System Reliability Analysis

Reliability Engineering Lifecycle

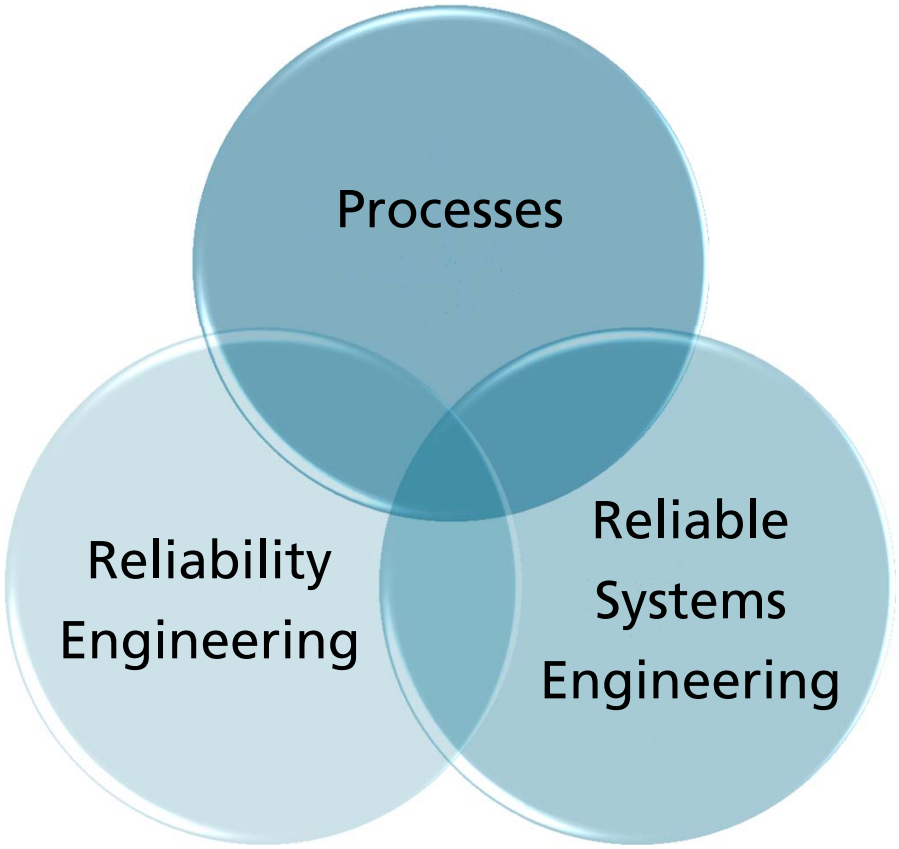
Dr. Mario Trapp

Fraunhofer IESE

mario.trapp@iese.fraunhofer.de

Fraunhofer IESE

The Elements of Engineering Reliable Systems



Processes

- Developing products following clearly defined processes is a necessary but not sufficient prerequisite
- Processes are the indispensable basis for the engineering of reliable systems

➔ **Fault Prevention**

Without processes you won't have your development under control and thus you won't have your system under control!

➔ Sound project management is an indispensable prerequisite for reliable systems engineering

➔ Assuring sufficient resources for quality assurance is a prerequisite

➔ Clear phases with well-defined quality gates ensure quality

➔ Supporting processes like configuration and change management are indispensable quality over the whole lifecyle

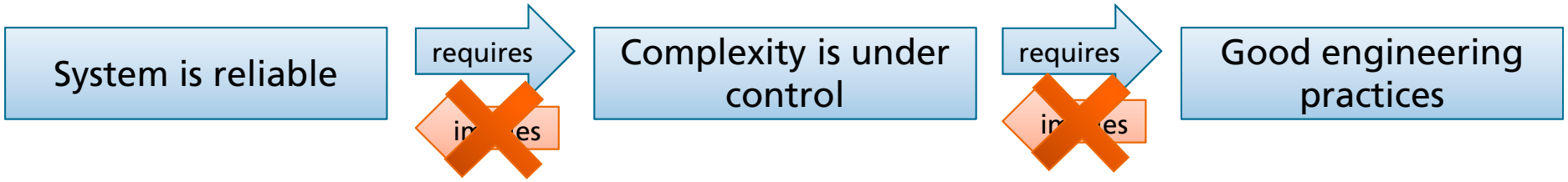
➔ …

Processes won't be considered in detail in this workshop!

Reliable Systems Engineering

Complex systems must be engineered

- From requirements to implementation to validation
- All phases of the development lifecycle must fulfill additional requirements to ensure the development of reliable systems
- In each phase a fault is not discovered, the cost for its removal increases by a factor of 10!
- A good engineering decreases additional effort for reliability
- A good engineering is the prerequisite to ensure system reliability



➔ **Systems Engineering Principles will be part of this section**

Reliability Engineering

- Describes the activities of
  - analyzing and assessing the system's risks
  - analyzing the cause-effect-relationships
  - selecting appropriate counter measures
  - verifying that the selected measures are sufficient
  - using fault forecasting mechanisms
- Direct Impact on Engineering
  - All counter measures are applied in the systems engineering part
  - Fault Removal (e.g., selection of QA-methodologies)
  - Fault Tolerance (e.g., impact of system architecture)

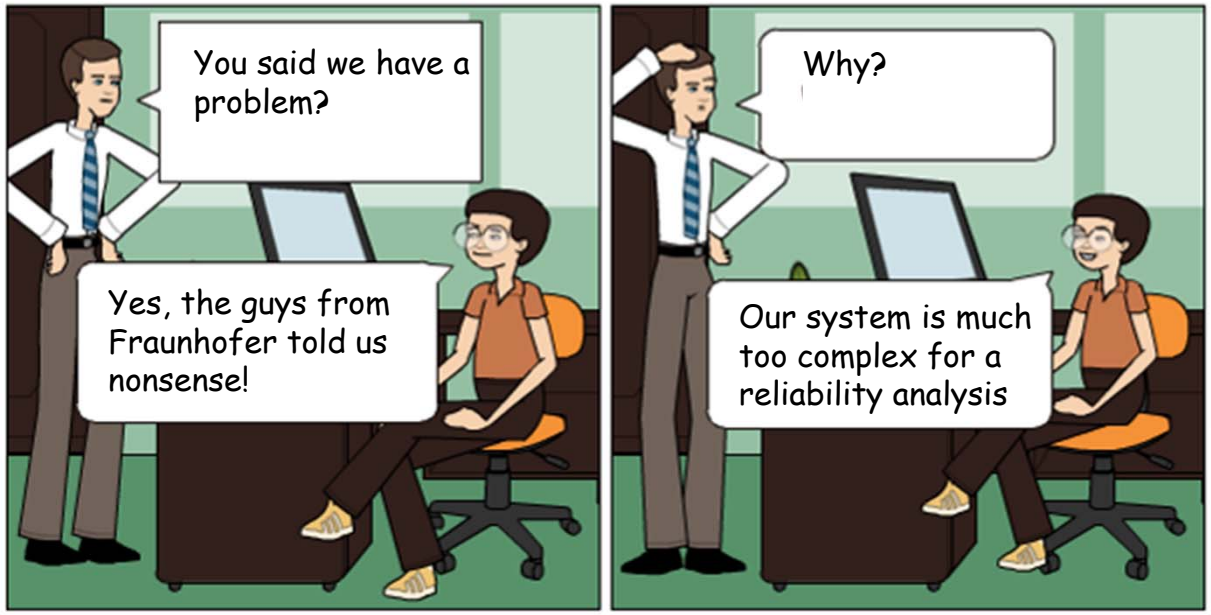
➔ **Will be the main topic of this workshop!**

Engineering reliable systems

- Why Engineering?
- The process point of view
- The methodology point of view – the required minimum

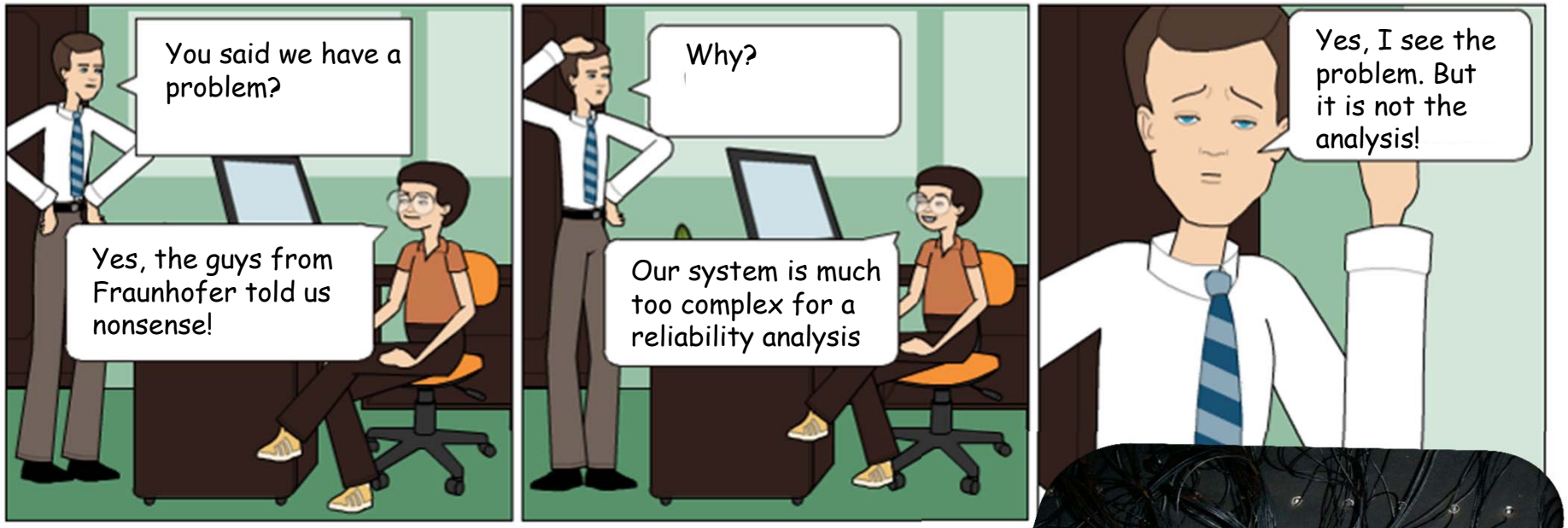
Why Engineering?

Why Engineering?



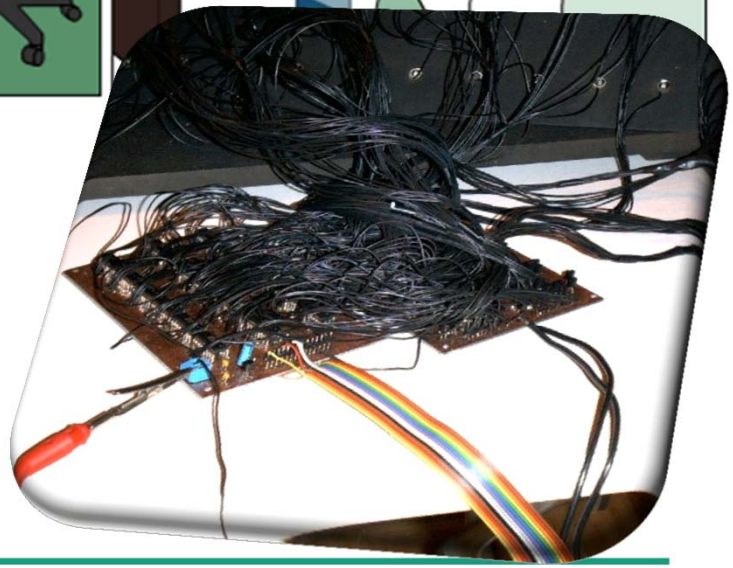
Why Engineering?

Why Engineering?

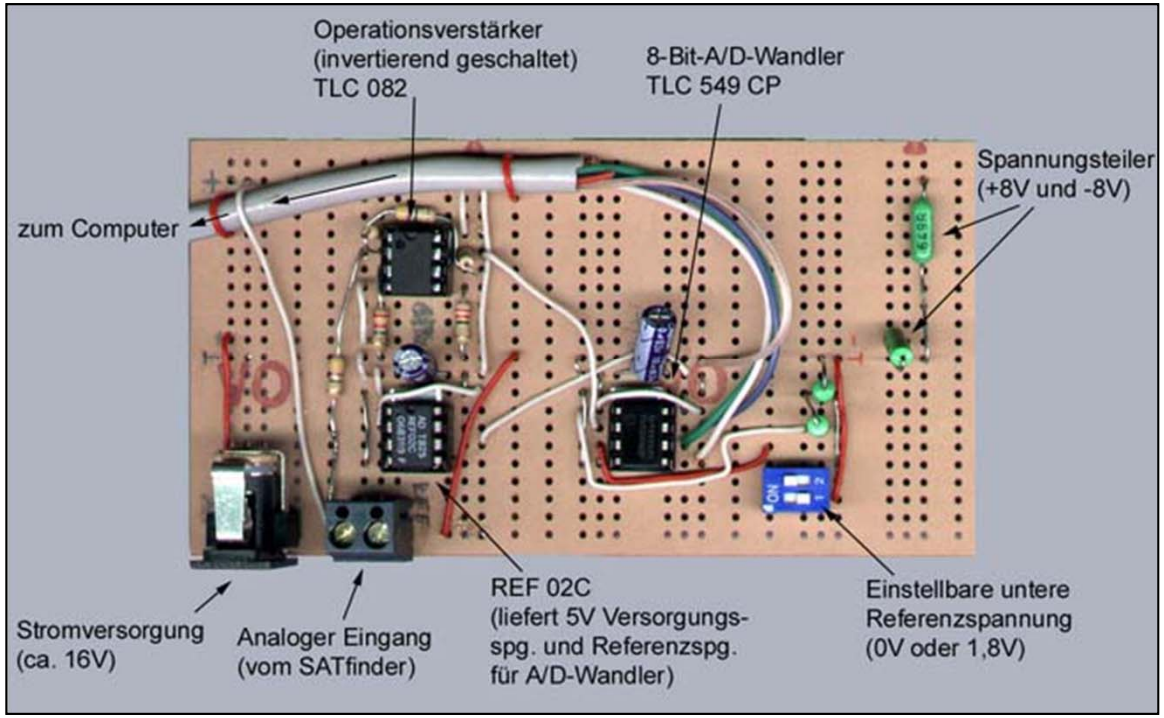


If you are not able to perform a reliability analysis due to complexity reasons, you have a real problem:

You lost control over your system!

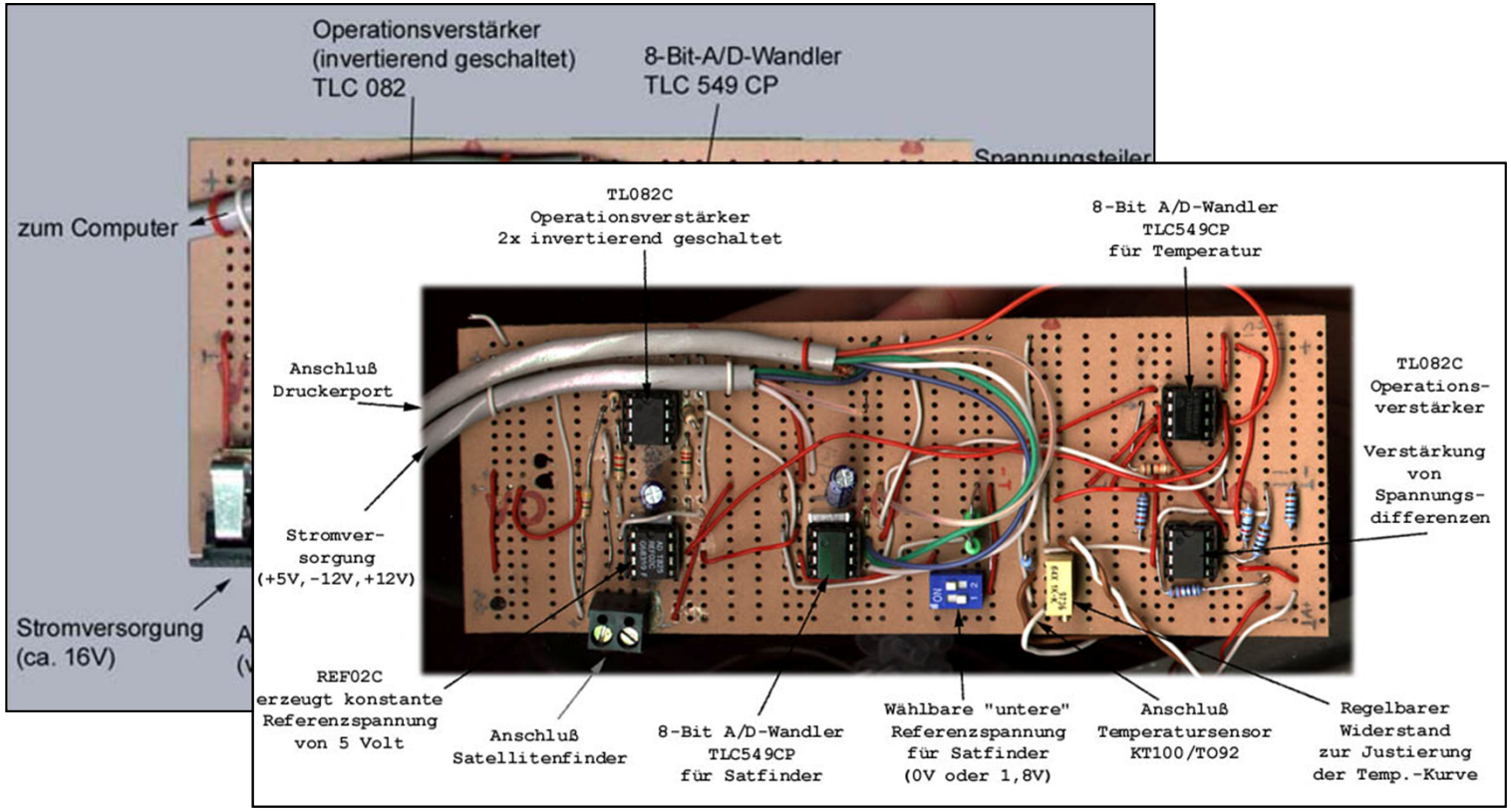


How the story began

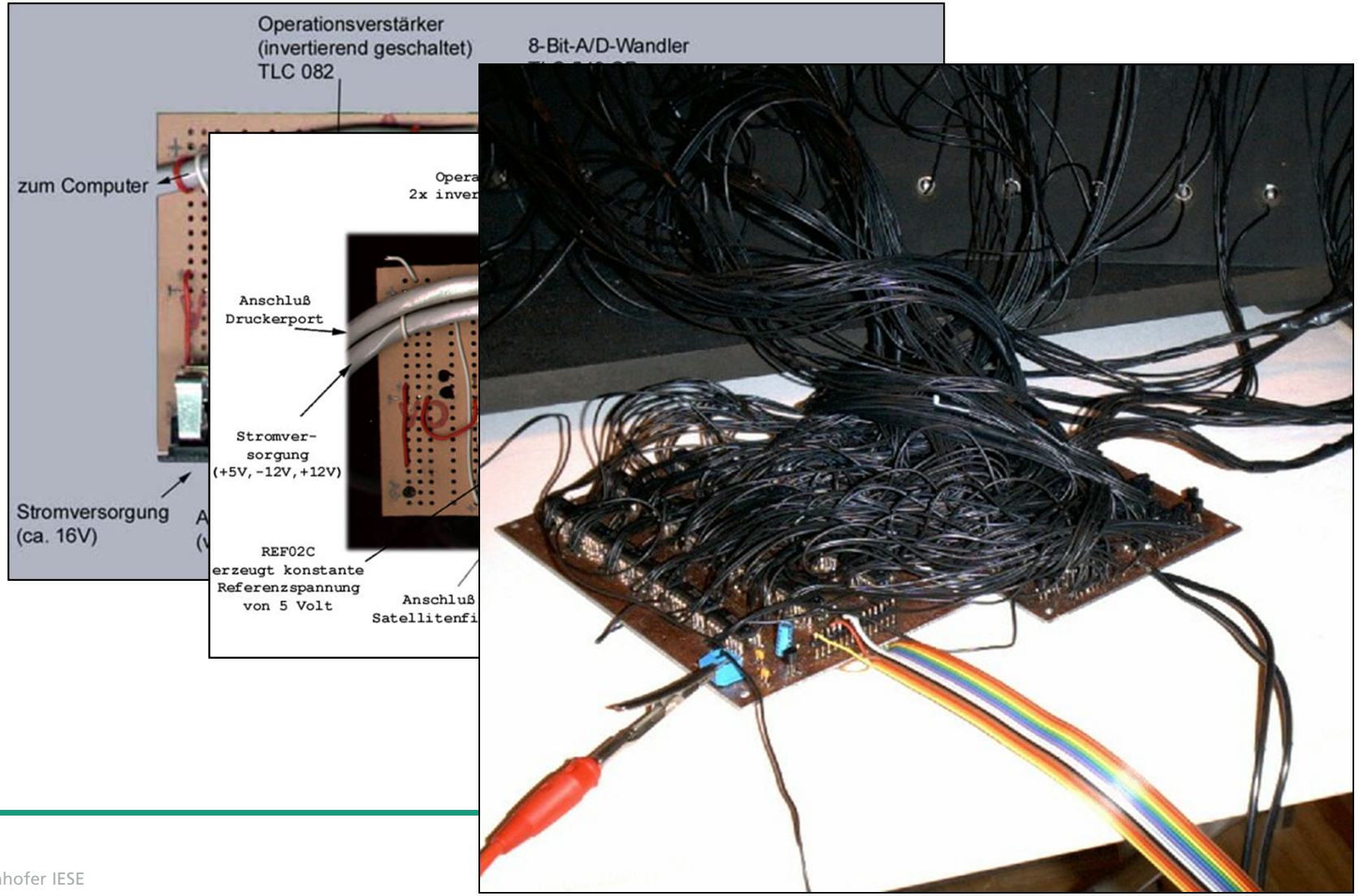


The big mistake

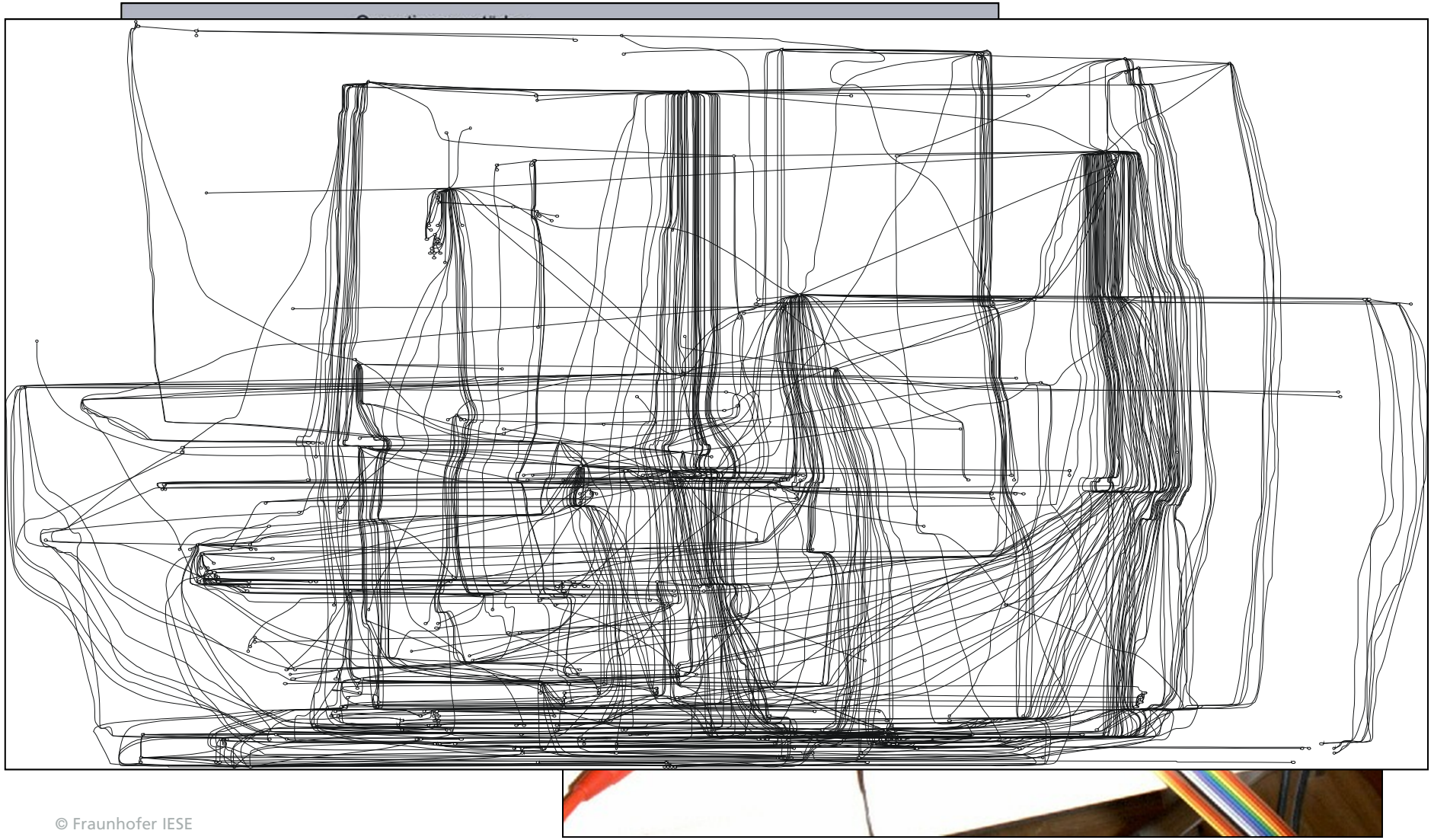
The point of no return



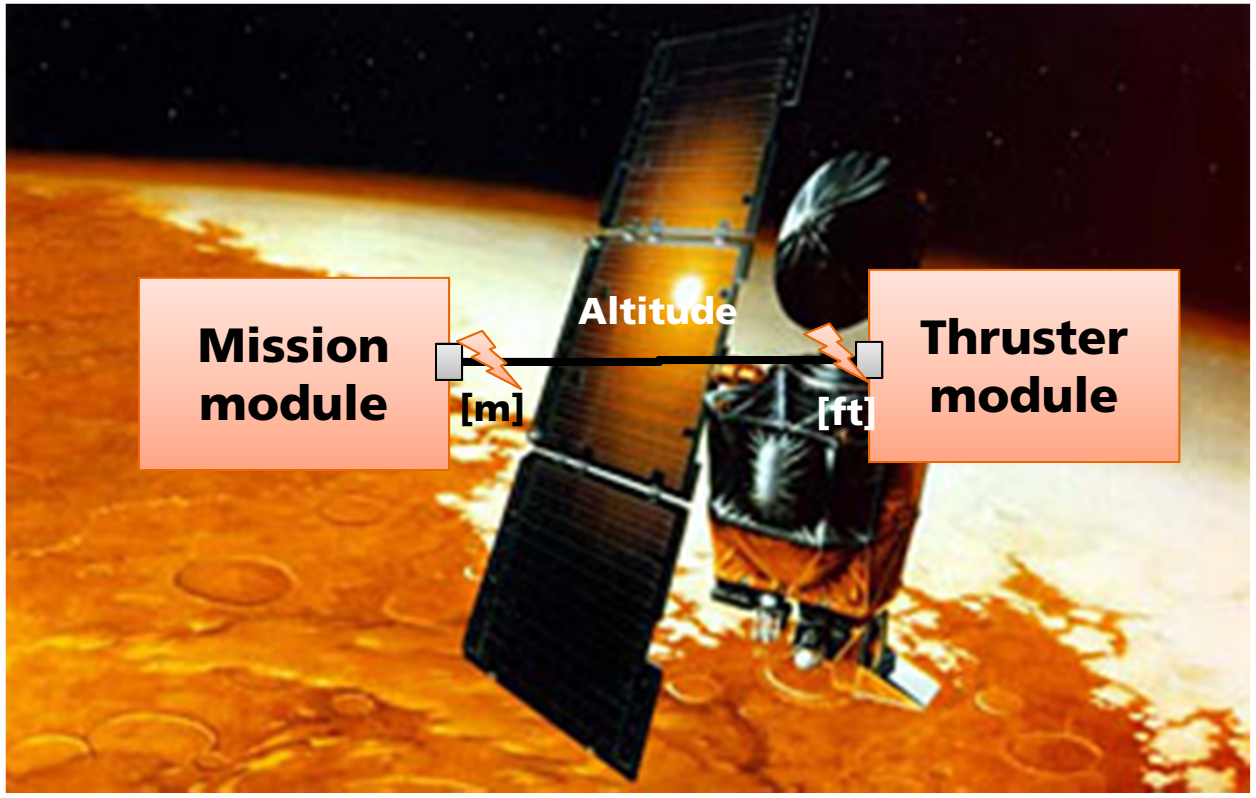
And suddenly it's too late



And suddenly it’s too late



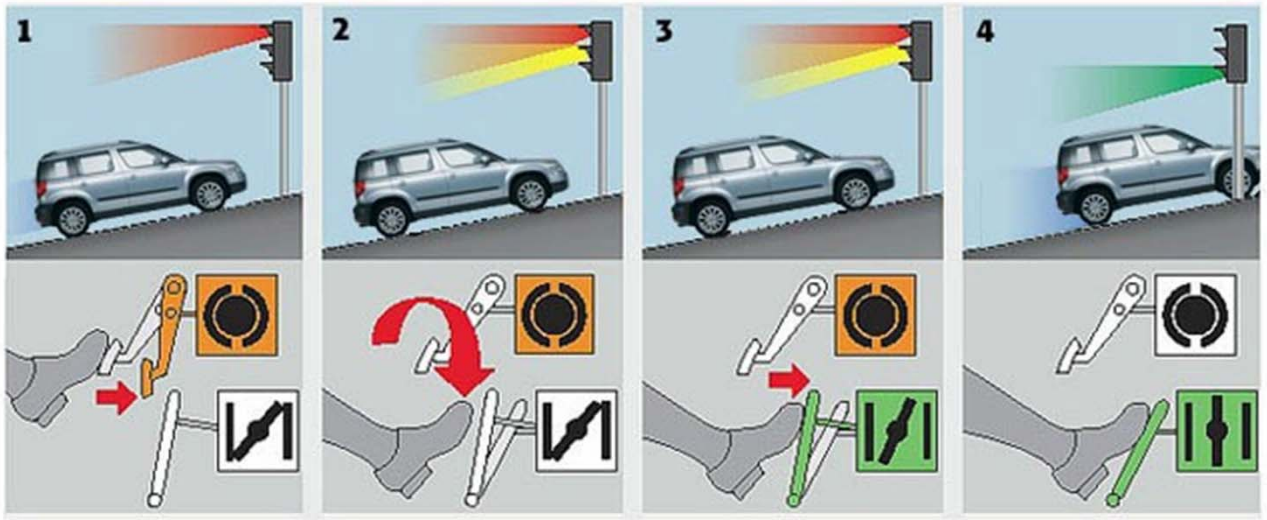
Loss of MARS Climate Orbiter (MCA) - Revisited

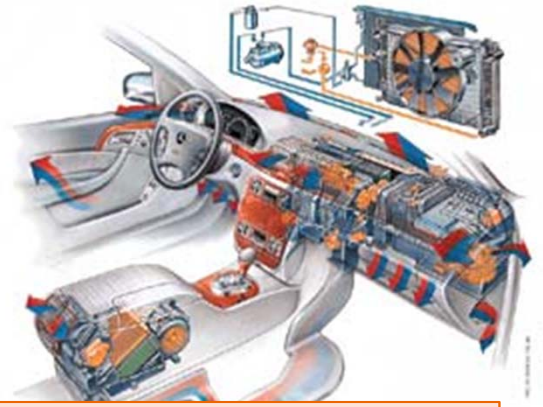


➔ **under /non-specified interfaces**

- → clear reliability impact
- → Obviously no reliability engineering but systems engineering problem

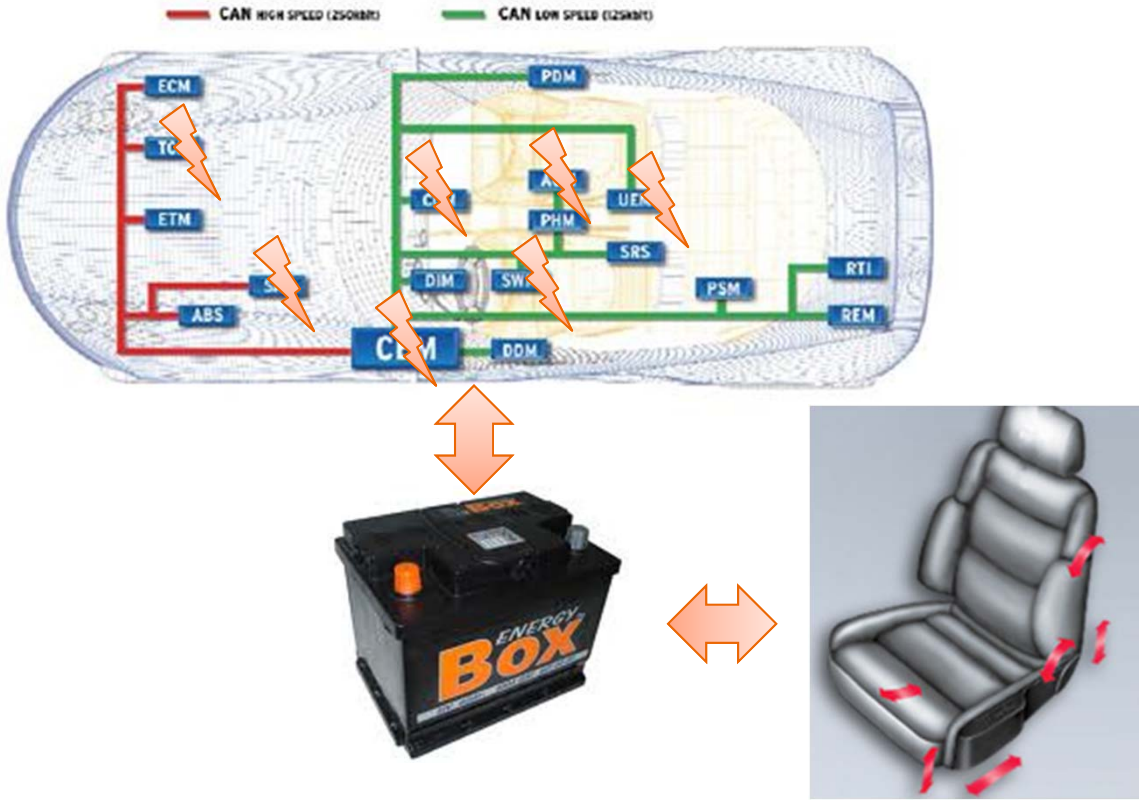
Feature Interaction





➔ **Complexity leads to undesired feature interactions**

Interference



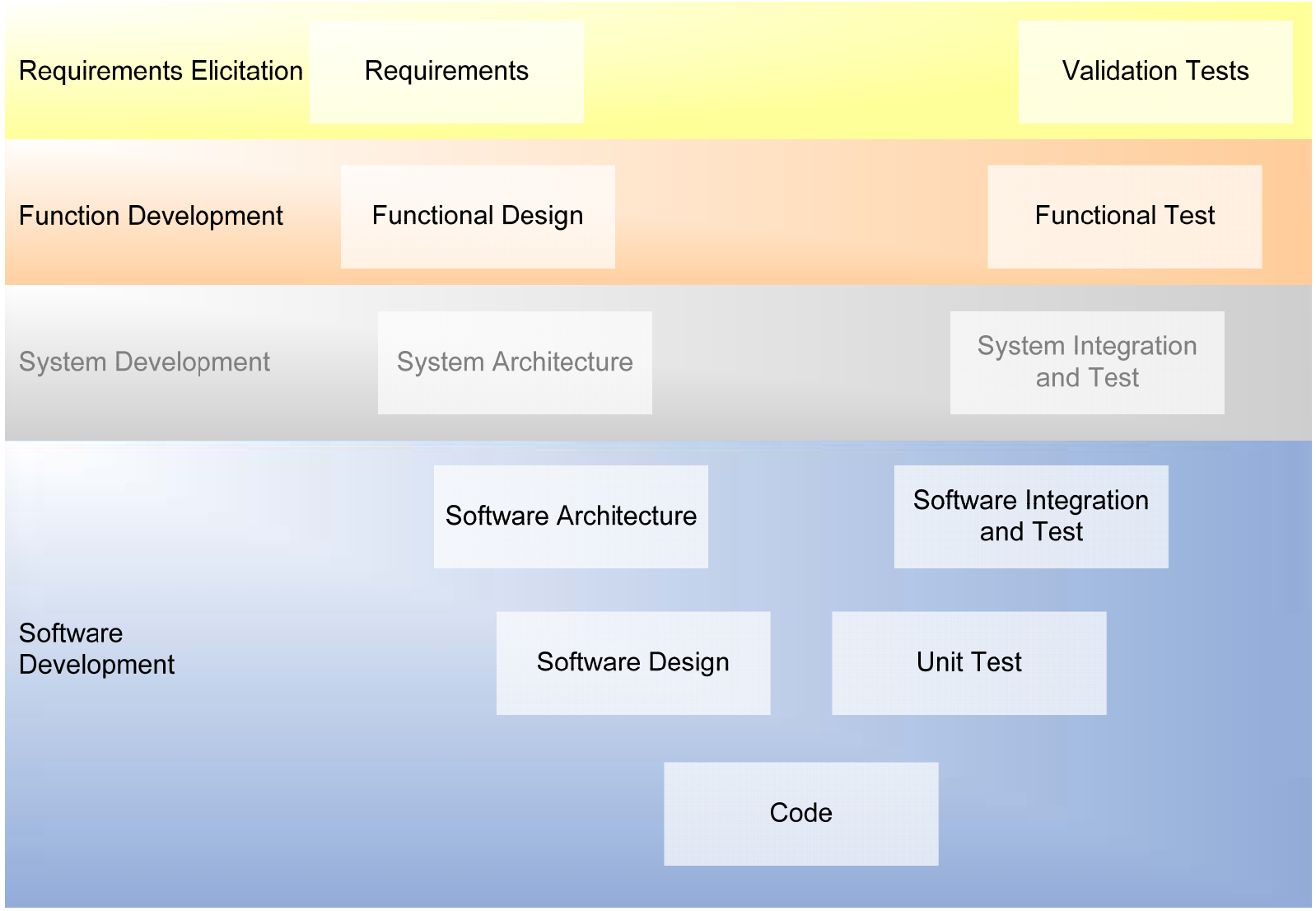
➔ System compexity leads to sub system interference

→ Cannot be manually identified / managed

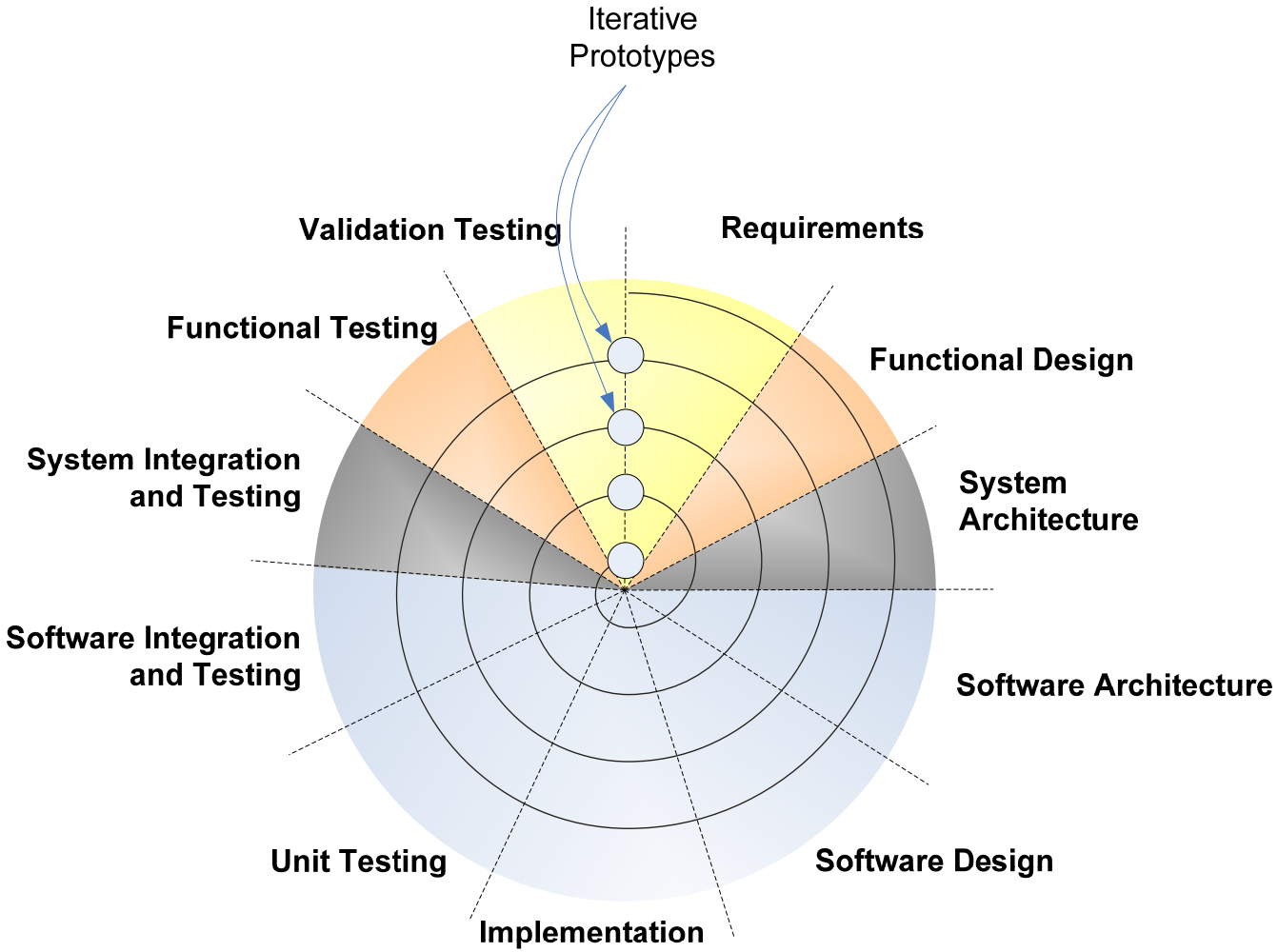
Engineering reliable systems

- Why Engineering?
- **The process point of view**
- The methodology point of view – the required minimum

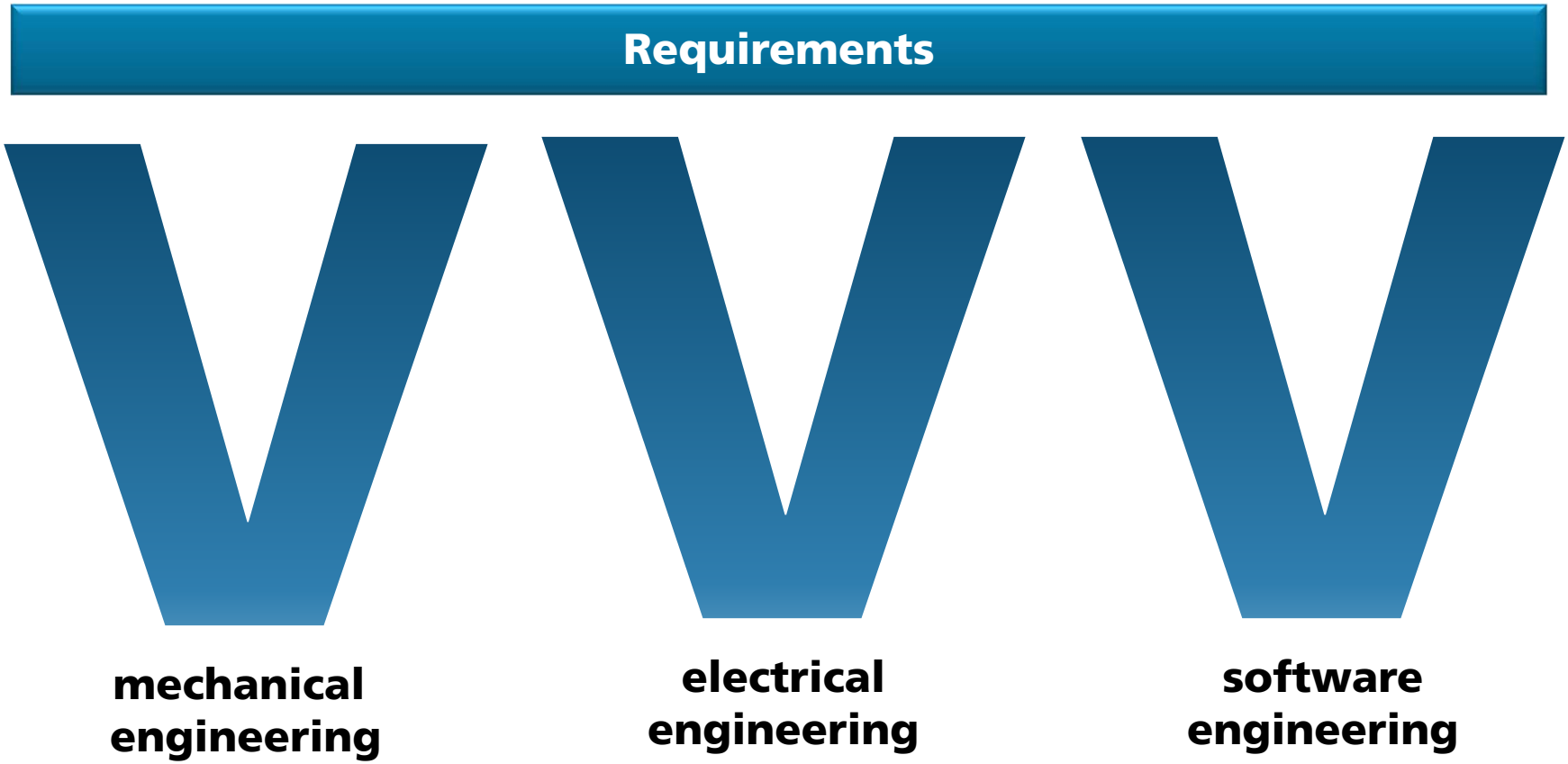
V-Productmodel (the SW-Viewpoint)



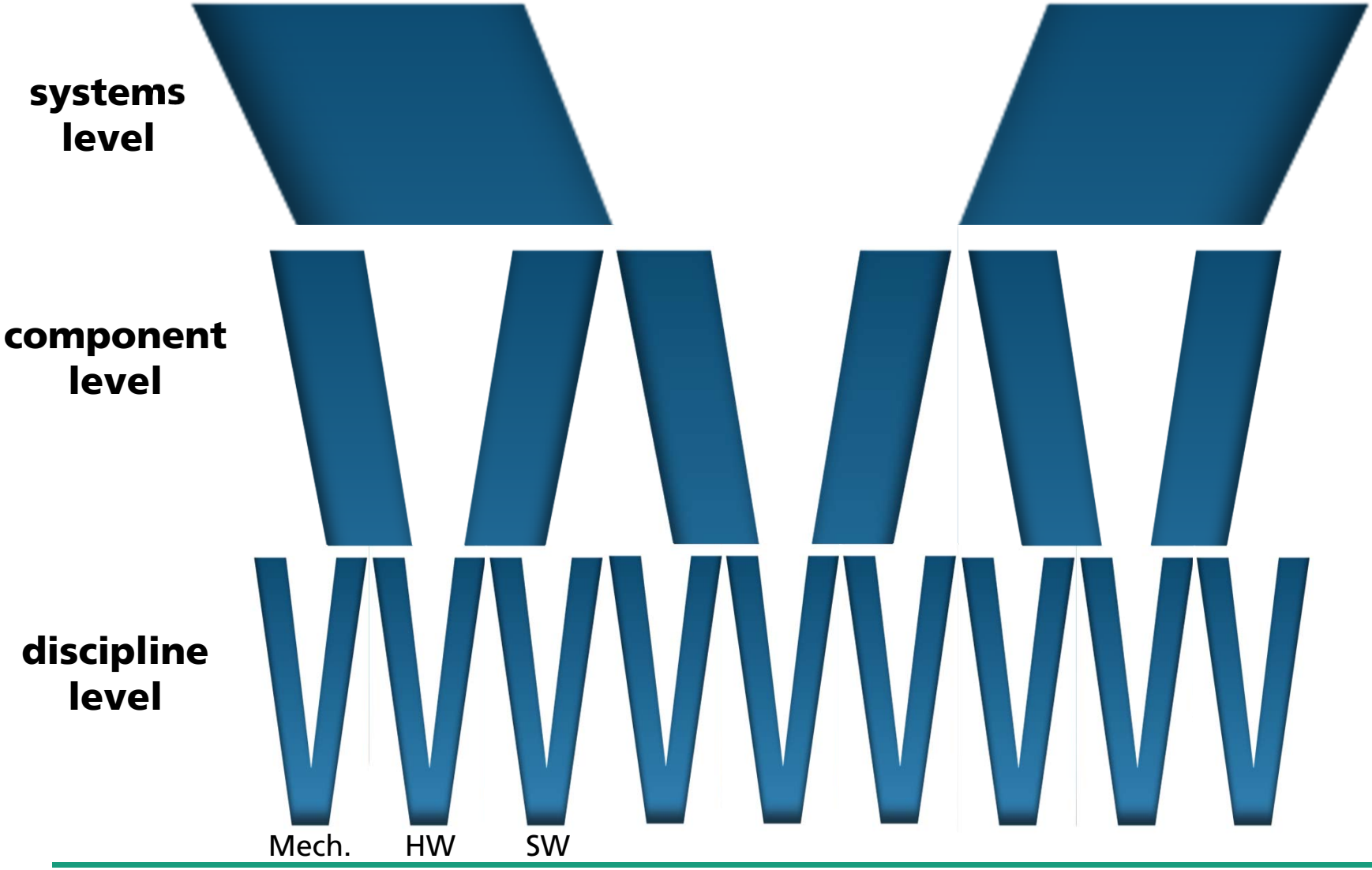
The Development Phases (SW-Viewpoint)



The System‘s viewpoint



From parallel to fractal „Vs“



Fraunhofer IESE

Engineering reliable systems

- Why Engineering?
- The process point of view
- **The methodology point of view – the required minimum**

Required Abstraction levels

Fraunhofer IESE

Required Information

- System Architecture
- Clear, precise specification of subsystem interfaces
- Clear, precise yet abstracting specification of subsystem functionality
- Clear, precise specification of composition
- Clear, precise specification of interactions

- the more detailed the specification
  - the less faults will be in the system
  - the easier will be the reliability analysis

- In case of insufficient specification
  - An reliability analysis might not be possible
  - In this case it is not a problem of the analysis methodology
  - But it is a problem of your system (this means you don't have the system complexity under control!)

Recommendation

Use SysML to specify your system

- SysML is intuitive yet formal enough to facilitate automated analyses for fault prevention / fault removal
- models include all information for fault forecasting
- Significantly reduces development complexity by
  - Abstraction
  - Modularization and hierarchy
  - Separation of concerns

- With a sound modeling technique you will
  - Efficiently prevent faults
  - Simplify/enable fault removal
  - Simplify/enable fault forecasting

Conclusion

- With sophisticated reliability analysis techniques we are able to analyze very complex systems,
- but we can hardly be better than the information we get as input!