Verlässliche Systeme

4. Kapitel
Störungen

Prof. Matthias Werner
Professur Betriebssysteme

4.1 Störungsbegriff

Fehler

- **Fehler** (allgemein): Abweichung von der Spezifikation (Erwartung)
- Modern wird von **Störung** (*impairment*) gesprochen: ganzheitliche Sicht, die auch Last, Schwachstellen, etc. berücksichtigt.

Definitionen nach LAPRIE [1]

- **Ausfall:** (*failure*) Entsteht, wenn der erbrachte Dienst vom erwarteten abweicht. Ausfälle werden durch Fehler verursacht.
- **Fehler(zustand):** (*error*) Abweichung eines Systemzustandes vom korrekten Fall; Abweichung von erwarteten Berechnungsergebnis (falsches Ergebnis).
- **Fehler(ursache):** (*fault*) tatsächliche oder angenommene Ursache eines Errors

[1] Beachten Sie bitte, dass in der deutschen (Umgangs-)Sprache alle Konzepte mit „Fehler“ bezeichnet werden.

Charakterisierung von Fehlern

Große Vielfalt bei Typen und Ursachen

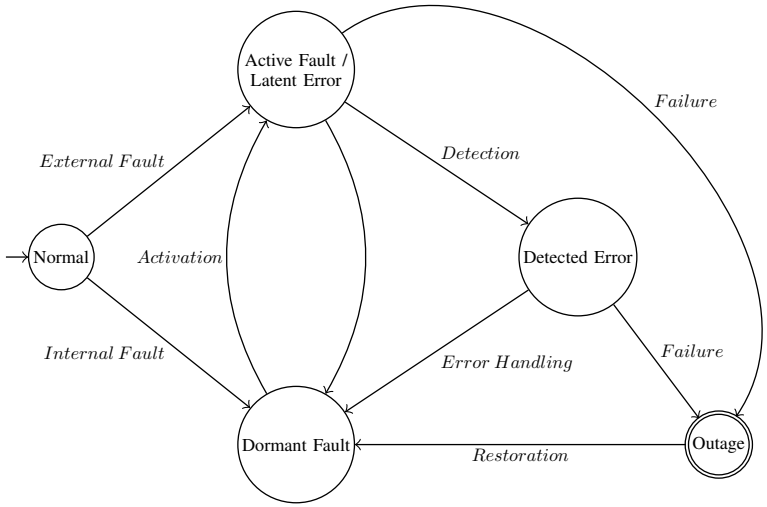
- **Fehlerart:**
  - **Zufallsfehler** (*accidental faults*) vs. **Absichtsfehler** (*intentional faults*)
- **Betrachtungen zur Quelle**
  - Phänomenologische Ursachen: **physische** Fehler vs. **menschlicher** Fehler
  - Systemgrenze: **interne** Fehler vs. **externe** Fehler
  - Zeit der Entstehung: **Entwurfsfehler** (*design faults*) vs. **Betriebssfehler** (**operational faults**)
- **Dauerhaftigkeit:**
  - **Permanente** Fehler vs. **temporäre** Fehler

Zeitliche Charakterisierung

- Eine Fehlerursache ist **aktiv**, wenn sie einen Fehlerzustand erzeugt
- Ein nicht aktiver Fehler ist ein **latenter** oder **inaktiver** Fehler
  ➡ häufig Wechsel zwischen aktiver und latenter Fehlerursache
- Zeitweilige externe zufällige Fehler werden auch **transiente Fehler** genannt
- Zeitweilige interne zufällige Fehler werden auch **intermittierende Fehler** genannt
- **Beispiele**:
  - Speicherfehler bei bestimmten Mustern, Systemüberlastung
  - Beliebige konzeptabhängige Fehler mit unbekannter Aktivierung

Fehlerautomat

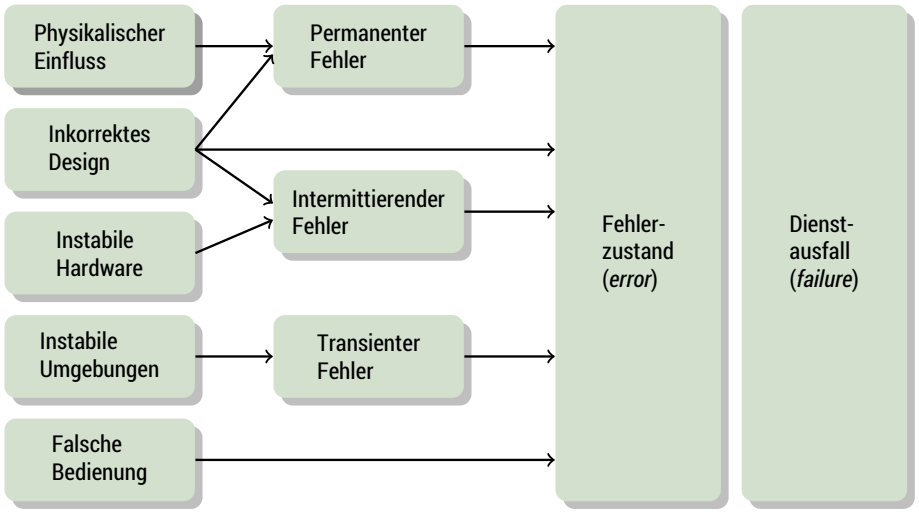
- Die zeitlichen Verhältnisse können mit Hilfe eines Fehlerautomaten dargestellt werden



Diskussion

- Ein externer Fehler kann ein Designfehler sein – nicht alle Umstände wurden berücksichtigt
  - **Beispiel**: Performance- oder Zeitfehler ➡ Abweichung von erwarteter Last oder Zeit
- Designfehler werden während Systementwicklung, -veränderung oder der Geschäftsprozessentwicklung und -einführung erzeugt
- Physische Fehler sind **zufällige Fehler**
- Beabsichtigte Fehler und Designfehler sind menschliche Fehler
- Viele Spezialbezeichnungen, z.B. **Bug**
  - **Heisenbug** – intermittierender Softwarefehler
  - **Bohrbug** – Permanenter Softwarefehler
  - **Mandelbugs** – erscheint chaotisch aufgrund vieler Abhängigkeiten

Quelle von Fehlern



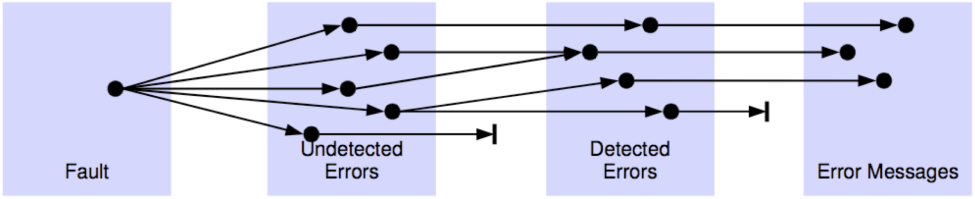
(nach Siewiorek/Swarz)

Fehlerzustände (Errors)

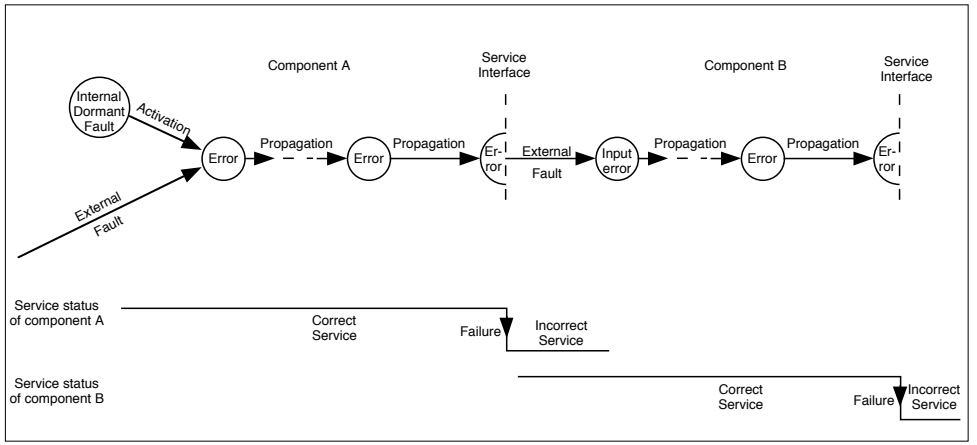
- Systemzustand, kein Ereignis
- Wird zum Ausfall abhängig von
  - Redundanz (beabsichtigter oder unbeabsichtigter)
  - Systemaktivität
  - Nutzerdefinition eines Ausfalls
    Beispiel: maximale Unterbrechungszeit, akzeptierte Verzögerung, Retransitionsrate
  - Latent (nicht erkannt) vs. erkannt

Fehlermeldungen (Errormessages) (HANSEN & SIEWIOREK)

- Gleiche Fehlerursache kann zu verschiedenen Fehlerzuständen führen
- Erkannte Fehler werden nicht unbedingt protokolliert



Fortpflanzung



- **Quelle:** [Avi+04]

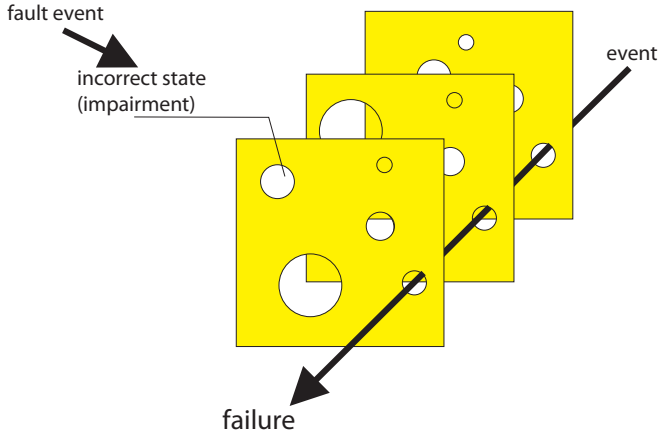
Fault-Failure-Modell

Alternative Sicht: KOPETZ

Es werden keine Zustände, sondern nur **Ereignisse** betrachtet.

- **fault**: Fehlerereignis
- **failure**: Ausfallereignis, kann gleichzeitig der *fault* der nächsthöheren Abstraktionsebene sein

Multiple Causes (D. ORLANDELLA and J. REASON)



- Argument: Mitunter kann nicht **die** Fehlerursache festgestellt werden, sondern Zusammenwirken mehrerer Ursachen
- Beschreibung von Störpotentialen ➡ **Schwächen**)
- Schweizer-Käse-Modell

4.2 Störungssemantik

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 Systemsmodell und Fehlerspezifikation ➡ **Fehlermodell**
- Zwei Aspekte:
  - Fehlersemantik
  - Auftreten von Fehlern
- Beginnen mit ersterem

Modell und Wirklichkeit

Ein Modell ist immer eine Abstraktion und nur bedingt aussagekräftig.

Abdeckung

Unter **Abdeckung** (*coverage*) eines Fehlermodells versteht man die (ggf. gewichtete) Anzahl von Fehlern, die im Modell beschreibbar sind, zu der (ggf. gewichtete) Anzahl von tatsächlich auftretenden Fehlern

- Wesentlich zur Beurteilung von Tests
- Einem Test liegt (mindestens **implizit**) stets ein bestimmtes Fehlermodell zugrunde
- Meist wird nur ein Fehlermodell gegen ein anderes, umfassenderes bewertet
- Die tatsächliche Abdeckung ist nur empirisch bestimmbar

Fehlermodelle

Fehlermodelle können auf verschiedenen Abstraktionsebenen beschrieben werden, z.B.:

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

Wir werden einige Modelle auf der Schaltereben und der Systemebene näher betrachten.

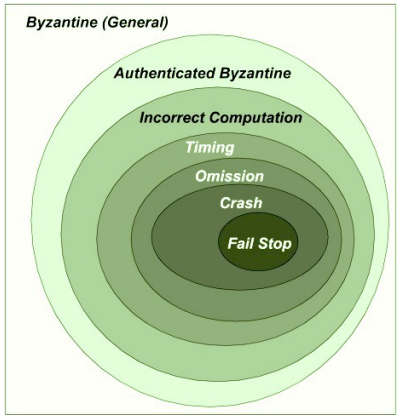
Fehlermodelle auf Schalterebene

- Fehlermodelle auf Schalterebene werden vor allen in zwei Bereichen eingesetzt:
  - Schaltungs- und Schaltkreisentwurf (für uns nicht so interessant)
  - Kommunikation (auch in allgemeinerem Sinn, z.B. persistente Daten)
- Es werden logische Signale auf Leitungen betrachtet
  - **Stuck-at-X-Fehler**
    ➡ Signal ist immer X ($X \in \{0, 1, valid, \ldots\}$)
  - **Bridging-Fehler** (Kurzschlüsse)
    ➡ dominantes Signal setzt sich durch
  - **Stuck-at-open-Fehler**
    ➡ unbestimmtes Signal

Fehlermodelle für verteilte Systeme

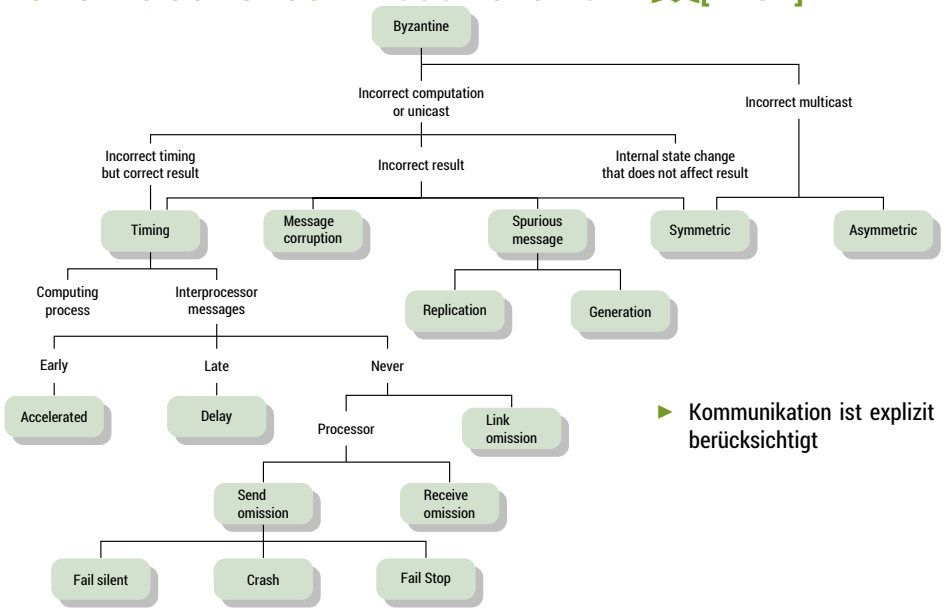
- Es wird nur das **Verhalten** am Interface eines Systems/Systemkomponente betrachtet
- Verhalten wird durch (Sequenz von) **Serviceelementen** beschrieben: Tuple von Wert und Zeit
- Modell von MALEK et al. (📖[LMJ91]):
  - Angekündigter Ausfall (*fail stop*)
  - Ausfallfehler (*crash*)
  - Auslassungsfehler (*omission*)
  - Zeitfehler (*timing*)
  - Wertefehler (*incorrect computation/communication*)
  - Beliebiger Fehler (*arbitrary/ Byzantine*)[1]

  Link-Fehler werden auf Komponentenfehler (beim Sender oder Empfänger) abgebildet.



[1] Beim *authenticated Byzantine fault* wird eingeschränkt, dass kein Knoten eine Nachricht eines anderen unbemerkt ändern kann.

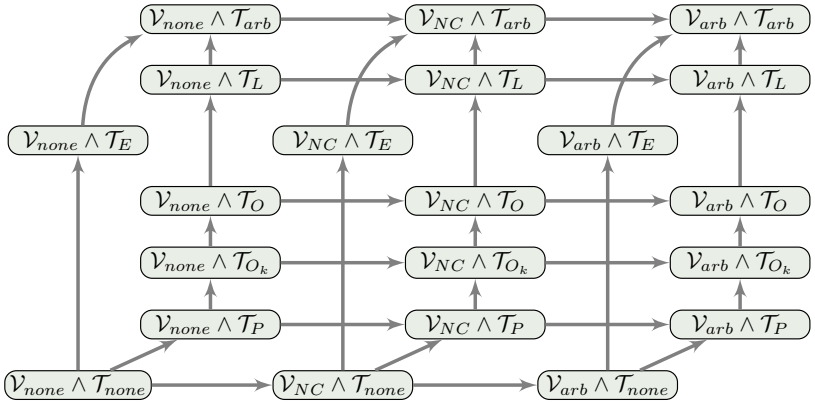
Fehlerhierachie nach BLOUGH und TORII 📖[BT97]



- Kommunikation ist explizit berücksichtigt

Halbordnung der Systemfehler (📖[Pow95])

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



$none$ – kein; P – permanent; O_k – Sequenz Auslassung; O – Auslassung; L – zu spät; E – zu früh; NC – nicht codebrechend; arb – beliebig

- Es reicht meist nicht aus die **Fehlerwirkungen** zu kennen
- Aussagen über das **Auftreten** notwendig
  - Wie viele Fehler gleichzeitig (maximal)?
  - Wie häufig?
  - Unabhängigkeit von Fehlern
- Betrachten **Unabhängigkeit** von Fehlern
  - Treten Fehler unabhängig voneinander auf?
  - Sind Fehler korreliert? Wie?
  - Sind Fehler intendiert (Angriff)?

4.3 Fehlerauftreten

Statistische Annahmen

- Wie bereits betrachtet: statistische Annahmen häufig über Mittelwerte beschrieben
  - Mittlere Zeit bis zu einem Ausfall (MTTF, vgl. Kapitel 3):

Mean time to failure

Es sei $f(t)$ die Dichte der Wahrscheinlichkeit, dass bis $t_0 + t$ **kein Fehler** auftritt.

$$\bar{t}_F = \int_0^\infty t \cdot f(t) dt$$

  - Für exponentiell verteilte Ausfälle: $\bar{t} = MTTF = \frac{1}{\lambda}$

Fehlerereignisse als stochastischer Prozess

- Ankunft von Fehlern (oder auch Last/Requests) wird häufig als **stochastischer Prozess** beschrieben
- **Wiederholung:**

Definition 4.1 (Stochastischer Prozess)

*Ein **stochastischer Prozess** oder zufälliger Prozess ist eine Menge $\{X(t)\}$ von Zufallsvariablen, die einen gemeinsamem Wertebereich (Zustandsraum) und einen gemeinsamen Parameter t als Index besitzen.*

- Jede Indexinstanz $X(t)$ ist für ein festes t eine Zufallsvariable
- Typischerweise wird t als Zeit interpretiert (insbesondere bei Ankunftsprozessen)

Poisson-Prozess

- Zählen Ereignisse eines diskreten Prozesses bis $t_0 + t$
- Anzahl sei $N(t)$
- Es soll gelten:
  - $N(t_0) = 0$
  - Unabhängigkeit in sich nicht überlappenden Intervallen
  - Wahrscheinlichkeit eines Ereignisses hängt nur von t ab, nicht von t_0.
  - $\lim_{t \to 0} \Pr\{\text{Ereignis in } t\} \sim t$

Dann spricht man von einem **Poisson-Prozess** und für $N(t)$ gilt:

$$\Pr\{N(t) = m\} = \frac{(a \cdot t)^m}{m!} e^{-a \cdot t}$$

Poisson-Prozess (Forts.)

- $N(t)$ gibt Anzahl von auftretenden Fehlern im Intervall der Länge t
- Speziallfall: Zeit bis zum **ersten** Fehler

$$\begin{aligned} \Pr\{N(t) = m = 0\} &= \frac{(a \cdot t)^m}{m!} e^{-a \cdot t} \\ &= \frac{(a \cdot t)^0}{0!} e^{-a \cdot t} \\ &= e^{-a \cdot t} \end{aligned}$$

➡ entspricht bekannter Überlebenswahrscheinlichkeit für konstante Fehlerraten (Exponentialverteilung)

Diskussion: Fehler und Last

Wechelwirkung zwischen Last und klassischen Fehlern

- Meist positiv korreliert:
  - Steigende Last belastet System (Wearout)
    ➡ Ausfallrate steigt an
  - Steigende Last erhöht die Wahrscheinlichkeit der Wirksamkeit eines Fehlerzustandes
    ➡ Ausfallrate steigt an
  - (Erkannte) Fehler führen zu Recovery-Massnahmen
    ➡ Last steigt
- Rückkopplungeffekt möglich
- Mitunter **senkt** Last auch Ausfallrate

Referenzen

[LMJ91] Luiz A. Laranjeira, Miroslaw Malek und Roy Jenevein. „On tolerating faults in naturally redundant algorithms“. In: *Reliable Distributed Systems, 1991. Proceedings., Tenth Symposium on*. Sep. 1991, S. 118–127

[Avi+04] Algirdas Avizienis u. a. „Basic Concepts and Taxonomy of Dependable and Secure Computing“. In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), S. 1–23

[BT97] Douglas. M. Blough und Tatsuhiro. Torii. „Fault-injection-based testing of fault-tolerant algorithms in message-passing parallel computers“. In: *Fault-Tolerant Computing, 1997. FTCS-27. Digest of Papers., Twenty-Seventh Annual International Symposium on*. Juni 1997, S. 258–267

[Pow95] David Powell. „Failure Mode Assumptions and Assumption Coverage“. In: *Predictably Dependable Computing Systems*. Hrsg. von Brian Randell u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, S. 123–140