# Root Cause Analysis liefert
# SRE-Methodenleitfaden
# (Root Cause Analysis as a Guide to SRE Methods)

Timm Grams

Fachhochschule Fulda
Fachbereich Elektrotechnik und
Informationstechnik

# Natürliches Software Engineering (NSE)

1. Sobald du eine Ahnung davon hast, was zu tun ist, hacke dir dein Programm zusammen. Das nennt man *Realisierung*.
2. Fertige einen Auszug daraus. So erhältst du *Konzept* und *Entwurf*.
3. Erstelle dann die *Spezifikation.* Darin erklärst du die überraschenden Eigenschaften deines Programms zu Features.
4. Mache schließlich dem Kunden klar, dass das, was du liefern kannst, er im Grunde seines Herzens auch haben wollte. Diese anspruchsvolle Aufgabe heißt *Requirements Engineering*.

# Was die Erfahrung uns sagt

Unfälle geschehen, weil

- die *Dokumentation* fehlt oder lückenhaft ist,
- die *Spezifikation* nicht gründlich durchdacht wurde,
- die *Tests* nur an der Oberfläche gekratzt haben, und
- die *Verifikation* ganz weggelassen worden ist.

Die Untersuchung von Unfällen liefert die besten Gründe für “unnatürliche” und mühsame Software Engineering Methoden.

# Zweck dieses Beitrags

## Das Problem

Viele SRE-Methoden. Der IEC Standard 61 508 zählt 66 auf.

Mehr oder weniger nützlich. Mehr oder weniger mühsam.

## Das Ziel

*Methodenleitfaden:* Unter welchen Bedingungen zahlen sich welche Methoden aus?

## Die Methode

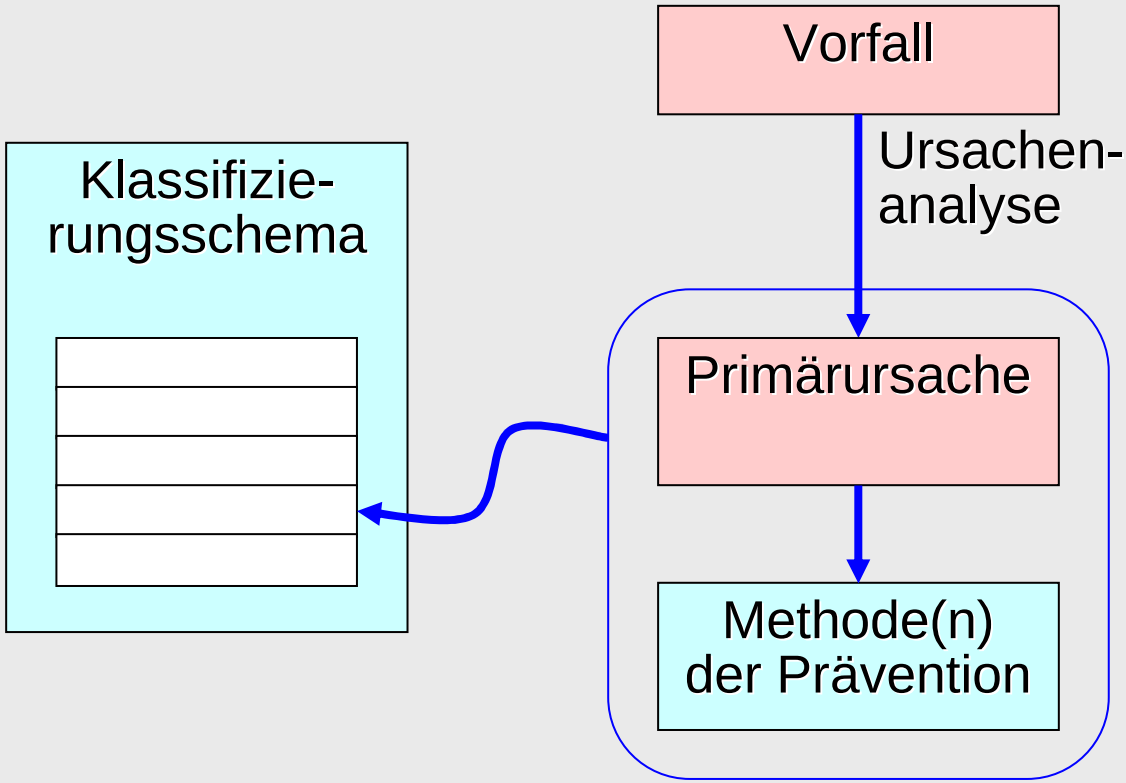
Root Cause Analysis als Klassifizierungsinstrument.

# Liste von SRE-Methoden (unvollständig)

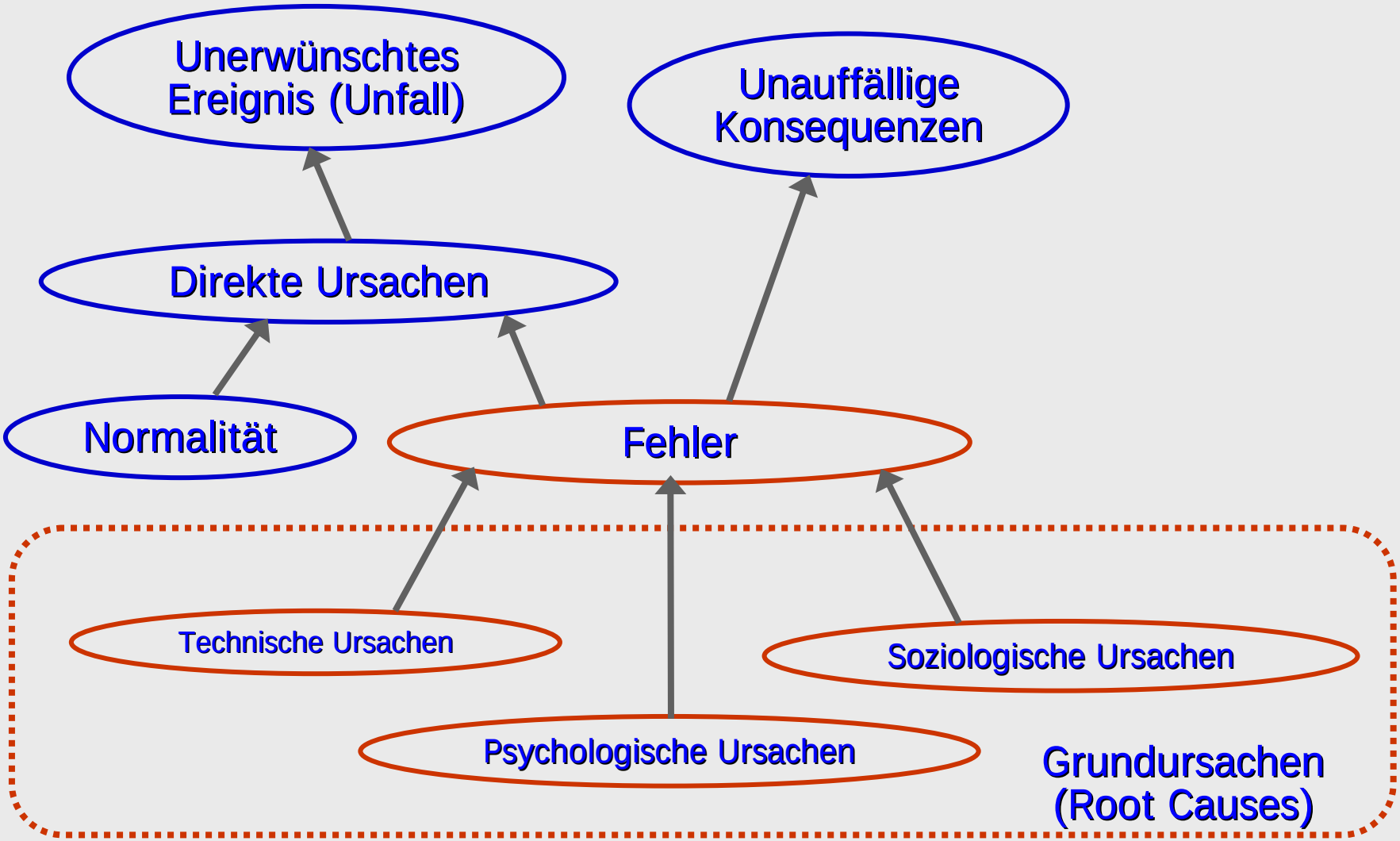
Ausfalleffekt-Analyse (FMEA)
Fehlerbaum-Analyse (FTA)
Ereignisbaum-Analyse (ETA)
Zuverlässigkeitsblockdiagramme
Zuverlässigkeitswachstumsmodelle (RGM)
Ursachen-Wirkungs-Diagramme
Code-Inspektion
Reviews
Walk-Through
Formale Inspektion (Fagan Inspection)
Metriken
Risikograph
Konfigurationsmanagement
Petri-Netze
HAZOP (Hazards and Operability Analysis)
Simulation

Unified Modeling Language (UML)
Structured Analysis and Design Technique (SADT)
Datenflussdiagramm
Zustandsdiagramme
Strukturierte Programmierung
Objektorientierte Programmierung
Regelkreis des selbstkontrollierten Programmierens
Beweisgeleitete Programmierung
Diversität, n-Versionen-Programmierung
Fehlerbuchführung (Defect Records)
Testen nach Regeln
White-Box-Test/Black-Box-Test
Äquivalenzklassentest, Grenzwertanalyse
Funktionsabdeckungstest
Statische Testmethoden
Regressionstest
Model-Checking

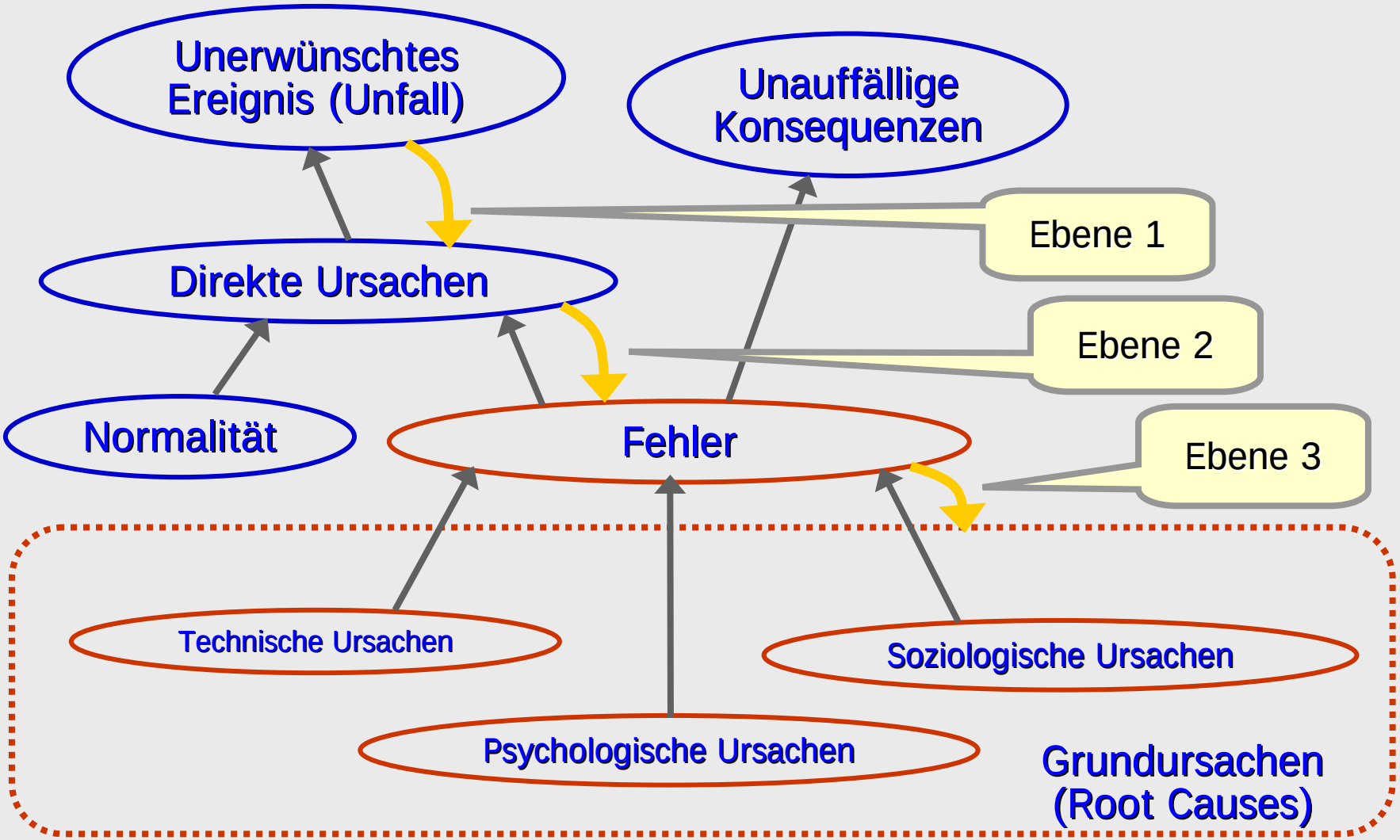
# Die Klassifizierungsmethode



# Drei Ebenen der Ursachenanalyse



# Drei Ebenen der Ursachenanalyse



# Drei Ebenen der Ursachenanalyse

1 Analyse der direkten Ursachen – Logik der Ursachen

Unfallhergang

Ereignisketten

2 Aufdeckung von Fehlern auf Basis normativer Modelle

Probleme der normativen Modellierung

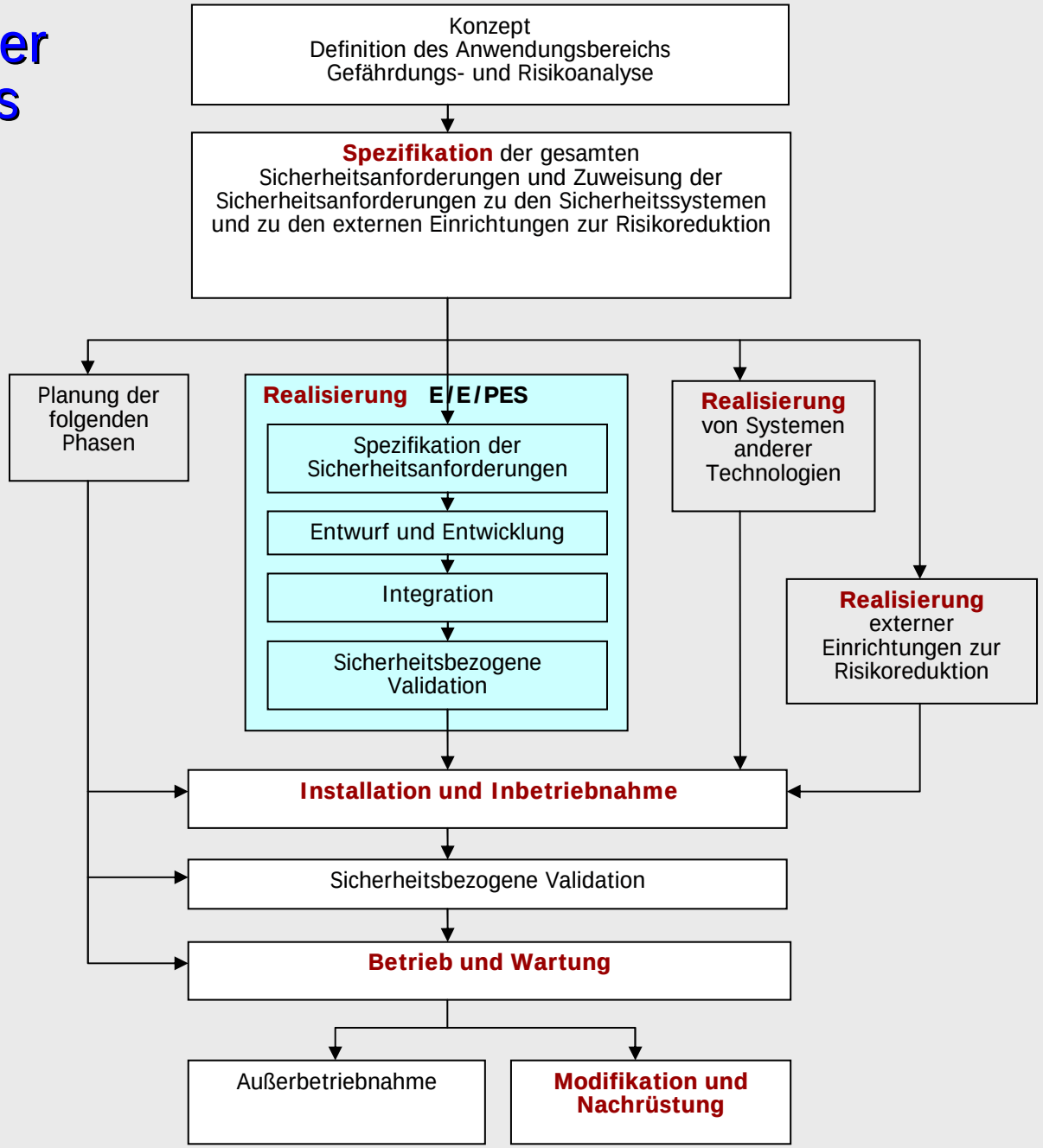
- Übersehen organisatorischer Faktoren
- Subjektivität der Klassifizierung und Ursachenzuweisung
- Übersimplifizierung
- Unwissen

3 Root Cause Analysis – Lernen durch Generalisierung

Basiert auf technischen, psychologischen und soziologischen Taxonomien

- Schlechte Realisierung technischer Aktivitäten
- Mängel in der Sicherheitskultur
- Ineffektive Organisationsstruktur

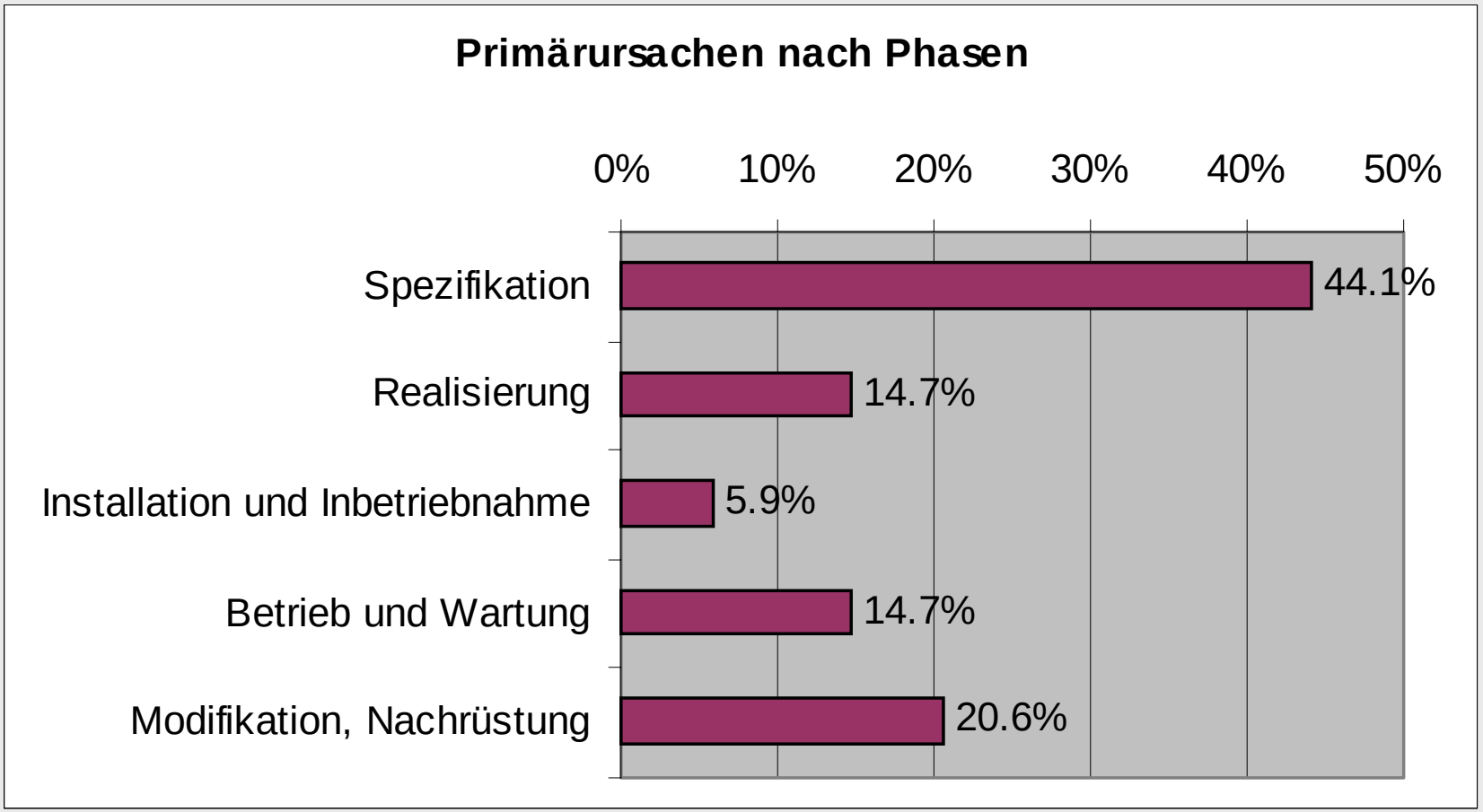
# Sicherheitsbezogener Gesamtlebenszyklus IEC 61 508



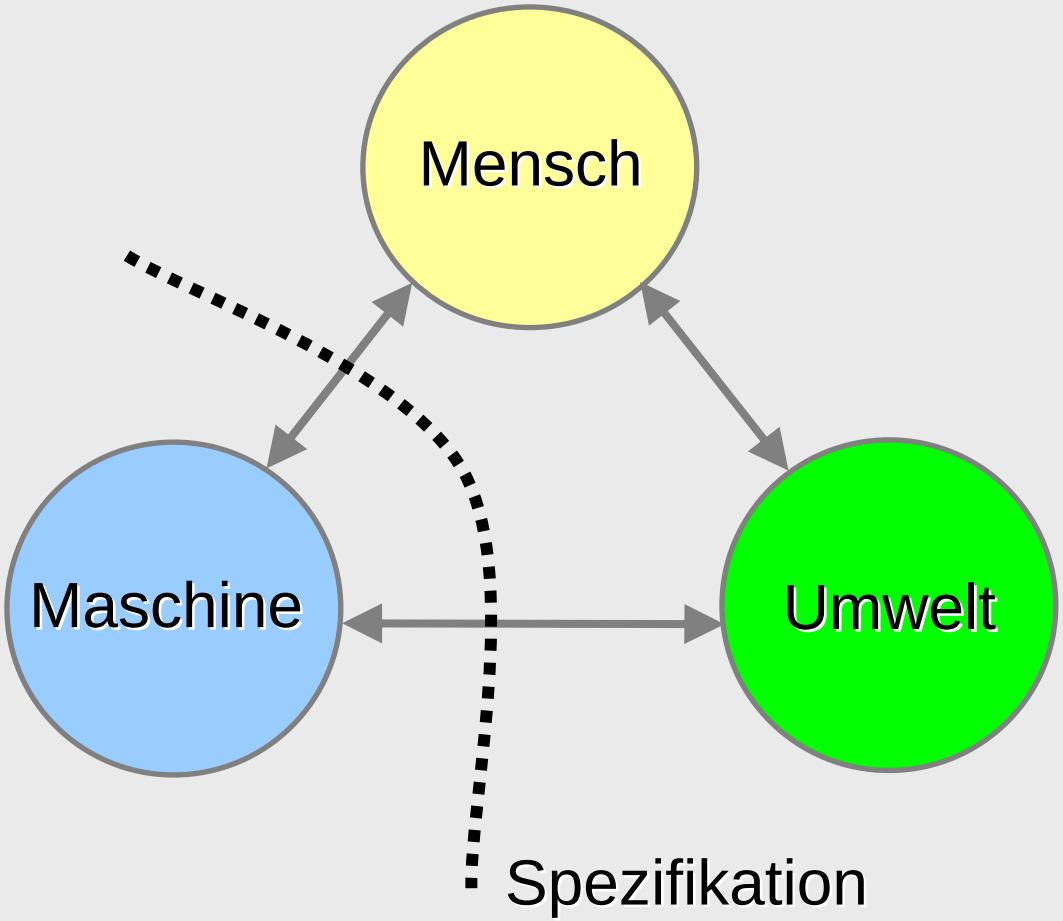
# Klassifizierungsschema

| *Phase* | *Beschreibung* |
|---|---|
| 1 | *Spezifikation*<br>- Funktionale Sicherheitsanforderungen<br>- Sicherheitsbezogene Zuverlässigkeitsanforderungen |
| 2 | *Realisierung*<br>- Entwurf<br>- Implementierung (Codierung)<br>- Validierung und Verifizierung |
| 3 | *Installation und Inbetriebnahme* |
| 4 | *Betrieb und Wartung* |
| 5 | *Änderungen nach Inbetriebnahme*<br>- Modifikation, Wiederverwendung und Nachrüstung<br>- Außerbetriebnahme |

# Klassifizierung der Ursachen
## (HSE-Studie, 34 Unfälle)



# Fehlerklassifizierung
## Mensch-Maschine-Umwelt-System



# Spezifikation = Norm

Fehler = Verstoß gegen die Spezifikation

Technischer Fehler

Maschine erfüllt die Spezifikation nicht.

Bedienfehler

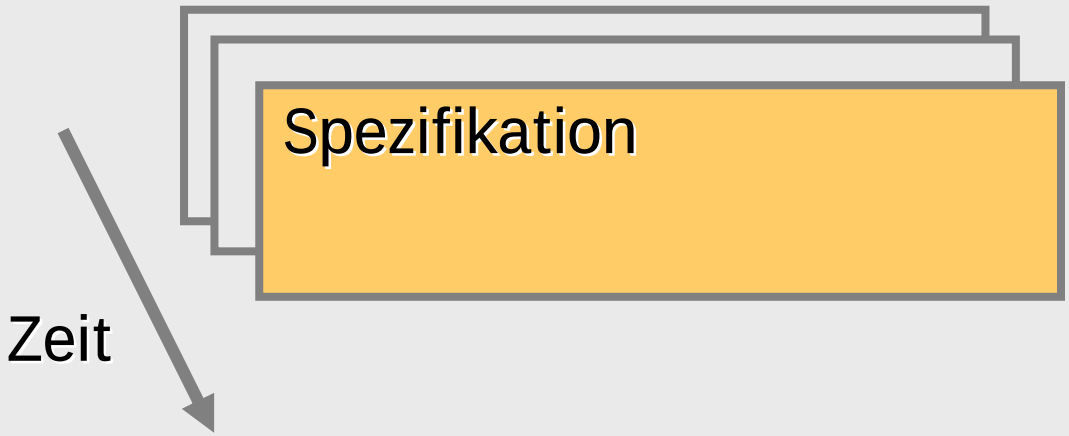
Mensch hält sich nicht an die Spezifikation („Bedienungs-anleitung“).

# Fehlerklassifizierung

- Spezifikationsfehler (nicht existent a priori)
- Technischer Fehler
  - Eingebauter Fehler (inhärenter Fehler)
    - Konstruktionsfehler
      - Programmierfehler
      - Entwurfsfehler in der Hardware
    - Fertigungsfehler
  - Ausfall (physikalischer Fehler)
- Bedienfehler (fälschlich: „menschliches Versagen“)

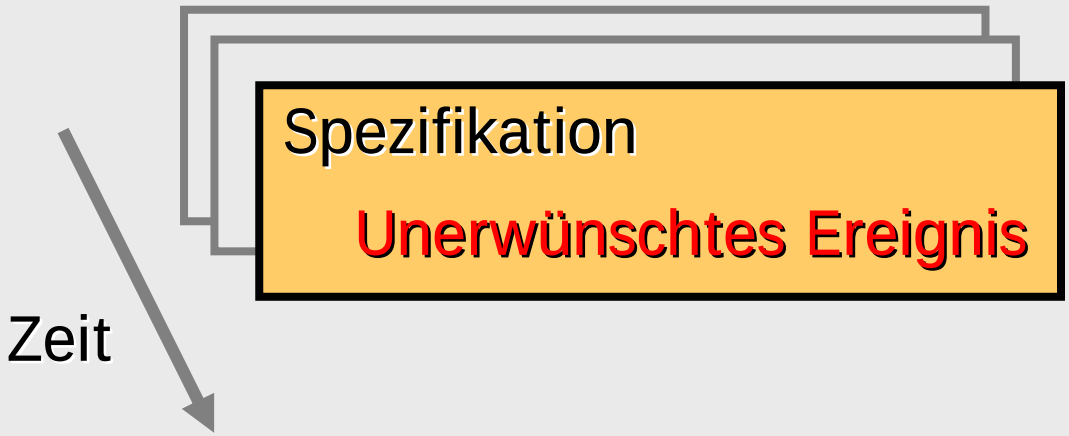
# Spezifikationsfehler sind Fehler a posteriori

Spezifikationsfehler werden in einem Prozess des *Versuchs und der Fehlerbeseitigung* (trial and error) sichtbar. Sicherheitsspezifikationen ändern sich mit der Zeit.



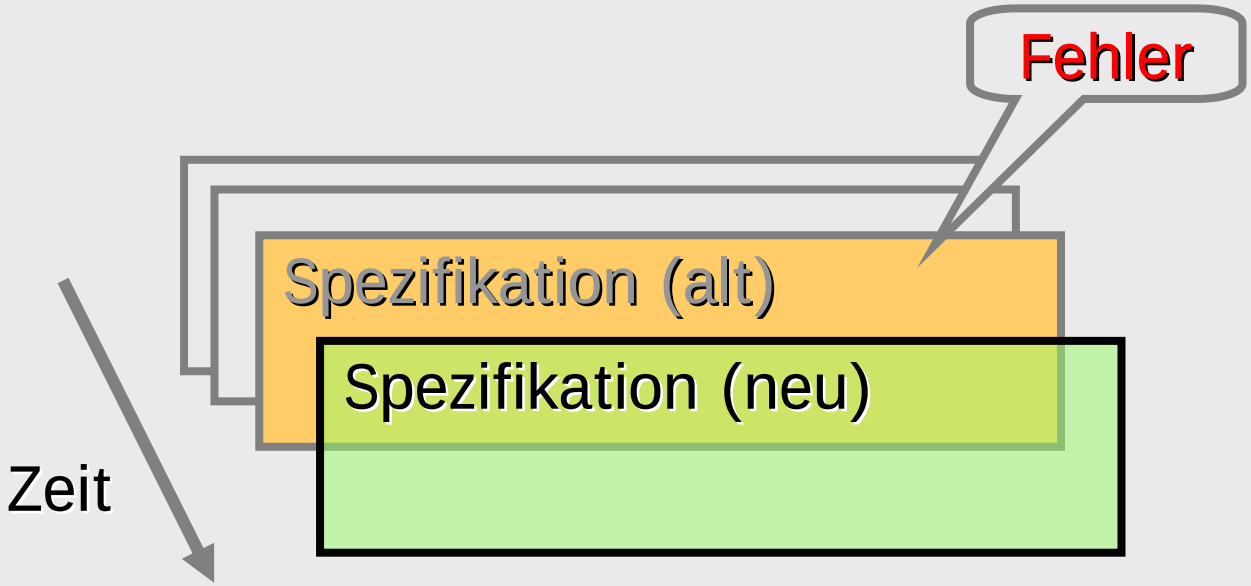
# Spezifikationsfehler sind Fehler a posteriori

Spezifikationsfehler werden in einem Prozess des *Versuchs und der Fehlerbeseitigung* (trial and error) sichtbar. Sicherheitsspezifikationen ändern sich mit der Zeit.



# Spezifikationsfehler sind Fehler a posteriori

Spezifikationsfehler werden in einem Prozess des *Versuchs und der Fehlerbeseitigung* (trial and error) sichtbar. Sicherheitsspezifikationen ändern sich mit der Zeit.



# Zuverlässigkeitsvorhersagen?

Fehler im Sinne der Zuverlässigkeit und Sicherheit sind Verstöße gegen die *gültige* Spezifikation.

Eine in der Zukunft liegende Änderung der Spezifikation kann *nicht* in Rechnung gestellt werden.

Spezifikationsfehler bilden *keine* Fehlerkategorie im Sinne der technischen Zuverlässigkeit.

Damit fällt etwa die Hälfte aller Fehler aus der Zuverlässigkeitsbetrachtung heraus!

# Zuverlässigkeitsvorhersagen?

Karl Raimund Popper:

Wir können nicht wissen, was wir in Zukunft wissen werden. Denn sonst wüssten wir es ja bereits jetzt.

(Das soll uns nicht entmutigen - aber kritisch gegenüber Prognosen jeder Art machen!)

# Die wichtigeren Fragen ...

Wie lernen wir möglichst viel aus Fehlern der Vergangenheit?

Welche Methoden hätten zur Vermeidung der Unfälle der Vergangenheit beitragen können?

Mit welchen (auf die Lebenszyklen bezogenen) Methoden lassen sich Unfälle zukünftig am besten vermeiden?

# … und eine Antwort

Klassifizierung und Bewertung präventiver Methoden mittels Grundursachenanalyse.

Das wird im Aufsatz an zwei Beispielen durchexerziert.

- Strandung des Kreuzfahrtschiffs „Royal Majesty“
- Scheitern des Jungfernflugs der Ariane 5

Weiteres Material ist aus einer HSE-Studie

# Klassifikation der Methoden auf Basis der Ursachenanalyse

| ***Lebenszyklusphase*** | | ***Methoden*** |
|---|---|---|
| **1** | *Spezifikation:* Funktionale Sicherheitsanforderungen, sicherheitsbezogene Zuverlässigkeitsanforderungen | Hazards and Operability Analysis (HAZOP)<br>Fehlerbaumanalyse (FTA)<br>Ereignisbaumanalyse (ETA) |
| **2.1** | *Realisierung:* Entwurf | Modularisierung: Hierarchische Zerlegung, Information Hiding |
| **2.2** | *Realisierung:* Implementierung (Codierung) | Modularisierung: Schnittstellenbeschreibungen wie Spezifikationen behandeln, mathematische und logische Notation von äußerster Genauigkeit verwenden, Dokumentation |
| **2.3** | *Realisierung:* Validierung und Verifizierung | Schnittstellenanalyse: Strenge Modulprüfung (werden alle Anforderungen erfüllt?)<br>Fehlereffektanalyse (FMEA)<br>Grenzwertanalyse, Äquivalenzklassentest |
| **3** | *Installation und Inbetriebnahme* | Inspektion und Funktionstests müssen so explizit wie möglich spezifiziert werden |
| **4** | *Betrieb und Wartung* | Design for Maintenance |
| **5** | *Änderungen nach Inbetriebnahme:* Modifikation, Wiederverwendung und Nachrüstung | Dieselben wie in der Realisierungsphase (Entwurf, Implementierung, Validierung und Verifizierung)<br>Regressionstests, Schnittstellenanalyse |

# Schlussfolgerungen in Kürze

Root Cause Analysis (Grundursachenanalyse)

liefert Aussagen zur Fehleranfälligkeit der Lebenszyklusphasen;

zeigt die Wirksamkeit von SRE-Methoden bzgl. der Phasen.

Spezifikationsfehler sind die häufigsten Fehler.

Zuverlässigkeitsvorhersagemethoden sind blind gegenüber Spezifikationsfehlern.

Anspruchsvolle SRE-Methoden wie Diversität und beweis-geleitetes Programmieren sind gegen Spezifikationsfehler machtlos.

Wirksame Methoden sind HAZOP, FMEA, FTA, ETA, genaue Spezifizierung und Dokumentation, Modularisierung.