Safety Engineering

COMP6226: Software Modelling Tools and Techniques for Critical Systems

Dr A. Rezazadeh (Reza)
Email: ra3@ecs.soton.ac.uk or ar4k06@soton.ac.uk

November 24

University of Southampton

Overview

- What is Safety?
- Safety-Related Concepts
- Safety-critical System Development
- Potential Influence of Software on Safety
- Hazard Analysis
- Safety-critical System Development
- Hazard and Risk Management Paradigm
- Hazard Likelihood Categories
- Hazard and Risk Management
- The Safety Life-cycle
- Safety Requirements Specification
- Safety Case
- Safety Case: Key Components
- Designing for Safety

University of Southampton

Objectives

- To understand Safety and Safety related concepts
- Safety-critical System Development
- Hazard Analysis
- Apply Hazard and Risk Management in the context of Software System

What is Safety?

- Safety is a property of a system to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment.
- The system may be software-controlled so that the decisions made by the software and subsequent actions are safety-critical.
  - Therefore, the software behaviour is directly related to the overall safety of the system.
- **Safety is not reliability!**
  - Reliability is the probability that a system will perform its intended function satisfactorily.
- **Safety is not security!**
  - Security is protection or defence against attack, interference, or espionage.

Safety-Related Concepts

- Safety is freedom from accidents or losses.
- An accident is an undesired and unplanned (but not necessarily unexpected) event that results in (at least) a specified level of loss (loss of life, damage to property or the environment).
- A hazard is a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead potentially to an accident (loss event).
- Risk is a combination of the likelihood of an accident and its severity

$$risk = P(a) * S(a)$$

- P: Probability S: Severity

Safety-Related Concepts

- A failure is the non-performance of a system or component, a random fault
  - Random failures are physical failures brought on by excessive stress on the device.
  - Failures are events – e.g., a component failure
- An error is a systematic fault
  - A systematic fault is a design error
  - Errors are states or conditions
- A fault is either a failure or an error

Potential Influence of Software on Safety

- A required function does not occur
  - Failure of the software to perform a required function; that is, the function is never executed, or no output is produced.
- An undesired event occurs
  - The software performs a function not required. (i.e. getting the wrong answer, issuing the wrong control instruction, or doing the right thing but under inappropriate conditions).
- An incorrect sequence of required events
  - The software possesses sequencing problems. For example, failing to ensure that two events happen at the same time, at different times, or in a particular order.

Potential Influence of Software on Safety – Cont.

- Timing failures in event sequences
  - The software exceeds maximum time constraints between events, fails to ensure minimum time constraints between events or possesses duration failures.
- An incorrect response to a safety-critical event
  - The software fails to recognise a hazardous condition requiring corrective action, fails to initiate a fault tolerant response to a recognized safety-critical function, or produces the wrong response to a hazardous condition or failure mode.

Hazard Analysis

- It is less expensive and far more effective to build in safety early than try to tackle it later
- The Hazard Analysis ties together hazards, faults, and safety measures
- Hazard analysis represents the heart of an effective safety programme.
  - The objective is to discover potential hazards and causes which may arise given a systems operational environment.

Safety-critical System Development

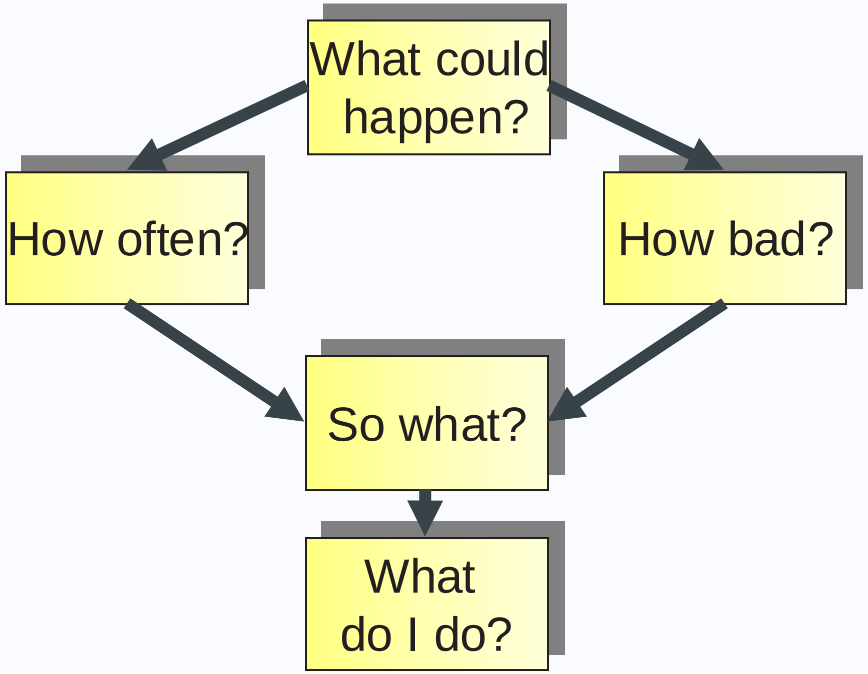
1. Identify the Hazards
2. Determine the Risks
3. Define the Safety Measures
4. Create Safe Requirements
5. Create Safe Designs
6. Implement Safety
7. Assure the Safety Process
8. Validate & Verify through formal modelling, Exhaustive Testing, static and adynamic analysis, model checking and so on.

high level goals

Safety Analysis

Hazard and Risk Management Paradigm

- Example Hazard Severity Categories
  - **Category I**: Catastrophic; may cause death or system loss.
  - **Category II**: Critical; may cause severe injury, severe occupational illness, or system damage.
  - **Category III**: Marginal; may cause minor injury, minor occupational illness, or minor system damage.
  - **Category** IV**:** Negligible; will not result in injury, occupational illness, or system damage.



Hazard Likelihood Categories

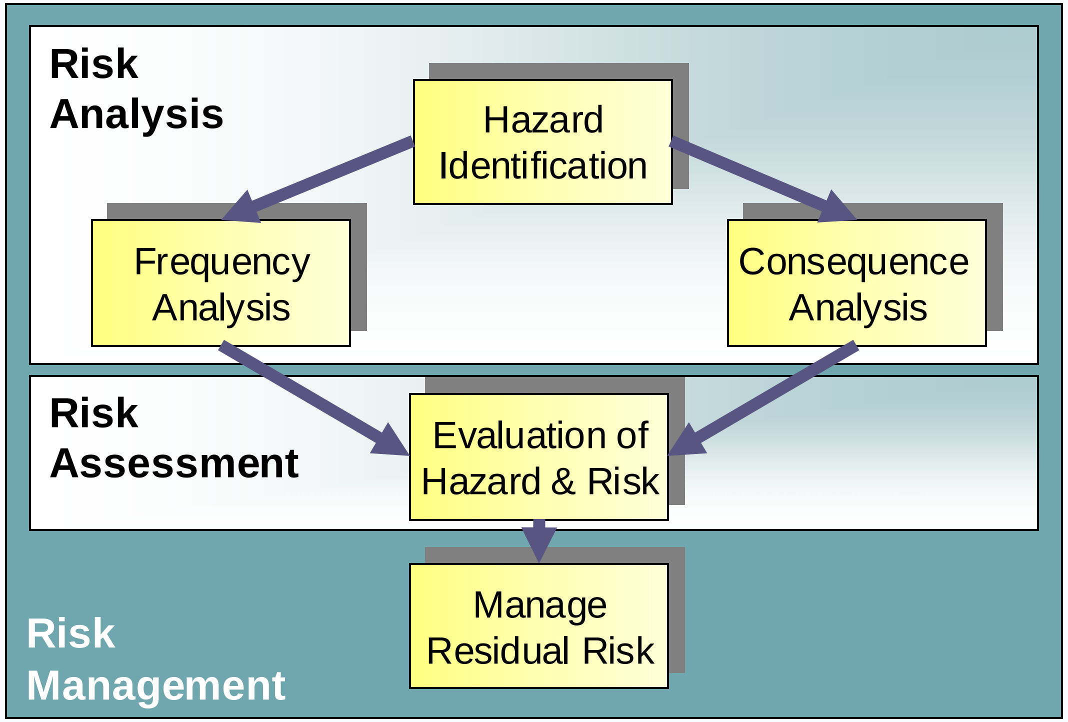
- Based on Leveson (Leveson, N.G. 1995):
  - Frequent: Likely to occur frequently to an individual item, continuously experienced throughout the fleet or inventory.
  - Probable: Will occur several times during the life of an individual item, frequently throughout the fleet or inventory.
  - Occasional: Likely to occur sometimes during the life of an individual item, several times throughout the fleet or inventory.

Hazard Likelihood Categories – Cont.

- Based on Leveson (Leveson, N.G. 1995):
- Remote: Unlikely to occur but possible during the life of an individual item but reasonably expected to occur in a fleet or inventory.
- Improbable: Extremely unlikely to occur to an individual item; possible for a fleet or inventory.
- Physically Impossible: Cannot occur to an individual item or in a fleet or inventory.

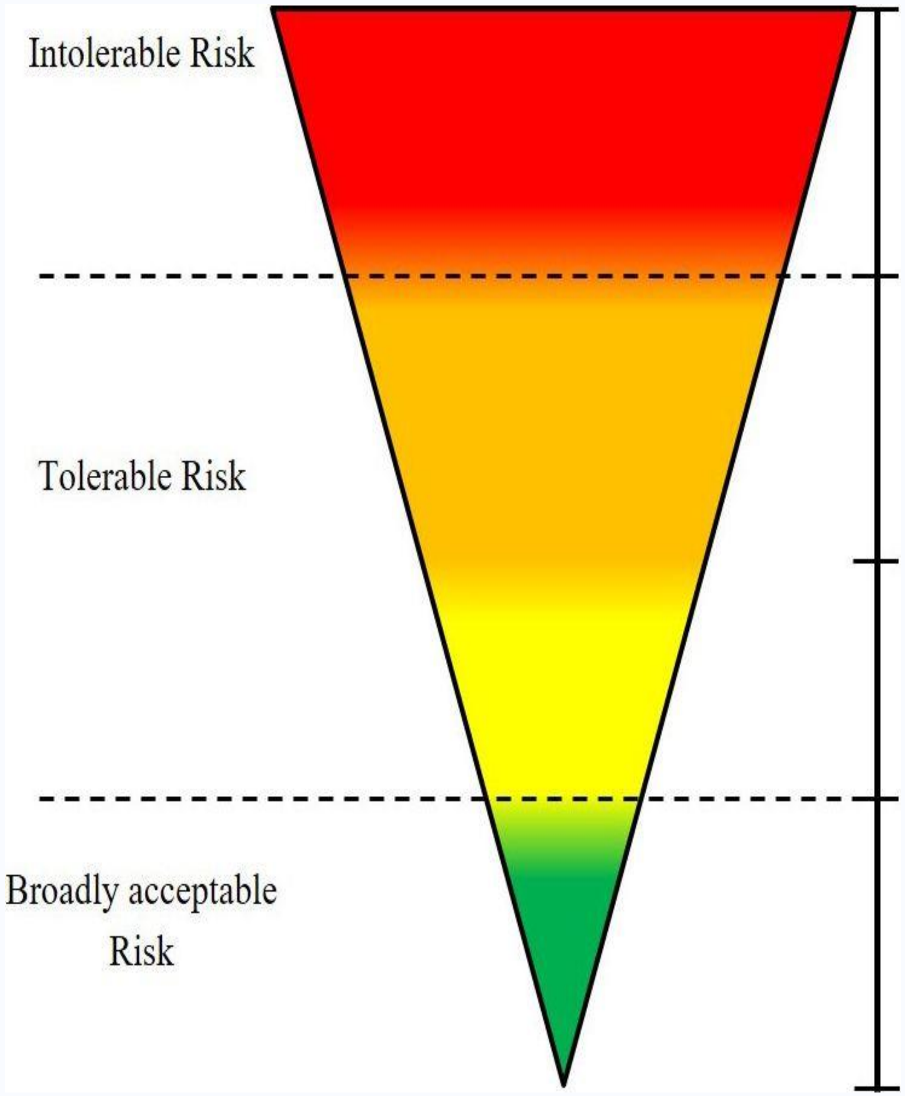
Hazard and Risk Management

- When is a System Safe Enough?
  - We identified the Hazards and ensured there were adequate Safeguards, consistent with the ALARP (As Low as Reasonably Practicable) principle
  - The cost emphasis of ALARP – an encouragement to add safeguards until increased benefits through risk reduction cannot be justified



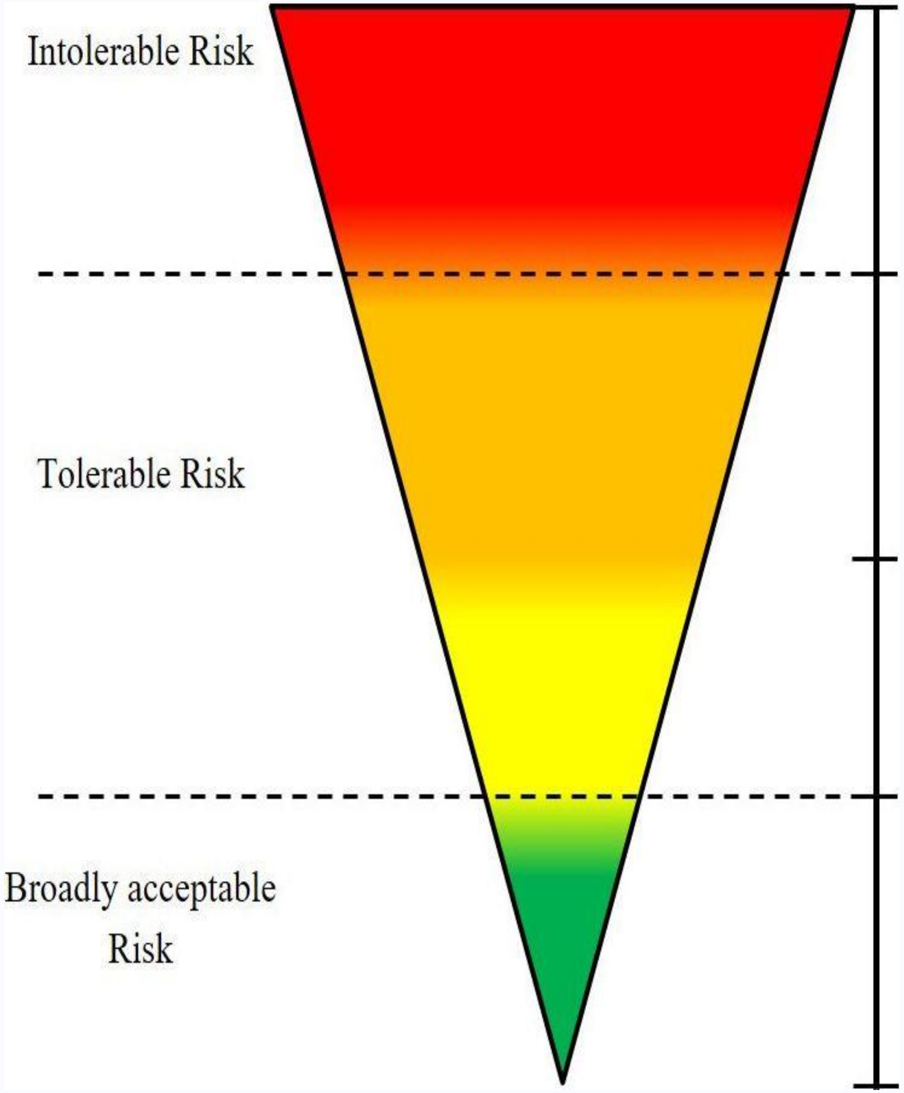
ALARP - As low as reasonably practicable

- "ALARP" is short for "as low as reasonably practicable".
- Reasonably practicable involves weighing a risk against the trouble, time and money needed to control it.
- Thus, ALARP describes the level to which we expect to see risks are controlled or reduced.



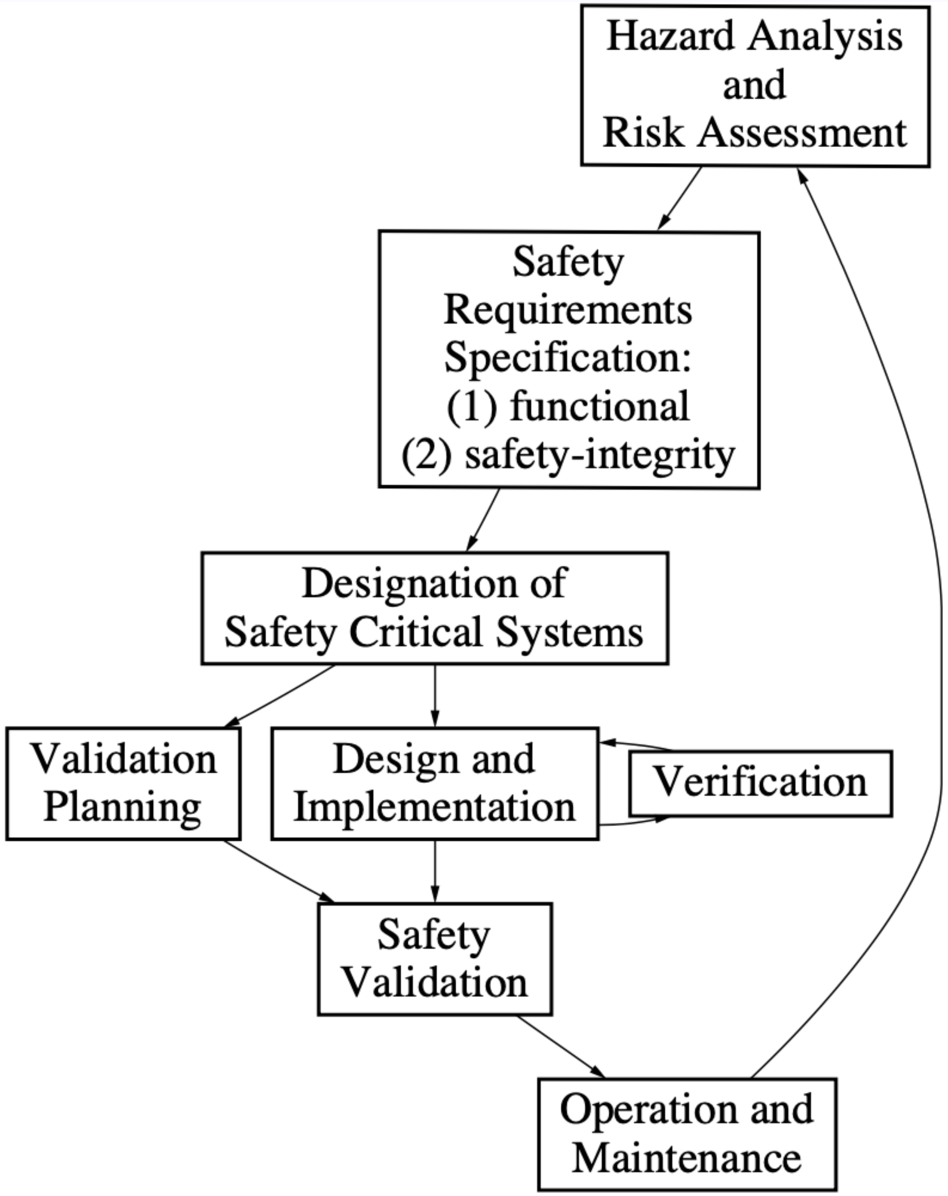
ALARP - As low as reasonably practicable

- **Unacceptable Region:**
  - Risks that are deemed unacceptable and must be reduced, regardless of the cost or effort involved.
- **ALARP Region:**
  - Risks that are tolerable but should be reduced to a level that is as low as reasonably practicable.
- **Broadly Acceptable Region:**
  - Risks that are so low that no further action is required, and they are considered to be adequately controlled.



The Safety Life-cycle

- You must identify the hazards of the system
- You must identify faults that can lead to hazards
- The Hazard Analysis feeds into the Requirements Specification
- You must define safety control measures to handle hazards



Hazard Analysis

- Hazard analysis represents the heart of an effective safety programme.
- The objective is to discover potential hazards and causes which may arise given a systems operational environment.

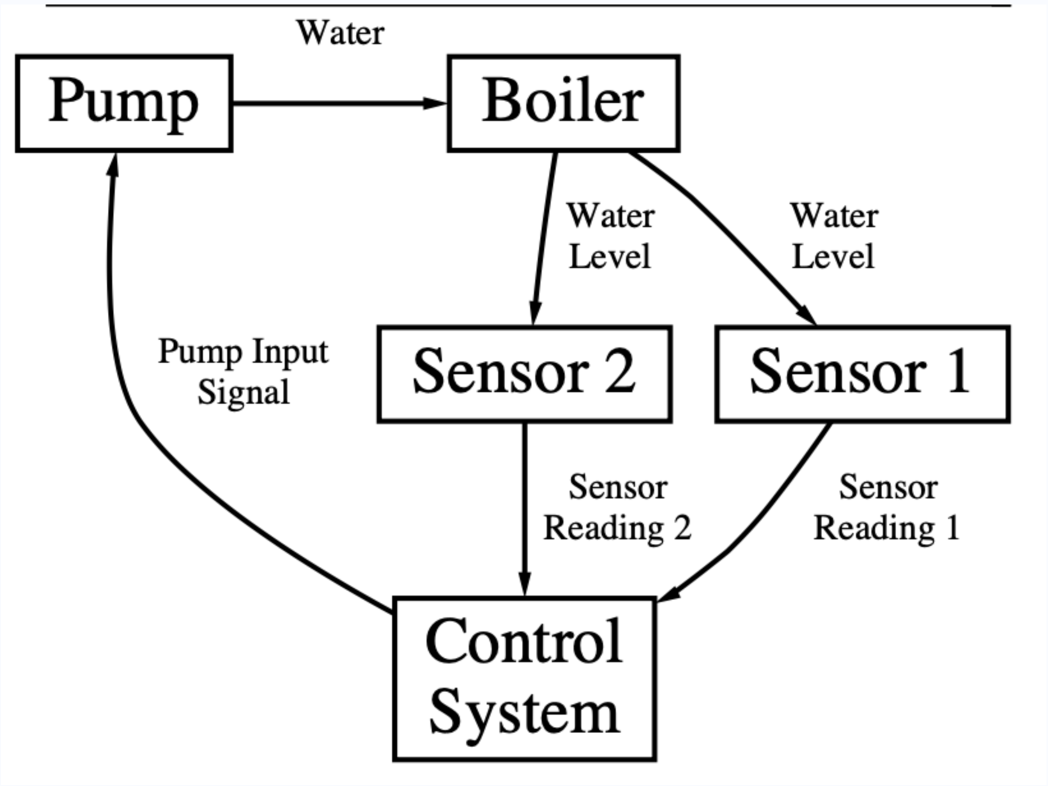
1. Hazard identification: determining what hazards might exist during operation of the system.
2. Hazard classification: order hazards with respect to hazard level (severity & likelihood of arising).
3. Hazard decomposition or causal analysis: an analysis of the individual hazards is carried out in order to determine root cause(s).

Hazard Analysis: Fault Trees

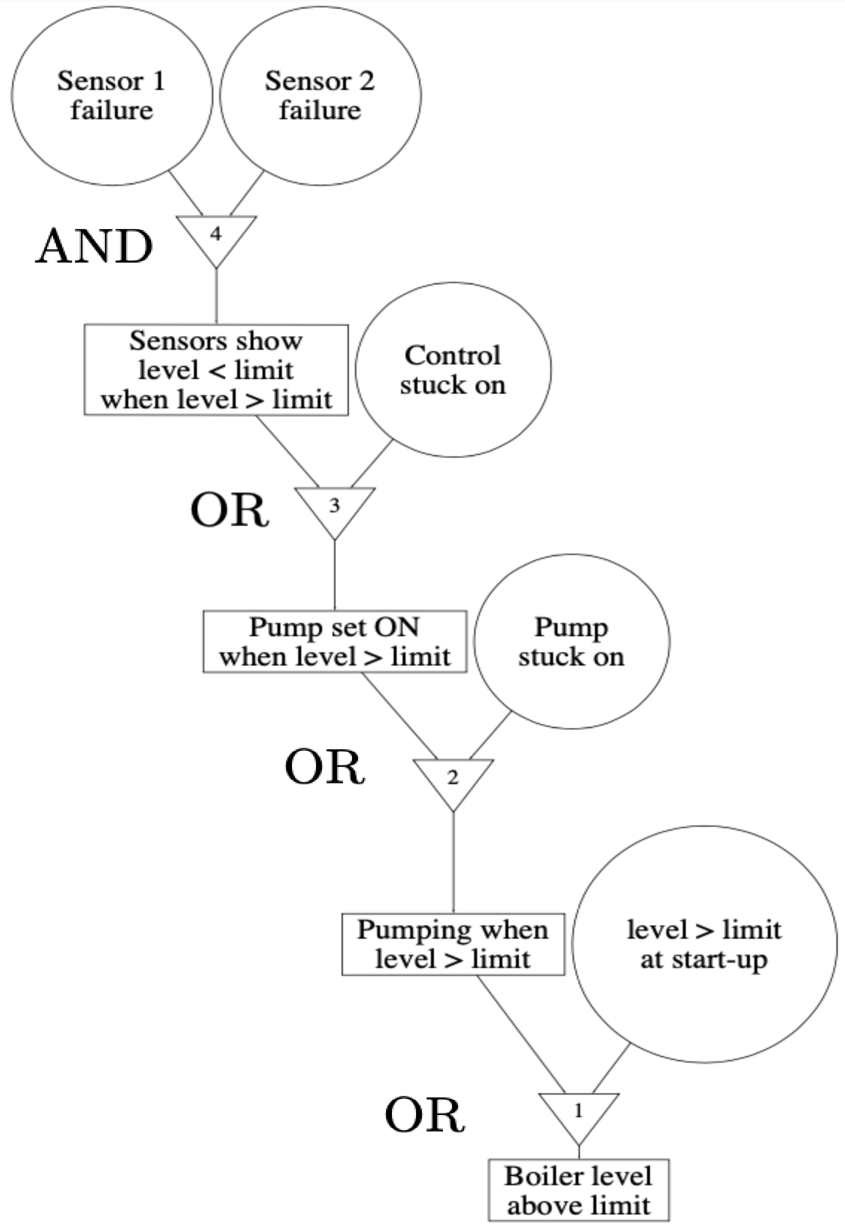
- There are many techniques for analysing hazards in order to identify the root causes.
- **Fault trees** are one such technique.
- Fault trees are used to show the causal links between systems events and particular system faults.
- Building blocks:
  - Basic events denoted by circles;
  - Intermediate events denoted by rectangles;
  - System fault denoted by the tree root;
  - Events are connected by AND- and OR-gates.

Example: A Simple Boiler System

- As an example, consider the following fault: “Boiler water level higher than safe limit.”



A Fault Tree for the Boiler System



Risk Assessment

- Risk assessment follows on from hazard analysis and is involved in classifying the acceptability or safety integrity level of each hazard.
- This classification is based upon the:
  - hazard level for each of the identified hazards and
  - likelihood of an accident resulting from the hazard.
- During hazard analysis worst-case effects are usually only considered and simple qualitative methods are employed.
- In contrast risk analysis involves more quantitative analyses, e.g. statistical analysis based upon historical data.

Safety Integrity Level

1. Intolerable: The system must be designed in such a way so that either the hazard cannot arise or, if it does arise, it will not result in an accident.

2. As low as reasonably practicable (ALARP): The system must be designed so that the probability of an accident arising because of the hazard is minimized subject to other considerations such as cost, delivery and so on.

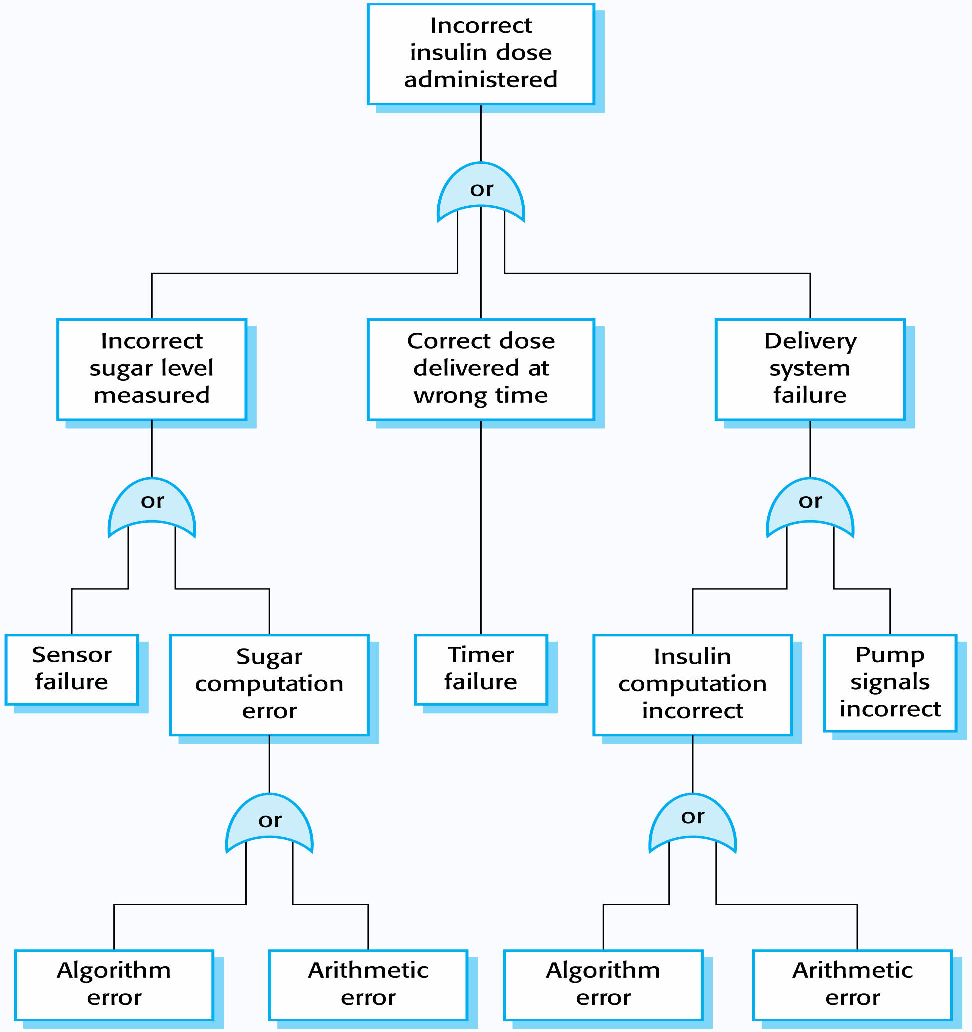
3. Acceptable: While the system designers should take all possible steps to reduce the probability of this hazard arising, these should not increase costs, delivery time or other non-functional system attributes.

Safety Requirements Specification

- Hazard analysis & risk assessment identify safety-critical functions and associated safety integrity levels which provide the basis for a safety criteria used by the system designers, i.e. a statement of what has to be achieved with respect to each of the hazards (but not how):

1. Hazard avoidance: design so that the hazard cannot arise, e.g. The "dead-man's handle" eliminates the hazard of a run-away train.
2. Hazard probability reduction: design to ensure that hazards cannot arise as a result of any single failure, e.g. install both software and hardware interlocks.
3. Accident prevention: design to detect hazards and remove them before an accident arises, e.g. automatic detection of aircraft engine fire linked with an automatic engine shut-down mechanism.

Insulin Pump – Software Fault Tree



Fault tree analysis

- Three possible conditions that can lead to delivery of incorrect dose of insulin
  - Incorrect measurement of blood sugar level
  - Failure of delivery system
  - Dose delivered at wrong time
- By analysis of the fault tree, root causes of these hazards related to software are:
  - Algorithm error
  - Arithmetic error

Insulin Pump - Software Risks

- Arithmetic error
  - A computation causes the value of a variable to overflow or underflow;
  - Maybe include an exception handler for each type of arithmetic error.
- Algorithmic error
  - Compare dose to be delivered with previous dose or safe maximum doses.
  - Reduce dose if too high.

Examples of safety requirements

SR1: The system shall not deliver a single dose of insulin that is greater than a **specified maximum dose** for a system user.

SR2: The system shall not deliver a daily cumulative dose of insulin that is greater than a specified maximum daily dose for a system user.

SR3: The system shall include a **hardware diagnostic** facility that shall be executed at least four times per hour.

SR4: The system shall include an **exception handler** for all of the exceptions that are identified in Table 3.

SR5: The **audible alarm** shall be sounded when any hardware or software anomaly is discovered and a diagnostic message, as defined in Table 4, shall be displayed.

SR6: In the event of an alarm, **insulin delivery** shall be **suspended** until the user has reset the system and cleared the alarm.

Safety Case

- Safety critical systems require certification by a regulating authority, e.g. FAA, HSE,
- A safety case documents the safety justification for a system and provides basis for the certification process –
- A safety case records all the safety activities and should therefore be created early on during the development of a system.
- A safety case must also be maintained throughout the operational life of a system.

Safety Case: Key Components

- A high-level rigorous safety argument, e.g.:
  - if X fails then Y will prevent Z from occurring ...
- Supporting evidence, e.g. test data, mathematical proofs showing that:
  - Y follows from the failure of X and that Y will prevent Z.
- All assumptions need to be made explicit; an unjustified assumption represents a flaw in the safety argument.
- Safety cases are typically multidisciplinary, consequently hard to manage.

Designing for Safety

- Design for Safety
  - Prevention
  - Detection and Control
- Design Trade Offs
  - Safety Vs Other features e. g. Fault Tolerance
  - Issues involved are moral, legal, financial
- Vulnerability to simple design errors
  - Tendency to neglect small errors
  - "Small Errors have small consequence” – not in Software

Reference

- Chapter 12 of Software Engineering by Ian Sommerville (10ed)
- Engineering a Safer World: Systems Thinking Applied to Safety by Nancy Leveson
- Safeware: System Safety and Computers, Leveson, N.G. Addison-Wesley, 1995.
- "Safety-Critical Computer Systems" Storey, N. Addison-Wesley, 1996.

YOUR QUESTIONS