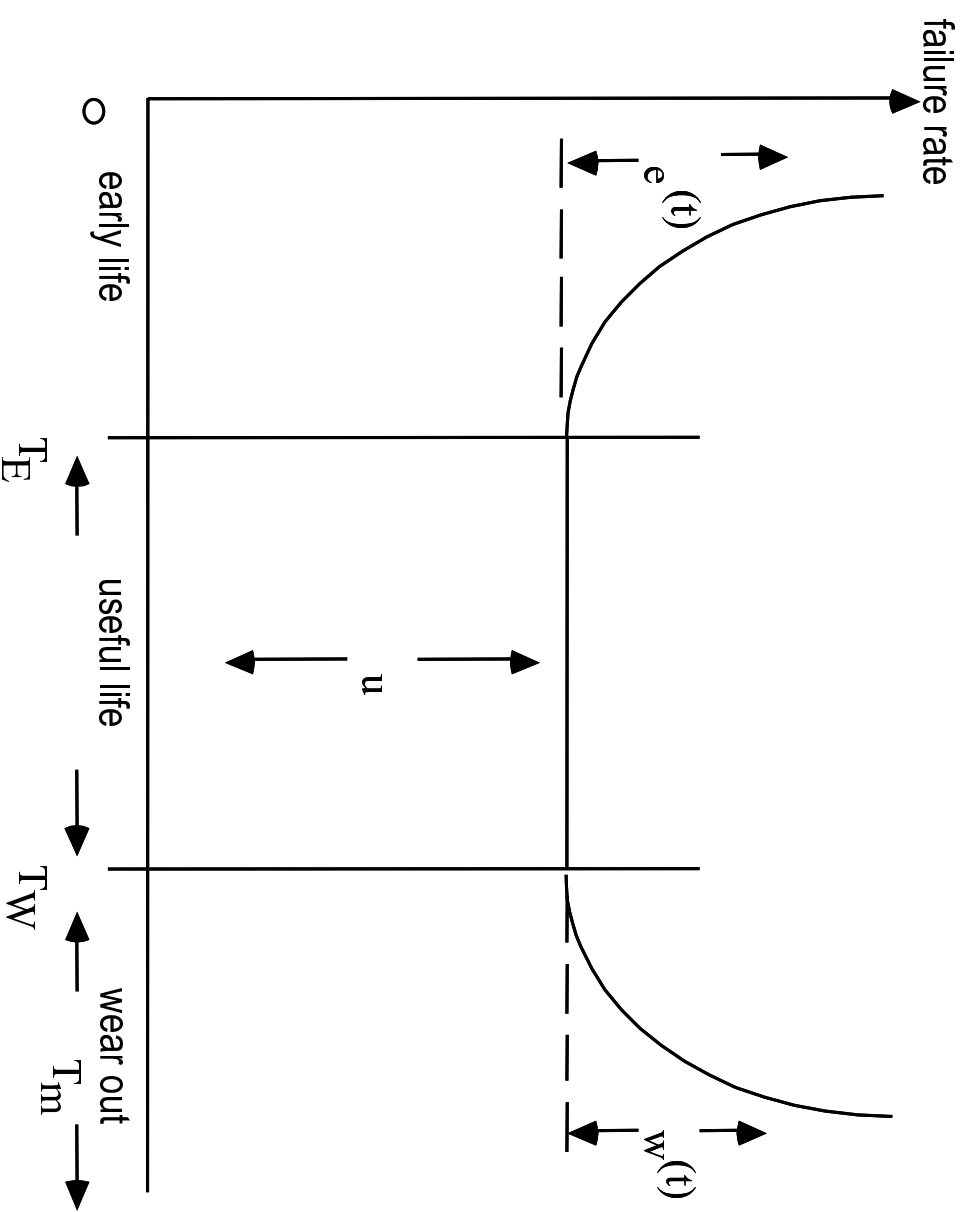
Eine Überlebenswahrscheinlichkeit eines Gerätes entspricht der Anzahl der Geräte einer Charge, die nach der Zeitspanne noch funktionieren.

Zum Zeitpunkt $t = 0$ funktionieren laut Definition noch alle Geräte.

$$R(t) = e^{-\lambda \cdot t}$$

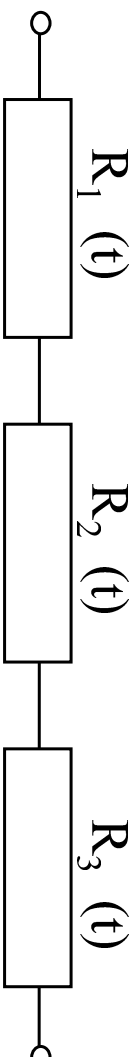


Badewannenkurve



Serienschaltung

Der Ausfall eines Modules führt zum Gesamtausfall



$$R_s(t) = R_1(t) \cdot R_2(t) \cdot R_3(t) = e^{-(\lambda_1 + \lambda_2 + \lambda_3)t}$$

Allgemein für eine Serie von n Modulen

$$R_s(t) = \prod_{i=1}^n R_i(t) = e^{-\lambda_s t}, \quad \lambda_s = \sum_{i=1}^n \lambda_i$$

λ_s : Systemfehlerrate

λ_i : Fehlerrate der einzelnen Moduln

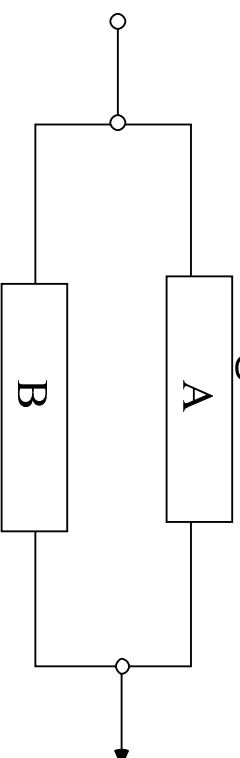
Bei n identischen Modulen:

$$R_s(t) = (R(t))^n$$

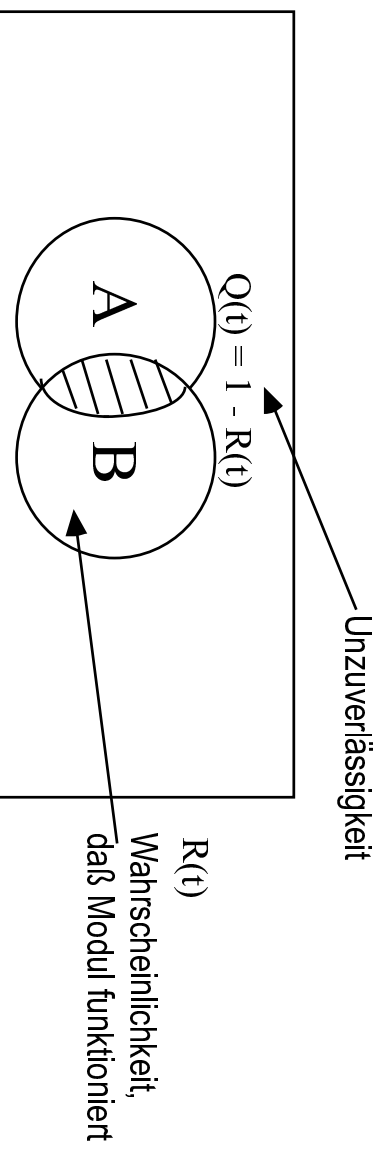
Parallelschaltung

Jedes Modul funktioniert unabhängig von den anderen.

Das System funktioniert, wenn wenigstens ein Modul funktioniert.



$$R_{p}(t) = R_A(t) + R_B(t) - R_{AB}(t) = R_A(t) + R_B(t) - R_A(t)R_B(t)$$



Venn Diagram

Parallelschaltung (II)

- **Allgemeiner Fall für n Module**

$$R(t) = 1 - (1 - R_1(t)) \cdot (1 - R_1(t)) \cdots (1 - R_1(t))$$

- **n identische parallele Module**

$$R_{rp}(t) = 1 - (1 - R_m(t))^n$$

Beispiel: $R_A = R_B = 0,5$, $R_P = 1 - (0,5)^2 = 0,75$

Gemittelte Zeiten

Da Raten eher „unhandlich“ sind, gibt es gemittelte Zeiten als Verlässlichkeitsmaße.

- **MTTF** (Mean Time To Failure, mittlere Zeit bis zum Ausfall)

$$MTTF = \int_0^{\infty} R(t) dt = \frac{1}{\lambda} \text{ für Exponentialverteilung}$$

- **MTTR** (Mean Time To Repair, mittlere Zeit der Reparatur)

$$MTTR = \frac{1}{\mu}, \quad \mu: \text{Reparaturrate}$$

- **MTBF** (Mean Time Between Failure, mittlere Zeit zwischen zwei Ausfällen)

$$MTBF = MTTF + MTTR$$

Verfügbarkeit

- **Verfügbarkeit (Availability)** $A(t)$ eines Systems ist die Wahrscheinlichkeit, daß das System zu einem gegebenen Zeitpunkt t funktioniert (seinen Service zufriedenstellend liefert)
- **Steady-state Verfügbarkeit** A_s eines Systems ist der Anteil der Lebenszeit des Systems, in dem es funktioniert.

$$\bullet A_s = \frac{\text{Uptime}}{\text{total Time}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} = \frac{\mu}{\mu + \lambda}$$

λ : Fehlerrate

μ : Reparaturrate

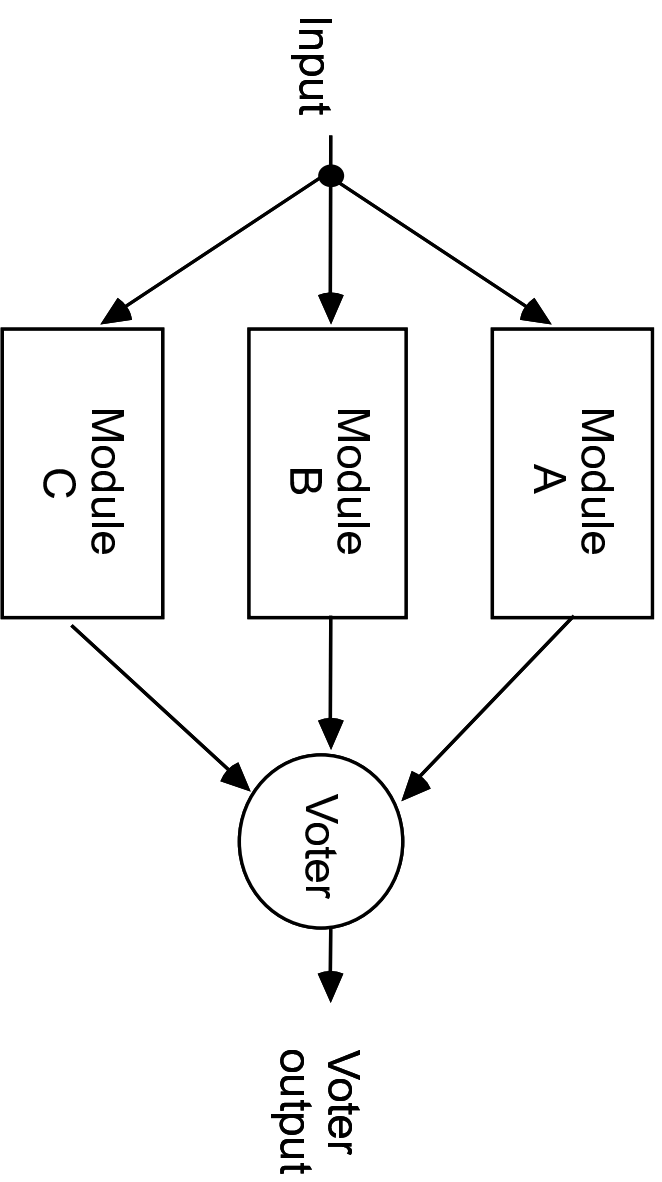
Weitere Maße

Außer den genannten gibt es noch eine Reihe weiterer Maße, die aber eher in Spezialfällen angewendet werden und daher nicht so stark verbreitet sind.

- **Responsivität (responsiveness)** $\mathcal{R}(t, D)$ ist die Wahrscheinlichkeit, daß ein zum Zeitpunkt t gestarteter Dienst bis zum Zeitpunkt $t + D$ korrekt ausgeführt wird.
- **Missionsdauer (mission time)** $MT(r)$ ist die Zeit, bis die Zuverlässigkeit unter ein gegebenes Niveau r sinkt.
- **Instandhaltbarkeit (maintainability)** $M(t)$ ist die Wahrscheinlichkeit, daß ein System in einen funktionfähigen Zustand innerhalb einer vorgegebenen Zeit t zurückkehrt.

Beispiel 1: TMR

Triple Modular Redundancy (TMR) ist eine Standardmethode für Hardware-FT.



Der Voter gibt ein korrektes Ergebnis, wenn er selbst korrekt arbeitet und wenigstens zwei der drei Module korrekt funktionieren.

Beispiel 1: Zuverlässigkeit von TMR

- **Berechnung der Zuverlässigkeit:**

- $R_{TMR} = R_{Voter} \cdot R_{2-aus-3}$

$$R_{TMR} = R_V (R_m^3 + 3R_m^2(1 - R_m))$$

- R_V : Zuverlässigkeit des Voters
 - R_m : Zuverlässigkeit jedes Moduls

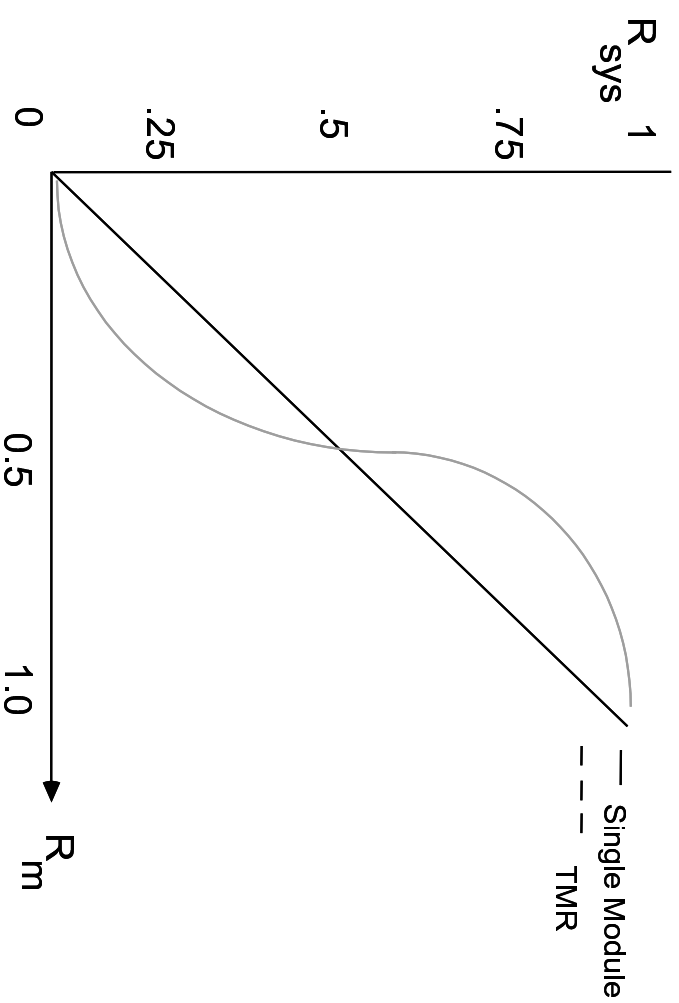
- **Wann lohnt sich TMR?**

- Es muß gelten: $R_{TMR} > R_m$

d.h. $R_V (R_m^3 + 3R_m^2(1 - R_m)) > R_m$

Beispiel 1: Zuverlässigkeit von TMR (II)

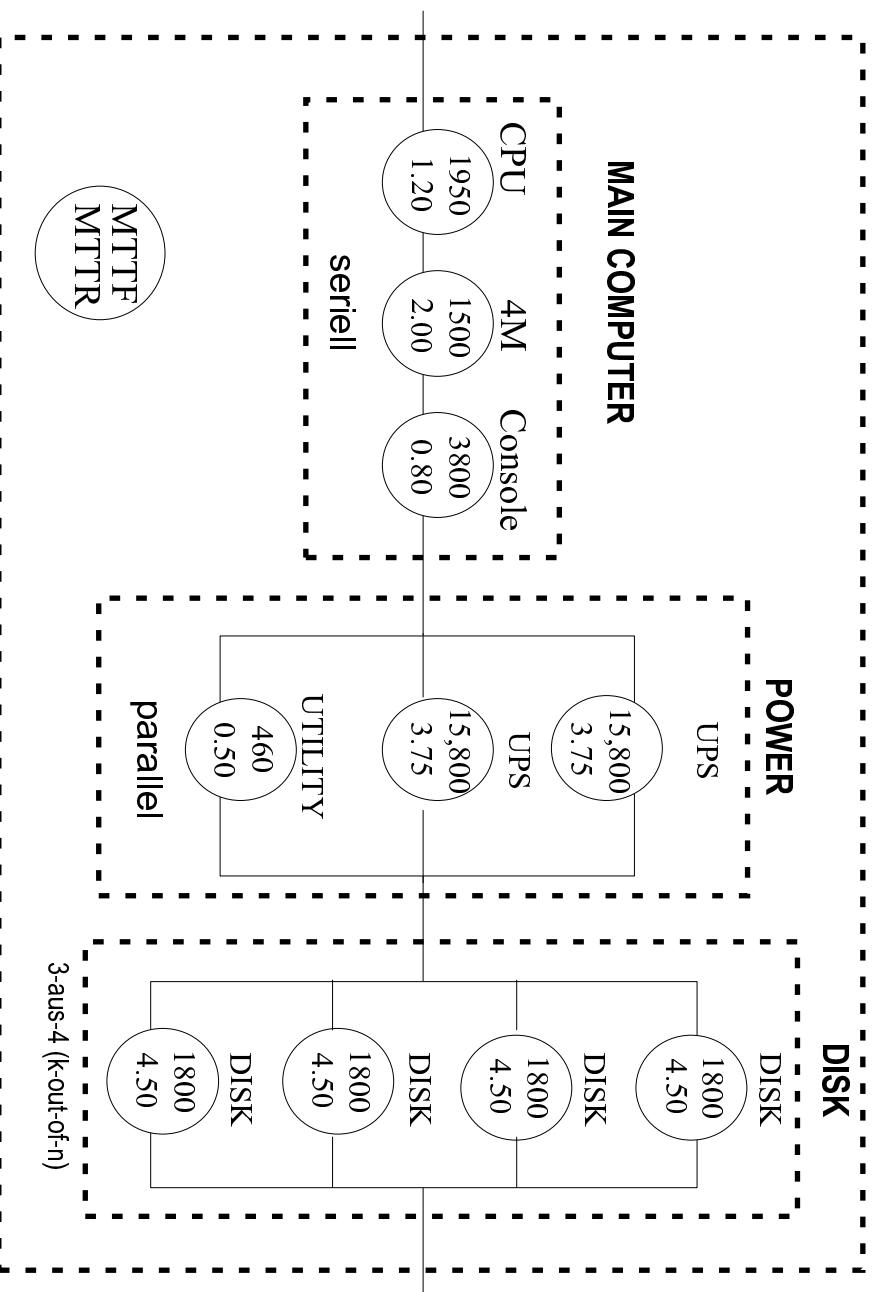
- Annahme: Voter ist perfekt ($R_V = 1$)



- TMR ist nur dann besser, wenn $R_m > 0,5$
- Voter sollte immer sehr zuverlässig sein ($R_m > 0,9$)

Beispiel 2

Berechnung von Gesamt-MTTF, MTTR und Verfügbarkeit.
(Zeitangaben in Stunden)



Beispiel 2: Serienelemente

- Gegeben ist MTTF und MTTR jedes Elements
- Fehlerrate insgesamt: $\lambda_S = \lambda_1 + \lambda_2 + \lambda_3$
- $MTTF_S = \frac{1}{\lambda_S} = \frac{1}{\sum_{i=1}^3 \frac{1}{MTTF_i}}$
- $MTTF_S = \left(\frac{1}{1950h} + \frac{1}{1500h} + \frac{1}{3800h} \right)^{-1}$
- $MTTF_S = 692.3h$

Beispiel 2: Serienelemente (II)

Verfügbarkeit

- $A_1 = \frac{1950h}{1950h+1.2h} = 0.99938499$
- $A_2 = \frac{1500h}{1950h+2.0h} = 0.99866844$
- $A_3 = \frac{3800h}{3800h+0.8h} = 0.99978952$
- $A_S = \prod_{i=1}^3 A_i$
- $A_S = 0.99784418$

MTTR

- $MTTR = \frac{1-A}{A} \cdot MTTF$
- $MTTR = 1.4975h$

MTBF

- $MTBF = MTTF + MTTR$
- $MTBF = 693.2h + 1.5h = 694.7h$

Beispiel 2: Parallelelemente

- $A_1 = A_2 = \frac{MTT F_{1/2}}{MTT F_{1/2} + MTT R_{1/2}}$
- $A_1 = A_2 = \frac{15800h}{15800h + 3.75h} = 0.9997627$
- $A_3 = \frac{460h}{460h + 0.5h} = 0.9989142$
- Parallelschaltung, Nichtverfügbarkeit: $U = \prod_{i=1}^n U_i$
- Verfügbarkeit: $A = 1 - U = \prod_{i=1}^3 1 - A_i$
- $A = 1 - (0.0002373)(0.0002373)(0.0010858)$
- $A = 0.99999999939$

Beispiel 2: Parallelelemente (II)

- MTTR

- $\mu = \mu_1 + \mu_2 + \mu_3$

- $MTTR = \frac{n}{1} \frac{1}{\sum_{i=1}^n \frac{1}{MTTR_i}}$

- $MTTR = \left(\frac{1}{3.75h} + \frac{1}{3.75h} + \frac{1}{0.5h} \right)^{-1}$

- $MTTR = 0.3947368h$

- MTTF

- $MTTF = \left(\frac{A}{1-A} \right) \cdot MTTR$

- $MTTF = \left(\frac{0.999999999939}{1-0.999999999939} \right) \cdot 0.3947368$

- $MTTF = 6,456,492,154h$

Beispiel 2: k -aus- n -Parallelelemente

- Seien n identische Module parallel, alle haben die gleiche MTTF und MTTR
- Lediglich k Module werden für die korrekte Funktion benötigt
- $k = 1$: Berechnung wie Parallelschaltung
- $k = n$: Berechnung wie Serienschaltung
- **Zuverlässigkeit**
- $R = \Pr(\text{System funktioniert für die Zeit } t)$
- $R = \Pr(n \text{ Module funktionieren oder } (n-1) \text{ Module funktionieren oder } \dots \text{ oder } k \text{ Module funktionieren})$
- Beachte: Die einzelnen Bedingungen schließen einander aus \Rightarrow Unabhängigkeit
- $R = \Pr(n \text{ Module funktionieren}) + \Pr(n-1 \text{ Module funktionieren}) + \dots + \Pr(k \text{ Module funktionieren})$

Beispiel 2: k -aus- m -Parallelelemente (II)

Wahrscheinlichkeit, daß x von n Modulen funktionieren:

$$P_{x\text{-von-}n} = \binom{n}{x} R^x (1 - R)^{n-x}$$

$$P_{x\text{-von-}n} = \frac{n!}{(n-x)!x!} R^x (1 - R)^{n-x}$$

Daher ist die Zuverlässigkeit einer k -aus- n -Schaltung (Wahrscheinlichkeit, daß k oder mehr Module funktionieren):

$$R = \sum_{x=k}^n \frac{n!}{(n-x)!x!} R^x (1 - R_m)^{n-x}$$

Für 3 von 4 und $R_m = 0.9$:

$$R = R_m^4 + \binom{4}{3} R_m^3 (1 - R_m) = 0.9^4 + 4 \cdot 0.9^3 \cdot 0.1 = 0.9477$$

Beispiel 2: k -aus- m -Parallelelemente (III)

- MTTR

$$MTTR = \frac{MTTR_M}{n-k+1}$$

- MTTF

$$MTTF = MTTF_m \left(\frac{MTTF_m}{MTTR_m} \right)^{n-k} \left(k \cdot \binom{n}{k} \right)^{-1}$$

$$MTTF = MTTF_m \left(\frac{MTTF_m}{MTTR_m} \right)^{n-k} \left(\frac{(n-k)!(k-1)!}{n!} \right)$$

- Verfügbarkeit

$$A = \frac{MTTF}{MTTF+MTTR}$$

Beispiel 2: k -aus- m -Parallelelemente (IV)

- Im gegebenen Beispiel
 - $n = 4$
 - $k = 3$
 - $MTTF_m = 1800h$
 - $MTTR_m = 4.5h$
- $MTTR = \frac{4.5h}{4-3+1} = 2.25h$
- $MTTF = 1800h \left(\frac{1800h}{4.5h} \right)^{n-k} \left(\frac{(4-3)!(3-1)!}{4!} \right) = 60,000h$
- Verfügbarkeit
$$A = \frac{1800h}{1800h+4.5h} = 0.9975062$$

Beispiel 2: Gesamtsystem

- Gesamtsystem ist eine Serienschaltung aus
 - Rechner
 - Stromversorgung
 - Disk-System

- MTTTF

$$MTTTF = \left(\sum_{i=1}^3 \frac{1}{MTTTF_i} \right)^{-1}$$

$$MTTTF = \left(\frac{1}{692.3h} + \frac{1}{6.46 \cdot 10^7 h} + \frac{1}{6.0 \cdot 10^4 h} \right)^{-1}$$

$$MTTTF = 685.25h$$

Beispiel 2: Gesamtsystem (II)

- Verfügbarkeit

$$A = \prod_{i=1}^3 A_i$$

$$A = 0.9978$$

- MTTR

$$MTTR = \left(\frac{1-A}{A}\right) MTTF$$

$$MTTR = 1.51h$$

- MTBF

$$MTBF = MTTF + MTTR$$

$$MTBF = 676.76$$

Beispiel 2: Gesamtsystem (III)

- Wahrscheinlichkeit, daß das System eine Woche fehlerlos läuft.
- 1 Woche = $7d \cdot 24\frac{h}{d} = 168h$
- $R(168h) = e^{-\frac{168h}{685.25h}}$
- $R(168h) = 0.78257 \approx 78.3\%$
- In einem Pool mit 5 Rechnern dieser Art wird im Mittel nach einer Woche einer einen Fehler gehabt haben

Beispiel 2: Gesamtsystem (IV)

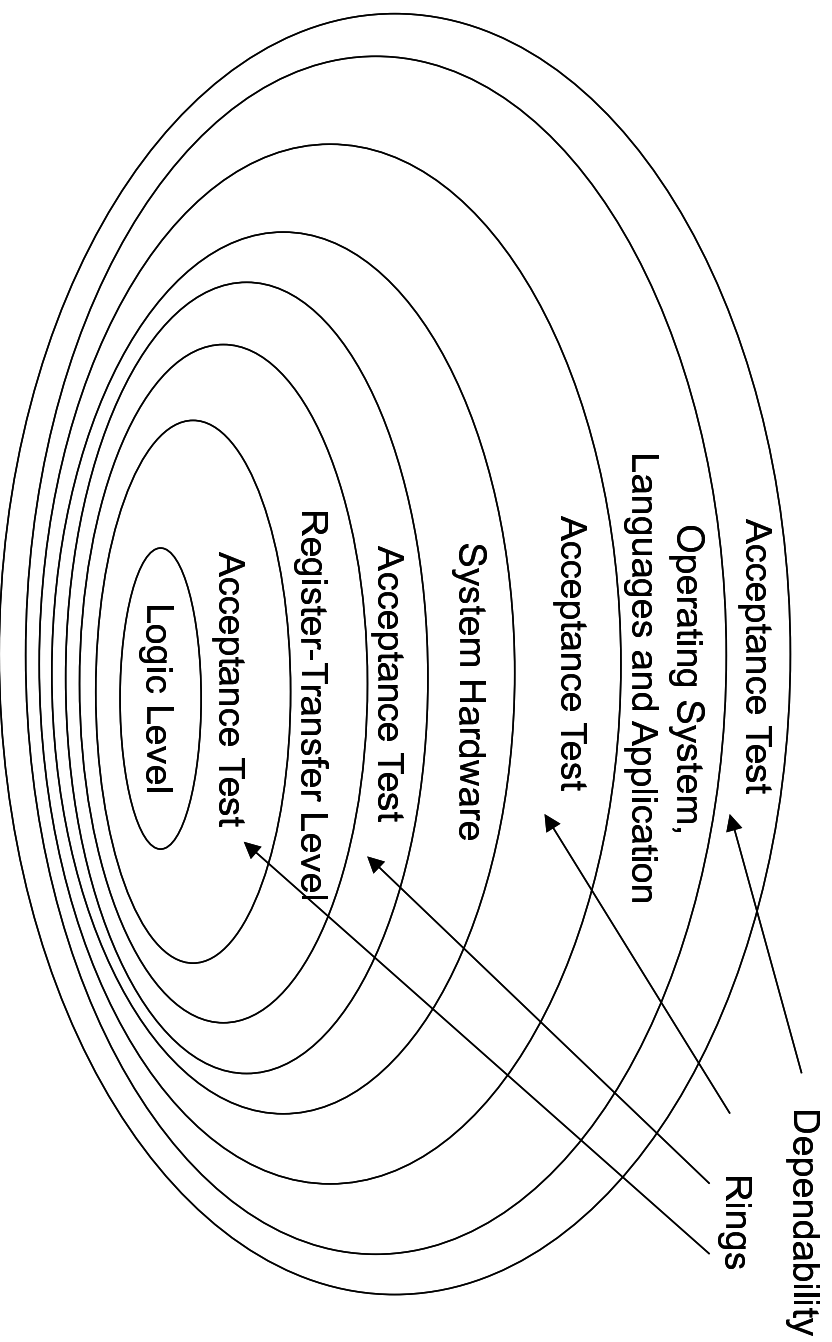
- Wie lange darf ein Experiment auf einem Rechner laufen, wenn maximal eines von zehn Experimenten von einem Fehler betroffen sein soll?
- $R(MT) > 0.9$
- $R(MT) > e^{-\frac{MT}{685.25h}} > 0.9$
- $-\frac{MT}{685.25h} > -0.1053$
- $MT(0.9) < 72.2h$

Fehlertoleranter Systementwurf: Strategien

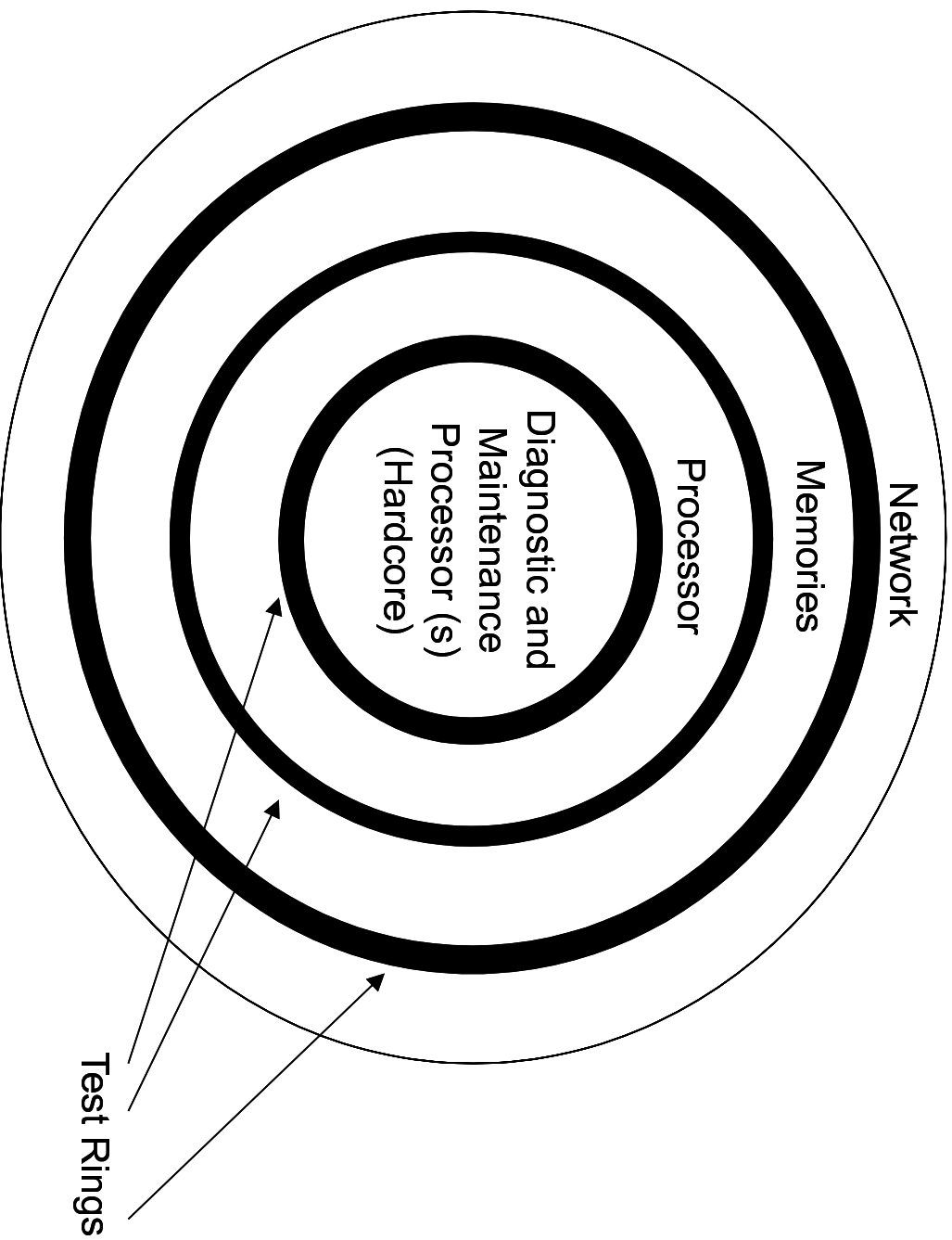
- **Eindämmung** (*containment*). Verhinderung, daß Fehler sich über gewisse Grenzen hinaus verbreiten
- **Diagnose** (*diagnosis*). Identifizierung des Modules, das für einen Fehler verantwortlich ist
- **Maskierung** (*masking*). Entstandene Fehler werden dynamisch korrigiert
- **Reparatur/Rekonfiguration** (*repair, reconfiguration*). Ersetzen, entfernen oder umgehen eines fehlerhaften Modules
- **Recovery** (*Wiederaufsetzen*). Hinführen eines Systems (nach einem Fehler) in einen Zustand, der für das weitere Arbeiten akzeptabel ist.

Eindämmung

- Zuverlässigkeitsringe (*dependability rings*)



Dependability rings für Standardssystem



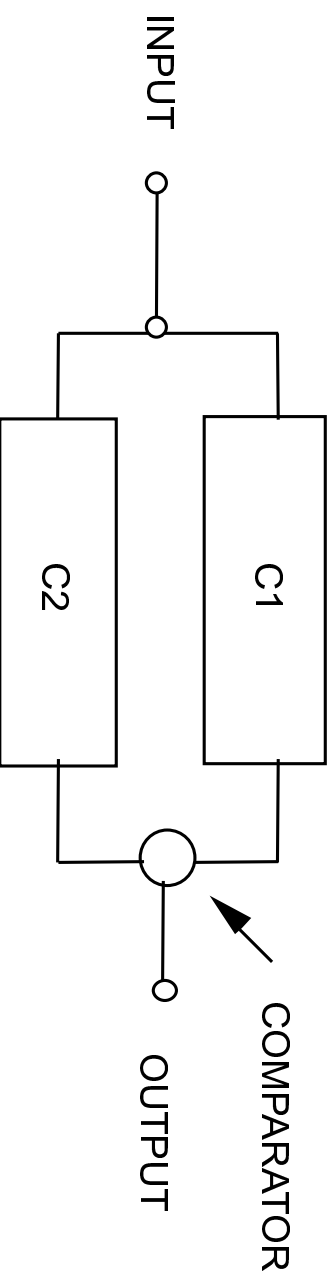
Fehlerdiagnose

Fehlerdiagnose = Fehlererkennung (*detection*) + Fehlerortung (*location*)

- Fehlererkennung durch
 - Replication checks
 - Timing checks
 - Reversal checks
 - Coding checks
 - Reasonableness checks
 - Structural checks
 - Diagnostic checks
 - Algorithmic checks

Replication Checks

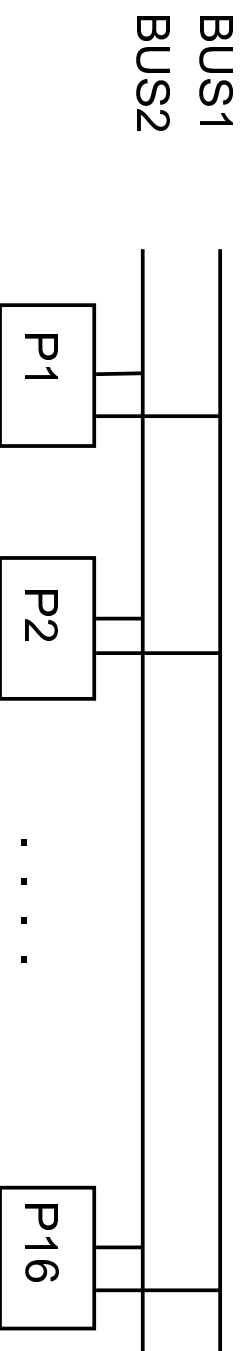
- Alles wird mehrmals gemacht
- Gründlich, aber teuer
- Varianten
 - Identische Replikation
 - Verschiedene Designs
 - Wiederholte Ausführung
 - Vergleich mit Standardausführung



Timing Checks

- Überprüfung, ob Zeitbedingungen eingehalten werden
- Varianten
 - Extra Zeitüberwachungseinheit (Watchdog)
 - Passive gegenseitige Überwachung
 - Aktive gegenseitige Überwachung

Beispiel Tandem-Computer: „I‘am alive“ im Sekundentakt, „Are you okay?“
zweiseitig



Reversal Checks

- Ausgaben hängen (in der Regel) deterministisch von Eingaben ab
- Berechnung der Eingaben aus erhaltenen Ausgaben
- Vergleich gegen Eingabe
- Beispiele
 - Wiedereinlesen nach Schreiben
 - Mathematische Funktionen, wie
 - * $(\sqrt{x})^2 \stackrel{!}{=} x$
 - * $\mathbf{A} \cdot \mathbf{A}^{-1} \stackrel{!}{=} \mathbf{I}$

Coding Checks

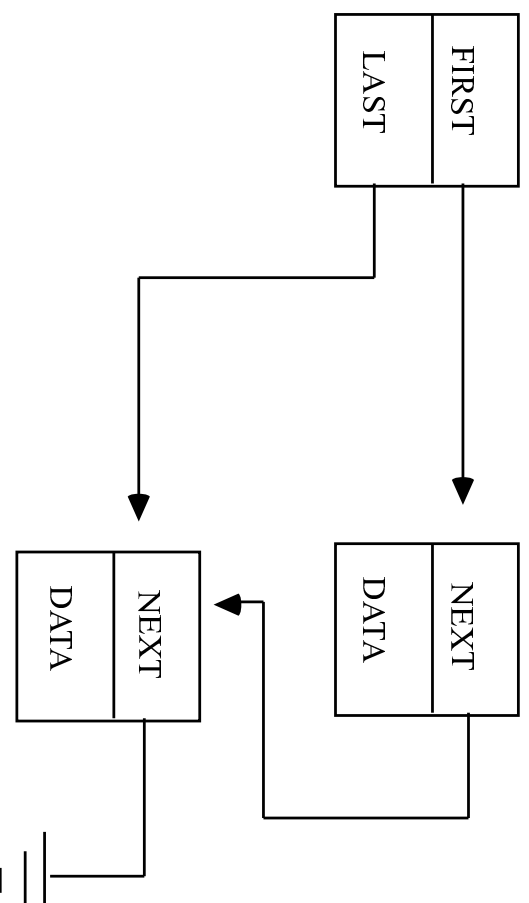
- Redundante Datendarstellung
- Beispiele
 - Paritätsbit
 - Berger-Code (Anzahlen von 1 oder 0)
 - Checksumme
 - Hamming-Code
 - Cyclic Redundancy Check

Reasonableness Checks

- Nutzung des gesunden Menschenverstands (Plausibilitätschecks)
- Ausnutzung von Wissen über das interne Design und Strukturen
- Beispiele
 - Bereichschecks (z.B. $0^\circ \leq \alpha < 360^\circ$, Arrayindex im vorgegeben Bereich)
 - Konsistenzüberprüfungen (z.B. Würde jede Eingabe verarbeitet?)
 - Typenchecks (Ist Ergebnis Integer?)
 - :

Structural Checks

- Überprüfung der Konsistenz von Datenstrukturen oder Systemstrukturen
- Beispiele
 - Anzahl der Elemente
 - Redundante Pointer
LIST HEAD
- Liste der PnP-Geräte



Diagnostic Checks

- Überprüfung, ob für eine bekannte Menge von Eingaben die korrekten Ausgaben erfolgen
- Typischer Einsatz in Hardwaretestprogrammen (z.B. BIOS)
- Beispiele
 - Speichertests
 - Exceptiontests
 - Belastungstests

Algorithmic Checks

- Überprüfen, ob Invarianten eines Algorithmus unberührt bleiben
- Beispiele
 - Sortierung: Anzahl der Einträge, Checksumme
 - Checksumme bei Matrizenmultiplikation

$$\begin{bmatrix} 1 & 2 \\ 3 & 5 \\ 4 & 7 \end{bmatrix} \times \begin{bmatrix} 2 & 4 & \dots & 6 \\ 1 & 3 & \dots & 4 \end{bmatrix} = \begin{bmatrix} 4 & 10 & \dots & 14 \\ 11 & 27 & \dots & 38 \\ 15 & 37 & \dots & 52 \end{bmatrix}$$

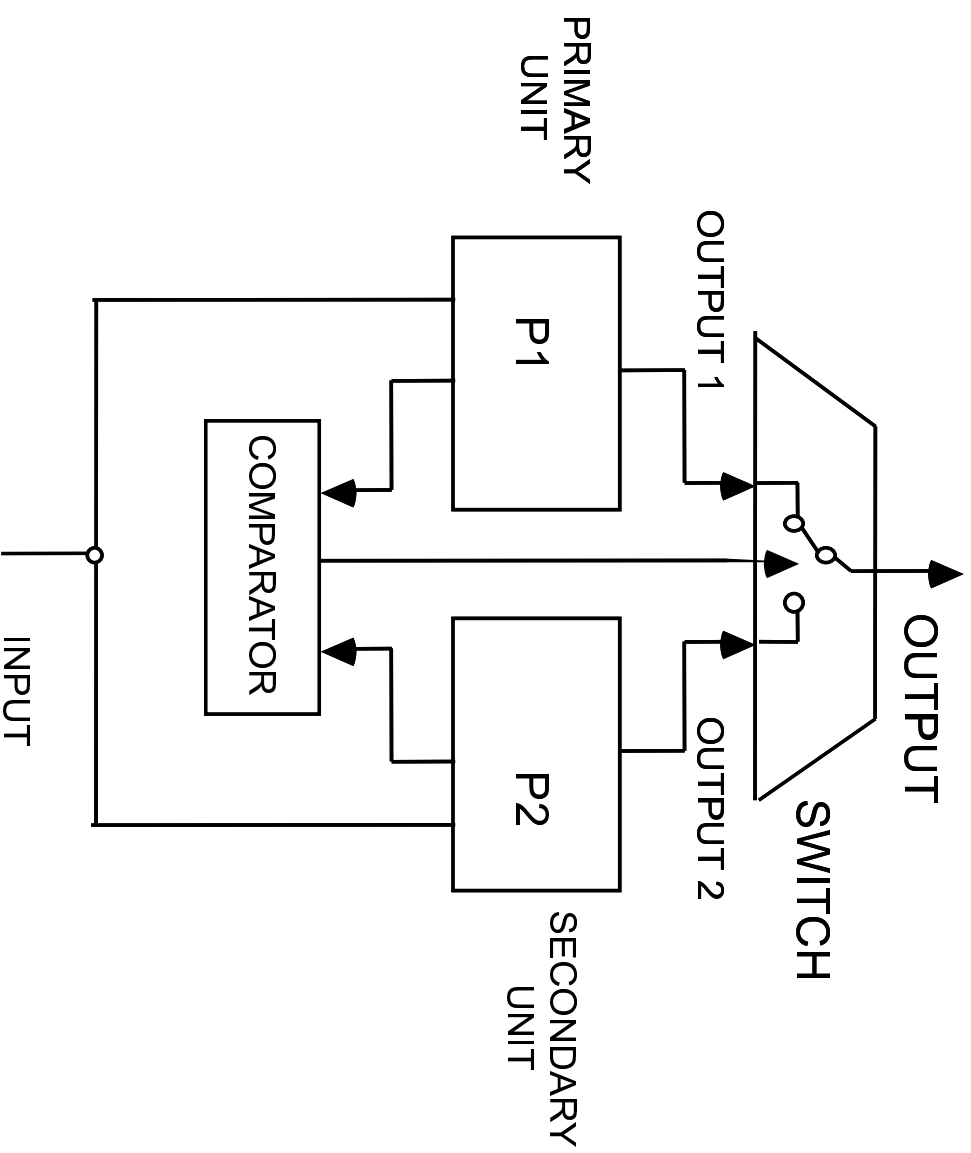
Fehlermaskierung

- Bei der Fehlermaskierung kommt es darauf an, daß ein Fehler nicht weitergegeben wird.
- Beispiele
 - Mittelwertbildung (nicht nur arithmetisch!)
 - Voting
 - Diverse Checks + Retry
 - Bei der Software-FT: n Version Programming (NVP)
- Achtung: In zeitabhängigen Systemen auf den *Determinismus der Replika* zu achten.

Recovery/Reconfiguration in Hardware

- Nutzung redundanter (in der Regel identischer) Module
- Redundanz kann sowohl zur Fehlererkennung als auch zur Fehlerbehebung (Recovery) genutzt werden
- Teuer, aber z. T. unverzichtbar
- Hochverfügbare Systeme scheitern selten an der HW: Verfügbarkeiten von 0.99 bis 0.999999999999

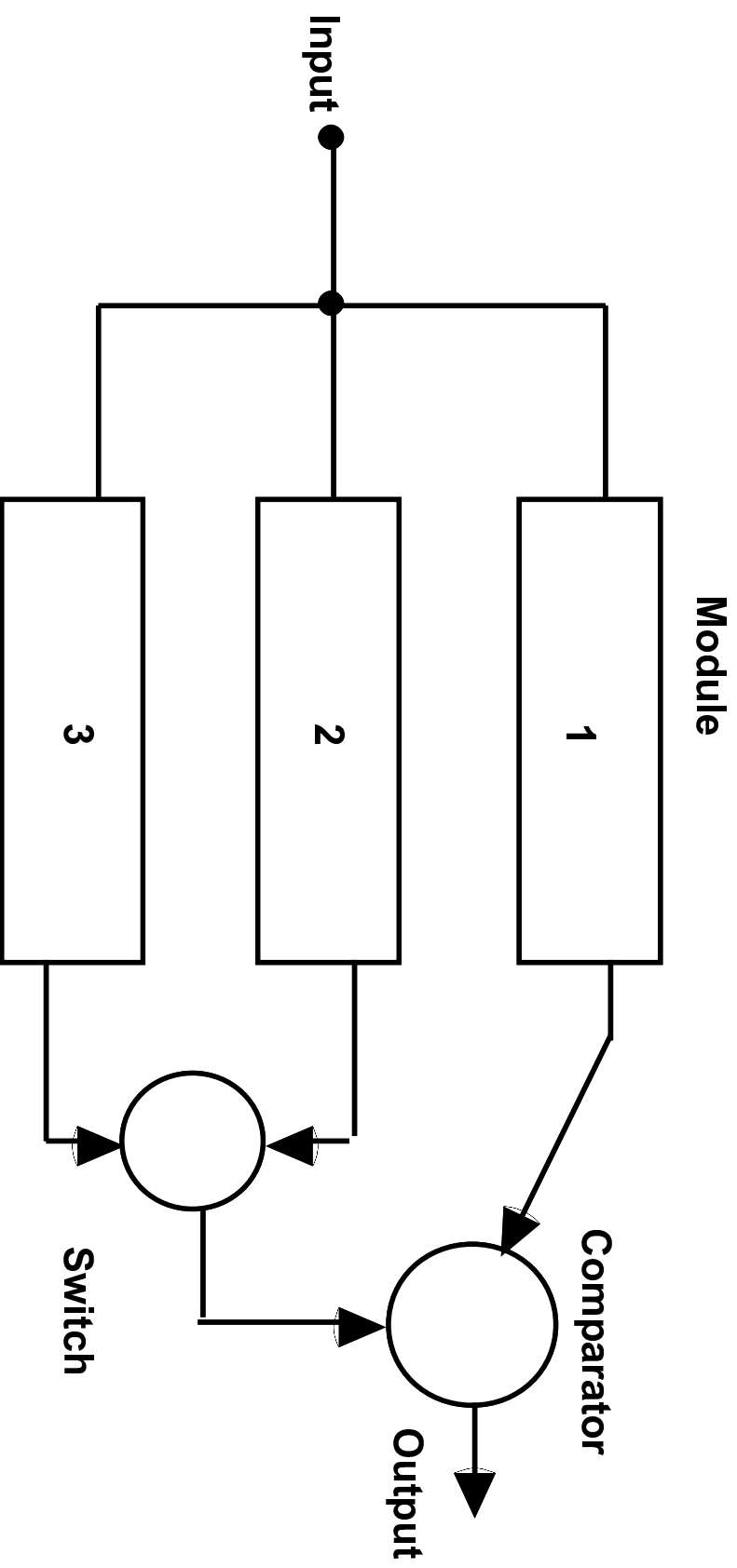
Duplex-System



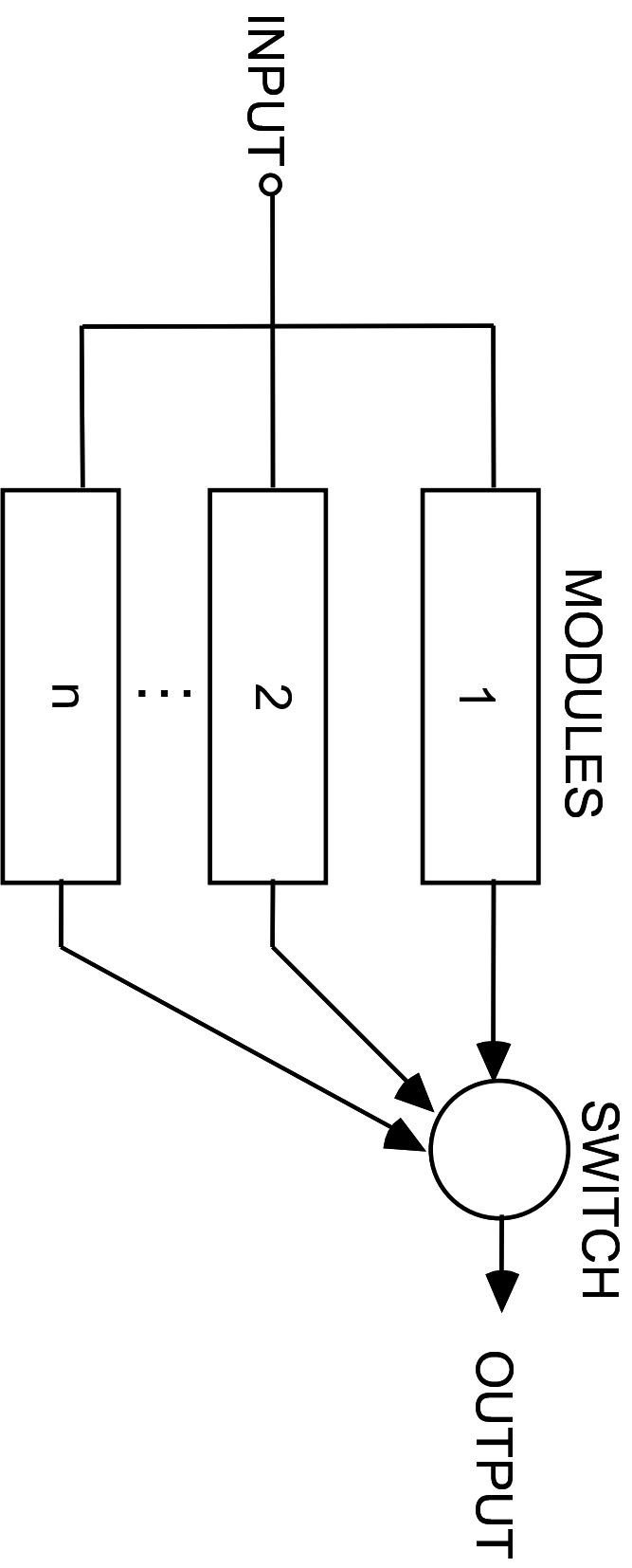
Duplex-System (II)

- Wenn eine Abweichung zwischen beiden Modulen auftritt, kann das fehlerhafte Modul mit einer Diagnosemethode bestimmt werden, z.B.:
 - Start eine Eigendiagnoseprogramms
 - Watchdog-Timer
 - Diagnose durch einen externen Arbiter
- Zuverlässigkeit
$$R = (R_m^2 + 2CR_m(1 - R_m))R_k$$
 - R_m : Zuverlässigkeit eines Modules
 - R_k : Zuverlässigkeit der Steuerung, des Comparators, ...
 - C : Coverage factor, steht für die kombinierte Wahrscheinlichkeit einer erfolgreichen Fehlererkennung und Neukonfiguration

Duplex mit Stand-by-Modul



Mehrfache Standby-Module

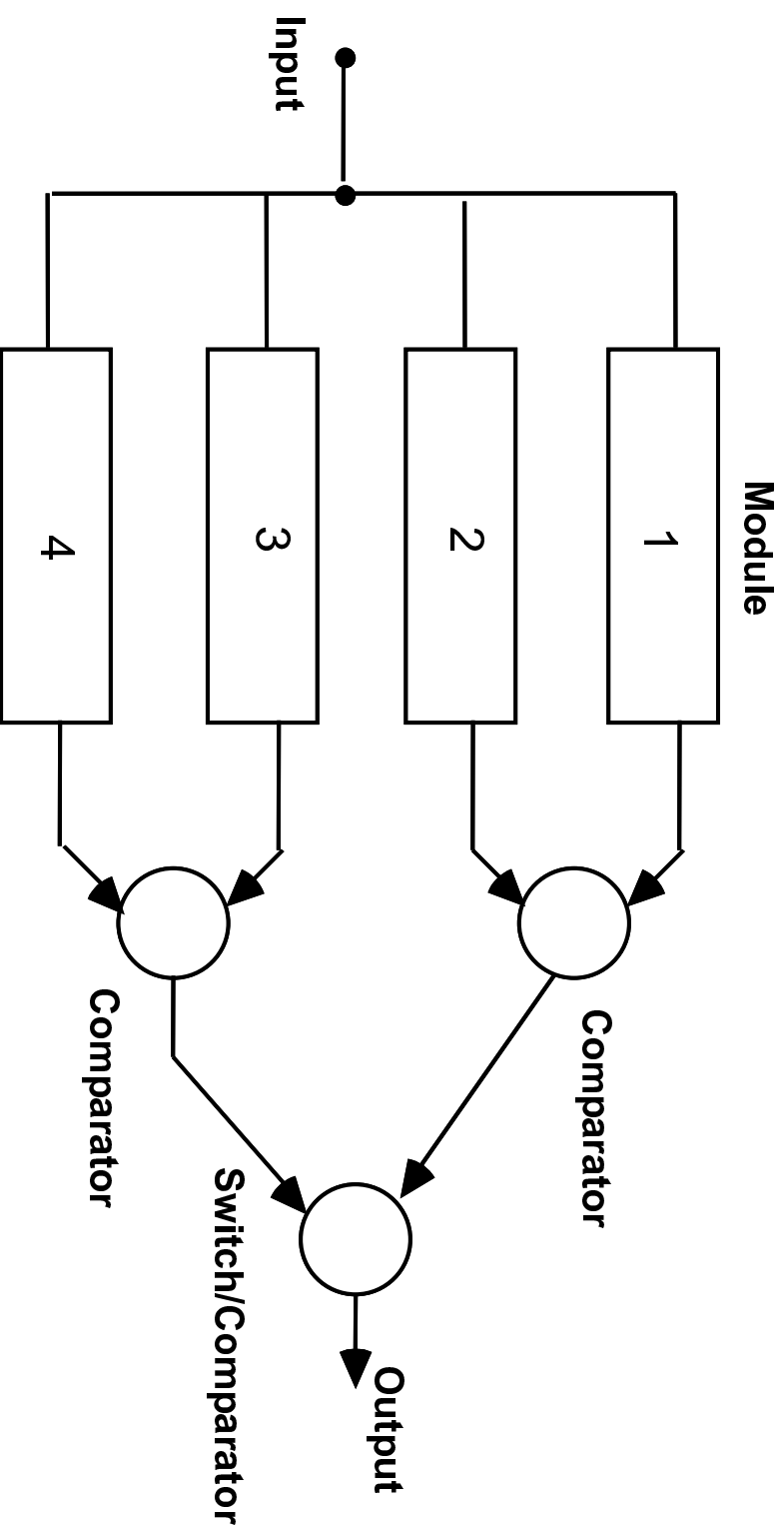


Standby Module

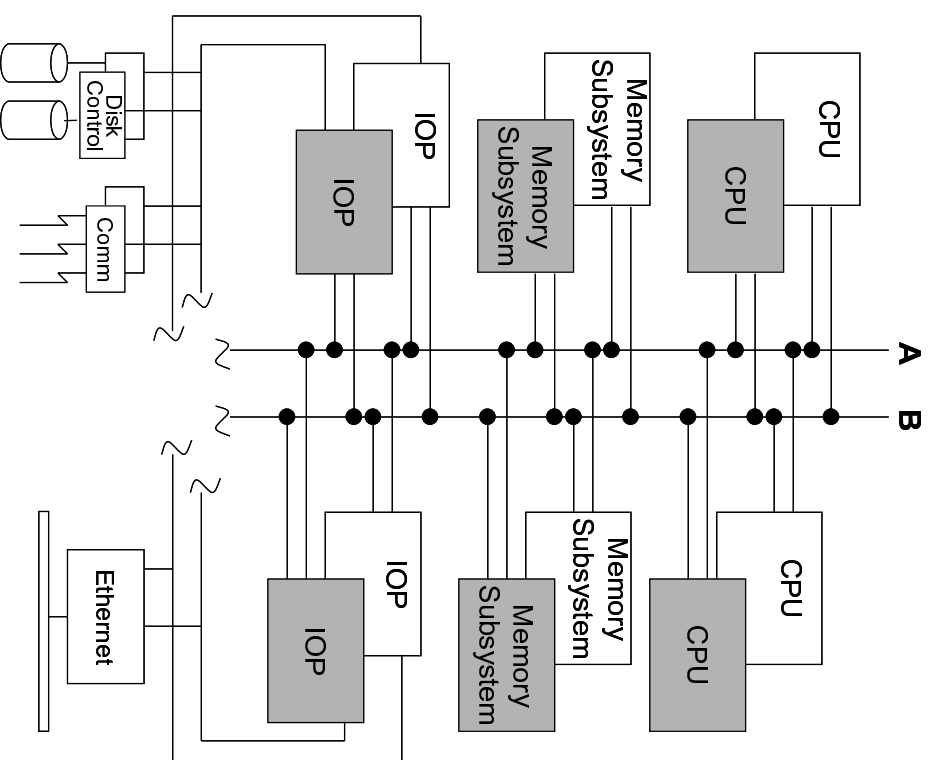
Es werden verschiedene Arten der Redundanz unterschieden:

- **Hot Standby:** Alle Eingaben werden bei der redundanten Einheit zeitgleich gelesen und verarbeitet
- **Warm Standby:** Eingaben werden (zeitversetzt) mitgeführt. Bei Übernahme durch die redundante Einheit muß diese in den entsprechenden Zustand gebracht werden
- **Cold Standby:** Redundante Einheit ist passiv/aus bis zur Übernahme
- Keine klare Unterscheidung zwischen „warm“ und „cold“, daher wird ersteres in der Literatur mitunter weggelassen.

Pair und Spare

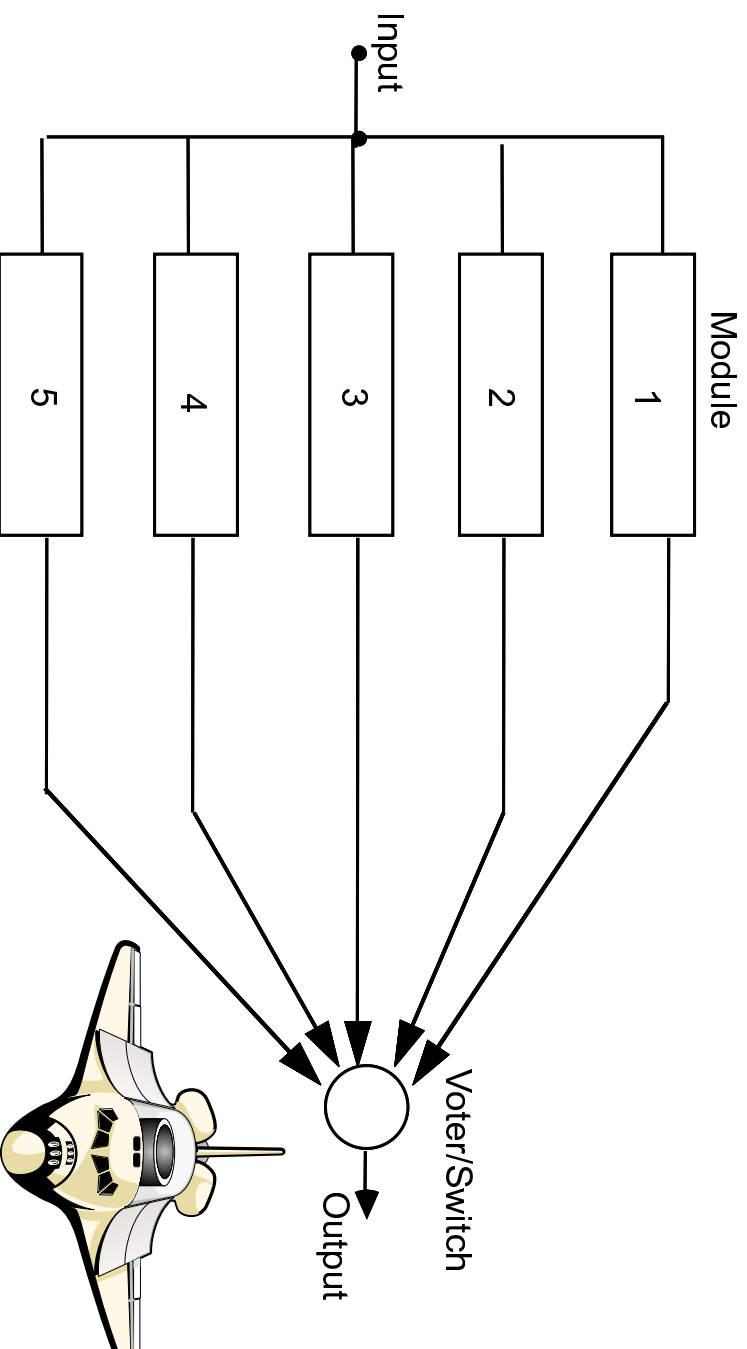


Beispiele für Hardwareredundanz



- Stratus XA/R Serie 300
- Prinzip: „Pair and Spare“
- Alternative Module arbeiten gleichzeitig über zwei Busse

Beispiele für Hardwareredundanz (II)



- Space Shuttle: TMR mit zwei Standby-Modulen
- Aktive Module: 1,2 und 3
- Modul 4: warm standby, Modul 5: cold standby