Agenda

		Welcome DESY	Hott
		Motivation	Trapp
		Basic Terms: - Safety, Reliability, Risk - Fault, Error, Failure - …	Trapp
		Engineering Reliable Systems	Trapp
12:30	14:00	Lunch break	---
		Stochastic Processes	Kemmann
		FTA (CFT-Addon)	Kemmann

System Reliability Analysis

A Motivation

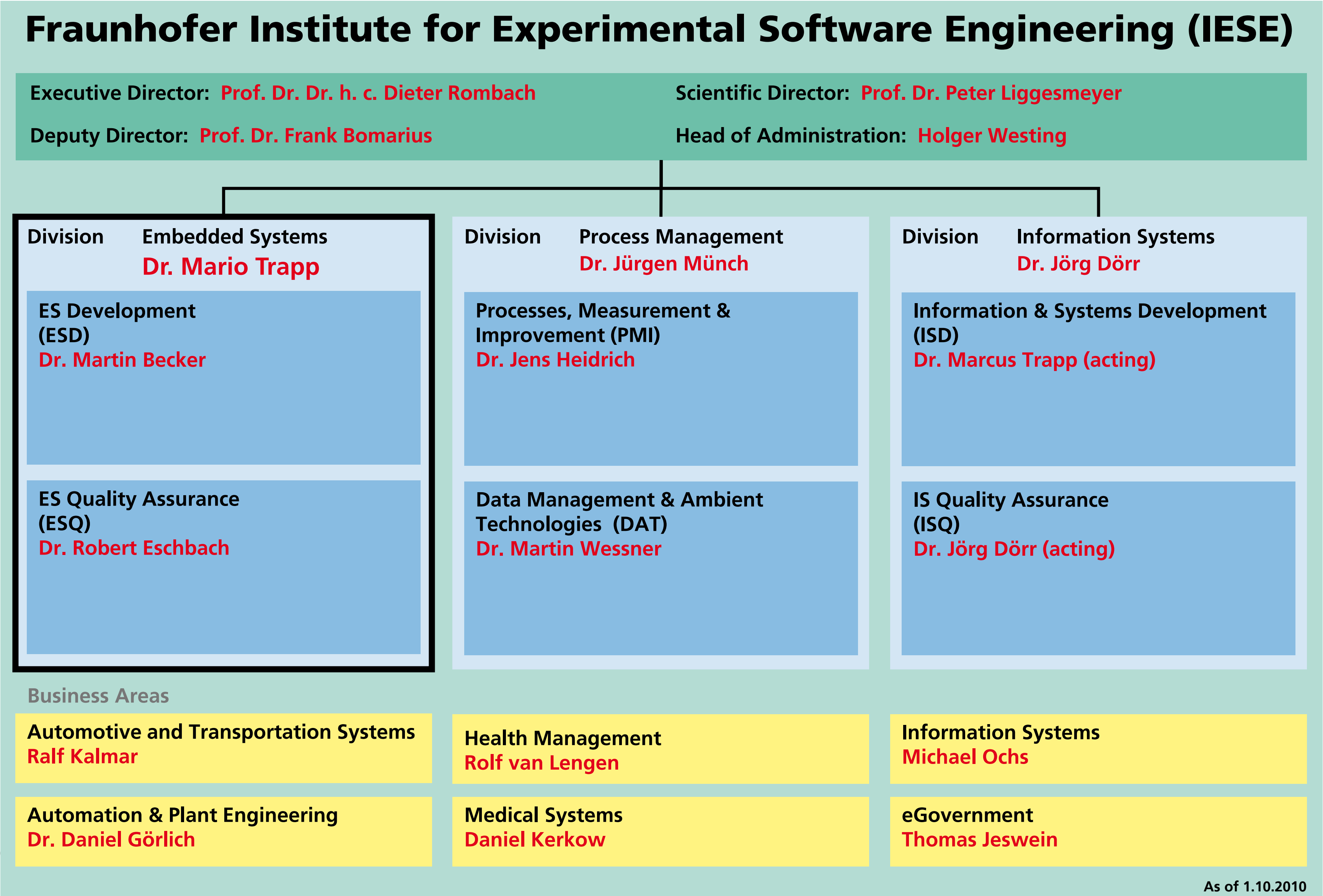
Dr. Mario Trapp

Fraunhofer IESE

mario.trapp@iese.fraunhofer.de

Fraunhofer
IESE

My Role @ IESE



Fraunhofer Institute for Experimental Software Engineering (IESE)

Executive Director: Prof. Dr. Dr. h. c. Dieter Rombach

Scientific Director: Prof. Dr. Peter Liggesmeyer

Deputy Director: Prof. Dr. Frank Bomarius

Head of Administration: Holger Westing

Division Embedded Systems
Dr. Mario Trapp

ES Development
(ESD)
Dr. Martin Becker

ES Quality Assurance
(ESQ)
Dr. Robert Eschbach

Division Process Management
Dr. Jürgen Münch

Processes, Measurement & Improvement (PMI)
Dr. Jens Heidrich

Data Management & Ambient Technologies (DAT)
Dr. Martin Wessner

Division Information Systems
Dr. Jörg Dörr

Information & Systems Development
(ISD)
Dr. Marcus Trapp (acting)

IS Quality Assurance
(ISQ)
Dr. Jörg Dörr (acting)

Business Areas

Automotive and Transportation Systems
Ralf Kalmar

Automation & Plant Engineering
Dr. Daniel Görlich

Health Management
Rolf van Lengen

Medical Systems
Daniel Kerkow

Information Systems
Michael Ochs

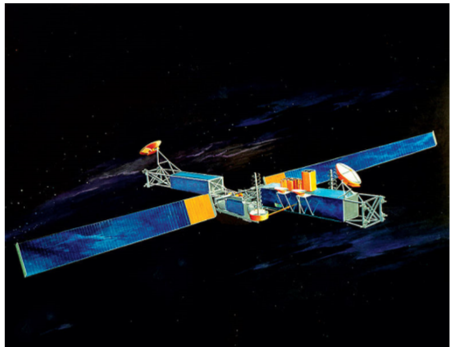
eGovernment
Thomas Jeswein

As of 1.10.2010

Loss of MARS Climate Orbiter (MCA)

- MCA should to enter orbit at an altitude of 140.5–150 km (460-500 k ft.)
- It entered orbit at 57 km (190,000 ft.) and was destroyed
- Reason:
  - The contractor for the craft's thrusters did use english units
  - NASA did use SI units
  - Instead of 150.000 m the craft was set to a target altitude of 150.000 ft
- Typical problem in complex systems
  - Single modules completely reliable
  - Composition of modules leads to failures
  - Most important reasons:
    - Missing or wrong specifications
    - Misssing analyses of integrated system

Milstar Satellite



- Should be placed to geosynchronous orbit
- A role rate filter was constantly zeroed
- Lead to a loss of roll axis control → yaw and pitch control
- Satellite was placed in much too low, unusable orbit

- Reasons:
  - A roll rate filter was included in the beginning of the project
  - Later it was decided not to use the filter
  - For consistency reasons the software code remained in the system but was set to a constant: zero.

  ➔ Typical problem: errors through changes

Ariane 5

- maiden flight of the Ariane 5 launcher ended in failure
- 30s after lift off:
  complete loss of guidance and altitude information
- 40 s after initiation of the flight sequence:
  Ariane 5 veered off its flight path, broke up, and exploded

- Reason:
  - Reused sofware module from Ariane 4
  - Not needed for Ariane 5, but reused for „commonality“
  - Not caught overflow exception led to failure of all redundant channels

Lufthansa-Crash in Warsaw

- reverse thrust can only be enabled if aircraft is on ground
- „aircraft on ground“ ≡ weight on both landing gears > *12* t
- delay of touch down of second LG due to strong side winds
- reverse thrust could not be activated
- system was completely correct but not safe

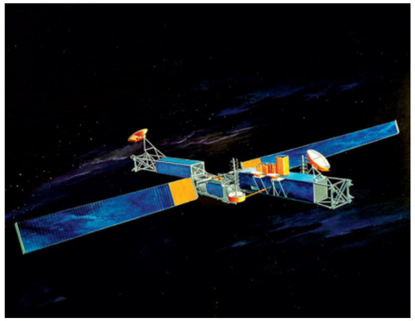
„Robot cannon kills 9, wounds 14“

- Robot guns automatically
  - Pick out targets
  - slew into position
- Human only has to “pull the trigger”
- Due to a “computer” failure, the cannon started shooting while turning around

The role of software – Some more examples

2003	3	Software failure contributes to power outage across the Northeastern U.S. and Canada.
2001	5	Panamanian cancer patients die following overdoses of radiation, amounts of which were determined by faulty use of software.
2000	4	Crash of a Marine Corps Osprey tilt-rotor aircraft blamed on “software anomaly.”
1997	225	Radar that could have prevented Korean jet crash hobbled by software problem.
1997	1	Software-logic error causes infusion pump to deliver lethal dose of morphine sulfate. Gish Biomedical reprograms devices.
1995	159	American Airlines jet, descending into Cali, Colombia, crashes into a mountain. Jury holds maker of flight-management system 17% responsible. A report from Aeronautica Civil of the Republic of Colombia, digitized by the University of Bielefeld in Germany found that the software presented insufficient and conflicting information to the pilots, who got lost.
1991	28	Software problem prevents Patriot missile battery from picking up SCUD missile, which hits U.S. Army barracks in Saudi Arabia.
1985	3	Software-design flaws in Therac-25 treatment machine lead to radiation overdoses in U.S. and Canadian patients.

A Summary

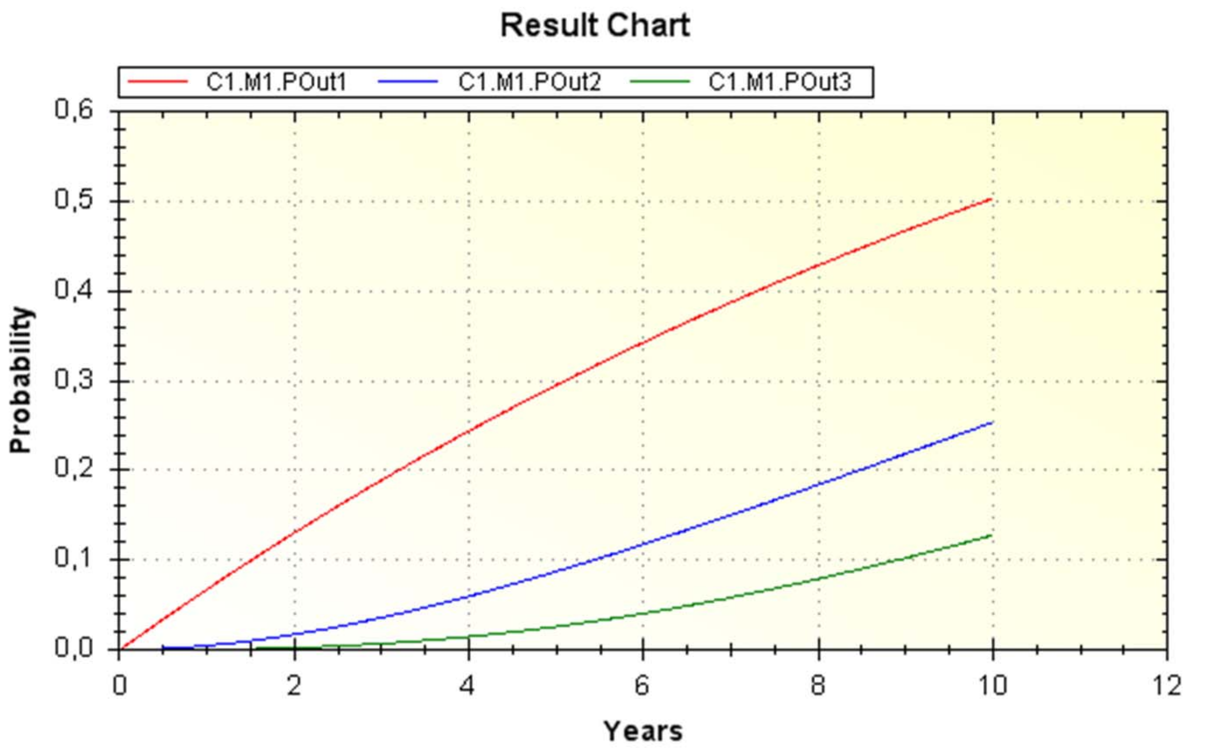


In most cases caused by

- hardly managed, underestimated **system complexity** without appropriate engineering and quality assurance processes
- erroneous **interaction** of sub systems
- wrong **reuse** of existing items in new context
- unsystematic **changes** of the system during and after development

Typical Misconceptions

1. Let‘s use redundancy



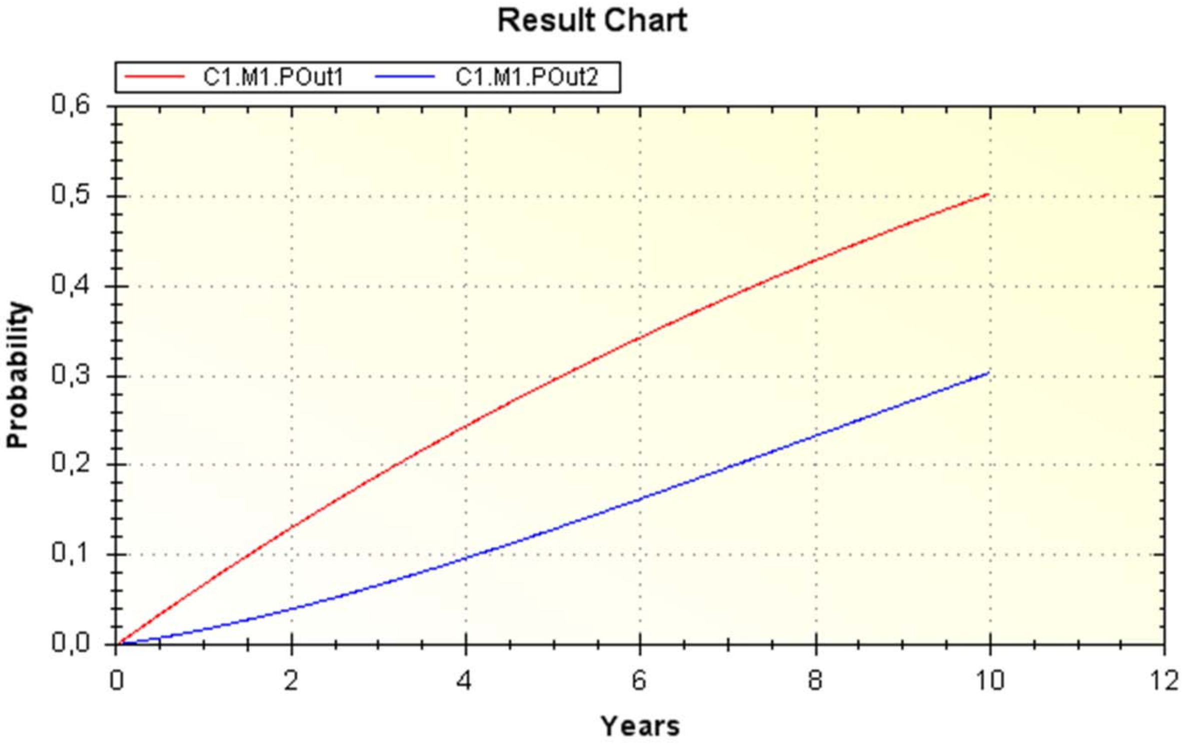
1 Channel

2 Channels

3 Channels

Typical Misconceptions (cont).

1. Let‘s use redundancy

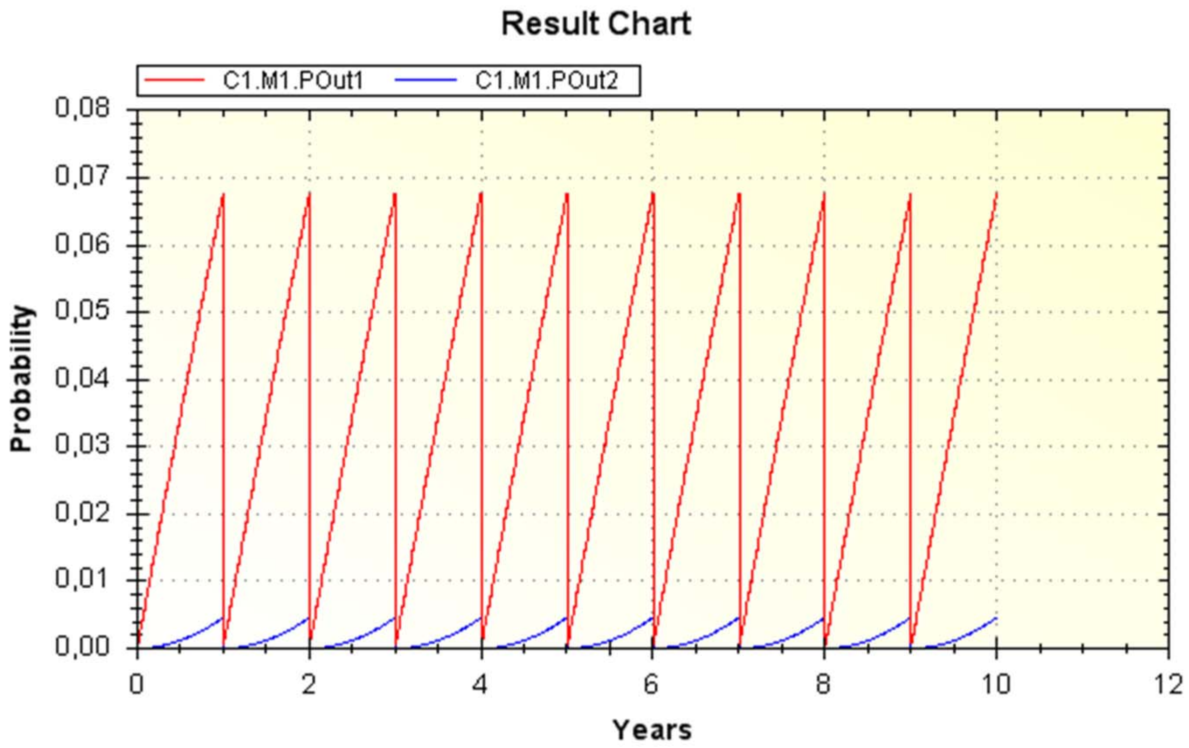


🟥 1 Channel

🟦 2 Channels with common causes

Typical Misconceptions (cont).

1. Let‘s use redundancy

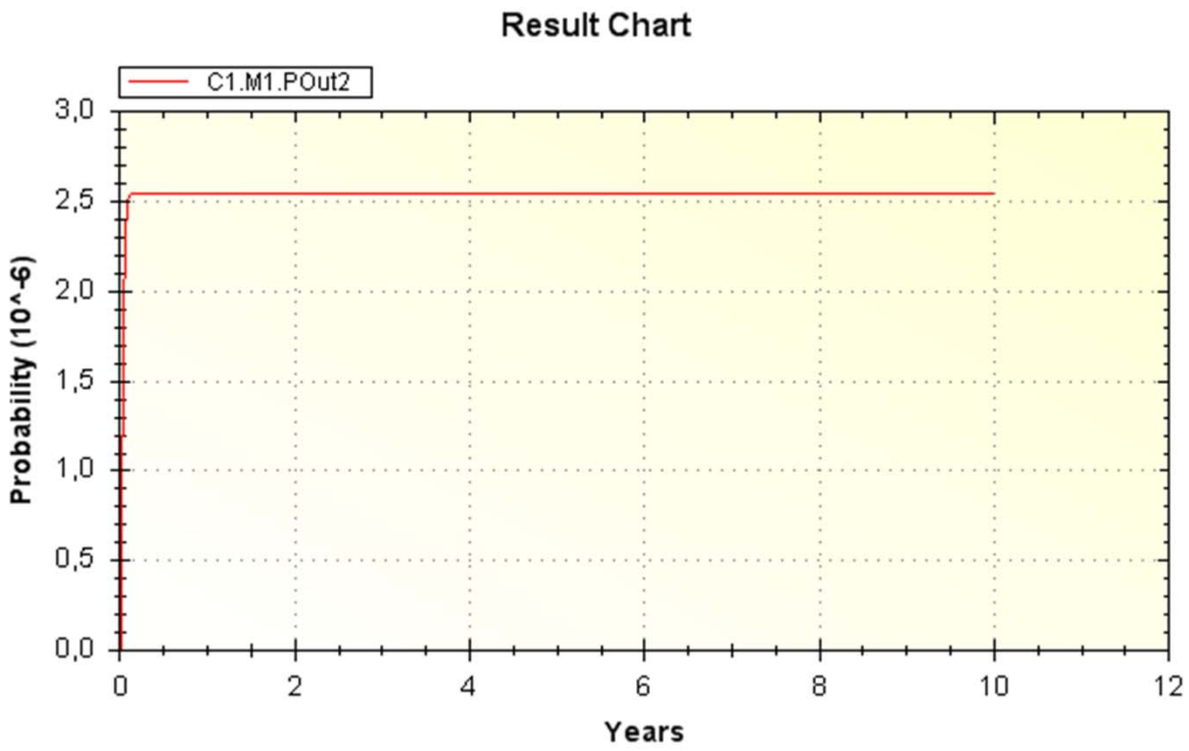


🟥	1 Channel
🟦	2 Channels

with regular replacement intervals

Typical Misconceptions (cont).

1. Let‘s use redundancy



——— 2 Channels

With fault revelation within one week

Typical Misconceptions (cont).

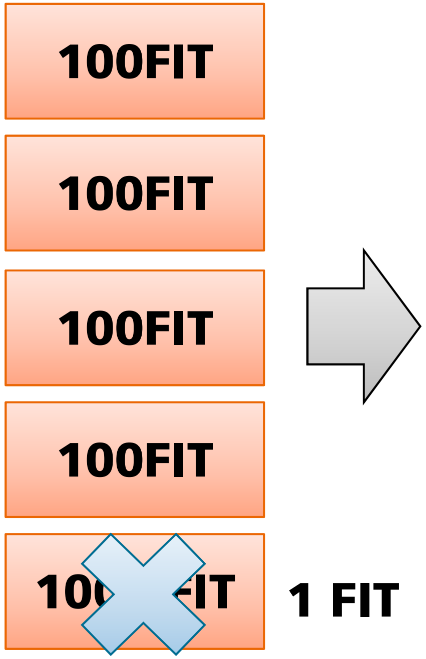
1. Let‘s use redundancy

- Redundancy $\nRightarrow$ independence
- Redundancy alone is usually not sufficient
- There is a series of approaches to be combined with redundancy
- Which approach is appropriate cannot be said without having analyzed the system and its failures modes

➔ Without analyses the necessity and effectiveness of measures remains unclear

Typical Misconceptions (cont).

2. Then let‘s use one of the long list of other counter measures



100FIT

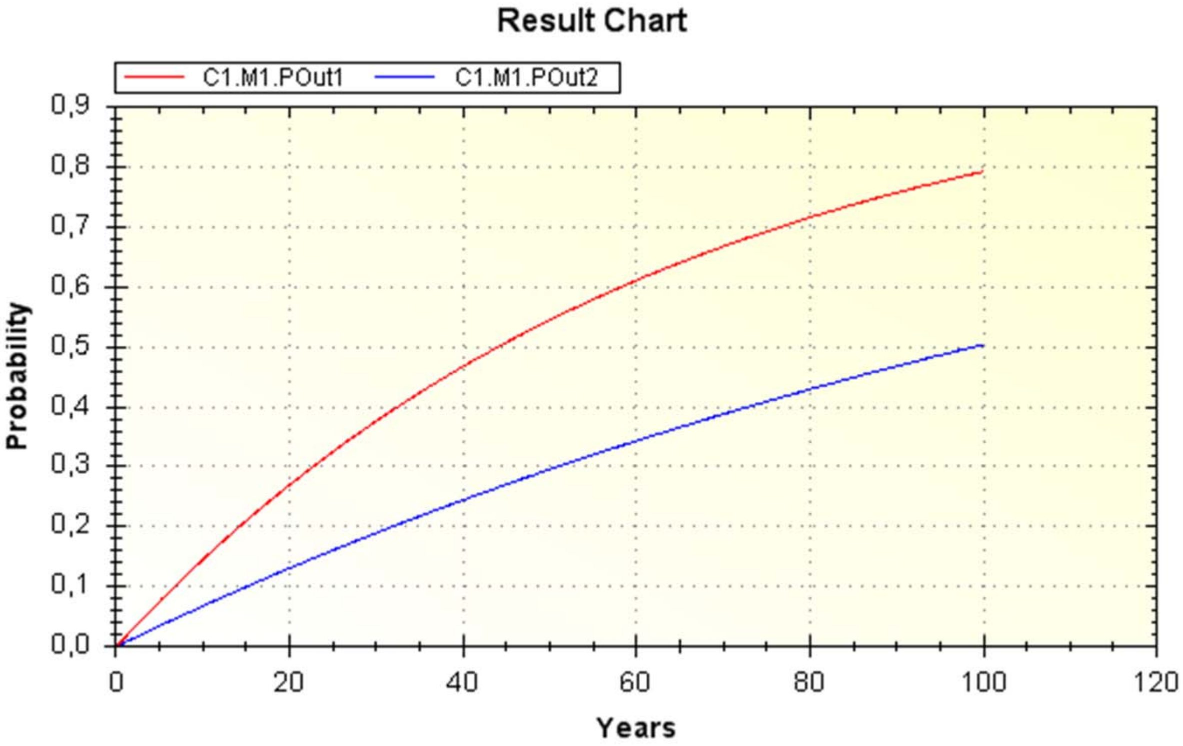
100FIT

100FIT

100FIT

~~100FIT~~

1 FIT



Typical Misconceptions (cont).

2. Then let‘s use one of the long list of other counter measures

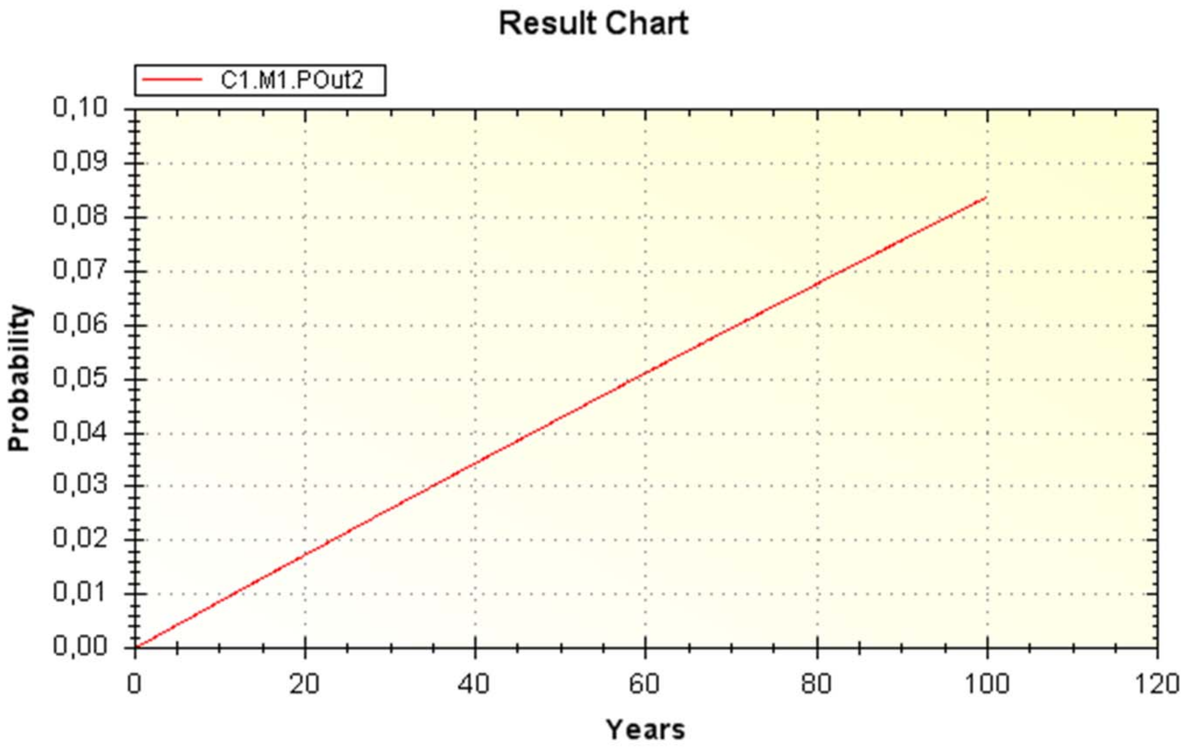
100FIT

100FIT

100FIT

100FIT

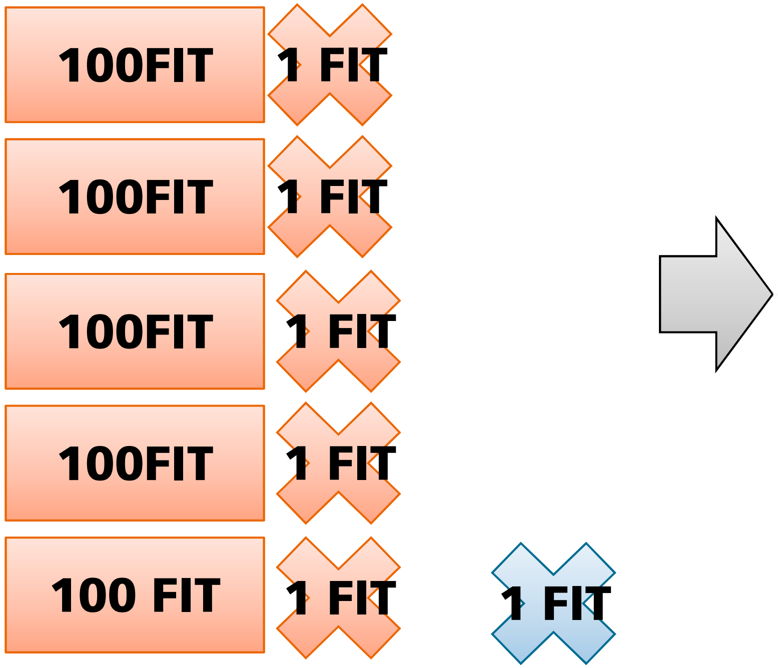
100 FIT



Typical Misconceptions (cont).

2. Then let‘s use one of the long list of other counter measures

Variant 1 Variant 2



100FIT

1 FIT

100FIT

1 FIT

100FIT

1 FIT

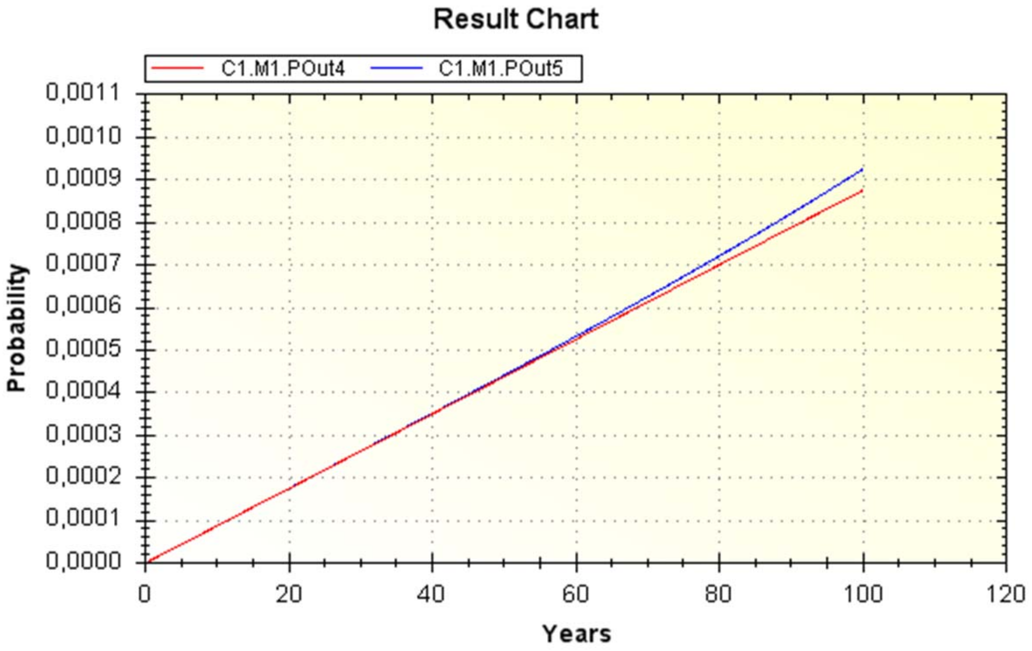
100FIT

1 FIT

100 FIT

1 FIT

1 FIT



Typical Misconceptions (cont).

2. Then let‘s use one of the long list of other counter measures

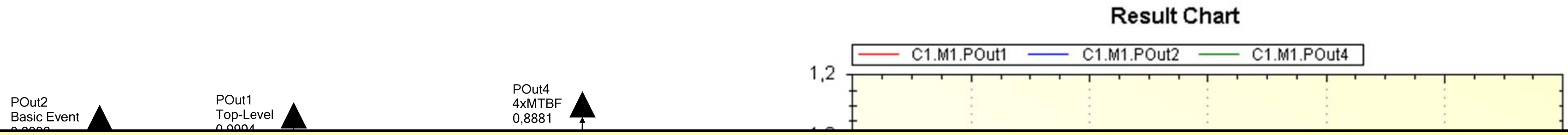
- Analyses are indispensable to
  - Understand the cause-effect-relationships
  - Understand the impact of component failures to system reliability
  - Identify the „big levers“ to efficiently apply counter measures

Typical Misconceptions (cont).

3. We know the MTTF/MTBF of our single parts – that‘s sufficient

$$MTBF = MTTF + MTTR$$

$$MTTF = E(t) = \int_{-\infty}^{\infty} t \cdot f(t)dt = \frac{1}{\lambda}\Big|_{\lambda=const}$$



- Calculation with MTTF (if at all) possible if and only if
  - exponential probability distributions are used
  - disjunctive combinations are used

- **in all other/most cases**
  - **failure rate and thus MTTF=E($\underline{t}$) (there is not *the* MTTF-value)**
  - **calculations are wrong and produce misleading results**

Typical Misconceptions (cont).

3. We know the MTTF/MTBF of our single parts – that‘s sufficient

- We are not interested in the reliability of parts
- But we are interested in the reliability of the systems

➔ Without appropriate analyses the reliability of complex systems cannot be estimated / calculated

System Reliability

Basic Terms and Definitions

Dr. Mario Trapp

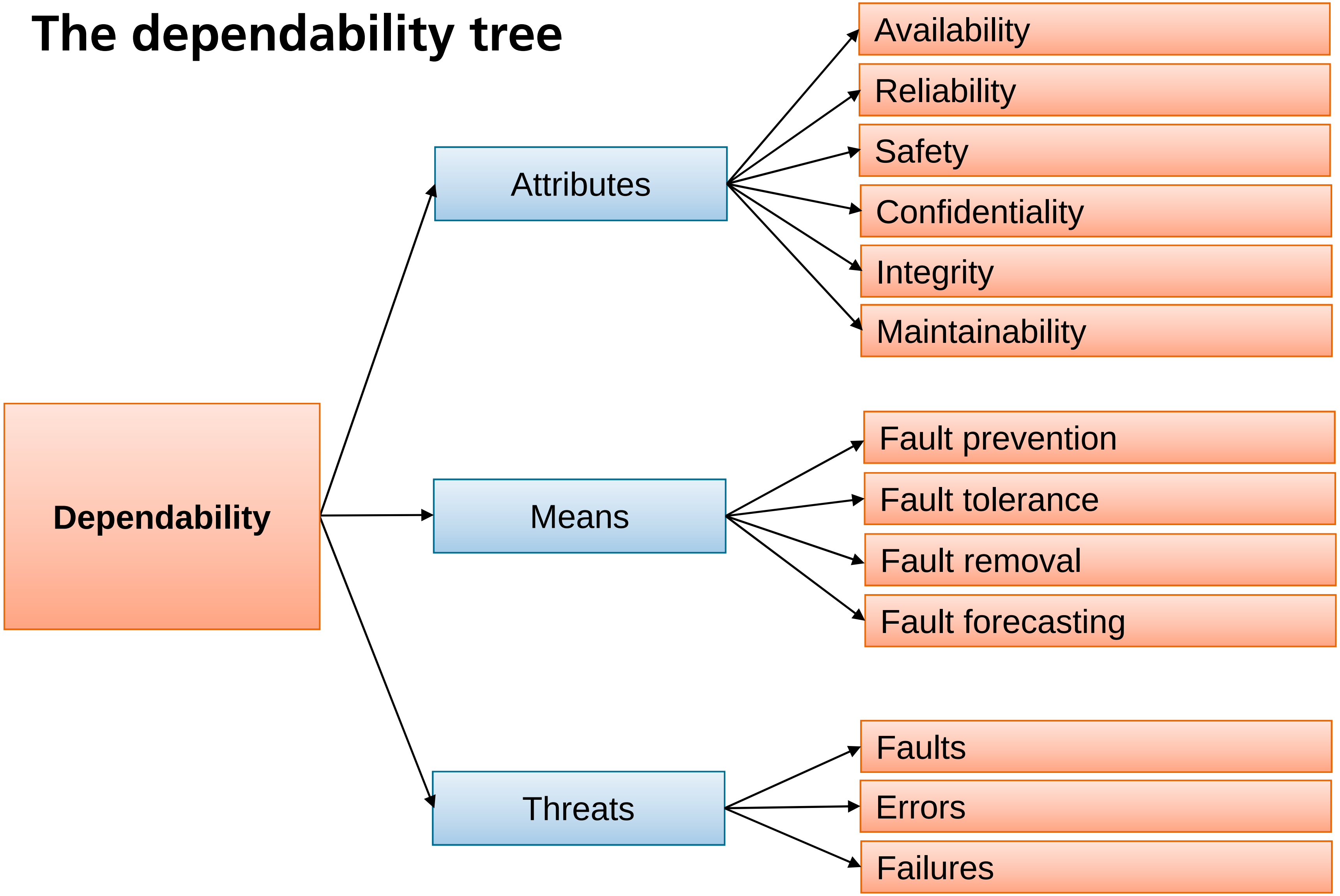
Fraunhofer IESE

mario.trapp@iese.fraunhofer.de

Dependability [Laprie, Randell et al.]

- **Dependability** of a computing system is the ability to deliver *service* that can justifiably be trusted.
- The **service** delivered by a system is its behavior as it is perceived by its *user*(s).
- A **user** is another system (physical, human) that interacts with the former at the service interface.
- The **function** of a system is what the system is intended for, and is described by the system specification.
- **Correct service** is delivered when the service implements the system function.
- A system **failure** is an event that occurs when the delivered service deviates from correct service. *(will be refined later!)*

The dependability tree



Dependability

Attributes

Availability

Reliability

Safety

Confidentiality

Integrity

Maintainability

Means

Fault prevention

Fault tolerance

Fault removal

Fault forecasting

Threats

Faults

Errors

Failures

Basic Terms

- **Safety** is freedom from unacceptable risk.
- **Availability A** is the property of a system, to fulfill its purpose at a given point of time / is the probability that the system fulfills its purpose at a given point of time.

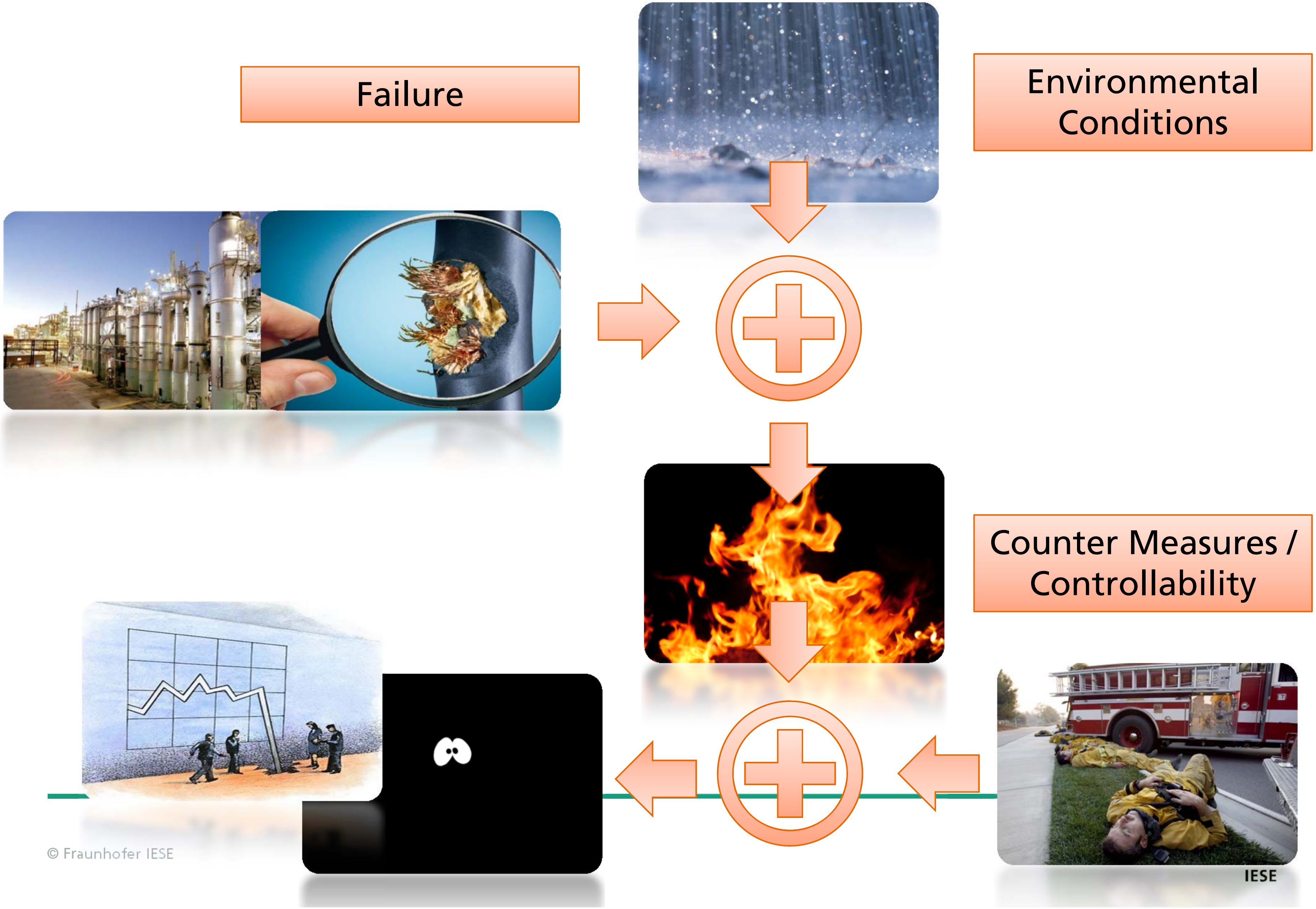
$$A = \frac{\text{TotalTime} - \text{OffTime}}{\text{TotalTime}} = \frac{MTBF}{MTBF + MTTR}$$

MTBF: Mean Time Between Failures (1/λ, λ: failure rate)
MTTR: Mean Time To Repair (1/μ , μ: repair rate)
MTTF: Mean Time To Failure (non-repairable systems, expected value)

- **Reliability R** is the property of an entity to fulfill its reliability requirements during or after a given time span under given application conditions. *R(t) is a function over the time t*.
- **Risk** is the combination of the probability that an undesired event / a failure occurs , the severity of the damage caused by this event and many other factors like exposure, environmental conditions, controllability, …

$$\text{Risk} = \boldsymbol{P}(\boldsymbol{e}) \circledast \boldsymbol{S}(\boldsymbol{e}) \circledast \ \ldots \leq \text{Risk}_{\text{max}}$$

The pathology of incidents and accidents

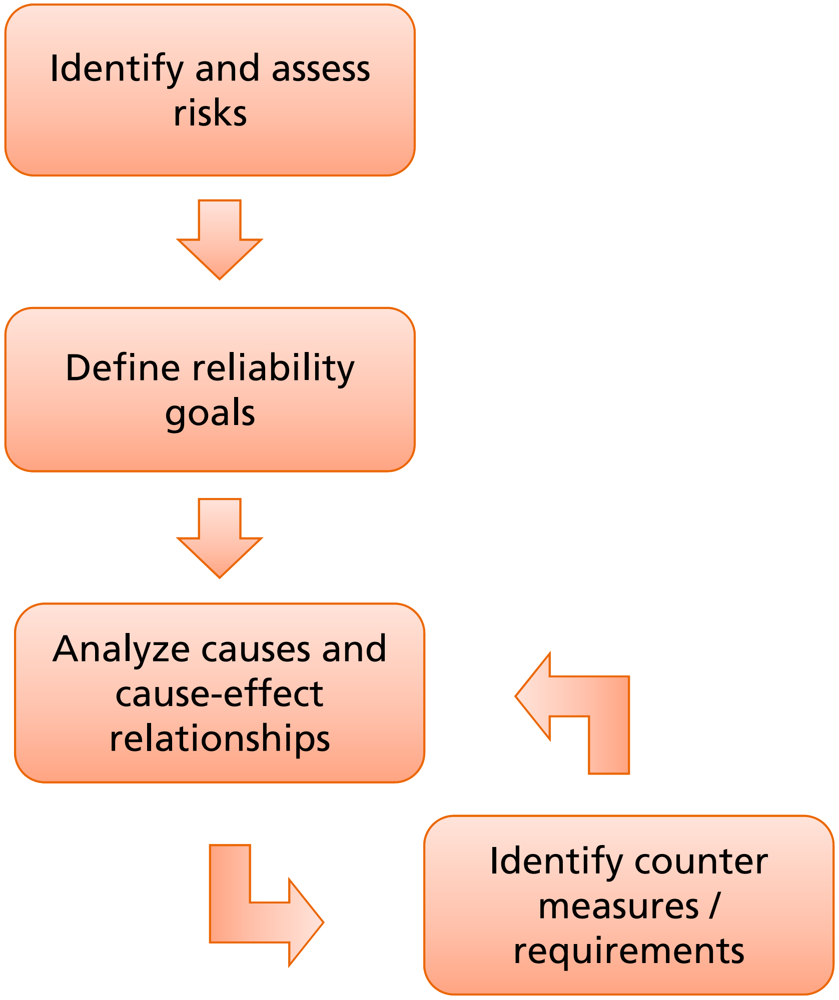


Failure

Environmental Conditions

Counter Measures / Controllability

The general idea



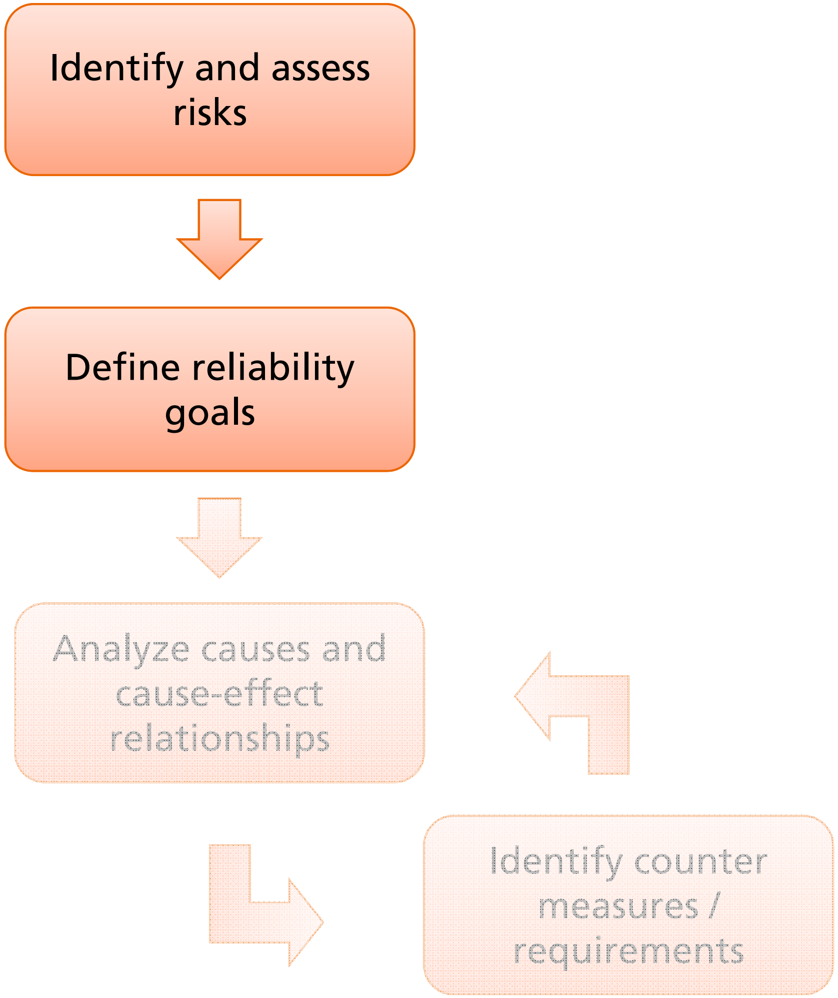
Identify and assess risks

Define reliability goals

Analyze causes and cause-effect relationships

Identify counter measures / requirements

The general idea



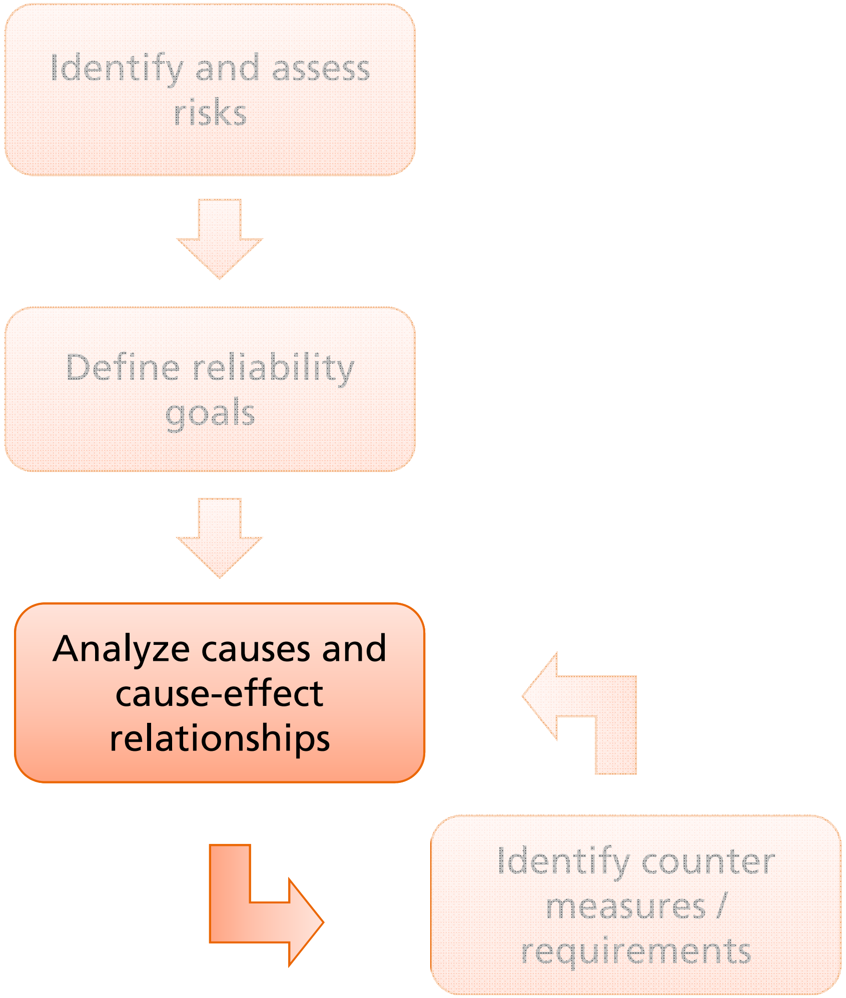
Identify and assess risks
Define reliability goals
Analyze causes and cause-effect relationships
Identify counter measures / requirements

From Risk to Reliability Goals

- Derivation of reliability goals:
  - Identify and assess risk of undesired events (failures)
    - Failure, external events/conditions, severity, …
  - Prioritize failures
  - Define reliability goals that must be fulfilled to avoid failures
    - Functional description (requirement)
    - Quantitative integrity level → e.g., probability
    - Generic goals like „the system must not fail" are possible but not reasonable since are neither achievable nor measurable

- Focus analysis and measures on the avoidance of violations of reliability goals
  - Understand failure causes and cause-effect relationships
  - Break-down reliability goals to reliability requirements on single components

The general idea

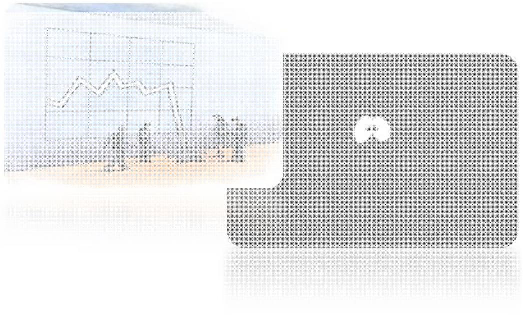


Identify and assess risks

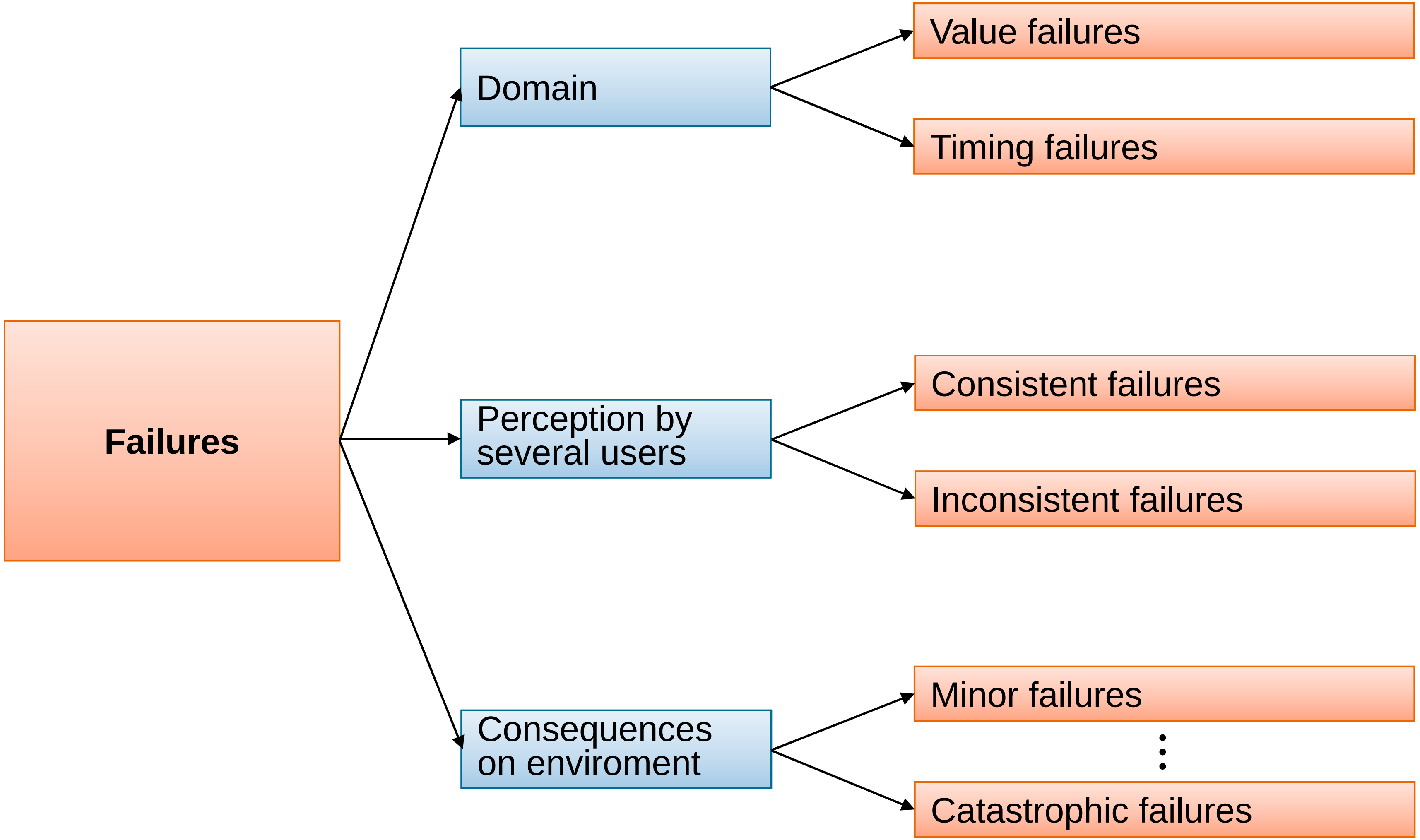
Define reliability goals

Analyze causes and cause-effect relationships

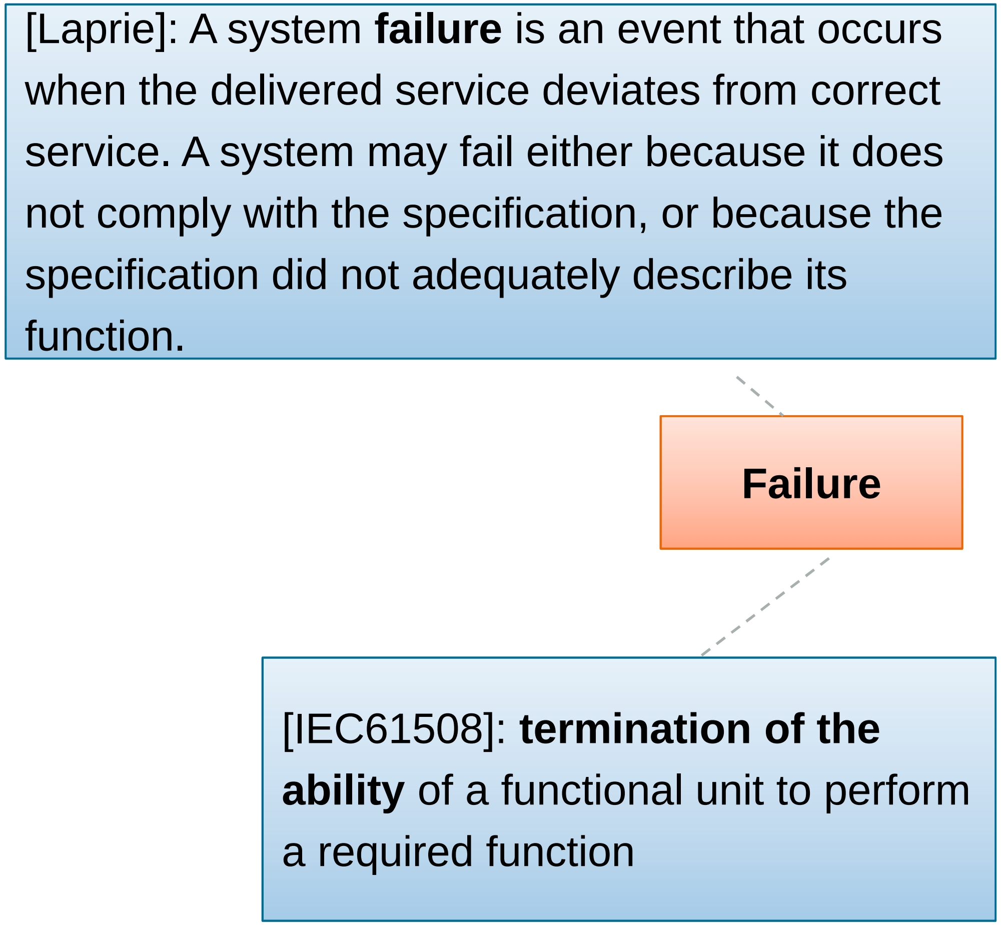
Identify counter measures / requirements



The failure modes



Causes: Mistake – Fault – Error – Failure



[Laprie]: A system **failure** is an event that occurs when the delivered service deviates from correct service. A system may fail either because it does not comply with the specification, or because the specification did not adequately describe its function.

Failure

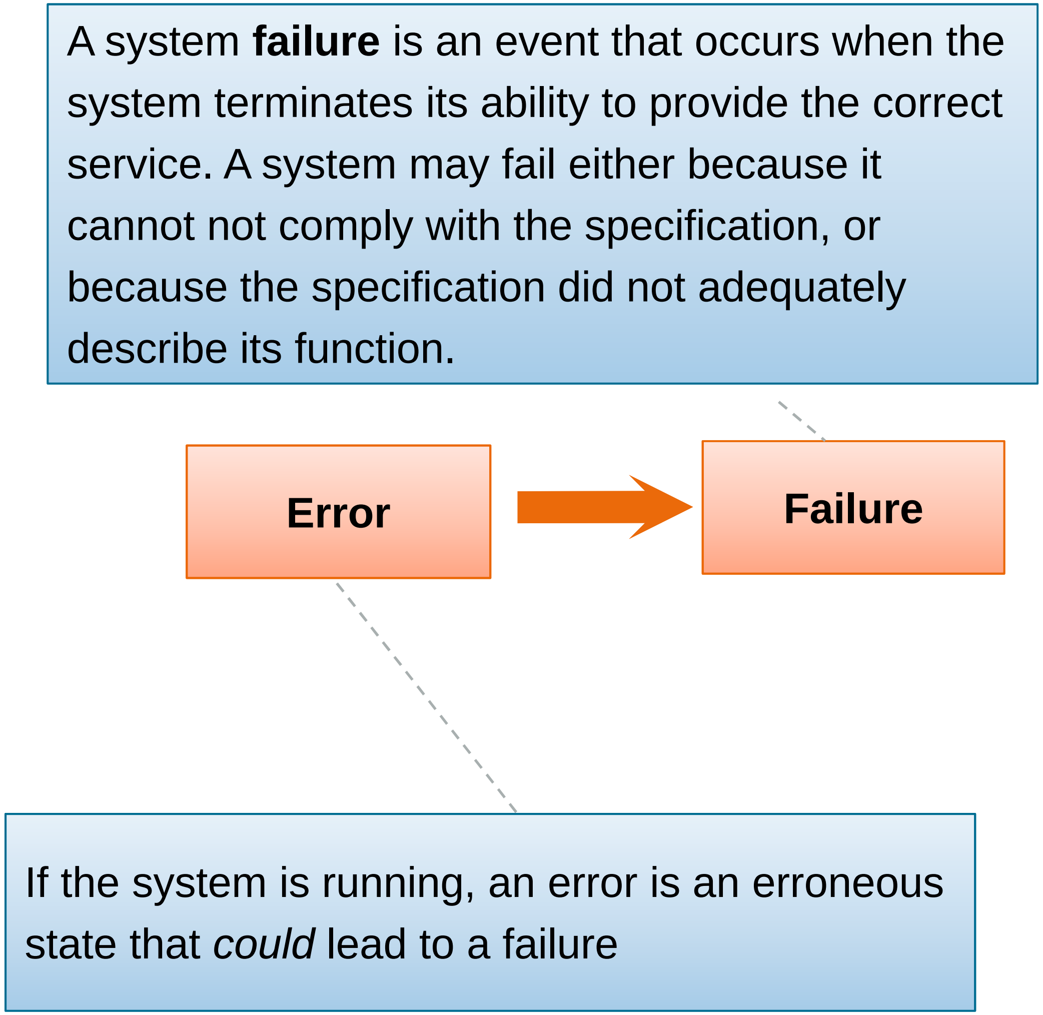
[IEC61508]: **termination of the ability** of a functional unit to perform a required function

Causes: Mistake – Fault – Error – Failure

A system **failure** is an event that occurs when the system terminates its ability to provide the correct service. A system may fail either because it cannot not comply with the specification, or because the specification did not adequately describe its function.

Failure

Causes: Mistake – Fault – Error – Failure



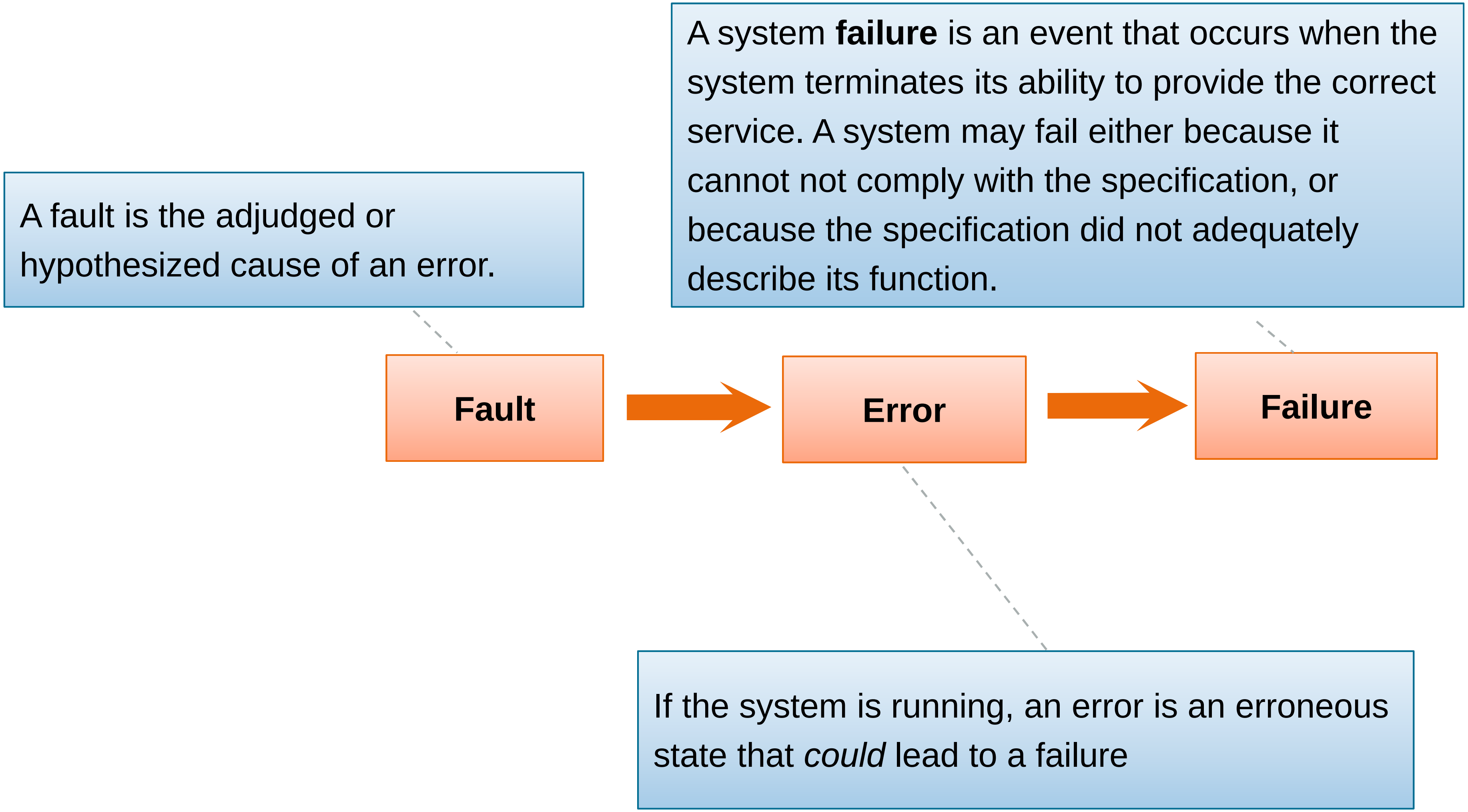
A system **failure** is an event that occurs when the system terminates its ability to provide the correct service. A system may fail either because it cannot not comply with the specification, or because the specification did not adequately describe its function.

Error

Failure

If the system is running, an error is an erroneous state that *could* lead to a failure

Causes: Mistake – Fault – Error – Failure



A fault is the adjudged or hypothesized cause of an error.

A system **failure** is an event that occurs when the system terminates its ability to provide the correct service. A system may fail either because it cannot not comply with the specification, or because the specification did not adequately describe its function.

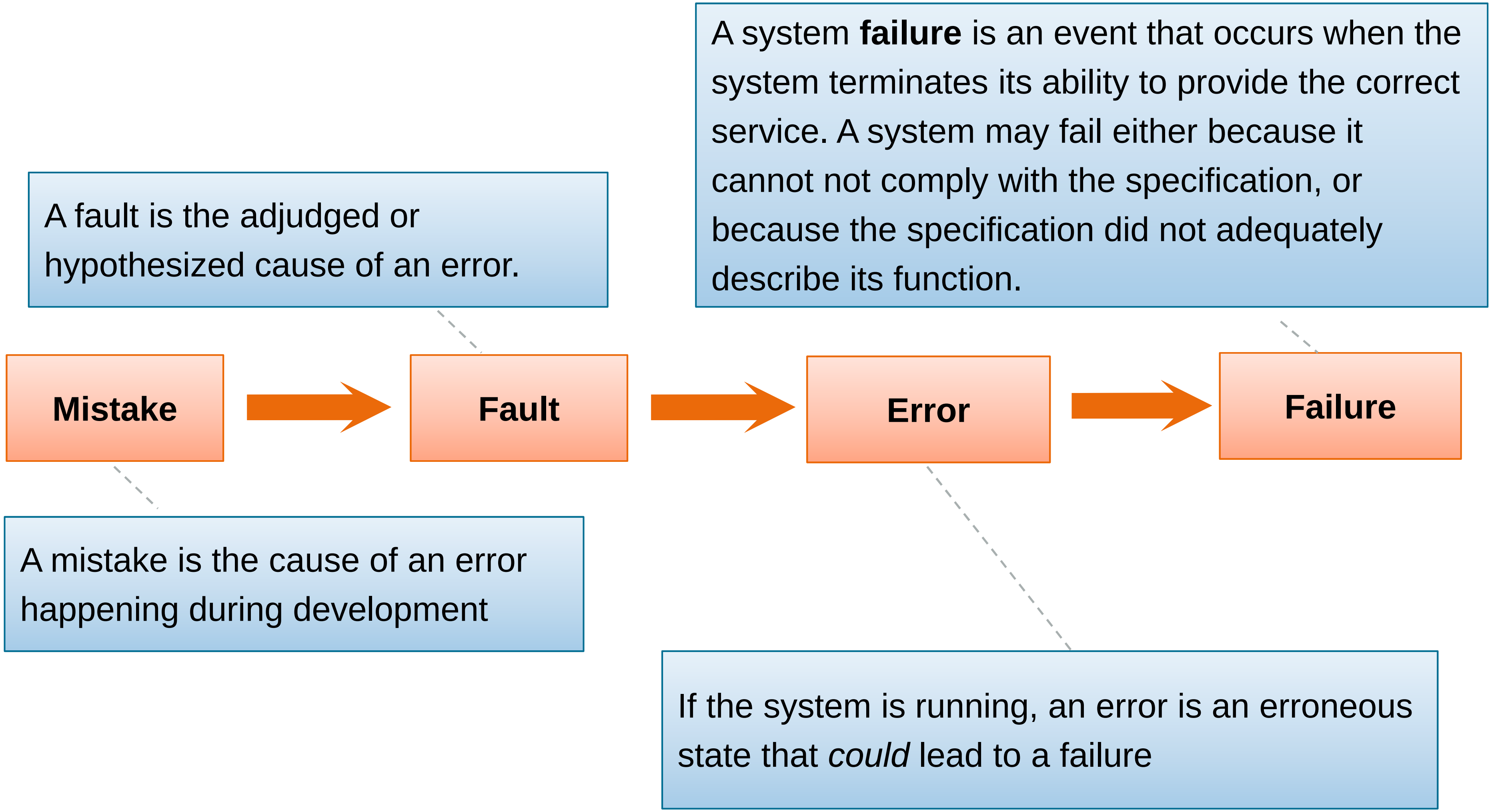
Fault

Error

Failure

If the system is running, an error is an erroneous state that *could* lead to a failure

Causes: Mistake – Fault – Error – Failure



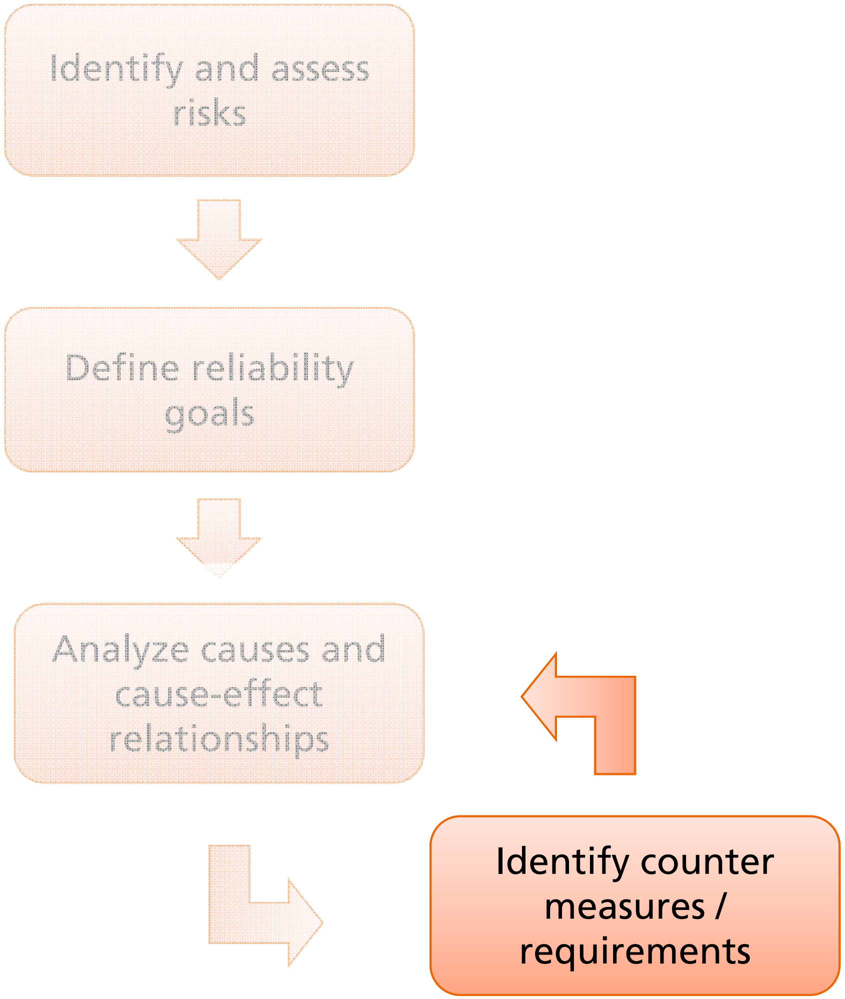
A fault is the adjudged or hypothesized cause of an error.

A system **failure** is an event that occurs when the system terminates its ability to provide the correct service. A system may fail either because it cannot not comply with the specification, or because the specification did not adequately describe its function.

A mistake is the cause of an error happening during development

If the system is running, an error is an erroneous state that *could* lead to a failure

The general idea



Identify and assess risks

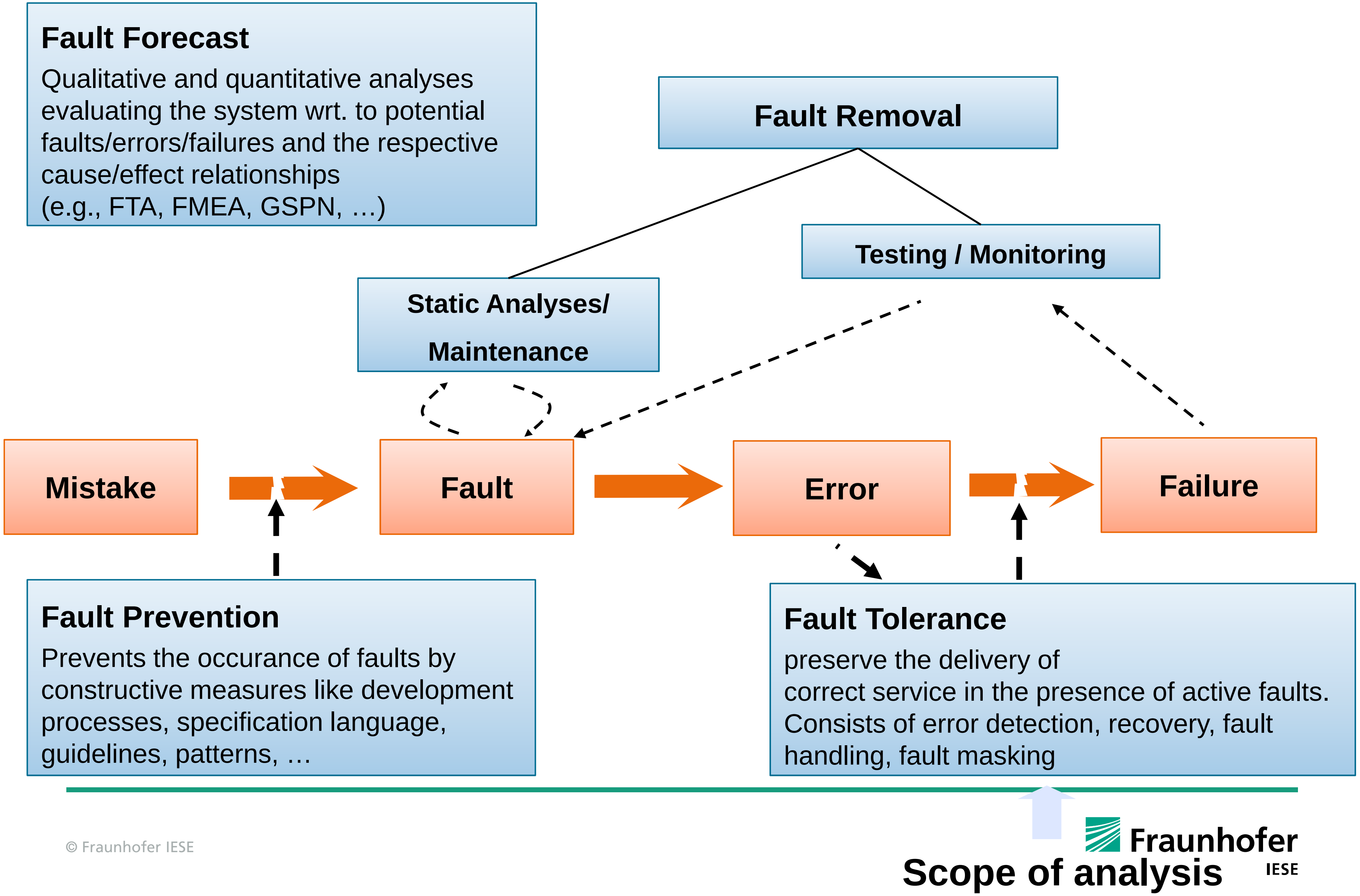
Define reliability goals

Analyze causes and cause-effect relationships

Identify counter measures / requirements

Fraunhofer
IESE

Possible Means to Improve Reliability



Fault Forecast

Qualitative and quantitative analyses evaluating the system wrt. to potential faults/errors/failures and the respective cause/effect relationships
(e.g., FTA, FMEA, GSPN, …)

Fault Removal

Static Analyses/ Maintenance

Testing / Monitoring

Mistake

Fault

Error

Failure

Fault Prevention

Prevents the occurance of faults by constructive measures like development processes, specification language, guidelines, patterns, …

Fault Tolerance

preserve the delivery of
correct service in the presence of active faults. Consists of error detection, recovery, fault handling, fault masking

Scope of analysis

Fault Tolerance

Error Detection

- originates an error signal or message within the system
- concurrent error detection takes place during service delivery
- preemptive error detection takes place while service delivery is suspended; it checks the system for latent errors and dormant faults

Recovery

- transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again
- Consists of error handling and fault handling

Handling faults and errors

Error handling

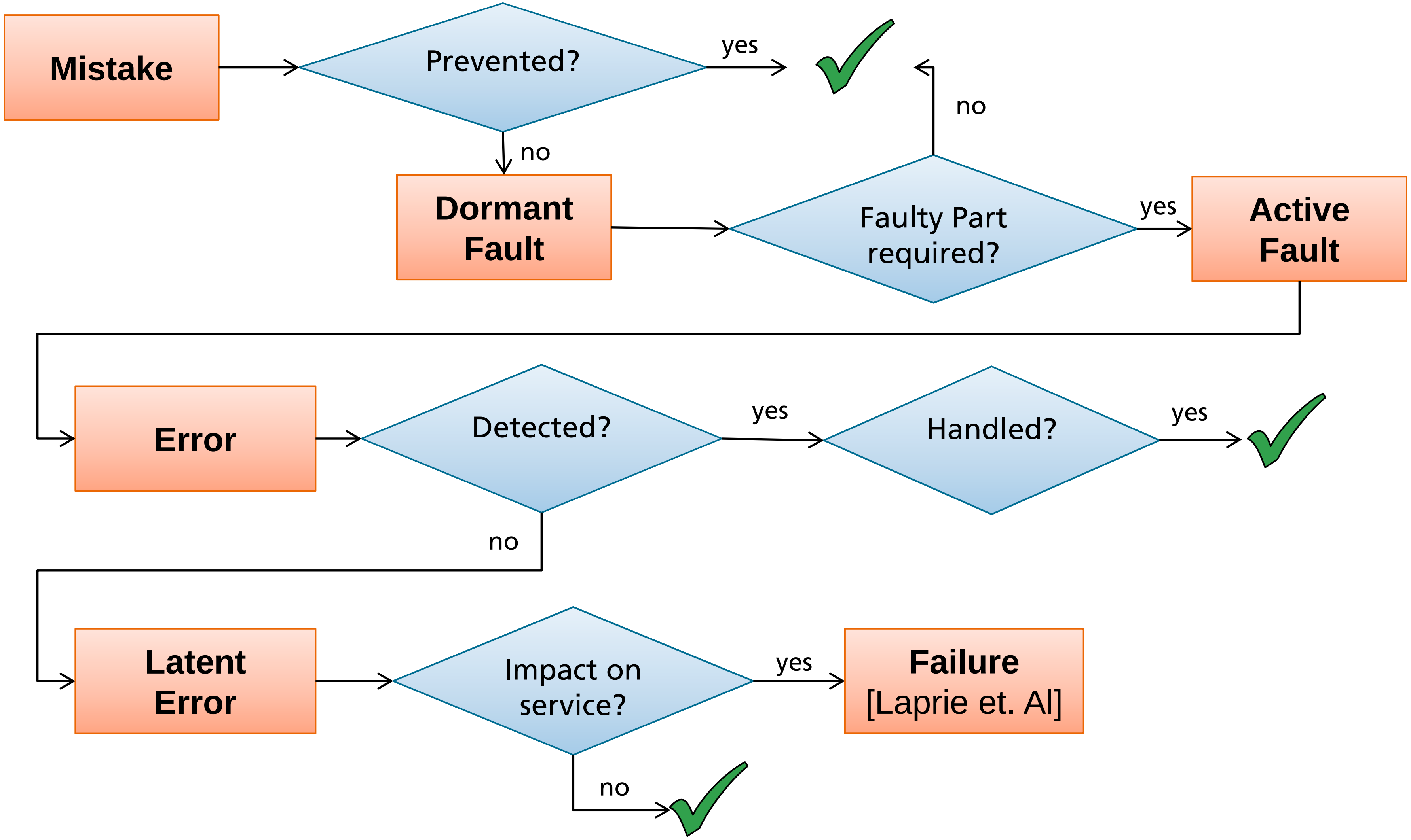
- eliminates errors from the system state
  - a) rollback: system state is set back to previously saved checkpoint
  - b) foll forward: system is set to a new, error-free state

Fault handling

- prevents located faults from being activated again
  1. fault diagnosis that identifies and records the cause(s) of error(s)
  2. Fault isolation that performs physical or logical exclusion of the faulty components from further participation in service delivery
  3. System reconfiguration that either switches in spare components or reassigns tasks among non-failed components
  4. system reinitialization that checks, updates and records the new configuration and updates system tables and records.

- Usually, fault handling is followed by corrective maintenance that removes faults isolated by fault handling.
- As apposed to fault tolerance maintenance requires the participation of an external agent.

The lifecycle from mistakes to failures



Mistake

Prevented?

yes

✓

no

Dormant Fault

Faulty Part required?

no

yes

Active Fault

Error

Detected?

yes

Handled?

yes

✓

no

Latent Error

Impact on service?

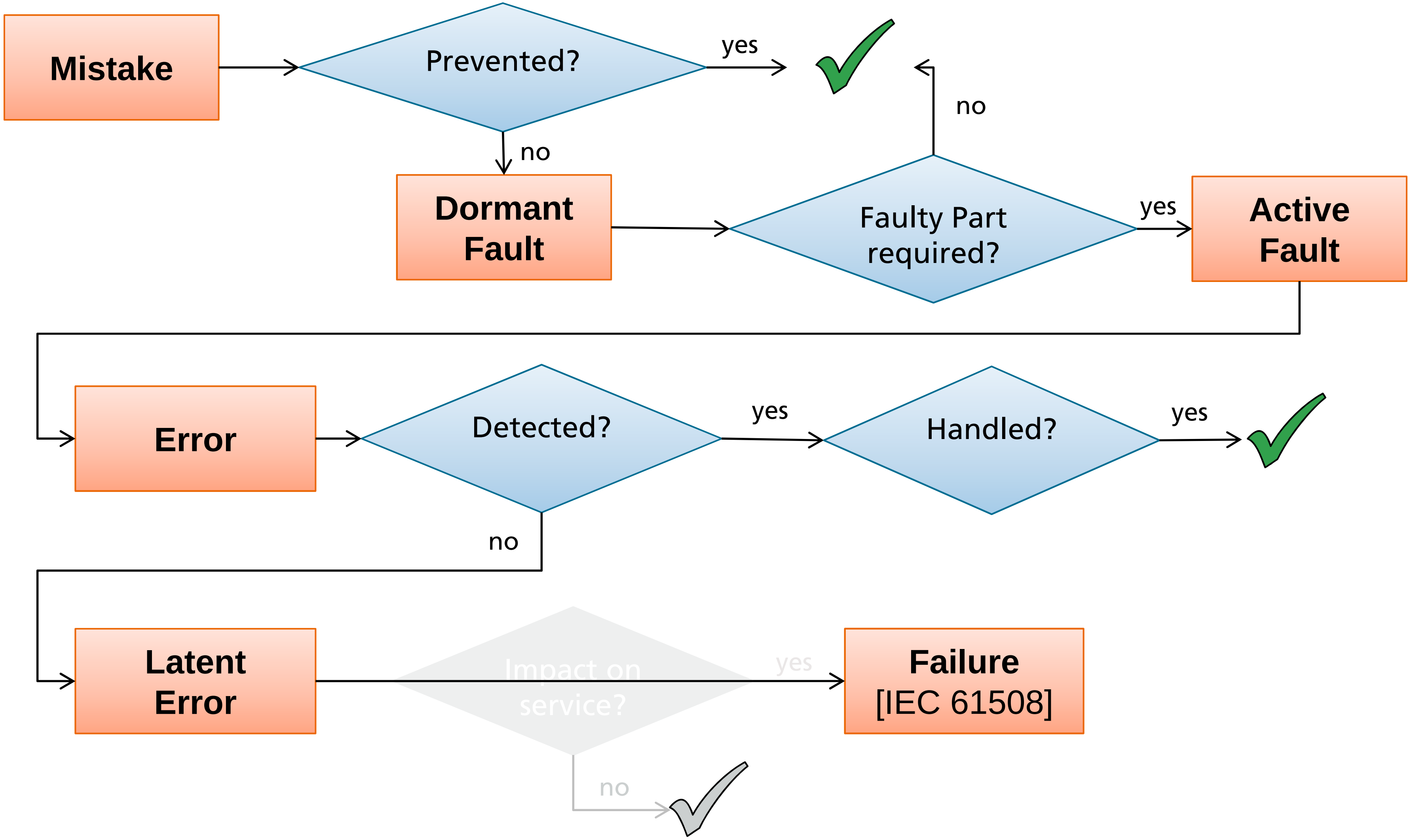
yes

Failure [Laprie et. Al]

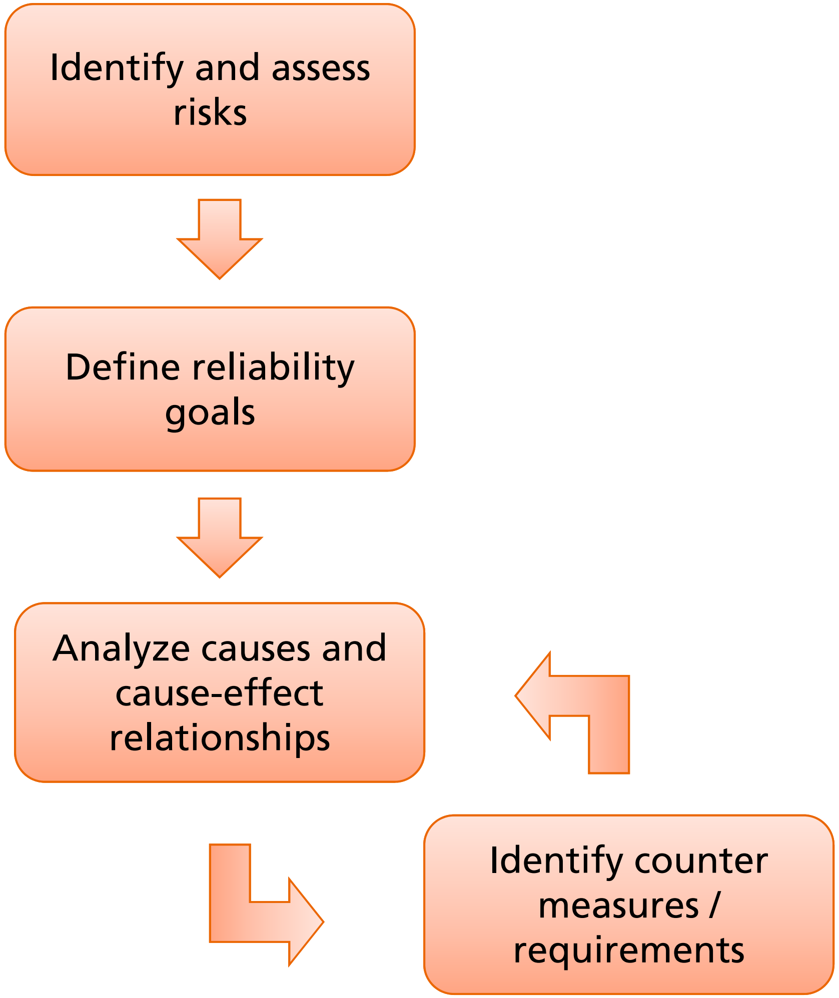
no

✓

The lifecycle from mistakes to failures



The general idea - revisited



Identify and assess risks

Define reliability goals

Analyze causes and cause-effect relationships

Identify counter measures / requirements

- Dependability
- Safety
- Reliability
- Availability
- Risk

- Failure
- Error
- Fault
- Mistake

- Fault Prevention
- Fault Removal
- Fault Forecast
- Fault Tolerance