Industrial Embedded Systems
- Design for Harsh Environment -

Dr. Alexander Walsch
alexander.walsch@ge.com

IN2244

Part III – Reliability Requirements

WS 2015/16

Technische Universität München

Fault, Error, Failure

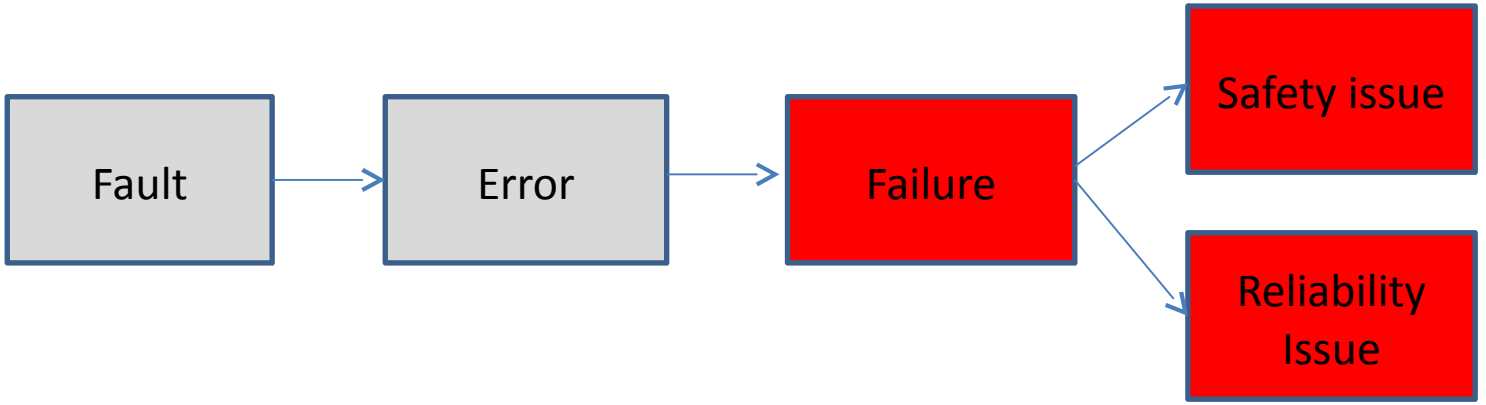
Fault (HW), Defect, Bug (SW)
abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function

Error (revealed fault)
a deviation from the correct value or state

Failure
Failure is defined as deviation from the specification. The designed function can not be executed anymore as specified.

Failure Mode
A function can fail in various ways. In our analysis we pick the failure mode that leads to the failure we investigate.

Fault, Error, Failure II



- Hardware faults can be random or systematic. Software defects are systematic
- Hardware faults can be thought of as physical faults, e.g. a bit flips, a wire breaks. Software defects are mistakes during development
- Faults and defects are dormant until the resource is used (think of a software task that executes specific code for the first time)
- Once it is used it may cause an error which is a deviation from the expected
- The error may make the system deviate from its specification. It is running outside its intended use

Failure Modes

Function:
A process variable is measured (input) and the temperature compensated reading transmitted using a 4 – 20 mA data communication interface (output).

The following failure modes and occurrences are known. What failure modes do influence our design most?

Failure Mode	Failure occurance
4 – 20 mA current signal stuck fail (output)	Low
4 – 20 mA current signal low fail (output)	Low
Sensor head fail (input)	Medium
Sensor head power failure	High
Other	low

Failure Modes and Effect Analysis (FMEA)

- System FMEA in requirements analysis (proposed system)
  - Also: Design FMEA (existing system)
- What are the failure modes and what is the effect:
  - System failure (e.g. power, communication, timeliness, erroneous) mode assessment
  - Plan how to prevent the failures
- How does it work?
  - Identify potential failure modes and rate the severity (team activity)
  - Evaluate objectively the probability of occurrence of causes and the ability to detect the cause when it occurs
  - Rank failure modes and isolate the most critical ones

FMEA II

- FMEA tools
  - Spreadsheet, proprietary (e.g. Reliasoft Xfmea)
- Risk ratings: 1 (best) to 10 (worst)
  - Severity (SEV) – how significant is the impact
  - Occurance (OCC) – likelihood of occurance
  - Detection (DET) – how likely will the current system detect the failure mode
- Risk Priority Number (RPN)
  - A numerical calculation of the relative risk of a particular failure mode
  - RPN = SEV x OCC x DET
  - Used to isolate the most risky functions and their failure modes
  - Qualtitative approach (risk ratings are relative numbers)

FMEA III

- Function/Component – What is the system going to do (functional/structural decomposition)?
- Failure – How could the function fail?
- Effect – What could be the outcome of the failure?
- Cause – What could be the cause of the failure?
- Control – How is the failure currently controlled?
- Control type – is this prevention or detection?

Function	Failure	Effect	SEV	Cause	OCC	Control	Control Type	DET	RPNi
Function1	Failure mode 1	Effect 1	2	Cause 1	9	Detection1	Detection	6	108
	Failure mode 2	Effect 2	8	Cause 2	2	Detection2	Detection	6	96
	Failure mode 3	Effect 3	1	Cause 3	3	Detection3	Detection	6	18
Function2	Failure mode 1	Effect 1	6	Cause 1	7	Detection1	Detection	6	252
	Failure mode 2	Effect 2	1	Cause 2	2	Detection2	Detection	6	12

FMEA Example
- See Whiteboard -

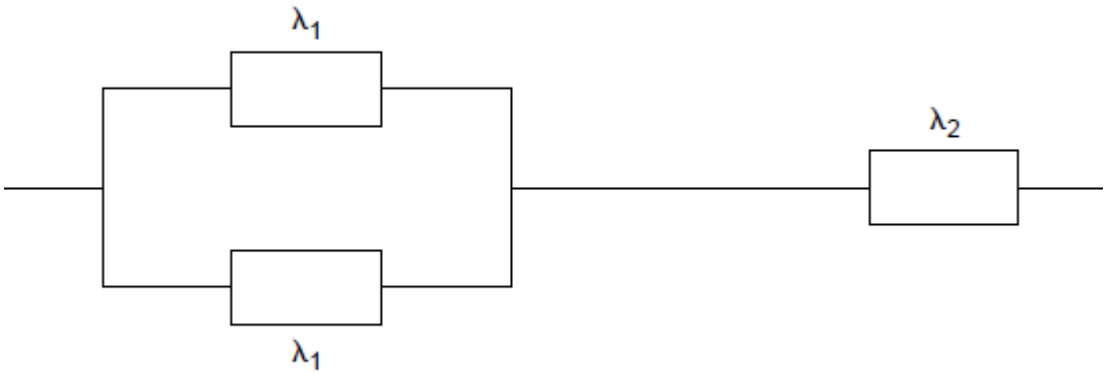
- We will take the software technical specification and derive possible failures, causes and detection mechanisms.
- The intent here is to specify additional non-functional software requirements (derived requirements).
- When thinking about software failures consider this:

Quality	Description of Quality
Accuracy	The term *accuracy* denotes the degree of freedom from error of sensor and operator input, the degree of exactness possessed by an approximation or measurement, and the degree of freedom of actuator output from error.
Capacity	The terms *capacity* denotes the ability of the software system to achieve its objectives within the hardware constraints imposed by the computing system being used. The main factors of capacity are Execution Capacity (timing) and Storage Capacity (sizing). These refer, respectively, to the availability of sufficient processing time and memory resources to satisfy the software requirements.
Functionality	The term *functionality* denotes the operations which must be carried out by the software. Functions generally transform input information into output information in order to affect the reactor operation. Inputs may be obtained from sensors, operators, other equipment or other software as appropriate. Outputs may be directed to actuators, operators, other equipment or other software as appropriate.
Reliability	The term *reliability* denotes the degree to which a software system or component operates without failure. This definition does not consider the consequences of failure, only the existence of failure. Reliability requirements may be derived from the general system reliability requirements by imposing reliability requirements on the software components of the application system which are sufficient to meet the overall system reliability requirements.
Robustness	The term *robustness* denotes the ability of a software system or component to function correctly in the presence of invalid inputs or stressful environmental conditions. This includes the ability to function correctly despite some violation of the assumptions in its specification.
Safety	The term *safety* is used here to denote those properties and characteristics of the software system that directly affect or interact with system safety considerations. The other qualities discussed in this table are important contributors to the overall safety of the software-controlled protection system, but are primarily concerned with the internal operation of the software. This quality is primarily concerned with the affect of the software on system hazards and the measures taken to control those hazards.
Security	The term *security* denotes the ability to prevent unauthorized, undesired and unsafe intrusions. Security is a safety concern in so far as such intrusions can affect the safety-related functions of the software.

Source:
Software Safety Hazard Analysis, J. Lawrence, LBLL

Reliability Block Diagram (RBD)

- We need two things to compare different architectures (in EE):
  - A probabilistic model – probability law
  - A notation – Reliability Block Diagram (RBD) which assume probabilistic independent blocks
  - Each block has a defined function, a failure mode with a failure rate
  - A system function can be spread across different blocks (think of blocks as components)



Source:
Smith: Reliability, Maintainability and Risk

RBD Example
- See Whiteboard -

Fault Tree Analysis (FTA)

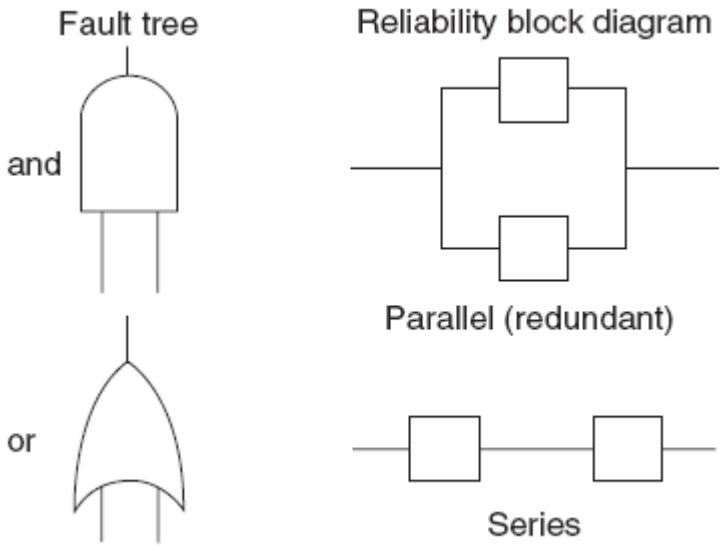
- Top event is failure mode (system or function)
- Devide system functions into sub-functions (functional decomposition) or system into components (component decomposition)
- Look into combinations of faults (strength of FTA)
- Tree like structure using combinatorical logic (can include probabilistic properties)
- Paths of Failure

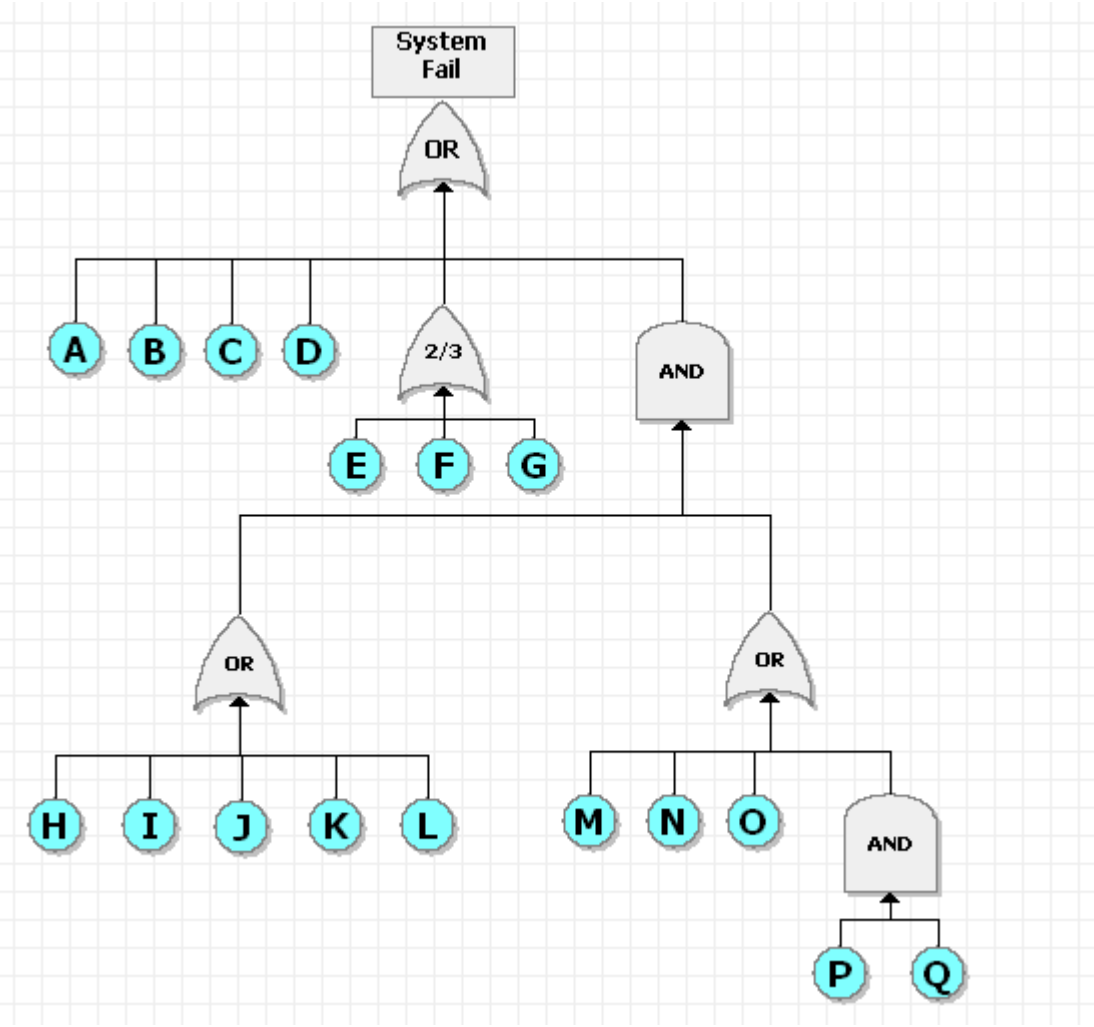
Outcome:

- Root cause event (external, internal) that (in combination) will lead to top event → failure modes of sub-functions or components
- Good system understanding – very useful if applied to existing systems to isolate reliability issues

FTA II

- FTA is semantically equivalent to Reliability Block Diagram (RBD)





Source:
Smith, Functional Safety

FTA Example
- See Whiteboard -