Sicherheit und Zuverlässigkeit in der Software-Entwicklung

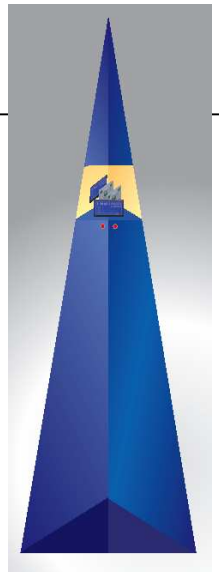
Sergio Montenegro

sergio@first.fhg.de

Holger Schlingloff

Holger.Schlingloff@first.fhg.de

Konzeption & Festlegung der Requirements



FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Was haben wir letztes Mal gemacht?

Was ist Verlässlichkeit

Gefahren

Beispiele von Unfällen

- Mars Polar Lander
- Berliner Feuerwehr

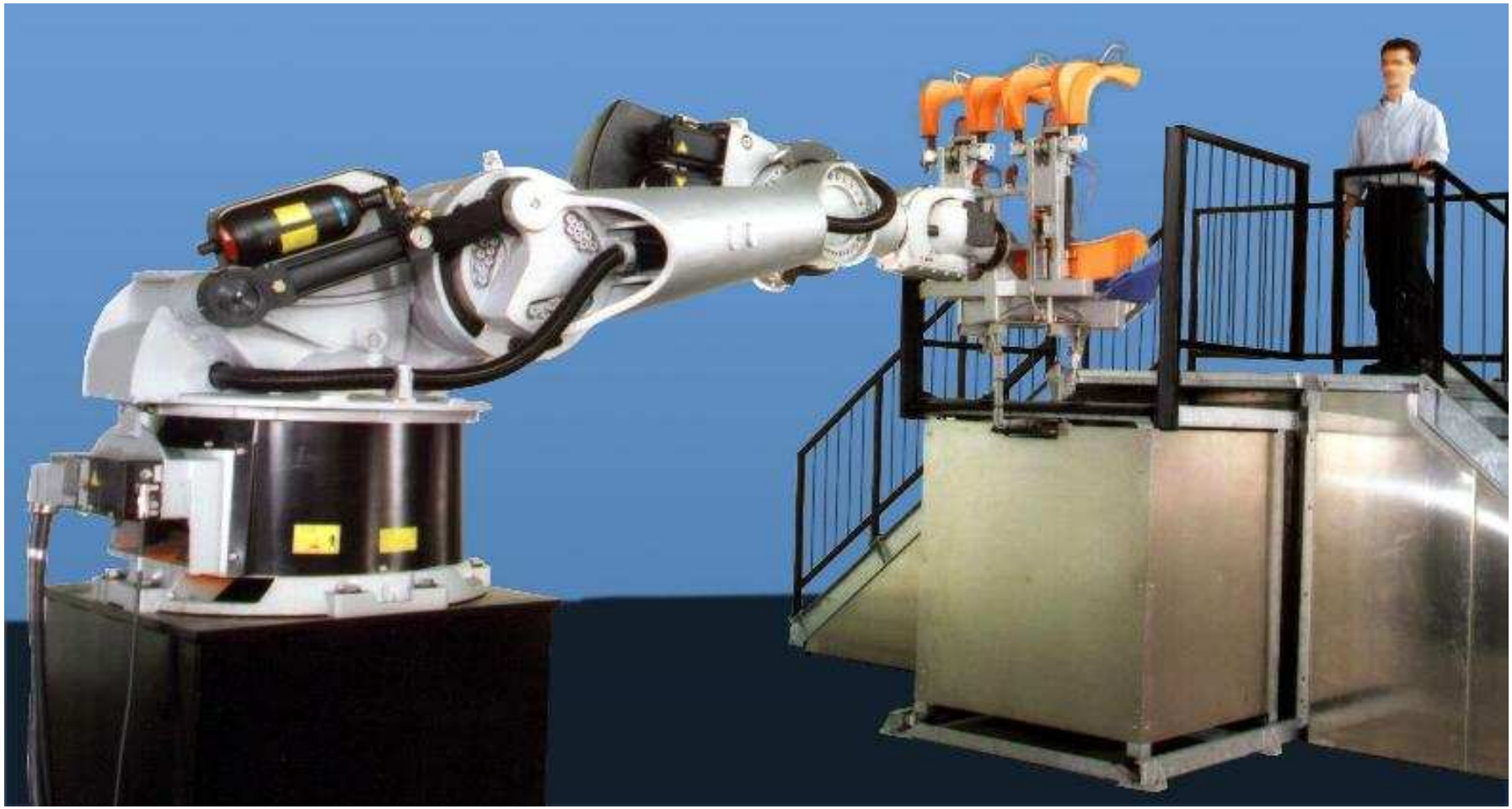
Unfallursachen

Inhalt für dieses Mal

- Fehlerbehandlung
- Fehlerauswirkungen, Fehlerfortpflanzung
- Fehlerprävention
- Begriffsbildungen Fehler
- Begriffsbildungen Sicherheit
- Normengerechte Vorgehensweisen

- Systemspezifikation und Anforderungsanalyse
- Analyseteam
- Differenz zwischen Requirements und Realisierung
- Konflikte und Lösungen im Anforderungsprozess
- Lasten- und Pflichtenhefte für eingebettete Systeme

Vertrauen...



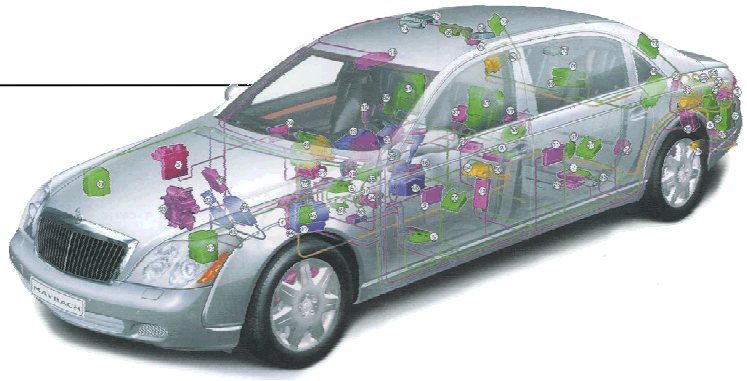
Warum funktionieren so viele komplexe Systeme nicht verlässlich?

Weil sie komplex sind!

Und die komplexeste Komponente ist...

So komplex, dass NIEMAND sie wirklich kennt oder versteht!

Software ist...



- **überall**
  integraler Bestandteil von Geräten / technischen Systemen

- **groß und komplex**

SAP/R3:	> 80.000.000 LOC (Zeilen Programmcode)
Microsoft Windows XP:	> 45.000.000 LOC (NT 2000: 20 M LOC)
Linux	> 30.000.000 LOC (Jun 2001 RH-7.1, RH-6.2: 15M)
Bank-Applikation:	> 3.000.000 LOC
Automobil:	> 2.000.000 LOC (High Dependability??)
Mobilfunktelephon:	> 1.000.000 LOC

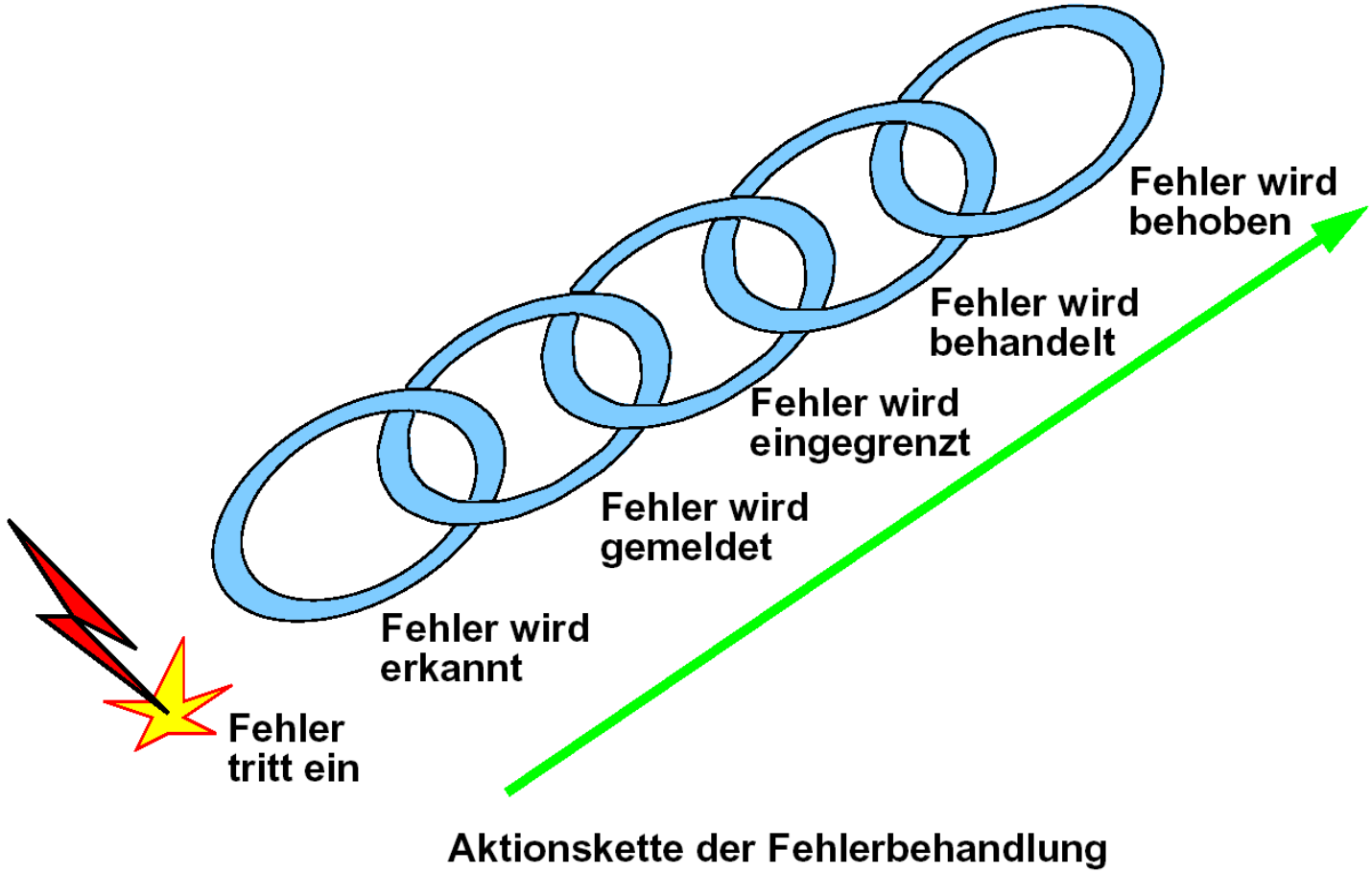
- **sehr vielgestaltig**

- **langlebig und anpassbar**

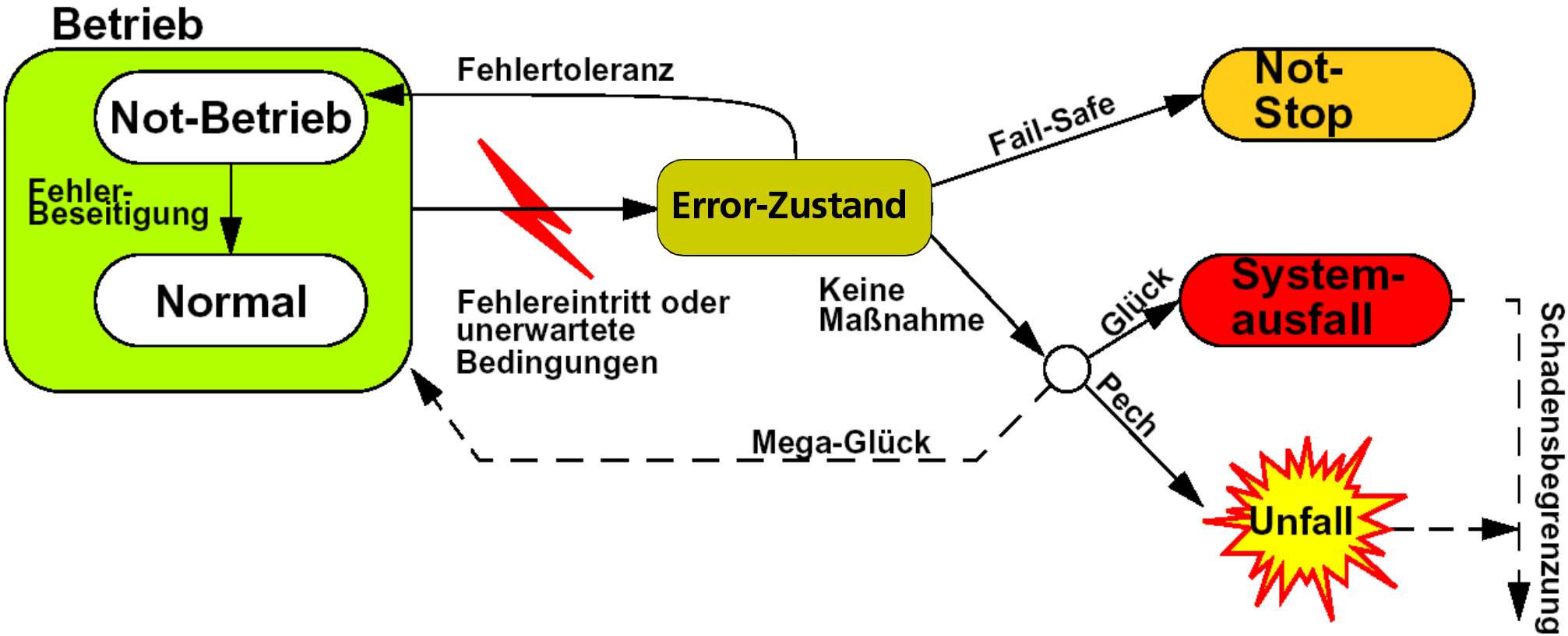
Software wird immer mehr zur qualitätsbestimmenden Komponente komplexer Systeme.

Quelle: nach SQ-Lab, FHG FIRST, 2003

Fehlerbehandlung Run-Time

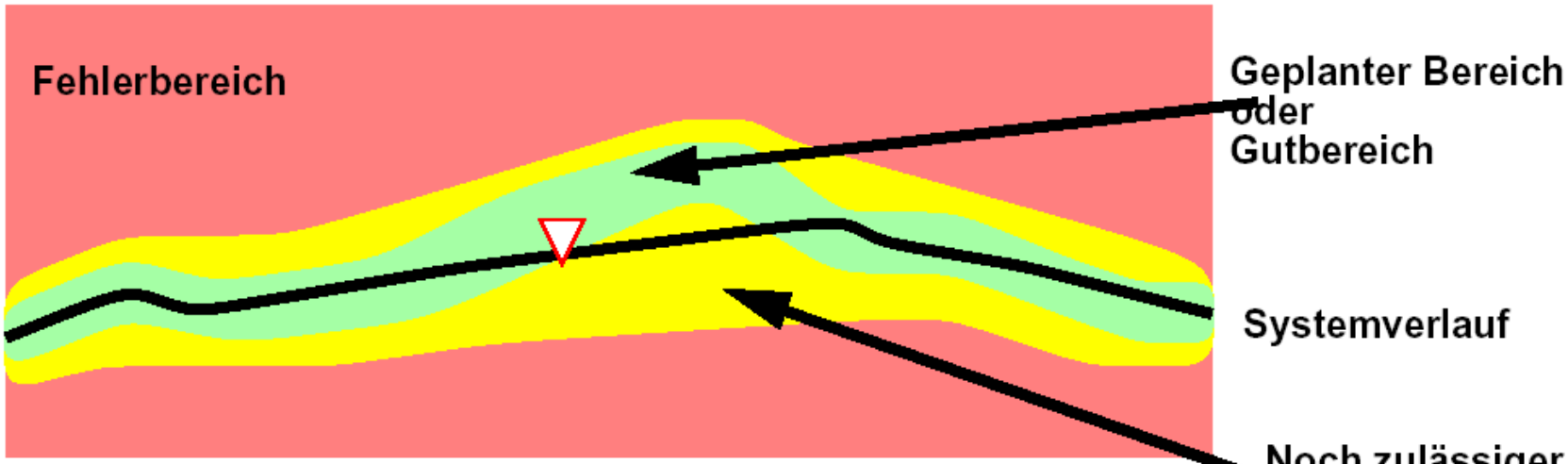


Fehler und Unfall



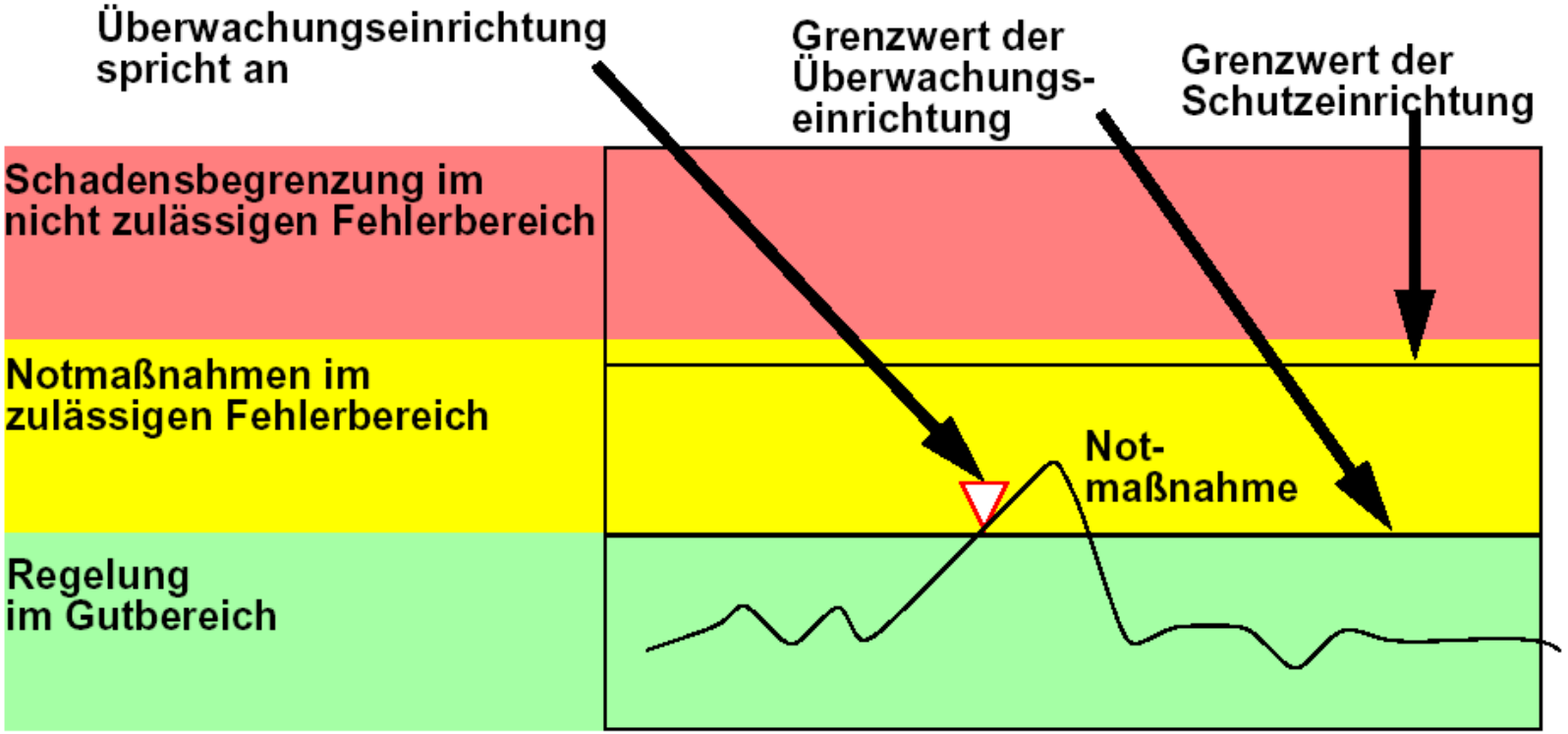
Der direkte Weg vom Fehler zum Unfall

Schmale Korridor



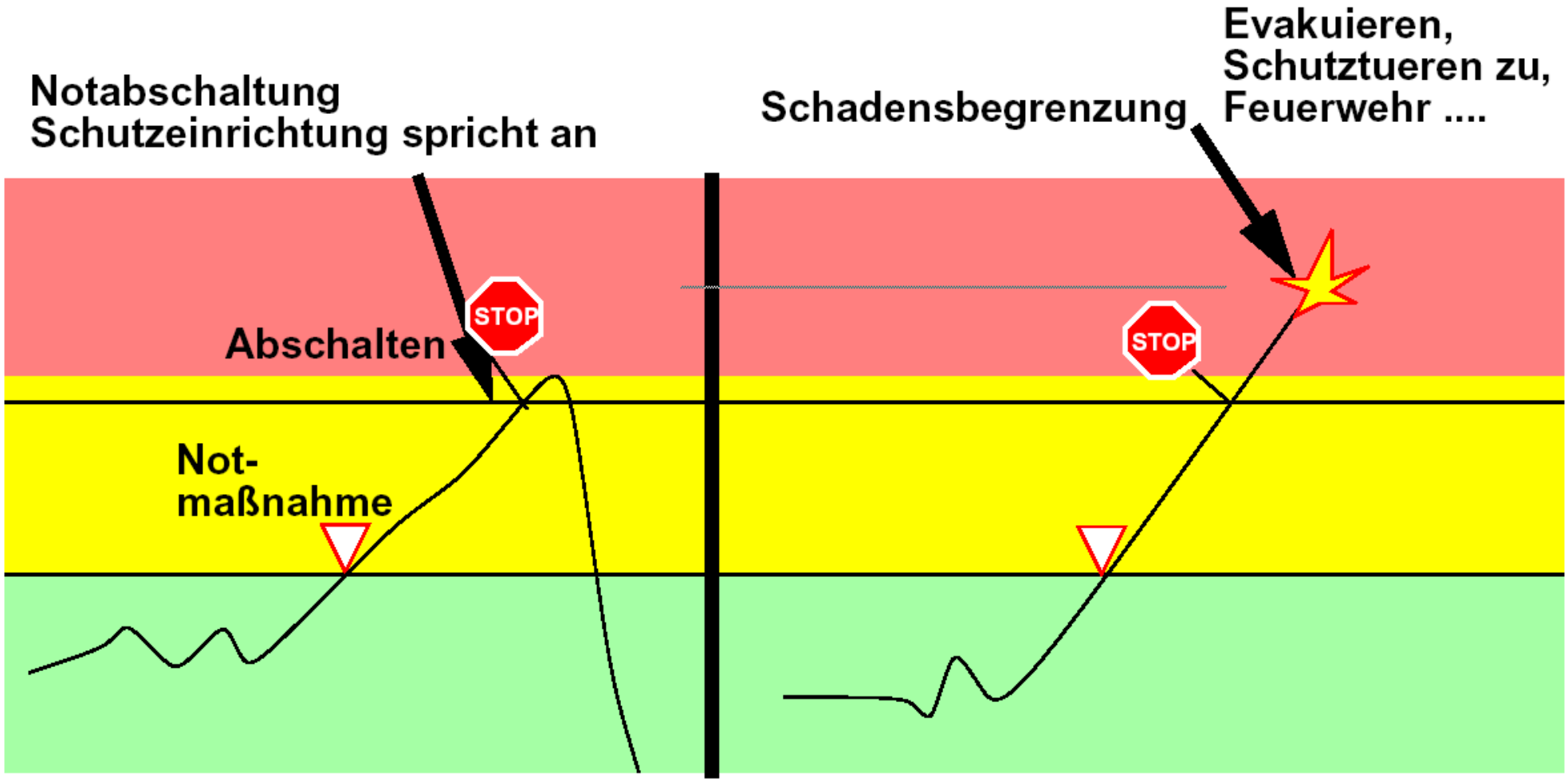
Der schmale Korridor vom geplanten Bereich

VID/VDE 2180



Überwachungseinrichtung nach VDI/VDE 2180

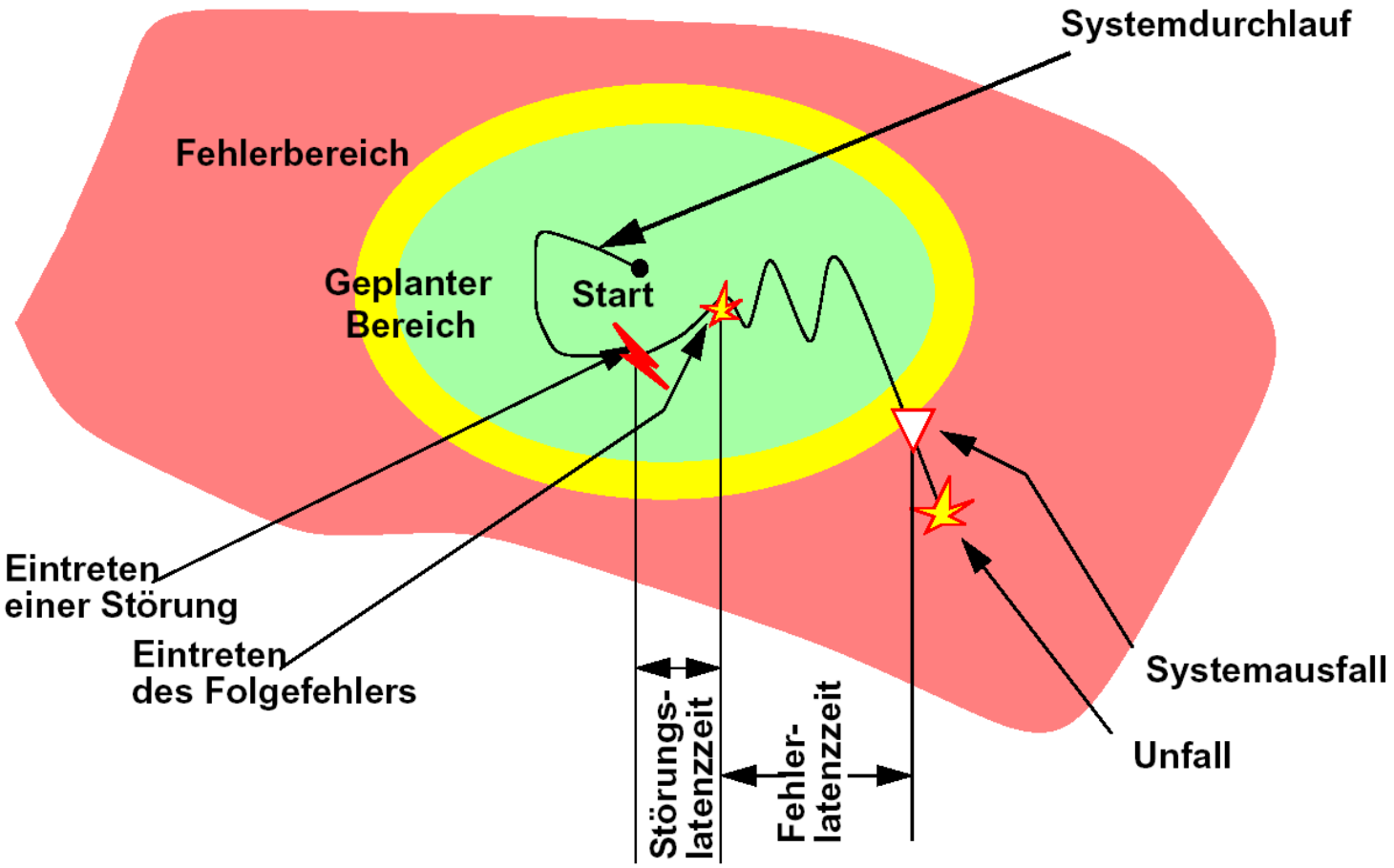
VID/VDE 2180



Wirkung von Schutzeinrichtungen nach VDI/VDE 2180

Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik

Von der Störung zum Unfall: Erkennung...



Von der Störung zum Unfall : Latenzzeiten

"The certain way to be wrong is to think you control it"

Möglichkeiten für Sicherheit:

1. **Fehler Prävention**
2. Fehler Erkennung
3. Fehler Behandlung
   - Meldung
   - Eingrenzung
4. Recovery
5. Fehler Beheben

Konzeption

1. Prävention:

- Hardware:
  - Entwicklungsfehler:
    - Simulation,
    - Reviews
    - Tests
    - Formale Methoden & Analyse
  - Ausfälle:
    - Abschirmung
    - Burn-in Phase

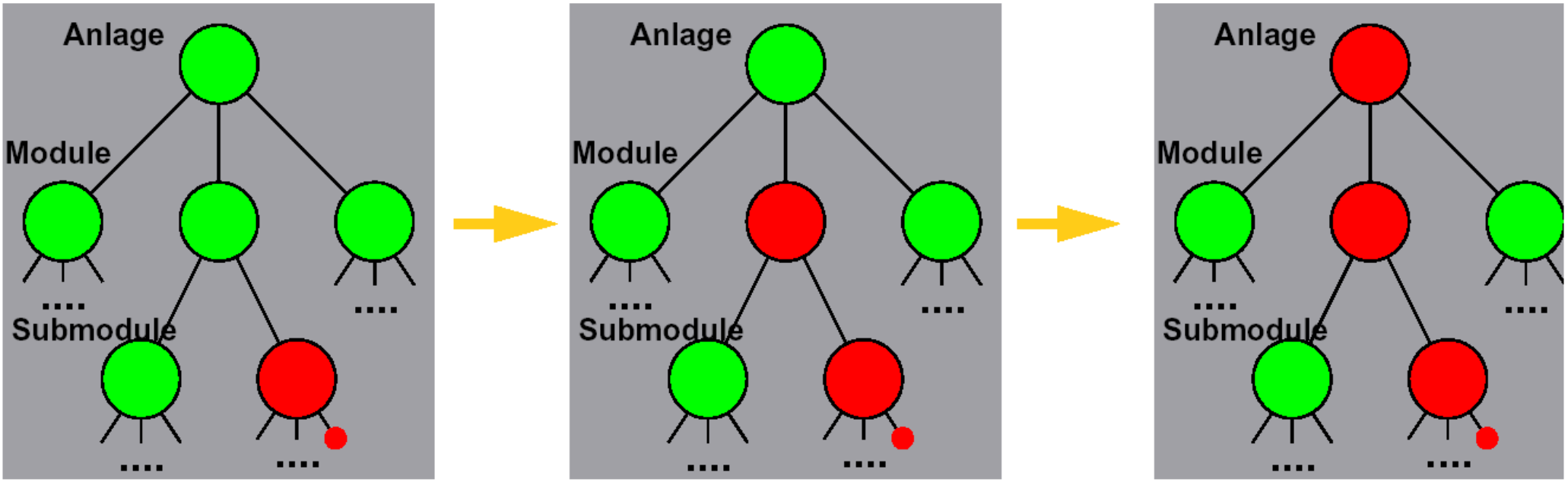
- Software:
  - Entwicklungsfehler
    - Reviews
    - Ausführliches testen
    - Test-Suite
    - Programmier Richtlinien
    - Formale Verifikation
    - Online Dokumentation & Diagramme
    - Einfachheit
  - Unerwartetes
    - ... Mehr denken? ....

"The certain way to be wrong is to think you control it"

Möglichkeiten für Sicherheit:

1. Fehler Prävention
2. **Fehler Erkennung**
3. Fehler Behandlung
   - Meldung
   - Eingrenzung
4. Recovery
5. Fehler Beheben

Fehlerfortpflanzung

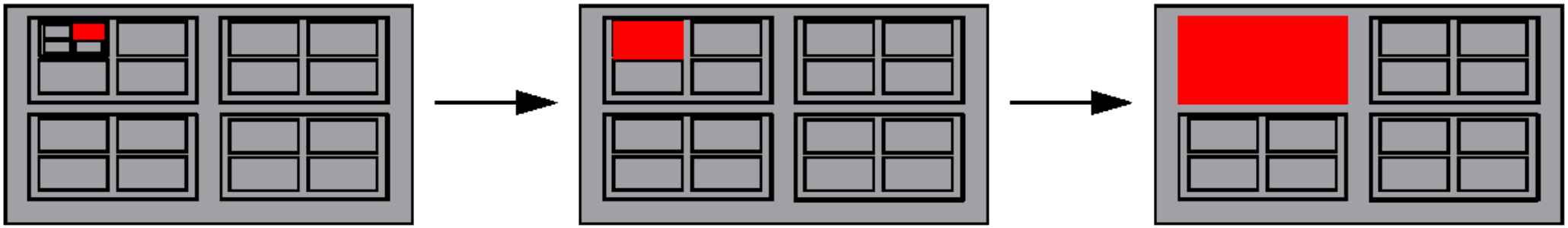


Funktionierendes Modul

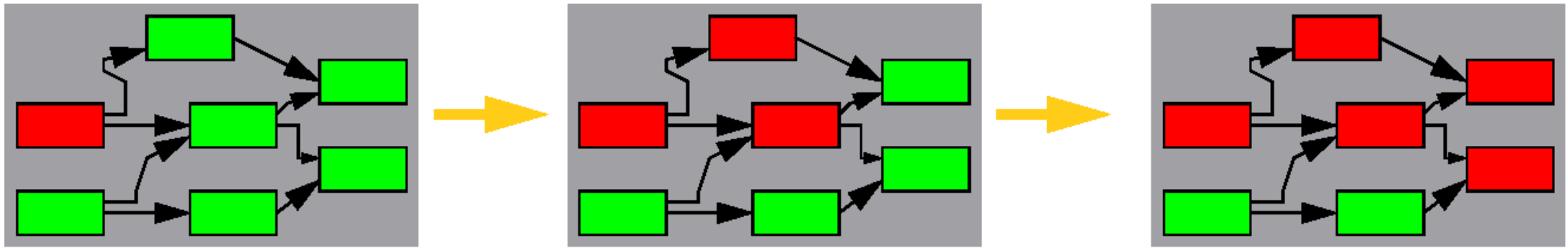
Ausgefallenes Modul

Fehlerfortpflanzung in der Modulhierarchie

Fehlerfortpflanzung



Fehlerfortpflanzung



System-Daten-Fluß.

Funktionierendes Modul

Ausgefallenes Modul

Fehlerfortpflanzung im Moduldatenfluß

Verlässlichkeit & Fehler / Ausfälle

Störung oder Irrtum -> Fehler -> Ausfall -> Störung -> Fehler -> Ausfall -> ....

Störung oder Irrtum -> Fehler -> Ausfall -> Unfall

Die kleinste Störung, wie z.B. ein Alpha-Teilchen trifft eine Speicherzelle und kann einen Fehler verursachen, z.B. ein Speicherbit kippt um.

Der Fehler kann einen Ausfall verursachen, beispielsweise wegen eines bit-Kippers im Programmcode verläuft sich die Steuerung und bleibt stehen.

Deswegen fällt ein Steuerrechner aus und man kann das Flugzeug nicht mehr lenken - es stürzt ab.

Und alles wegen eines subatomischen Partikels!

3. Behandlung

Umschaltung der Reserve-Bordrechner

Automatische Umschaltung zwischen aktivem und Monitor-Bordrechner (nach Fehlererkennung)

Redundante Busse

Schatten Speicher + Parity: Daten dupliziert

FPGA-Logic wird zyklisch und nach Fehlererkennung neu geladen.

Watchdog: Reset

2 Software Versionen (mit checksum)

4. Recovery

Wichtige Daten im Flash/EEPROM (Context memory) retten

Integration in Kommunikationsnetzwerk

Aktuellen Daten werde von dem (redundanten) laufenden Bordrechnern empfangen.

Aktueller Zustand wird erfaßt