Fehlertolerante Systeme

Einführung, Klassifikation, Fehlermodell

Peter Ulbrich

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

21. Oktober 2014

https://www4.cs.fau.de/Lehre/WS14/MS_AKSS/

Überblick

Was ist ein Fehler?

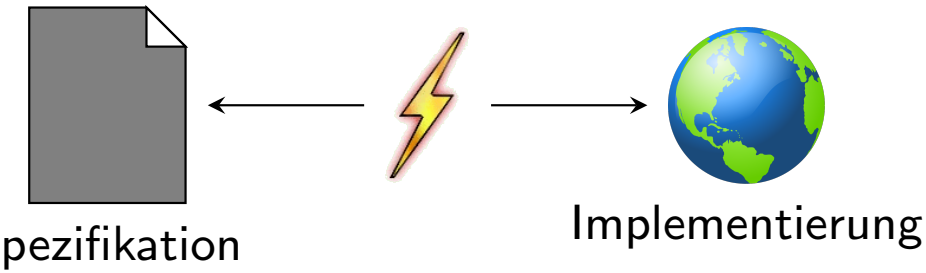
- Definition
- Sichtbarkeit
- Klassifikation

Gegenmaßnahmen

- Umgang mit Fehlern
- Redundanz

Definition: Fehler

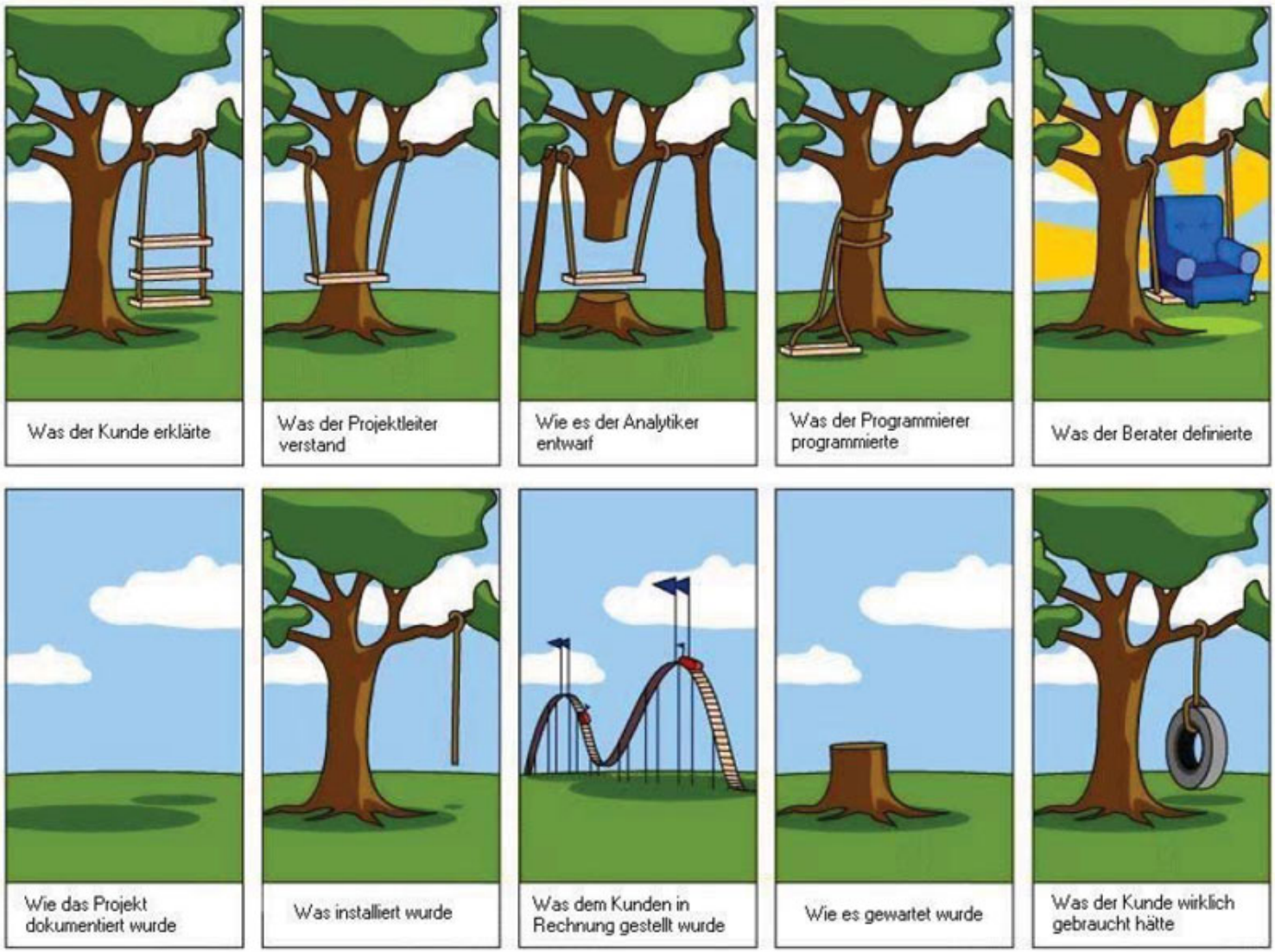
- laut DIN EN ISO 8402:1995-08 [4] ist ein Fehler die „Nichterfüllung einer festgelegten Forderung“



- Fehler kennen demzufolge viele Ausprägungen, . . .
  - sie können lediglich als störend empfunden werden
    - die eigentliche Funktion ist noch vorhanden, es geht aber Komfort verloren
  - sie können die Funktionalität beeinträchtigen
    - das Abspielen eines Videos „ruckelt“, die Bildrate wird nicht erreicht
  - sie können aber auch zum vollständigen Systemversagen führen
    - eine fehlerhafte Fluglageregelung kann den I4Copter abstürzen lassen
- **Wichtig:** der Bezugspunkt ist die Spezifikation $\rightsquigarrow$ **Verifikation**
  - Haben wir das System korrekt implementiert?

Abgrenzung: Keine Validierung

Haben wir das korrekte System implementiert?



Wann ist ein Fehler nun ein Fehler?

Das Problem beginnt, wenn der Schuh drückt!

```
int regelschritt() {

  int sensorwert = 0;
  sensorwert = leseSensor();

  int stellwert =
      regler(sensorwert);
  return stellwert;

}
```

Wann ist ein Fehler nun ein Fehler?

Das Problem beginnt, wenn der Schuh drückt!

(gutartiger) **Defekt**

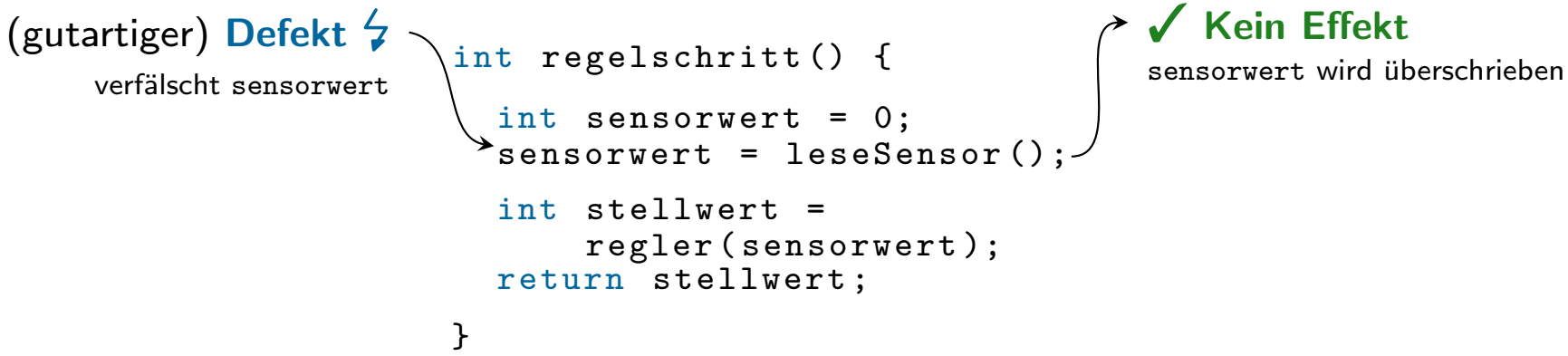
verfälscht `sensorwert`

```
int regelschritt() {

  int sensorwert = 0;
  sensorwert = leseSensor();

  int stellwert =
      regler(sensorwert);
  return stellwert;

}
```

- Defekte (engl. *faults*) sind die Quelle allen Übels
  - Ursachen: Software-Bugs, Produktionsfehler, äußere Einflüsse, ...
  - beziehen sich auf die Struktur

Wann ist ein Fehler nun ein Fehler?

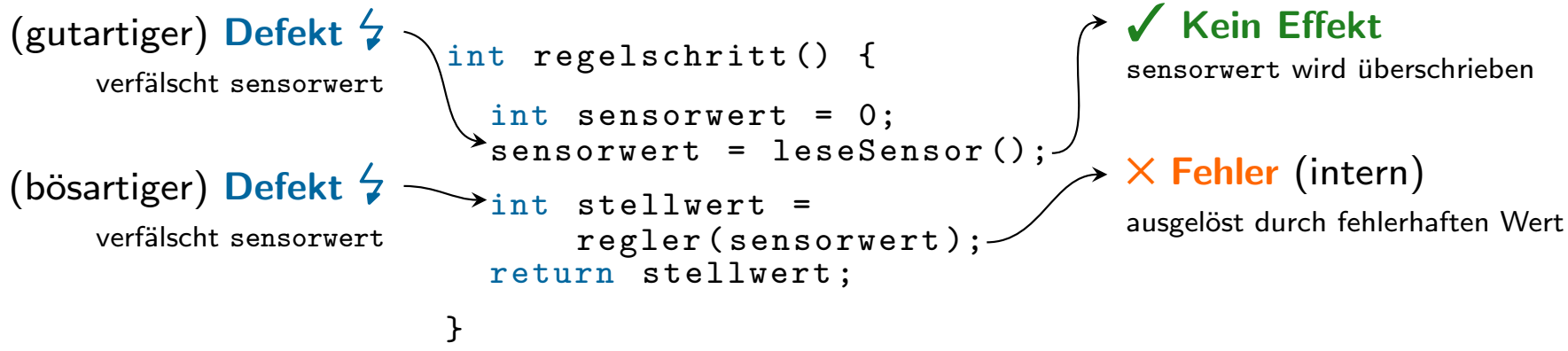
Das Problem beginnt, wenn der Schuh drückt!



- Defekte (engl. *faults*) sind die Quelle allen Übels
  - Ursachen: Software-Bugs, Produktionsfehler, äußere Einflüsse, ...
  - beziehen sich auf die Struktur
  - gutartige Defekte (engl. *benign faults*) führen nicht zu einem Fehler

Wann ist ein Fehler nun ein Fehler?

Das Problem beginnt, wenn der Schuh drückt!

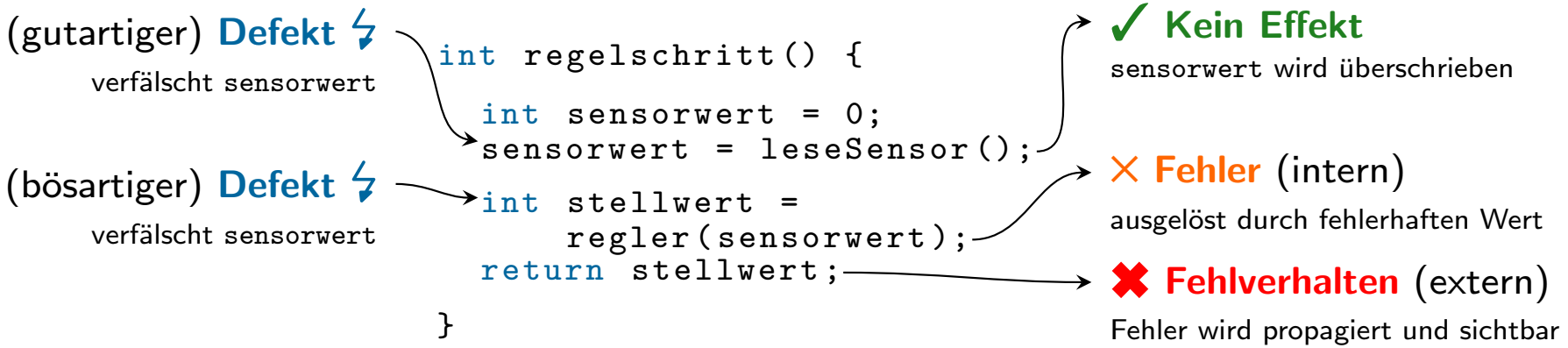


- Defekte (engl. *faults*) sind die Quelle allen Übels
  - Ursachen: Software-Bugs, Produktionsfehler, äußere Einflüsse, ...
  - beziehen sich auf die Struktur
  - gutartige Defekte (engl. *benign faults*) führen nicht zu einem Fehler
- die Manifestation eines Defekts ist ein innerer Fehler (engl. *error*)

  ⇝ der Defekt ist also bösartig (engl. *malign fault*)
  - beziehen sich auf den nicht sichtbaren, inneren Zustand

Wann ist ein Fehler nun ein Fehler?

Das Problem beginnt, wenn der Schuh drückt!

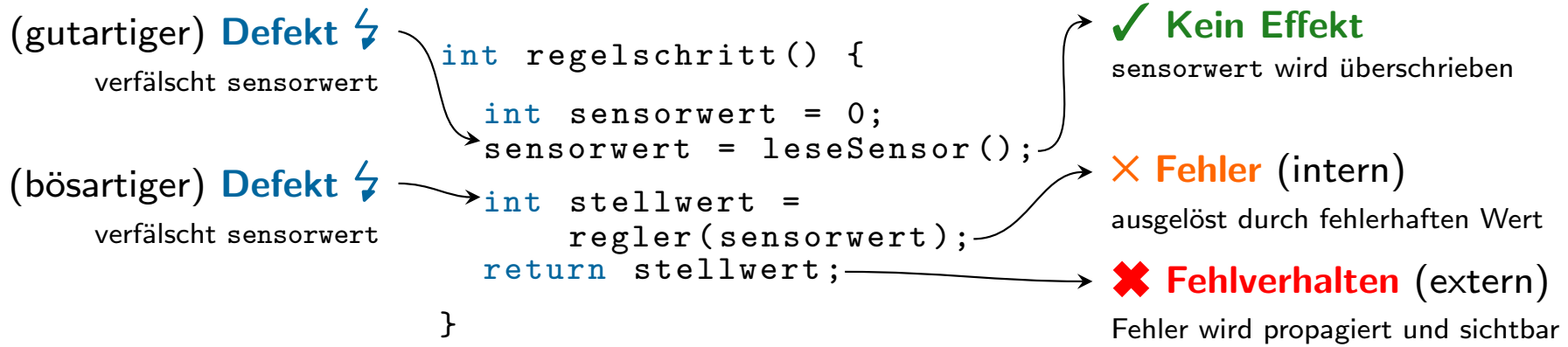


- Defekte (engl. *faults*) sind die Quelle allen Übels
  - Ursachen: Software-Bugs, Produktionsfehler, äußere Einflüsse, ...
  - beziehen sich auf die Struktur
  - gutartige Defekte (engl. *benign faults*) führen nicht zu einem Fehler
- die Manifestation eines Defekts ist ein innerer Fehler (engl. *error*)

  ⇝ der Defekt ist also bösartig (engl. *malign fault*)
  - beziehen sich auf den nicht sichtbaren, inneren Zustand
- außen sichtbare, innere Fehler heißen Fehlverhalten (engl. *failure*)
  - beziehen sich auf das beobachtbare Verhalten

Wann ist ein Fehler nun ein Fehler?

Das Problem beginnt, wenn der Schuh drückt!

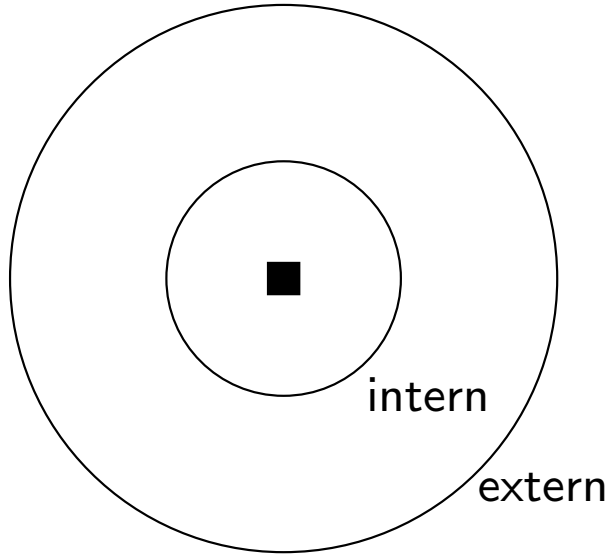


- Defekte (engl. *faults*) sind die Quelle allen Übels
  - Ursachen: Software-Bugs, Produktionsfehler, äußere Einflüsse, ...
  - beziehen sich auf die Struktur
  - gutartige Defekte (engl. *benign faults*) führen nicht zu einem Fehler
- die Manifestation eines Defekts ist ein innerer Fehler (engl. *error*)

  ⇝ der Defekt ist also bösartig (engl. *malign fault*)
  - beziehen sich auf den nicht sichtbaren, inneren Zustand
- außen sichtbare, innere Fehler heißen Fehlverhalten (engl. *failure*)
  - beziehen sich auf das beobachtbare Verhalten
  - man spricht von der fault ⇝ error ⇝ failure-Kette [11, Kapitel 1]

Es ist alles eine Frage der Sichtbarkeit

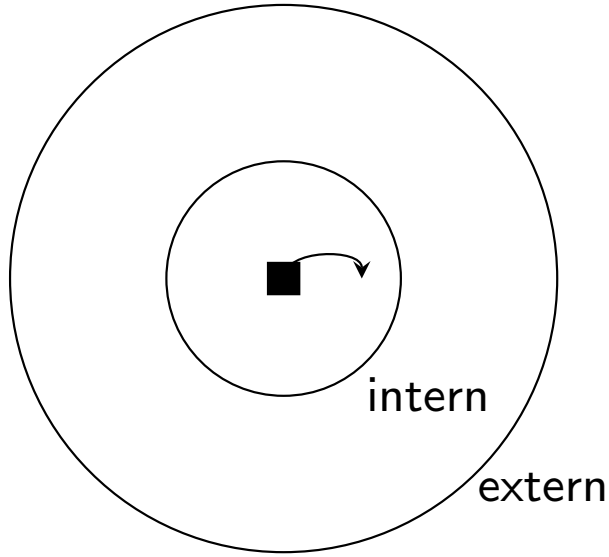
Was ich nicht weiß, macht mich nicht heiß!



Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

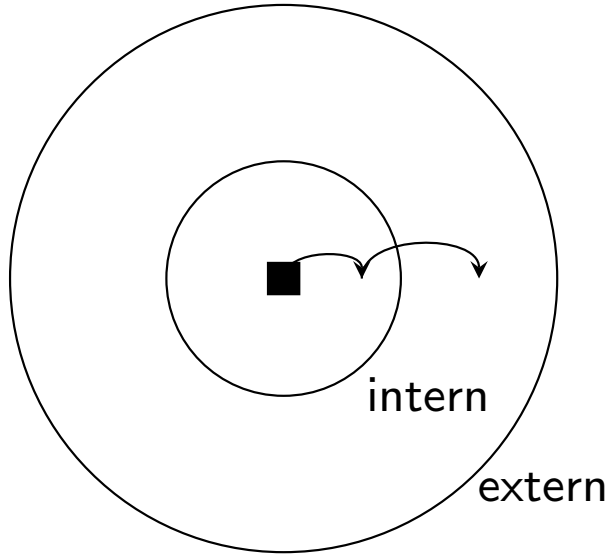
- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten



Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

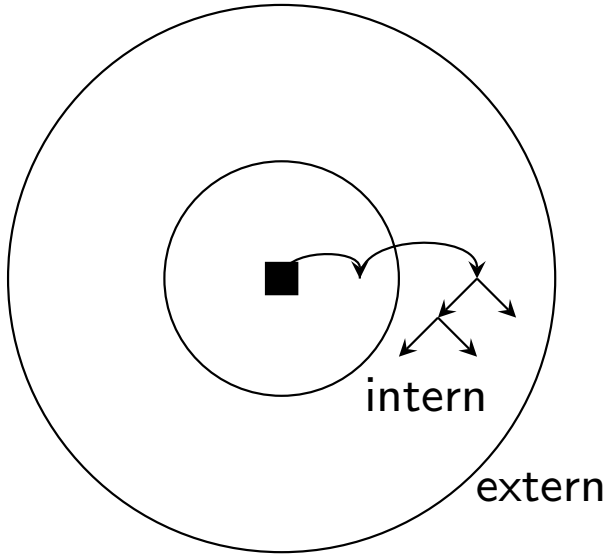
- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand



Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

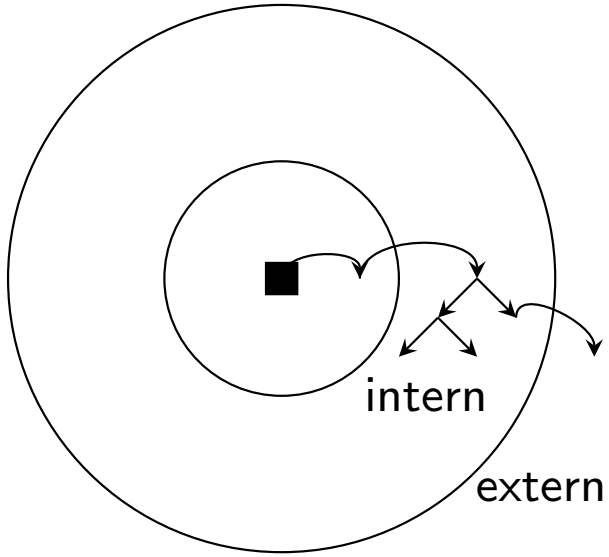
- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand
  - intern kann sich der Fehler weiter verbreiten



Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

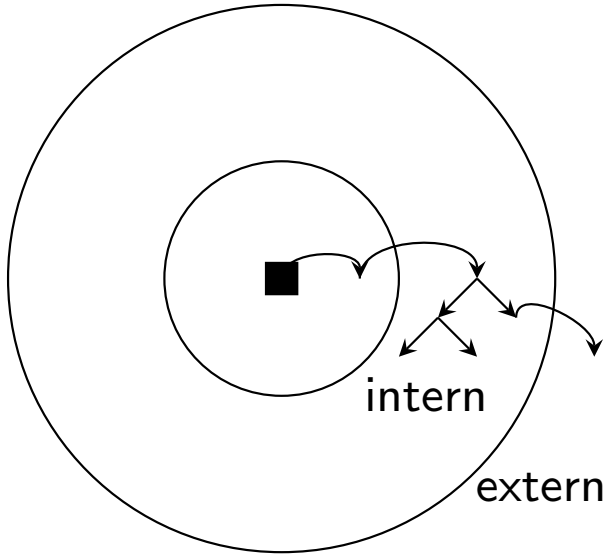
- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand
  - intern kann sich der Fehler weiter verbreiten
- wird der innerer Fehler nach außen gereicht, ist er als Fehlverhalten sichtbar



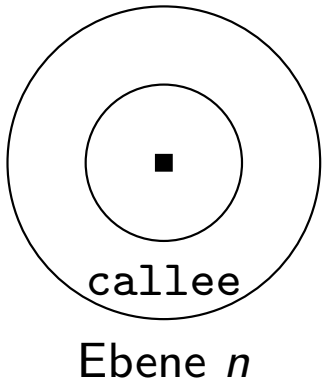
Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand
  - intern kann sich der Fehler weiter verbreiten
- wird der innerer Fehler nach außen gereicht, ist er als Fehlverhalten sichtbar



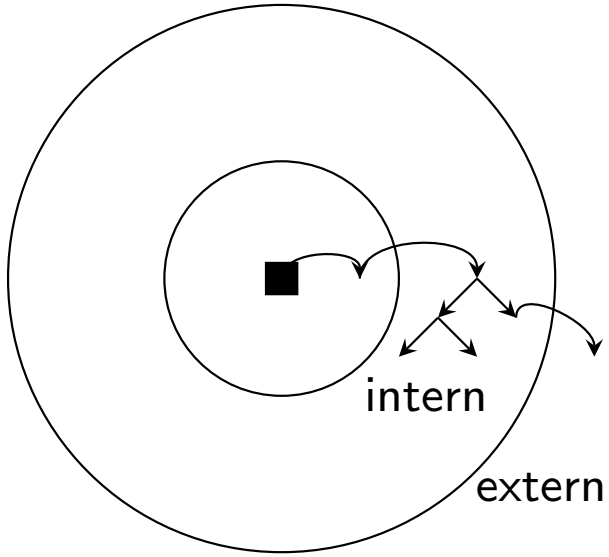
- über die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette pflanzen sich Fehler fort



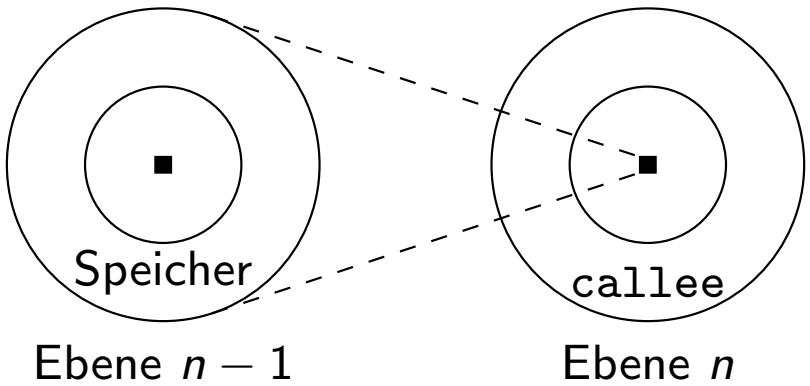
Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand
  - intern kann sich der Fehler weiter verbreiten
- wird der innerer Fehler nach außen gereicht, ist er als Fehlverhalten sichtbar



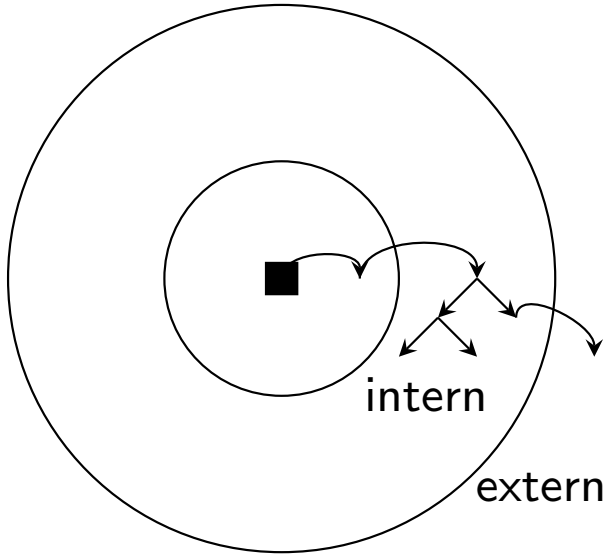
- über die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette pflanzen sich Fehler fort



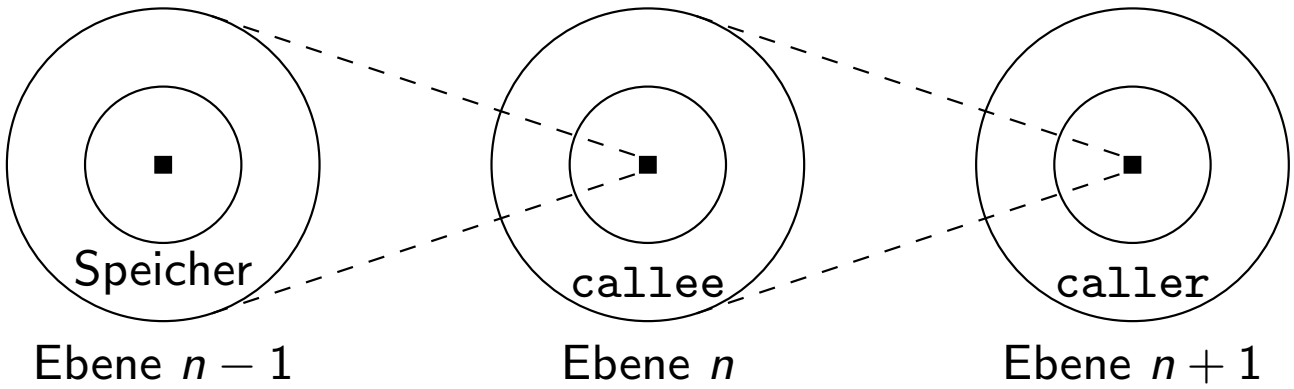
Es ist alles eine Frage der Sichtbarkeit

Was ich nicht weiß, macht mich nicht heiß!

- gutartige Defekte haben keinen Einfluss auf das korrekte Systemverhalten
- bösartige Defekte/innere Fehler beeinflussen den internen Zustand
  - intern kann sich der Fehler weiter verbreiten
- wird der innerer Fehler nach außen gereicht, ist er als Fehlverhalten sichtbar



- über die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette pflanzen sich Fehler fort



- und können schließlich zum vollständigen Systemversagen führen

Folgen von Fehlern

Wenn die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette zugeschlagen hat . . .

Nicht jedes beobachtbare Fehlverhalten muss auch entdeckt werden:

Folgen von Fehlern

Nicht jedes beobachtbare Fehlverhalten muss auch entdeckt werden:

☞ unerkannte Datenfehler (engl. *silent data corruption, SDC*)

- unbemerkte Fehlerfortpflanzung innerhalb oder außerhalb des Systems
  - fehlerhafte Berechnungsergebnisse oder Ausgabewerte
- sehr, sehr schwer ausfindig zu machen
  - das verursachte Fehlverhalten wird erst viel später sichtbar

Folgen von Fehlern

Wenn die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette zugeschlagen hat . . .

Nicht jedes beobachtbare Fehlverhalten muss auch entdeckt werden:

☞ unerkannte Datenfehler (engl. *silent data corruption, SDC*)

- unbemerkte Fehlerfortpflanzung innerhalb oder außerhalb des Systems
  - fehlerhafte Berechnungsergebnisse oder Ausgabewerte
- sehr, sehr schwer ausfindig zu machen
  - das verursachte Fehlverhalten wird erst viel später sichtbar
- die Fehlererkennung verhindert die unbemerkte Fehlerausbreitung
  - Zusicherungen (engl. *assertions*) übernehmen genau diese Aufgabe

☞ erkannte, nicht korrigierbare Fehler (engl. *detected unrecoverable error, DUE*)

- eine Fortpflanzung kann gezielt unterbunden werden ($\rightsquigarrow$ fail-stop)
- die Stellen, an diese Fehler auftreten, lassen sich vergleichsweise einfach herausfinden, z. B. durch eine Ablaufverfolgung (engl. *backtrace*)

Klassifikation nach dem Auftreten

- Fehler müssen nicht immer auftreten . . .

permanente Fehler (engl. *permanent fault/error/failure*)

- bestehen eine unbegrenzt lange Zeitdauer
- bis sie durch eine korrigierende Maßnahme behoben werden

sporadische Fehler (engl. *intermittent fault/error/failure*)

- treten unregelmäßig auf, häufen sich aber in vielen Fällen und . . .
- sind oft Vorboten drohender, permanenter Fehler

transiente Fehler (engl. *transient fault/error/failure*)

- treten wie sporadische Fehler unregelmäßig auf . . .
- münden i. d. R. aber nicht in einem permanenten Fehler

Klassifikation nach dem Auftreten

- Fehler müssen nicht immer auftreten . . .

  permanente Fehler (engl. *permanent fault/error/failure*)
  - bestehen eine unbegrenzt lange Zeitdauer
  - bis sie durch eine korrigierende Maßnahme behoben werden

  sporadische Fehler (engl. *intermittent fault/error/failure*)
  - treten unregelmäßig auf, häufen sich aber in vielen Fällen und . . .
  - sind oft Vorboten drohender, permanenter Fehler

  transiente Fehler (engl. *transient fault/error/failure*)
  - treten wie sporadische Fehler unregelmäßig auf . . .
  - münden i. d. R. aber nicht in einem permanenten Fehler

- Implikationen aus der fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette:
  - normalerweise: transiente Defekte $\not\rightsquigarrow$ permanentes Fehlverhalten
  - möglich: permanenter Defekt $\rightsquigarrow$ transientes Fehlverhalten
    - wenn sie nur unregelmäßig den inneren Sichtbarkeitsbereich verlassen

Softwarefehler (engl. *software bugs*) . . .

- sind permanente Defekte
  - manifestieren sich aber nicht unbedingt in einem inneren Fehler, von außen beobachtbarem Fehlverhalten oder einem Systemausfall
  - Beispiel: sog. Heisenbugs verursacht durch Nebenläufigkeitsfehler
    - auch Bohrbugs, Mandelbugs oder Schrödinbugs
    - treten manchmal auf, manchmal nicht $\rightsquigarrow$ sehr schwer zu reproduzieren

Softwarefehler (engl. *software bugs*) ...

- sind permanente Defekte
  - manifestieren sich aber nicht unbedingt in einem inneren Fehler, von außen beobachtbarem Fehlverhalten oder einem Systemausfall
  - Beispiel: sog. Heisenbugs verursacht durch Nebenläufigkeitsfehler
    - auch Bohrbugs, Mandelbugs oder Schrödinbugs
    - treten manchmal auf, manchmal nicht $\rightsquigarrow$ sehr schwer zu reproduzieren
- resultieren aus einer fehlerhaften Umsetzung der Spezifikation
  - in der Regel durch den Programmierer, Architekten, ...
  - Ursprung: Anforderungserhebung, Entwurf, Implementierung, ...
    - betroffen ist der komplette Zyklus der Softwareerstellung

Softwarefehler (engl. *software bugs*) ...

- sind permanente Defekte
  - manifestieren sich aber nicht unbedingt in einem inneren Fehler, von außen beobachtbarem Fehlverhalten oder einem Systemausfall
  - Beispiel: sog. Heisenbugs verursacht durch Nebenläufigkeitsfehler
    - auch Bohrbugs, Mandelbugs oder Schrödinbugs
    - treten manchmal auf, manchmal nicht $\rightsquigarrow$ sehr schwer zu reproduzieren
- resultieren aus einer fehlerhaften Umsetzung der Spezifikation
  - in der Regel durch den Programmierer, Architekten, ...
  - Ursprung: Anforderungserhebung, Entwurf, Implementierung, ...
    - betroffen ist der komplette Zyklus der Softwareerstellung
- sind systematische Fehler
  - Betreibt man mehrere Instanzen derselben Softwareversion
  - mehrfach unter identischen, äußeren Bedingungen,

  $\rightsquigarrow$ werden alle Instanzen dieselben beobachtbaren Fehler zeigen.
    - die äußeren Bedingungen sind allerdings nicht ohne Weiteres reproduzierbar
    - vgl. Trägerrakete Ariane 5 [9]: Ausfall von SRI 1 und SRI 2 aufgrund desselben Softwarefehlers

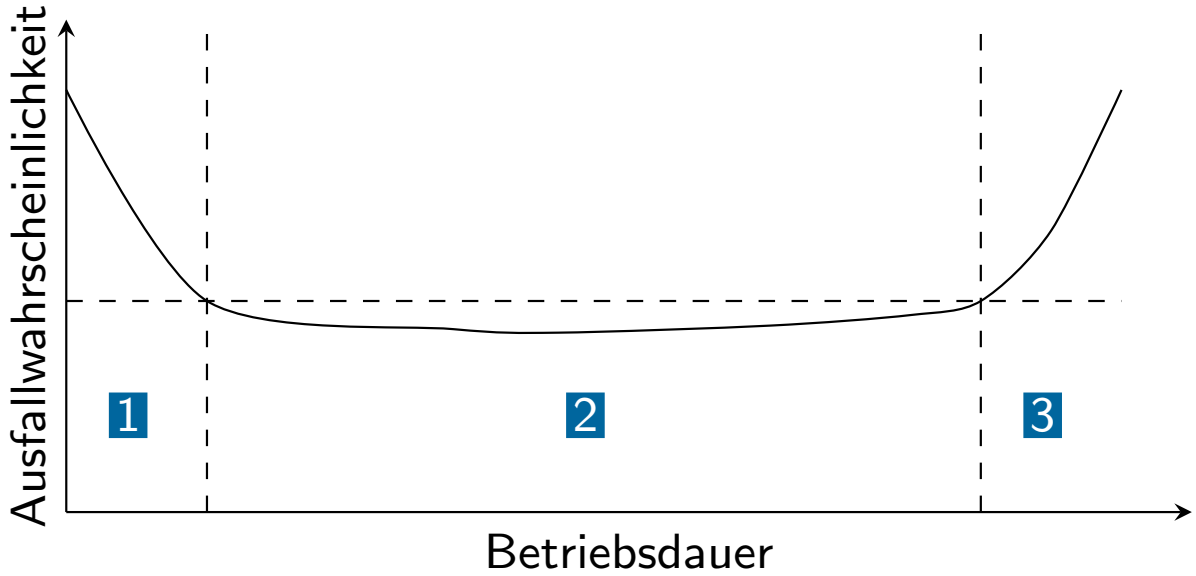
Hardwarefehler

- permanente Hardwarefehler sind . . .
  - extrinsischer Natur: herstellungsbedingte Materialfehler
    - z. B. fehlerhafte Dotierung eines Halbleiters oder Materialunreinheiten
    - treten meist zu Beginn der Lebenszeit auf ($\leadsto$ Säuglingssterblichkeit)
  - intrinsischer Natur: Verschleißerscheinungen
    - kündigen sich meist durch sporadische Fehler an
    - treten meist am Ende der Lebenszeit auf

Hardwarefehler

- permanente Hardwarefehler sind . . .
  - extrinsischer Natur: herstellungsbedingte Materialfehler
    - z. B. fehlerhafte Dotierung eines Halbleiters oder Materialunreinheiten
    - treten meist zu Beginn der Lebenszeit auf (↝ Säuglingssterblichkeit)
  - intrinsischer Natur: Verschleißerscheinungen
    - kündigen sich meist durch sporadische Fehler an
    - treten meist am Ende der Lebenszeit auf
- Umwelteinflüsse verursachen transiente Hardwarefehler
  - mannigfaltige Ursachen
    - radioaktive Strahlung
    - elektromagnetische Interferenz
    - instabile Spannungsversorgung
    - Fertigungsstreuung bei einzelnen Transistoren
    - Temperaturschwankungen führen zum temporären Materialdefekten
    - . . .
  - treten als schwer zu fassende „Bitkipper“ in Erscheinung

Wahrscheinlichkeit permanenter Hardwarefehler



1. erhöhte Säuglingssterblichkeit durch fertigungsbedingte Defekte
   - eine Einbrennphase (engl. *burn-in*) filtert fehlerhafte Elemente heraus
2. normaler, sinnvoll nutzbarer Betriebszeitraum
   - Ausfallrate nahe an der durchschnittlichen Ausfallwahrscheinlichkeit
3. durch Verschleiß bedingte Ausfälle
   - auch Halbleiterbauelement unterliegen einem Verschleißprozess
   - z. B. Elektromigration, Spannungsrisse durch thermische Belastungen, Verschleiß der Oxidschicht am Gate ...

Transiente Hardwarefehler

- Bitkipper durch Umladungen in Speicherzellen und Schaltkreisen
  - ☞ verursacht durch ionisierende Strahlung
    - Alphateilchen aus kontaminierten Chipgehäusen oder Lötkugeln
      - das waren „die ersten transienten Fehler“ [11, Kapitel 1.1]
      - direkte Erzeugung transienter Fehler durch Erzeugung von Elektronen und Löchern (engl. *holes*), die sich nicht rekombinieren
    - Neutronen aus kosmischer Strahlung

      primäre kosmische Strahlung galaktische und solare Partikel

      sekundäre kosmische Strahlung entsteht durch Wechselwirkung primärer Strahlung mit Atomen aus der Erdatmosphäre

      terrestrische kosmische Strahlung bezeichnet die Partikel kosmischer Strahlung, die schließlich die Erdoberfläche erreichen

Transiente Hardwarefehler

- Bitkipper durch Umladungen in Speicherzellen und Schaltkreisen
  - ☞ verursacht durch ionisierende Strahlung
    - Alphateilchen aus kontaminierten Chipgehäusen oder Lötkugeln
      - das waren „die ersten transienten Fehler“ [11, Kapitel 1.1]
      - direkte Erzeugung transienter Fehler durch Erzeugung von Elektronen und Löchern (engl. *holes*), die sich nicht rekombinieren
    - Neutronen aus kosmischer Strahlung

      primäre kosmische Strahlung galaktische und solare Partikel

      sekundäre kosmische Strahlung entsteht durch Wechselwirkung primärer Strahlung mit Atomen aus der Erdatmosphäre

      terrestrische kosmische Strahlung bezeichnet die Partikel kosmischer Strahlung, die schließlich die Erdoberfläche erreichen
- Verfälschung von Kommunikation auf Bussen
  - ☞ verursacht durch elektromagnetische Interferenz $\rightsquigarrow$ Rauschen
    - z. B. in Automobilen gibt es verschiedene Quellen für Wechselfelder
      - elektronischer Anlasser, Lichtmaschine, ...
    - eine sparsame elektronische Abschirmung macht dies zum Problem

Anfälligkeit eines Schaltkreises für transiente Fehler

- die transiente Fehlerate (engl. *soft-error rate, SER*) eines Schaltkreises hängt (stark vereinfacht) von folgenden Faktoren ab:

$$\text{SER} = C \times \text{Neutronenfluss} \times \text{Fläche} \times e^{-Q_{crit}/Q_{coll}}$$

Anfälligkeit eines Schaltkreises für transiente Fehler

- die transiente Fehlerate (engl. *soft-error rate, SER*) eines Schaltkreises hängt (stark vereinfacht) von folgenden Faktoren ab:

$$\text{SER} = C \times \text{Neutronenfluss} \times \text{Fläche} \times e^{-Q_{crit}/Q_{coll}}$$

C prozess- und schaltkreisspezifische Konstante

Fläche des Schaltkreises

Q_{crit} minimale für eine Fehlfunktion notwendige Ladung
- wird mit Hilfe von Simulationen bestimmt

Q_{coll} Effizienz der Ladungsaufnahme
- abhängig von der Dotierung und der Versorgungsspannung V_{CC}
- je größer das Bremsvermögen (engl. *stopping power*) eines Teilchens ist, desto größer ist auch Q_{coll}
  - das Bremsvermögen beschreibt die Energie, die ein Teilchen auf einer bestimmten Wegstrecke an die umliegende Materie abgeben kann

Anfälligkeit eines Schaltkreises für transiente Fehler

- die transiente Fehlerate (engl. *soft-error rate, SER*) eines Schaltkreises hängt (stark vereinfacht) von folgenden Faktoren ab:

$$\text{SER} = C \times \text{Neutronenfluss} \times \text{Fläche} \times e^{-Q_{crit}/Q_{coll}}$$

C prozess- und schaltkreisspezifische Konstante

Fläche des Schaltkreises

Q_{crit} minimale für eine Fehlfunktion notwendige Ladung
  - wird mit Hilfe von Simulationen bestimmt

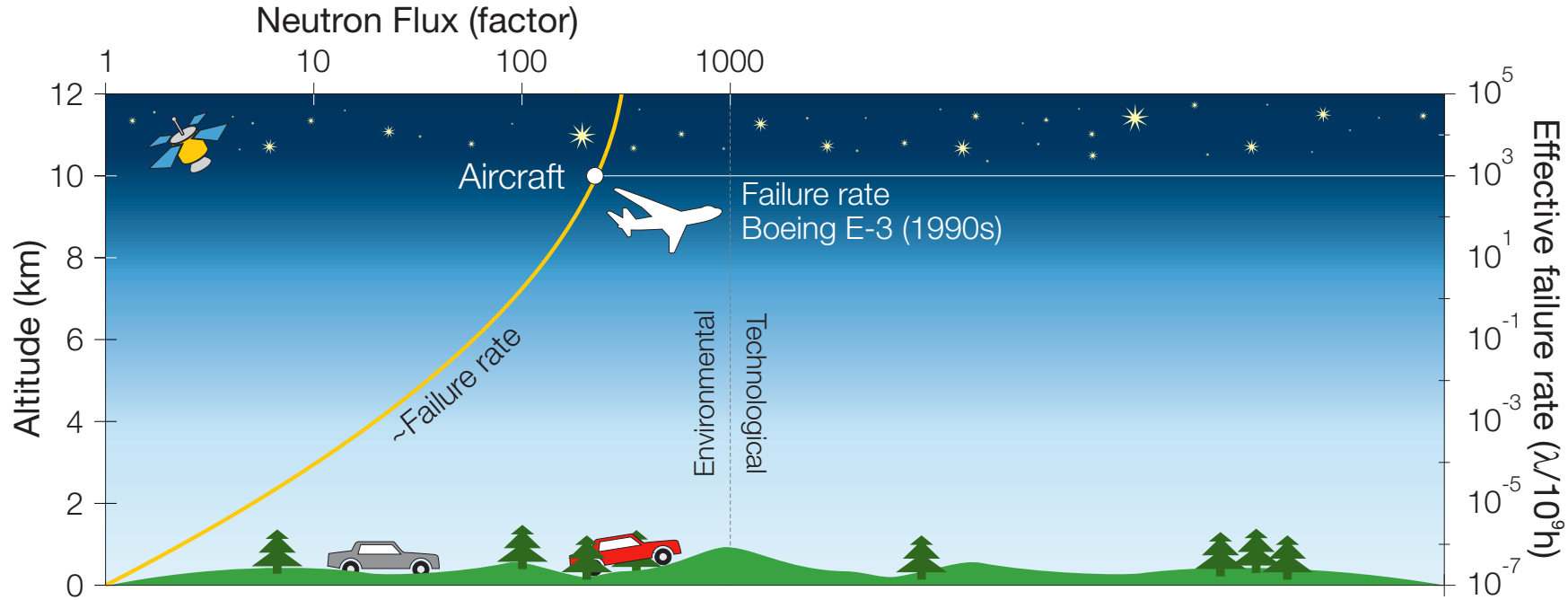
Q_{coll} Effizienz der Ladungsaufnahme
  - abhängig von der Dotierung und der Versorgungsspannung V_{CC}
  - je größer das Bremsvermögen (engl. *stopping power*) eines Teilchens ist, desto größer ist auch Q_{coll}
    - das Bremsvermögen beschreibt die Energie, die ein Teilchen auf einer bestimmten Wegstrecke an die umliegende Materie abgeben kann

- kleinere Halbleiterstrukturen sind Fluch und Segen zugleich

$\rightsquigarrow$ kleinere Fläche $\rightsquigarrow$ kleinere SER

$\rightsquigarrow$ kleinere Q_{crit} $\rightsquigarrow$ größere SER

Fehlerraten – Entwicklung und Tendenzen



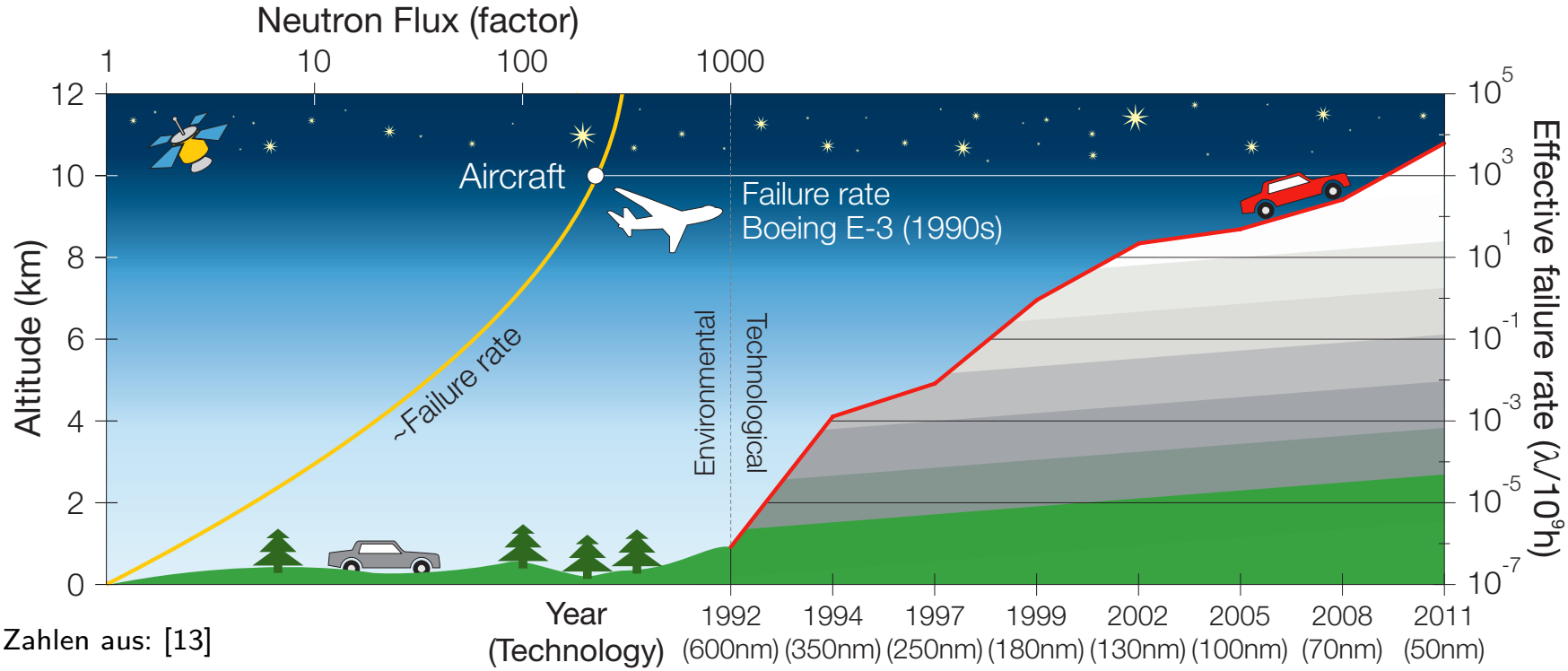
Zahlen aus: [13]

☞ Exakte Fehlerrate schwer zu ermitteln

- Fehlerrate pro Bit stagniert oder verbessert sich
  - deutet auf das Erreichen eines Sättigungsbereichs hin

  ⇝ jede elektrische Beeinflussung bedeutet einen Bitkipper
- Fehlerrate des Systems hängt indes von vielen Faktoren ab

Fehlerraten – Entwicklung und Tendenzen



Zahlen aus: [13]

☞ Exakte Fehlerrate schwer zu ermitteln

- Fehlerrate pro Bit stagniert oder verbessert sich
  - deutet auf das Erreichen eines Sättigungsbereichs hin

  ⇝ jede elektrische Beeinflussung bedeutet einen Bitkipper
- Fehlerrate des Systems hängt indes von vielen Faktoren ab

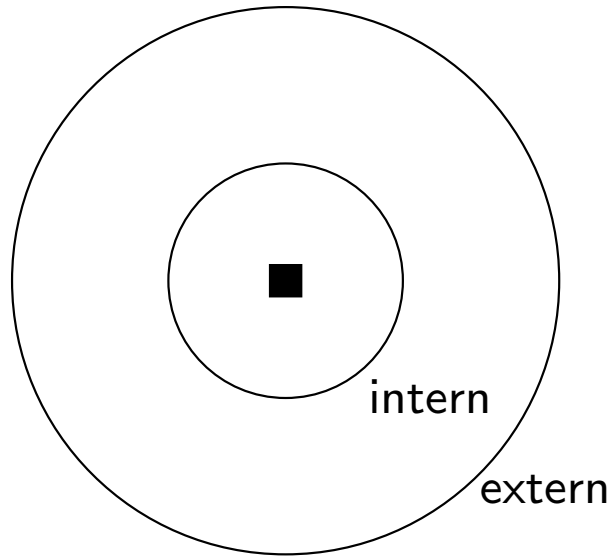
☞ Vorhersagen von Forschern und Herstellern [1, 5]

- Mehr Leistung und Parallelität ⇝ Auf Kosten der Zuverlässigkeit

Überblick

Maßnahmen zum Umgang mit Fehlern

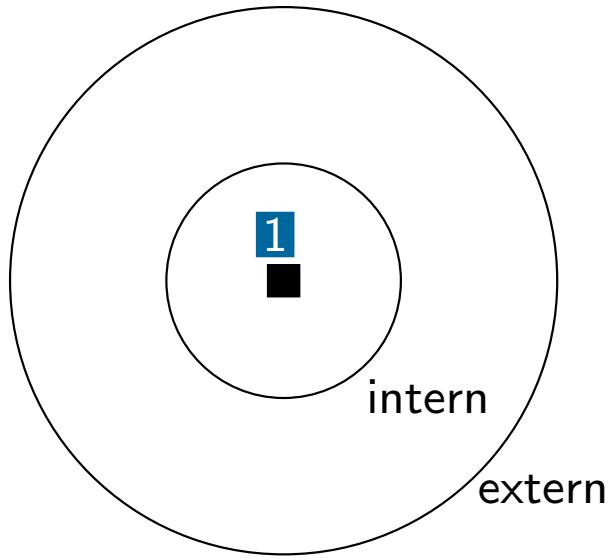
Versuchen die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette aufzubrechen



Maßnahmen zum Umgang mit Fehlern

1 Vorbeugung – versucht die Entstehung von Defekten in der Produktion zu verhindern

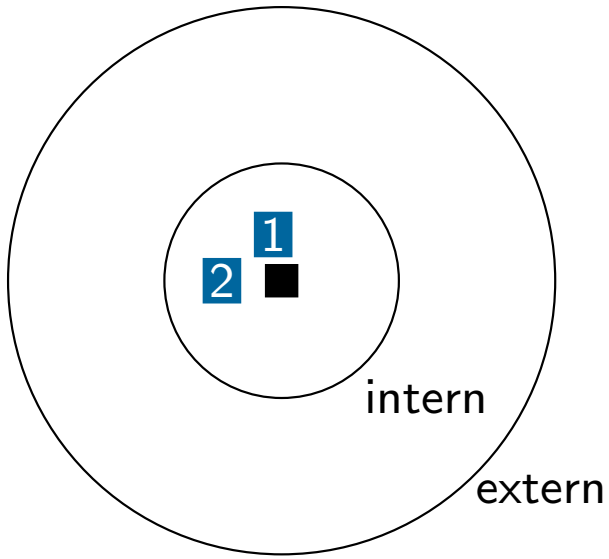
- z.B. durch Entwicklungsmethoden



Maßnahmen zum Umgang mit Fehlern

Versuchen die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette aufzubrechen

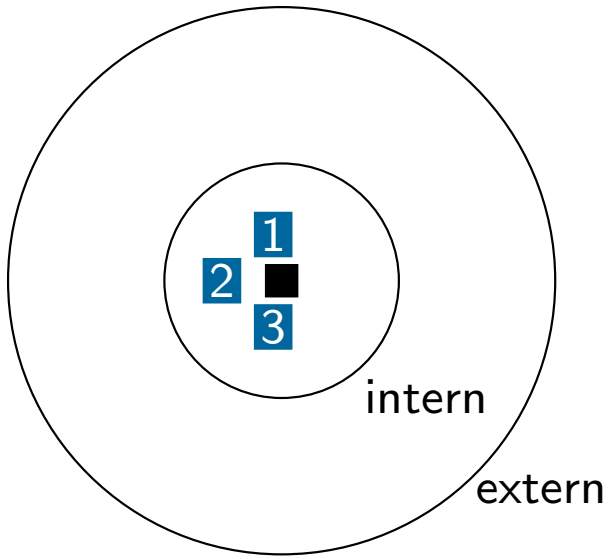
1. Vorbeugung – versucht die Entstehung von Defekten in der Produktion zu verhindern
   - z.B. durch Entwicklungsmethoden
2. Entfernung – vor der Auslieferung oder im Zuge einer planmäßigen Wartung
   - erfordert die Erkennung von Defekten $\mapsto$ Qualitätssicherung



Maßnahmen zum Umgang mit Fehlern

Versuchen die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette aufzubrechen

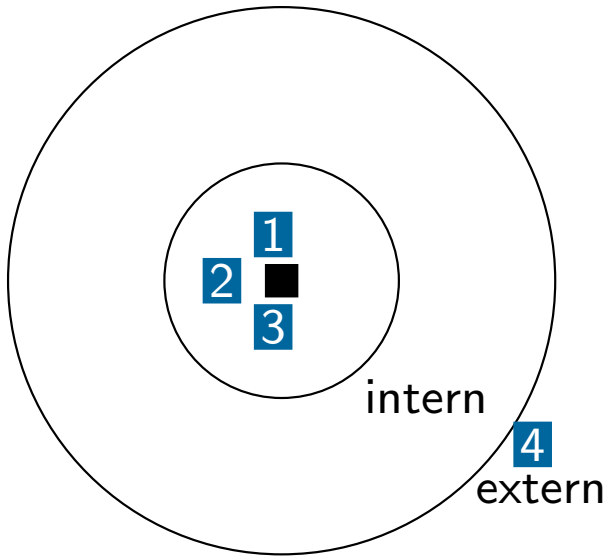
1. Vorbeugung – versucht die Entstehung von Defekten in der Produktion zu verhindern
   - z.B. durch Entwicklungsmethoden
2. Entfernung – vor der Auslieferung oder im Zuge einer planmäßigen Wartung
   - erfordert die Erkennung von Defekten $\mapsto$ Qualitätssicherung
3. Vorhersage – Wo treten evtl. Defekte auf?
   - ermöglicht die Entfernung oder ihre Umgehung



Maßnahmen zum Umgang mit Fehlern

Versuchen die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette aufzubrechen

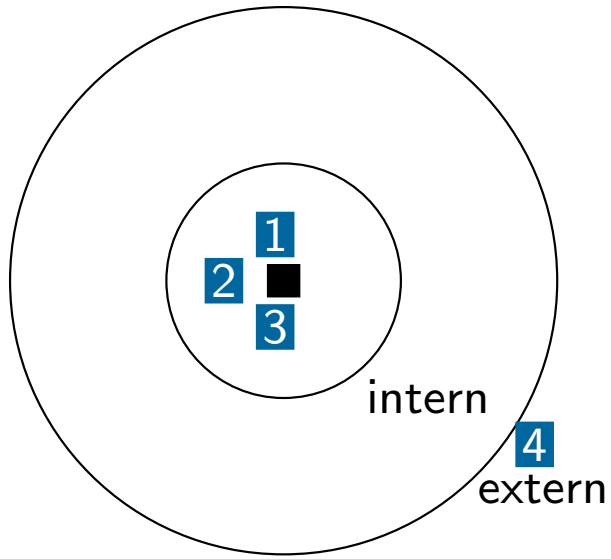
1. Vorbeugung – versucht die Entstehung von Defekten in der Produktion zu verhindern
   - z.B. durch Entwicklungsmethoden
2. Entfernung – vor der Auslieferung oder im Zuge einer planmäßigen Wartung
   - erfordert die Erkennung von Defekten $\mapsto$ Qualitätssicherung
3. Vorhersage – Wo treten evtl. Defekte auf?
   - ermöglicht die Entfernung oder ihre Umgehung
4. Toleranz – verhindert nicht den Defekt, aber die Fortpflanzung zum Fehlverhalten
   - z.B. durch Maskierung innerer Fehler



Maßnahmen zum Umgang mit Fehlern

Versuchen die fault $\rightsquigarrow$ error $\rightsquigarrow$ failure-Kette aufzubrechen

1. Vorbeugung – versucht die Entstehung von Defekten in der Produktion zu verhindern
   - z.B. durch Entwicklungsmethoden
2. Entfernung – vor der Auslieferung oder im Zuge einer planmäßigen Wartung
   - erfordert die Erkennung von Defekten $\mapsto$ Qualitätssicherung
3. Vorhersage – Wo treten evtl. Defekte auf?
   - ermöglicht die Entfernung oder ihre Umgehung
4. Toleranz – verhindert nicht den Defekt, aber die Fortpflanzung zum Fehlverhalten
   - z.B. durch Maskierung innerer Fehler



Ziel zuverlässiger Systeme

☞ Reduktion des vom Benutzer beobachtbaren Fehlverhaltens

Problem – Fehlervermeidung

- ungehemmte Fehlerfortpflanzung führt zum Systemversagen
  - unerkannte Datenfehler (engl. *silent data corruption*)
    - bedingen beispielsweise fehlerhafte Stellwerte für Aktoren
    - ihre Folgen treten häufig räumlich und zeitlich unkorreliert auf
  - erkannte, unkorrigierbare Fehler (engl. *detected unrecoverable errors*)
    - führen zu einem unmittelbaren, erkennbaren Systemversagen

☞ Vermeidung dieser Fehler ist je nach Anwendung erforderlich

- Problematik: eine entsprechend robuste Auslegung einzelner Komponenten ist häufig nicht möglich
  - diese Komponente müsste frei von konzeptionellen Fehler sein, also keinerlei Hardware- oder Softwaredefekte etc. enthalten
  - sie müsste auch allerlei widrigen äußeren Umständen trotzen
- Lösung: man benötigt ein System, das einzelne Fehler tolerieren kann
  - einzelne Komponenten können ausfallen ...
  - dies wird durch andere redundante Komponenten aufgefangen,

  ⇝ die gewünschte Funktionalität an der Schnittstelle bleibt erhalten
  - der Anwender bekommt davon möglichst nichts mit (⇝ Transparenz)

Arten von Redundanz

- Redundanz ist eine Grundvoraussetzung für Fehlertoleranz

strukturelle Redundanz

- Replikation $\rightsquigarrow$ hardwarebasierte Fehlertoleranzlösungen (typisch)
- mehrfache Auslegung: Prozessoren, Speicher, Sensoren, Aktoren, ...
- gleichartige Instanzen, agieren häufig simultan

funktionelle Redundanz

- mehrfache Herleitung desselben Sachverhalt auf verschiedenen Wegen
  - Ventilstellung $\rightsquigarrow$ Stellungsgeber bzw. Durchflussmengenmesser
- Funktionswächter (engl. *watchdog*) für bestimmte Parameter

Informationsredundanz

- zusätzliche Informationen (nicht zwingend erforderlich)
- Speicherung von Brutto- und Nettobetrag
- Typischerweise in Form von Codierung (Prüfsummen, CRC, ...)

zeitliche Redundanz

- über den Normalbetrieb hinausgehende Zeit
- z.B. Numerische Algorithmen, Schlupf in einem EZS, ...

Ziel der Redundanz

Was man mit dem Mehraufwand eigentlich bezweckt!

Fehlererkennung (engl. *fault detection*)

- Erkennen von Fehlern z. B. mithilfe von Prüfsummen

Fehlerdiagnose (engl. *fault diagnosis*)

- Identifikation der fehlerhaften redundanten Einheit

Fehlereindämmung (engl. *fault containment*)

- verhindern, dass sich ein Fehler über gewisse Grenzen ausbreitet

Fehlermaskierung (engl. *fault masking*)

- dynamische Korrektur von Fehlern z. B. durch Mehrheitsentscheid

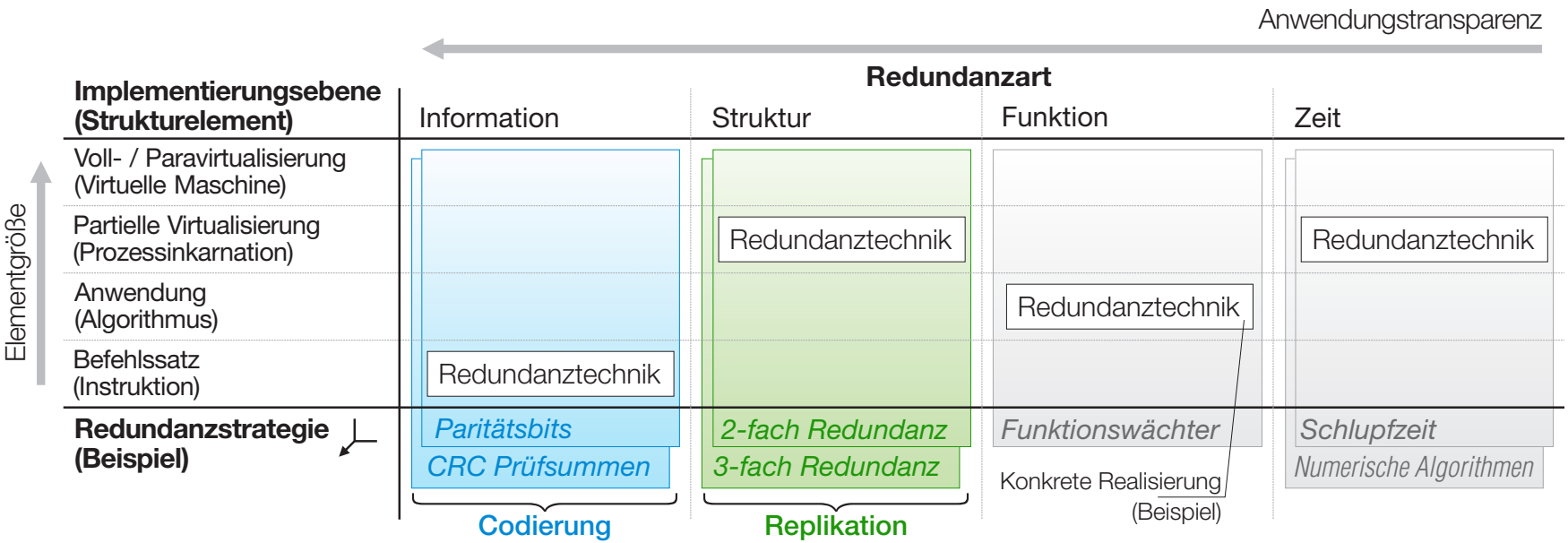
Wiederaufsetzen (engl. *recovery*)

- Wiederherstellen eines funktionsfähigen Zustands nach Fehlern
  - Reparatur (engl. *repair*) bzw. Rekonfiguration (engl. *reconfiguration*)

Fokus der Vorlesung: Fehlererkennung und Fehlermaskierung

Koordinierter Einsatz von Redundanz

Ein zweites Rechensystem nützt alleine nicht viel!



- Es gibt viele Implementierungsalternativen
- In der Praxis: Mischformen $\rightsquigarrow$ was ermöglicht Fehlererkennung?
- Funktionelle und zeitliche Redundanz erfordern umfangreiches Vorabwissen

Fokus der Vorlesung: strukturelle und Informationsredundanz