Universität Bremen

Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Systems of High Safety and Security

Lecture 4 from 05.11.2025: Hazard Analysis

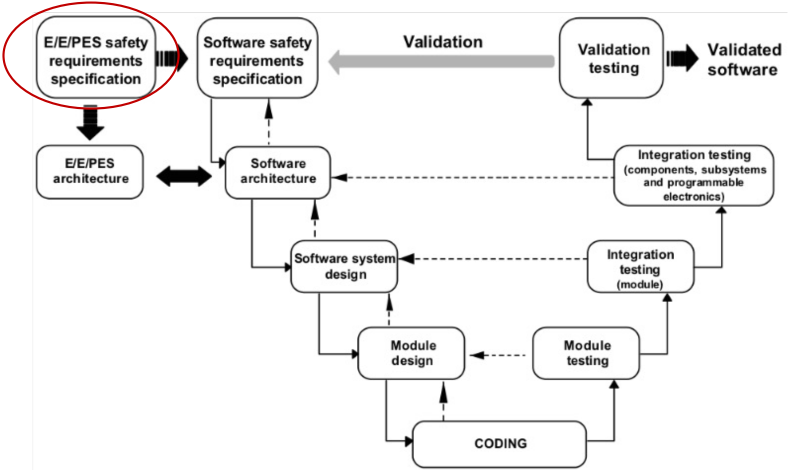
Winter term 2025/26

Christoph Lüth

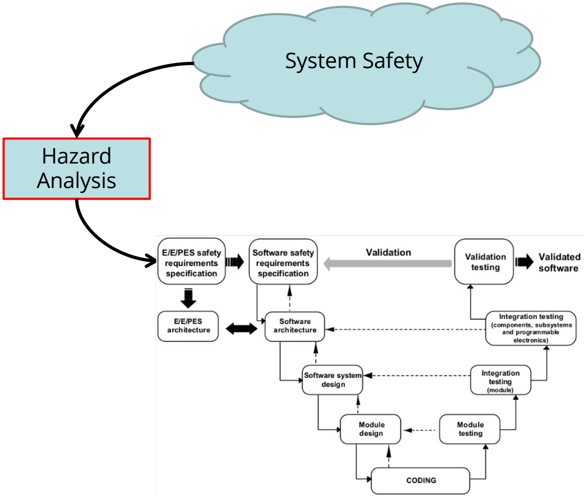
Roadmap

- Introduction
- Legal Requirements - Norms and Standards
- The Development Process
- Hazard Analysis
- The Big Picture: Hybrid Systems
- Temporal Logic with LTL and CTL
- Operational Semantics
- Axiomatic Semantics - Specifying Correctness
- Floyd-Hoare Logic
- A Simple Compiler and its Correctness
- Hardware Verification
- A Simple TinyRV32 Core
- Conclusions

Hazard Analysis in the Development Cycle


E/E/PES safety requirements specification
Software safety requirements specification
Validation
Validation testing
Validated software
E/E/PES architecture
Software architecture
Integration testing (components, subsystems and programmable electronics)
Software system design
Integration testing (module)
Module design
Module testing
CODING

The Purpose of Hazard Analysis



System Safety

Hazard Analysis

E/E/PES safety requirements specification

Software safety requirements specification

Validation

Validation testing

Validated software

E/E/PES architecture

Software architecture

Integration testing (components, subsystems and programmable electronics)

Software system design

Integration testing (module)

Module design

Module testing

CODING

- Hazard Analysis systematically determines a list of **safety requirements**.
- The realization of the safety requirements by the software product must be **verified**.
- The product must be **validated** w.r.t. the safety requirements.

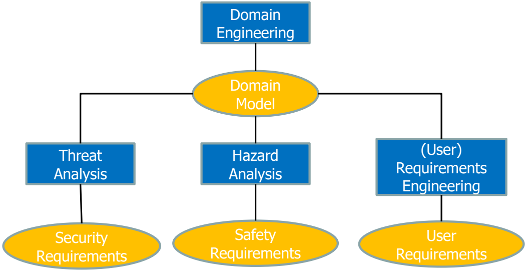
Hazard Analysis …

- … provides the basic **foundations** for **system safety**.
- … is performed to **identify** hazards, hazard **effects**, and hazard **causal** factors.
- … is used to determine **system risk**, to determine the significance of hazards, and to establish **design measures** that will eliminate or mitigate the identified hazards.
- … is used to **systematically** examine systems, subsystems, facilities, components, software, personnel, and their interrelationships.

Clifton Ericson: *Hazard Analysis Techniques for System Safety*.
Wiley-Interscience, 2005.

Side remark: User Requirements

- The objective of hazard analysis is to produce a **complete** and **consistent** set of safety requirements.
- Complementary to safety and security requirements, the **user requirements** express what the system should do from the end-user perspective.
- User requirements are systematically derived in two steps:
  - **Domain engineering**
  - **Requirements engineering**



See Bjørner D. (2010) Domain Engineering.
In: Boca P., Bowen J., Siddiqi J. (eds) Formal Methods: State of the Art and New Directions. Springer, London. https://doi.org/10.1007/978-1-84882-736-3_1

Form and Output of Hazard Analysis

The **output** of hazard analysis is a list of **safety requirements** and **documents** detailing how these were derived.

- Because the process is informal, it can only be **checked** by **reviewing**.
- It is therefore **critical** that
  - standard forms of analysis are used,
  - documents have a standardized form, and
  - all assumptions are documented.

Classification of Hazard Analysis

- **Top-down methods** start with an anticipated hazard and work backwards from the hazard event to potential causes for the hazard.
  - Good for finding causes for hazard;
  - good for avoiding the investigation of “non-relevant” errors;
  - bad for detection of missing hazards.

- **Bottom-up methods** consider “arbitrary” faults and resulting errors of the system and investigate whether they may finally cause a hazard.
  - Properties are complementary to top-down properties;
  - Not easy with software where the structure emerges during development.

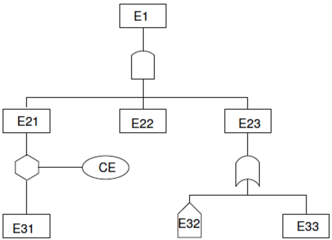
Hazard Analysis Methods

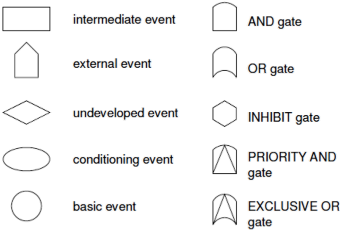
- **Fault Tree Analysis (FTA)** – top-down
- **Event Tree Analysis (ETA)** – bottom-up
- **Failure Modes and Effects Analysis (FMEA)** – bottom up
- Cause Consequence Analysis – bottom up
- HAZOP Analysis – bottom up
- Markov chains in combination with reachability analysis – top-down
    - Allows for stochastic modelling of complex world models and effective model checking, as long as models are not too large.

Fault Tree Analysis

Fault Tree Analysis (FTA)

- Top-down deductive failure analysis (of undesired states)
  - Define undesired top-level event (UE);
  - Analyze all causes affecting an event to construct fault (sub)tree;
  - Evaluate fault tree.





FTA: Cut Sets

- A **cut set** is a set of events that cause the top UE to occur (also called a **fault path**).
- Cut sets reveal critical and weak links in a system.
- Extension- **probabilistic** fault trees:
  - Annotate events with probabilities;
  - Calculate probabilities for cut sets.
  - Useful for hardware faults and unpredictable events in the environment.
- Cut sets can be calculated top down or bottom up.
  - MOCUS algorithm (Ericson, 2005)
  - Corresponds to the DNF of underlying formula.
  - Inhibit gate, priority and gate, exclusive or gate need to be transformed first into AND, OR, with event negation

Fault-Tree Analysis: Process Overview

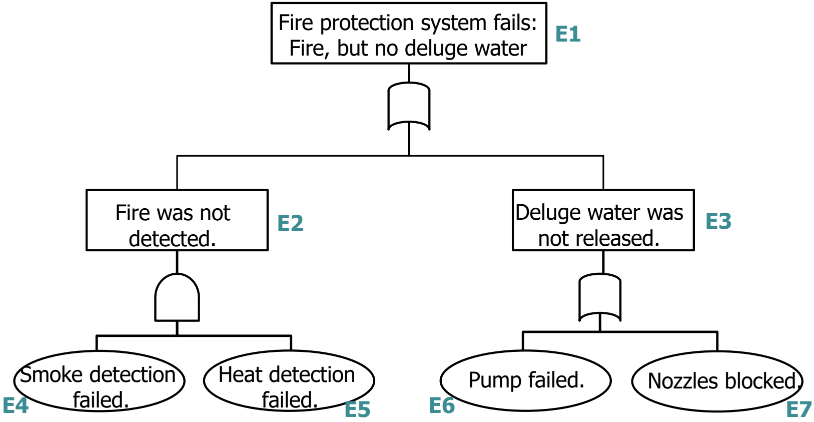
1. Understand system design.
2. For all identified hazards:
    1. Define top undesired event;
    2. Establish boundaries (scope);
    3. Construct fault tree;
    4. Evaluate fault tree (cut sets, probabilities);
    5. Validate fault tree (check if correct and complete);
    6. Modify fault tree (if required);
    7. Document analysis.

MOCUS Algorithm: Calculating the Minimal Cut Sets

1. Name all gates and events.
2. Place top event in the first row.
3. Replace top gate with inputs:
    1. Inputs of AND-gates are kept as lists (sets) in the row;
    2. Inputs of OR-gates are put into a new row each.
4. Move down the fault tree, replacing non-basic events with their inputs this way.
5. When only basic events remain: the remaining lists are the cut sets.
6. Remove all non-minimal cut sets and duplicate cut sets.

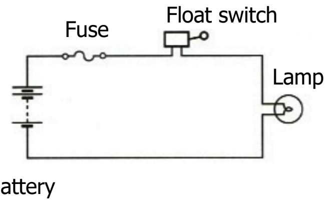
Fault Tree Analysis: First Simple Example

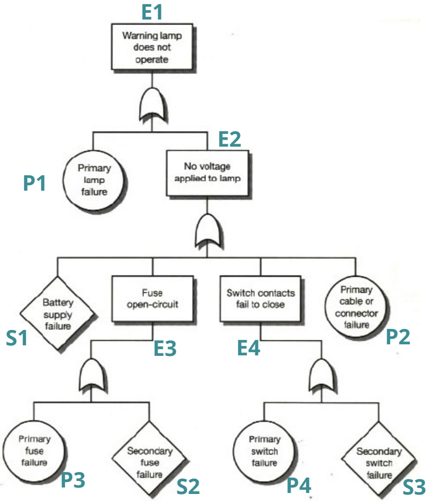
- Consider a simple **fire protection system** connected to smoke/heat detectors.



Fault Tree Analysis: Another Example

- A lamp warning about low level of brake fluid.
- Top undesired event: warning lamp off despite low level of fluid.



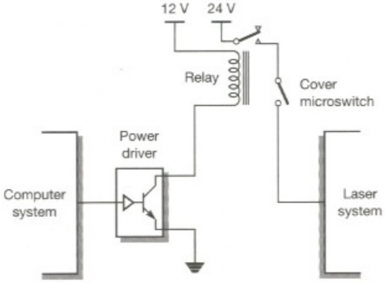


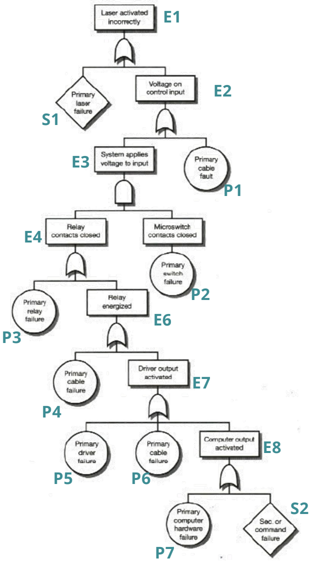
Source: N. Storey, Safety-Critical Computer Systems.

Fault Tree Analysis: Final Example

A laser is operated from a control computer system.

- The laser is connected via a relay and a power driver and protected by a cover switch.
- Top Undesired Event:
  Laser activated without explicit command from computer system.





Source: N. Storey, Safety-Critical Computer Systems.

Extended FTA: Consider Functional Insufficiencies

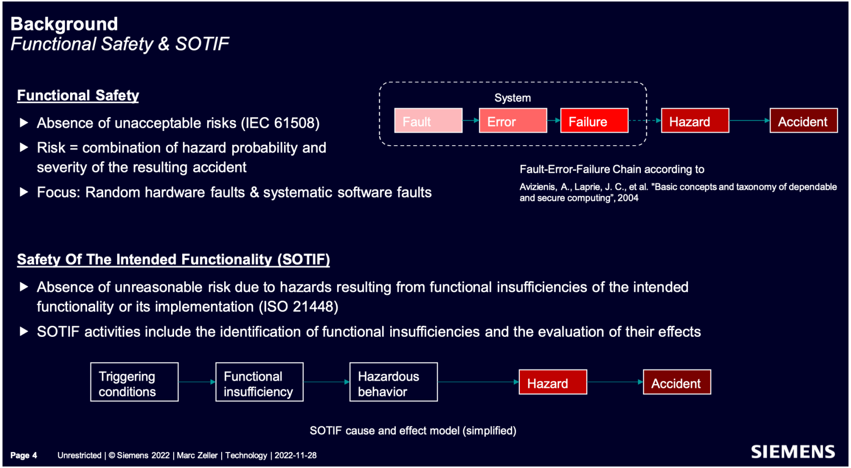
- **Background**
  - Whenever functionality based on machine learning is employed, their **actual functional behaviour** may deviate from the **intended functional behaviour**.
- **Example**
  - Deep Neural Networks has been trained to detect obstacles on railway tracks – this function OD is safety critical and needed for autonomous (driverless) trains.
  - The **intended functionality** is: OD outputs flag “obstacle is present” if and only if an obstacle on the track exists
  - Due to insufficient training of the DNN (the DNN may be implemented correctly!), **the actual functional behaviour** of OD may result in
    - False Negatives: OD signals “no obstacle” though there is one
    - False Positives: OD signals “obstacle” though there is none

Extended FTA: Consider Functional Insufficiencies

- The standard IS0 21448 has coined the term:
  - **Safety of the Intended Functionality (SOTIF)** in the context of autonomous road vehicles
  - A correctly implemented, but insufficiently trained DNN, for example, cannot guarantee the desired SOTIF, though the DNN implementation does not contain any bugs.

- Consequently, it is advisable to distinguish between different types of events in the nodes of a fault tree
  - Technical fault or software fault
  - Functional insufficiency (e.g. due to inadequate training)

Marc Zeller: **Component Fault and Deficiency Tree (CFDT): Combining Functional Safety and SOTIF Analysis.**
IMBSA 2022: 146-152



Background
Functional Safety & SOTIF

Functional Safety

- Absence of unacceptable risks (IEC 61508)
- Risk = combination of hazard probability and severity of the resulting accident
- Focus: Random hardware faults & systematic software faults

Fault-Error-Failure Chain according to

Avizienis, A., Laprie, J. C., et al. "Basic concepts and taxonomy of dependable and secure computing", 2004

Safety Of The Intended Functionality (SOTIF)

- Absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or its implementation (ISO 21448)
- SOTIF activities include the identification of functional insufficiencies and the evaluation of their effects

SOTIF cause and effect model (simplified)

Page 4 Unrestricted | © Siemens 2022 | Marc Zeller | Technology | 2022-11-28

SIEMENS

Marc Zeller: **Component Fault and Deficiency Tree (CFDT): Combining Functional Safety and SOTIF Analysis.** IMBSA 2022: 146-152.

FTA - Conclusions

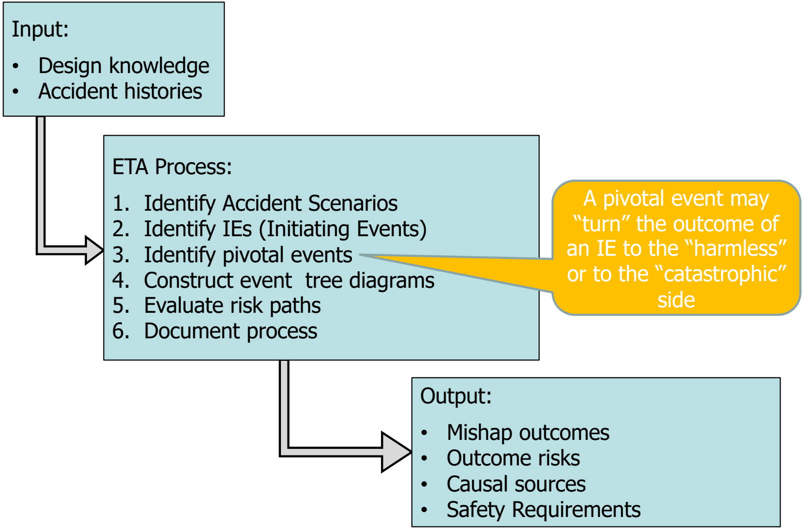
- **Advantages**:
  - Structured, rigorous, methodical approach;
  - Can be effectively performed and computerized, commercial tool support;
  - Easy to learn, do, and follow;
  - Combines hardware, software, environment, human interaction.
- **Disadvantages**:
  - Can easily become time-consuming and a goal in itself, rather than a tool to identify safety requirements
  - Modelling sequential timing and multiple phases is difficult.
  - Distinction between events and states always needs to be clarified

Event Tree Analysis

Event Tree Analysis (ETA)

- Bottom-up method
- Applies to a chain of cooperating activities
- Investigates the effect of activities failing while the chain is processed
- Depicted as binary tree; each node has two leaving edges:
  - Activity operates correctly
  - Activity fails
- Useful for calculating risks by assigning probabilities to edges
- Complexity: $\mathcal{O}(2^n)$

Event Tree Analysis - Overview



Input:

- Design knowledge
- Accident histories

ETA Process:

1. Identify Accident Scenarios
2. Identify IEs (Initiating Events)
3. Identify pivotal events
4. Construct event tree diagrams
5. Evaluate risk paths
6. Document process

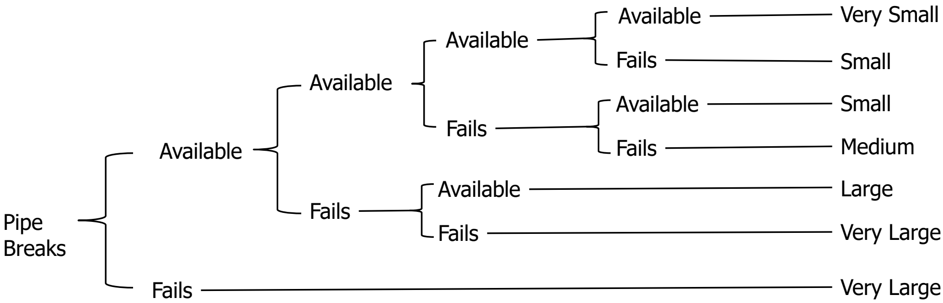
A pivotal event may "turn" the outcome of an IE to the "harmless" or to the "catastrophic" side

Output:

- Mishap outcomes
- Outcome risks
- Causal sources
- Safety Requirements

Example: Cooling System for a Nuclear Power Plant

Initating Event	*Pivotal Events*				*Outcome*
	Electricity	Emergency Core Cooling	Fission Product Removal	Containment	Fission Release



Probabilistic ETA:
Fire Detection/Suppression System for Office Building

Initating Event Probability	*Pivotal Events*			*Outcome*	*Prob.*
	Fire Detection Working	Fire Alarms Working	Fire Sprinkler Working		
Fire Starts P= 0.01	YES (P= 0.9)	YES (P= 0.7)	YES (P= 0.8)	Limited damage	0.00504
			NO (P= 0.2)	Extensive damage, People escape	0.00126
		NO (P= 0.3)	YES (P= 0.8)	Limited damage, Wet people	0.00216
			NO (P= 0.2)	Death/injury, Extensive damage	0.00054
	NO (P= 0.1)			Death/injury, Extensive damage	0.001

ETA - Conclusions

- **Advantages**:
  - Structured, rigorous and methodical;
  - Can be effectively computerized, tool support is available;
  - Easy to learn, do, and follow;
  - Combines hardware, software, environment and human interaction;
  - Can be effectively performed on varying levels of system detail.

- **Disadvantages**:
  - An ETA can only have one IE;
  - Can overlook subtle system dependencies – pivotal events are not identified;
  - Partial success/failure not distinguishable;
  - High branching complexity in presence of many pivotal events.

Failure Modes and Effects Analysis

Failure Modes and Effects Analysis (FMEA)

- Analytic approach to review potential failure modes, their causes, and their effects.
- Three approaches: **functional**, **structural** or **hybrid**.
- Applicable to hardware and software-related analyses.
- It analyzes
  - the failure mode,
  - the failure cause,
  - the failure effect,
  - its criticality,
  - and the recommended action,

and presents them in a **standardized table**.

Software Failure Modes

Guide word	Deviation	Example Interpretation
omission	The system produces no output when it should. Applies to a single instance of a service but may be repeated.	No output in response to change in input; periodic output missing.
commission	The system produces an output, when a perfect system would have produced none. One must consider cases with both, correct and incorrect data.	Same value sent twice in series; spurious output, when inputs have not changed.
early	Output produced before it should be.	Really only applies to periodic events; Output before input is meaningless in most systems.
late	Output produced after it should be.	Excessive latency (end-to-end delay) through the system; late periodic events.
Value (detectable)	Value output is incorrect, but in a way, which can be detected by the recipient.	Out of range.
value (undetectable)	Value output is incorrect, but in a way, which cannot be detected.	Correct in range; but wrong value

Criticality Classes

- Risk as given by the *risk mishap index* (MIL-STD-882):

Severity

1. Catastrophic
2. Critical
3. Marginal
4. Negligible

Probability

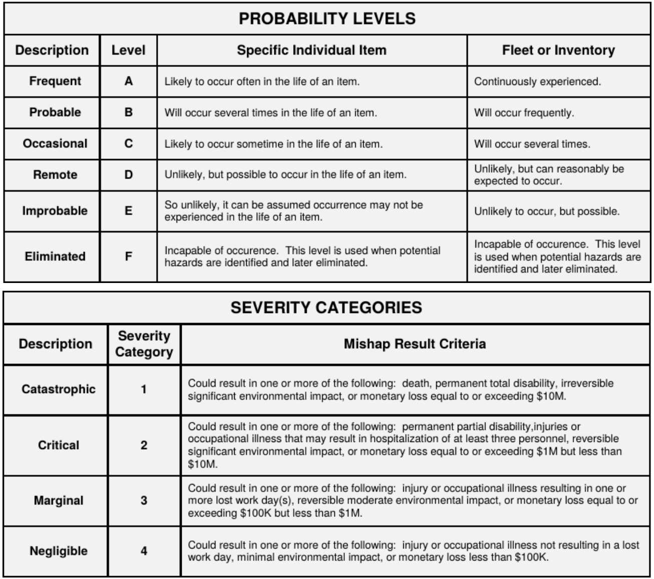
A. Frequent

B. Probable

C. Occasional

D. Remote

E. Improbable



PROBABILITY LEVELS			
Description	Level	Specific Individual Item	Fleet or Inventory
Frequent	A	Likely to occur often in the life of an item.	Continuously experienced.
Probable	B	Will occur several times in the life of an item.	Will occur frequently.
Occasional	C	Likely to occur sometime in the life of an item.	Will occur several times.
Remote	D	Unlikely, but possible to occur in the life of an item.	Unlikely, but can reasonably be expected to occur.
Improbable	E	So unlikely, it can be assumed occurrence may not be experienced in the life of an item.	Unlikely to occur, but possible.
Eliminated	F	Incapable of occurence. This level is used when potential hazards are identified and later eliminated.	Incapable of occurence. This level is used when potential hazards are identified and later eliminated.

SEVERITY CATEGORIES		
Description	Severity Category	Mishap Result Criteria
Catastrophic	1	Could result in one or more of the following: death, permanent total disability, irreversible significant environmental impact, or monetary loss equal to or exceeding $10M.
Critical	2	Could result in one or more of the following: permanent partial disability,injuries or occupational illness that may result in hospitalization of at least three personnel, reversible significant environmental impact, or monetary loss equal to or exceeding $1M but less than $10M.
Marginal	3	Could result in one or more of the following: injury or occupational illness resulting in one or more lost work day(s), reversible moderate environmental impact, or monetary loss equal to or exceeding $100K but less than $1M.
Negligible	4	Could result in one or more of the following: injury or occupational illness not resulting in a lost work day, minimal environmental impact, or monetary loss less than $100K.

Source:MIL-STD-822E, www.system-safety.org/Documents/MIL-STD-882E.pdf

FMEA Example: Airbag Control

- Consider an **airbag control system**, consisting of
  - the airbag with gas cartridge;
  - a control unit with
    - Output: Release airbag
    - Input: Accelerometer, impact sensors, seat sensors, …
- FMEA:
  - **Structural**: what can be broken?
    - Mostly hardware faults.
  - **Functional**: how can it fail to perform its intended function?
    - Also applicable for software.

Airbag Control (Structural FMEA)

ID	Mode	Cause	Effect	Crit.	Appraisal
1	Omission	Gas cartridge empty	Airbag not released in emergency	C1	SR-56.3
2	Omission	Cover does not detach	Airbag not released fully in emergency	C1	SR-57.9
3	Omission	Trigger signal not present in emergency.	Airbag not released in emergency	C1	Ref. To SW-FMEA
4	Commission	Trigger signal present in non-emergency	Airbag released during normal vehicle operation	C2	Ref. To SW-FMEA

Airbag Control (Functional FMEA)

ID	**Mode**	**Cause**	**Effect**	**Crit.**	**Appraisal**
5-1	Omission	Software terminates abnormally	Airbag not released in emergency.	C1	See 5-1.1, 5-1.2.
5-1.1	Omission	- Division by 0	See 5-1	C1	SR-47.3 Static Analysis
5-1.2	Omission	- Memory fault	See 5-1	C1	SR-47.4 Static Analysis
5-2	Omission	Software does not terminate	Airbag not released in emergency.	C1	SR-47.5 Termination Proof
5-3	Late	Computation takes too long.	Airbag not released in emergency.	C1	SR-47.6 WCET Analysis
5-4	Commission	Spurious signal generated	Airbag released in non-emergency	C2	SR-49.3
5-5	Value (u)	Software computes wrong result	Either of 5-1 or 5-4.	C1	SR-12.1 Formal Verification

FMEA - Conclusions

- Advantages:
  - Easily understood and performed;
  - Inexpensive to perform, yet meaningful results;
  - Provides rigour to focus analysis;
  - Tool support available.
- Disadvantages:
  - Focuses on single failure modes rather than combination;
  - Not designed to identify hazard outside of failure modes;
  - Limited examination of human error, external influences or interfaces.

Hazard Analysis as a Reachability Problem

The analysis whether “finally something bad happens” is well-known from **property checking** methods:

- Create a **world model** describing everything (desired or undesired, with environment, including human users) which might happen in the system under consideration.
- Specify a logical property P describing the undesired situations.
- Check the model whether a path – that is, a sequence of state transitions – exists such that P is fulfilled on this path.
- Calculate the probability that the path fulfilling P is executed (by stochastic model checking, e.g. with PRISM https://www.prismmodelchecker.org)
- Specify as safety requirement that mechanisms shall exist preventing paths leading to P from being taken.

Conclusions

The Seven Principles of Hazard Analysis

Source: Ericson (2005)

1) Hazards, mishaps and risk are not chance events.
2) Hazards are created during design.
3) Hazards are comprised of three components (HE, IM, T/T).
4) Hazards and mishap risk is the core safety process.
5) Hazard analysis is the key element of hazard and mishap risk management.
6) Hazard management involves seven key hazard analysis types.
7) Hazard analysis primarily encompasses seven hazard analysis techniques.

Summary

- Hazard Analysis is the **start** of the formal development.
- Its most important output are **safety requirements**.
- Adherence to safety requirements has to be **verified** during development and **validated** at the end.
- We distinguish different types of analysis:
  - Top-Down analysis (Fault Trees)
  - Bottom-up (FMEAs, Event Trees)
- It makes sense to **combine** different types of analyses, as their results are complementary.

Conclusions

- Hazard Analysis is a creative process, as it takes an informal input („system safety“) and produces a formal output (safety requirements).

- Its results cannot be formally proven, merely checked and reviewed.

- Review plays a key role.

- Therefore,
  - documents must be readable, understandable, auditable;
  - analysis must be in well-defined and well-documented format;
  - all assumptions must be well documented.