CSC313 High Integrity Systems/ CSCM13 Critical Systems

Course Notes

Chapter 4: Safety Criteria

Anton Setzer
Dept. of Computer Science, Swansea University

http://www.cs.swan.ac.uk/∼csetzer/lectures/critsys/current/index.html

December 4, 2017

Requirements vs Specification

- **Requirements** and **specifications** are often used interchangeably.
- We have not found a crystal clear statement about the difference between **requirements** and **specification** in the literature.
- We will present first our own understanding and then present definitions as found in different sources.

Requirements vs Specification (Our Understanding)

- **Requirements** is a description of what a system should do, written from the **point of view of a user**.
  So it is written from the perspective of an outside user.
- **Specification** is a precise description of what a system, or component of a system should do, written from the **perspective of a developer of it**.
  It is written in such a way that one can **verify** (using **testing** or **formal verification**) whether the system fulfils it or not.
  - Writing a specification requires an understanding of how this system is implemented.
- **Requirement Specification** is a specification of a system derived from the requirements.
  - It makes the requirements precise enough so that one can **verify** whether the system fulfils it.

Wikipedia and Sommerville on Requirements

- From Wikipedia on Requirements Analysis:
  - Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the **needs** or **conditions to meet for a new or altered product**,
  - taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating and managing software or system requirements.
- Sommerville (Software Engineering, 8th Edition, p. 118):
  - **Requirements of a system** are the descriptions of the services a system should carry out and of the constraints.

Requirements vs Specifications (From Web Page)

- Answers on the following forum page
  http://programmers.stackexchange.com/questions/121289/what-is-the-difference-between-requirements-and-specifications
  - "The sound-bite answer is that requirements are what your program should do, the specifications are how you plan to do it.
    Another way to look at it is that the requirements represent the application from the perspective of the user, or the business as a whole. The specification represents the application from the perspective of the technical team. Specifications and requirements roughly communicate the same information, but to two completely different audiences."
  - "Requirements document what is needed - they shouldn't specify the how, but the what.
    Specifications document how to achieve the requirements - they should specify the how.
    In many places these documents are not separate and are used interchangeably."

Example

- Consider a simple specification of the safety of a track segment in an interlocking system.
  - In reality one needs exceptions to this, such as
    - possibility of overriding it under special circumstances,
    - allowing in a station two trains in the same segment.
- **Requirement:**
  - The signals should guarantee that if the trains obey the signals there are never two trains in the same train segment.
- **Specification:**
  - Whenever train detection detects that there is a train in a segment seg, all signals guarding segment seg are set to red.
  - Whenever a signal sig is green, and segment seg is guarded by sig, all signals sig' $\neq$ sig guarding seg are red.
- Specification requires that you have made precise what "guarded by" means.

Two Kinds of Requirements

(Sommerville, 8th Edition, p. 118)

- **User requirements** are statements written by the future user of a software product, usually in natural language or using diagrams, of what services the system is expected to provide and the constraints under which it must operate.
  - Developed by the system buyer (usually a company), who invites bids by software developers.
- **System requirements** is a precise statement which sets out the system services and constraints in detail.
  - Developed by the software developer so that the system buyer (the client) understands and can validate what the software will do.
  - It may serve as a contract between the system buyer and software developer.

Requirements Document

- **Requirements document** = document, which attempts to describe the requirements of a system in an unambiguous manner.
- Requirements include:
  - **Functional requirements.**
  - **Non-functional requirements.**
  - **Domain Requirements.**
  - **Safety Requirements.**

Functional Requirements

- **Functional requirements** describe
  - the functions a system should provide,
  - how the system should react to particular inputs
  - how the system should behave in particular situations.
  - what the system should not do.
- **Example:** Requirements of an autopilot might include the following:
  - the need to measure acceleration;
  - the need to compute relative positions.

Non-Functional and Domain Requirements

- **Non-functional requirements**
  - describe constraints on the services or functions offered by the system.

  These can be considerations like
  - ease of maintenance,
  - size,
  - cost.
- **Domain requirements** or **Context of Operation** describes in which situation the system is to be used, e.g.
  - temperature,
  - education standard of operators.

Safety Requirements Document

- **Safety requirements** are requirements which ensure an adequate safety of a system.
- For critical systems usually a separate **safety requirements document** sets out the safety requirements of the system.
- As general requirements, safety requirements can be grouped as:
  - **functional safety requirements**,
  - **non-functional safety requirements**,
  - **context of operation**.

Safety Requirements

- **Functional safety requirements** = services, to be carried out by the system in order to guarantee safety.
  - **Example:** if the heat in the reactor of a nuclear power plant exceeds a certain temperature, an automatic shut down mechanism has to be activated.
- **Non-functional safety requirements** = other requirements. E.g.
  - reliability,
  - availability.
  - See below in Subsection (c).

Safety Requirements (Cont.)

- **Context of operation** describe the context, under which a system is supposed to operate. E.g.
  - **Location of a unit**.
    - **Example:** requirement that the outside temperature should not exceed 50 °C.
      Then to place the unit into a desert might not be appropriate.
  - **Administrative procedures**.

Importance of Requirements

- Most accidents, in which software is involved, can be traced to **incomplete requirements**:
  - incomplete specification,
  - incomplete or wrong assumption about the operation of the controlled system,
  - unhandled states of the controlled system,
  - unhandled environmental conditions.

Importance of Requirements

- **Coding errors** have more an effect on reliability and other qualities rather than on safety.
  - Maybe even the Therac-25 fulfilled its specification.

$\Rightarrow$ Importance of the specification process for critical systems system.

Example of Incomplete Requirements

- Consider the following line of the safety requirements document concerning the control of pumps used to fill a water tank.
  - Shut off the pumps, if the water level remains above 100 metres for more than 4 seconds.

Example of Incomplete Requirements

- Four possible interpretations:
  - Shut off the pumps if the **mean water level** over the past four seconds was above 100 metres.
  - Shut off the pumps if the **median water level** (average of minimum and maximum water level) over the past four seconds has been above 100 metres.
  - Shut off the pumps if the **rms t(root mean square)** water level over the past four seconds was above 100 metres. (Rms is another version of average, which often is very important).
  - Shut off the pumps if the **minimum water level** over the past four seconds was above 100 metres.

Example of Incomplete Requirements

- The last statement (minimum $\geq$ 100 metre) is the most literal.
- However, this might cause problems, if there are large waves in the tank.
  - Pumps might not be turned off, even if water had reached a dangerously high level.
- General problem with natural language specification.

Validation vs. Verification

- **Verification** is the process of verifying that a **software product** meets its **specification**.
- **Validation** means two processes:
  - The process of confirming that the **specification** guarantees the customer **requirements**.
  - The process of confirming that the **software product** fulfils the customer **requirements**.
- Detecting validation errors during the first kind of validation is much cheaper.
- However, often validation errors are only detected after developing a prototype of the software product or even after developing the final product.
  - Finding validation problems in the final product can be very expensive.

Validation Problem

- A substantial part of the **specifications** can be written **formally**.
- By **testing** we can guaranteea very **high level of correctness**.
- and by **formal verification** (using e.g. SPARK Ada) we can even **guarantee correctness** for some aspects of the specification.

Validation Problem

- Requirements refer to **reality**.
  - They use notions such as “safe”, “ease of use”.
- Therefore requirements can in general **not be written formally**.
- Therefore validation **cannot be carried out formally**.
  - In Sect. 3 we looked at **semi formal** methods for carrying out validation for safety requirements.
  - Allows to systematically probe whether the requirements are fulfilled.
  - But **systematic probing doesn’t guarantee that validation**.

Validation Problem

- The **validation problem** is the **gap** between **specification** and **requirements**, where we cannot obtain the same level of **guarantee** as in verification.

Faults and Errors

- A **fault** is a defect within a system.
- An **error** is a deviation from the required operation of the system or subsystem.
- A **system failure** occurs when the system fails to perform its required function.

Faults and Errors

- A fault can be for instance
  - a permanent failure of a hardware component,
  - a transient fault (e.g. a radioactive particle hitting memory and changing some bits of it),
  - a coding error,
  - a mistake in the design of a system (software or hardware).
- A fault may or may not result in an error.

Faults and Errors

- A fault may even go undetected,
  - e.g. if there is a bug in a software routine which is in reality never executed,
  - e.g. if there is enough **fault tolerance** built into the system so that the faults are masked and don't result in an error (e.g. using error detecting codes in memory; fault tolerance will be discussed in Sect. 5).
- or may be dormant for some time and suddenly result in an error (e.g. when the faulty software routine is executed for the first time).

System Failures

- An error may or may not lead to a **system failure**, depending on whether the system has been built in order to function even in the presence of errors.
  - E.g. a system might **reconfigure** itself and remove a module causing errors from the system.
    - Note that this means that **we have an error**, since the **system** in this case **deviates from its required operation** – its **ability to tolerate errors** is after removing the system **inhibited**.
- For instance the system of a computer might reconfigure itself if it discovers memory not working, so that the damaged memory is no longer used.
  Afterwards the computer still functions well, but with reduced memory.

Hazard

- **Hazard** = situation in which there is actual or potential danger to people or to the environment.
- Just potential causes of accidents,
  independent of the risk associated with it.
  - E.g. one hazard is that if one walks outside during a thunderstorm, one might be struck by a lightning.

Accidents and Incidents

- **Accident** = unintended event or sequence of events that causes death, injury, environmental or material damage.
  - If there is an accident, there must have been a hazard (unless it caused only material damage).
  - But a hazard might not result necessarily in accidents.
- **Incident** or **near miss** = unintended event or sequence of events that does not result in loss but, under different circumstances, has the potential to do so.
  - Often, before an accident happens, several incidents occur.
  - Incidents are warning signals.

Risk

- **Risk** = combination of the frequency or probability of a specified hazardous event and its consequence.
- If consequence can be quantified:
  Risk = product of frequency of an event and its consequence.

Risk

- **Example:**
  - A unit in a chemical plant might explode on average every 10 000 years.
    $\Rightarrow$ Frequency of an event is 1/10000 failures per year.
  - Assume that an explosion will on average kill 100 people.
    $\Rightarrow$ Consequence = 100 deaths per accident.
  - Therefore the risk is 100 deaths x 1/10000 failures per year
    = 0.01 deaths per year.

Risk (Cont.)

- Often risks with **low probability** and **high consequence** are overlooked.
  - **Example Therac-25**:
    - Medical-controlled radiation therapy machine, which massively overdosed 6 people because of a software bug.
    - Probability of an overdose occurring was estimated very low.
    - However, since its consequences of an overdose are extremely high, the risk is high,
      and a rather high **safety integrity level** should have been assigned to the control mechanisms for radiation.

Risk (Cont.)

- No **hazard-free area** exists in life.
  - If one leaves one's house, one might be hit a car – should we therefore stay at home for ever?
- Therefore need to **quantify risks** in order to identify high risk hazards, for which particular care should be taken.
- However, a **danger** is to use risk calculations in order to **justify not to take any safety measurements**.
- Often the numbers used in risk calculations are chosen in such a way that the **risk appears to be much lower than it is**.

Acceptable Risk

- I have seen one reference (source lost), which was referring to
  - US Department of Transportation, Federal Aviation Administration: System Design and Analysis Standard AC25-1309-1A, 1988.

  Acceptable risk =
  "1 life left during 1 billion hours of operation".
- 1 billion hours = approximately 100000 years (more precisely 114079 years).

(c) Dimensions of Dependability

- **Dependability** = property of a system that justifies placing one's reliance on it.
- **Dependable system** = system that has a high degree of dependability.
  So a critical system should be highly dependable.
- Dependability is a global property which can be decomposed into several **dimensions.**

Dimensions of Dependability

- To say that the computer system is dependable is too unspecific in order to be useful in requirements documents.
- Therefore we will in the following split this notion up into several dimensions.
  - Some of these dimension give rise to ways of measuring them, called **metrics**.
    - An example will be the measurement of availability and reliability below.
    - If we have a metric, we can verify this metric by testing, so one can check, if the system has met the specification.
- Others are of more qualitative nature.

Dimensions of Dependability (Cont.)

(A) The **four main dimensions of dependability** are the following:
- (i) Reliability
- (ii) Availability
- (iii) Safety.
- (iv) Security.

(B) Related dimensions to the four main ones are the following:
- (v) Maintainability
- (vi) System integrity.
- (vii) System recovery.
- (viii) Failsafe operation.
- (ix) Data integrity.

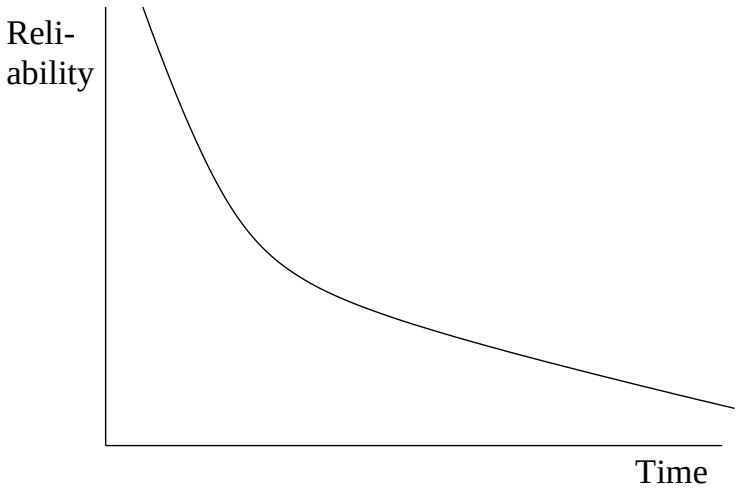
(A) 4 Main Dimensions

(i) Reliability

- **Reliability =** probability of a component or system to function correctly **continuously over a given period of time** under a given set of operation conditions.
- Measured by the probability of the system to operate continuously (**no interruptions**) over that given period.
  - Example would be: the probability of functioning continuously over 7 days is 0.99.
- Functioning correctly means;
  - **Operation as defined in the specification**.
  - **No maintenance** carried out.

Reliability

- Reliability necessarily decreases over length of time for which reliability is required.
- For satellites, high reliability over **extremely long time** required.
- For some military applications, reliability only limited to **duration of a particular mission**, could be **hours or even minutes**.
  - E.g. an anti-aircraft missile has to be reliable (as well with respect to safety features) from when it is fired until it reaches its target.



Example of Calculation of Reliability

- Assume that of 10000 computer chips leaving a factory having thorough testing, after 6 months 9998 are still working correctly.
  $\Rightarrow$ Reliability of this chip after 6 months is 9998/10000
- Related to reliability is **unreliability**, which is 1− reliability.
  - Easier to write down.
  - Unreliability of the above mentioned computer chip after 6 months is $2/10000 = 0.02\%$.

Reliability Requirement for Critical Systems

- Reliability is particular important for safety-related systems, if **continuous uninterrupted operation** is essential for maintaining safety:
  - A **flight critical aircraft system** has to work with high reliability throughout a flight.

Three Forms of Reliability

- **Hardware Reliability**
  - Probably of a hardware component to fail.
- **Software Reliability**
  - Probably of a software component to produce a correct output.
  - Different from hardware reliability, since
    - Software does not wear out.
    - After an error and recovery it can continue operating without being affected by the error.
- **Operator reliability.**
  - Probability for the operator to make an error.

(ii) Availability

- **Availability** = Probability that a system will be functioning correctly at any given time.
  - Measured by the proportion of time the device is functioning correctly.
  - Allows that the operation of the system might be interrupted.
- If a system is within 1000 hours of operation out of operation for 1 hour, then its average availability during the period is 999/1000.
- **Unavailability** = 1 - availability.
  - The unavailability for the above mentioned system is $1 - 999/1000 = 0.1\%$.

Availability and Critical Systems

- **High availability** important for certain **business critical systems**:
  - E.g. **computer systems in banks**,
  - **telephone exchanges**.
    - Telephone exchanges have nowadays requirements of a downtime of a few hours throughout their lifetime of several decades:
      5 hours unavailability within 2 decades gives an unavailability of $\frac{5h}{(24*365*20)h} = 2.85 \cdot 10^{-5}$.
    - Usually, if they are unavailable, they are unavailable for a few seconds, or less than a second, and the customers usually don't even notice this.

Availability and Critical Systems

- High availability can be important for systems which are **activated in emergency cases only**:
  - Fire alarm.
  - Nuclear reactor shutdown system.

Example Availability vs. Reliability

- Assume **system A fails once per year** and **system B fails once per month**.
  $\Rightarrow$ **System A is more reliable than system B.**
- Assume additionally that it takes **3 days to restart system A** and **10 minutes to restart system B**.
  $\Rightarrow$ Unavailability of system A is 3 days/year, unavailability of system B is 120 minutes/year.
  $\Rightarrow$ Assuming that we are measuring availability over the period of at least one year, and that it will usually take about a year for the problem to occur, **System B is more available than system A.**

Dependency on Time Intervals

- If the period of time measured is substantially less than a year, say 1/2 year or only a month, and if with high probability at least 1 year for the problem to occur, one obtains however that System A is more available, since it will work with high probability continuously over this period.

(iii) Safety

- **Safety** = property of a system, that it will not endanger human life or the environment.
- **Safety-related system** = system by which the safety of equipment or a plant is guaranteed.
- The above dimensions contribute to this dimension, but there are aspects of safety which are not guaranteed by high availability and reliability.
  - For instance, if people are killed when being to close to a rocket when it is started, since there are no protective measurements, then the rocket is not safe, even if it is highly reliable and available.

(iv) Security

- **Security** = property of a system to protect itself from external attacks that may be accidental or deliberate.
  - So whereas **safety** is safety from threats from the **inside**, **security** is safety from threats from the **outside**.
- **Examples of attacks:**
  - viruses,
  - unauthorised use of system services,
  - unauthorised access to system data,
  - unauthorised modification of system data.

Security (Cont.)

- Lack of security may **compromise safety, availability and reliability of a system**.
  - For instance, if a terrorist is able to get access to the computer system controlling a nuclear power station, this will endanger the safety of that plant.
- Security is especially important for
  - **military applications,**
  - **financial software,**
  - systems containing **important personal data** (e.g. university database of student marks, medical data in a hospital).

Damages Caused by External Attacks

- **Denial of service.**
  System forced into a state where its normal services become unavailable.
- **Corruption of programs or data.**
  Affect system's behaviour.
- **Disclosure of confidential information.** Might affect secret data which is not to be exposed, or system related data, which allows future attacks.

Terminology

- **Exposure** = possible loss or harm in a computing system.
- **Vulnerability**= Weakness of a computer-based system that may be exploited to cause loss or harm.
- **Attack** = exploitation of a system vulnerability.
- **Threat** = circumstance that have potential to cause loss or harm.
- **Control**= protective measure that reduces system vulnerability.
- **Survivability**= ability of a system to continue to deliver service while it is under attack.

Relation to Safety Notions

- Notions from security correspond approximately to notions in safety as follows (one can argue for different correspondences):

Security	Safety
exposure	risk
vulnerability	hazard
attack	incident/accident
threat	fault

(B) Related Dimensions

(v) Maintainability

- **Maintenance** = action taken to retain a system in or return it to its designed operating condition.
- **Maintainability** = ability of a system to be maintained.
- Major problems: **maintenance-induced failures.**
- Maintainability can be expressed in a
  - qualitative
    - e.g. by expressing the skills needed by the workers carrying out maintenance,
  - or quantitative manner (MTTR, see next slide).

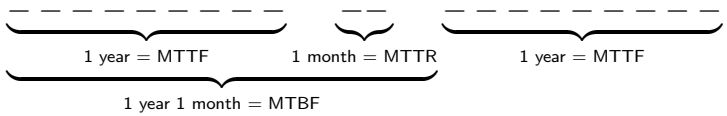
Maintainability vs. Availability

The **metrics** relating maintainability and availability are as follows:

- **Mean time to repair** (**MTTR**) = average time it takes to repair a system.
  - **Example:** if it takes on average 2 h to repair a car then its MTTR is 2h.
- **Mean time to failure** (**MTTF**) = average time it takes for a failure to appear.
  - If a turbine fails on average after 3 years its MTTF is 3 years.
- **Mean time between failures** (**MTBF**) is MTTR + MTTF.
  - If for the turbine it takes 1 month to repair it then its MTBF is 3 years + 1 month.

MTBF

- So we have



Maintainability vs. Availability

$$\begin{aligned}
\text{Availability} &= \frac{\text{time system is operational}}{\text{total time}} \\
&= \frac{\text{MTTF}}{\text{MTBF}} \\
&= \frac{\text{MTTF}}{\text{MTTF+MTTR}} \\
\text{Unavailability} &= 1 - \text{Availability} \\
&= \frac{\text{time system is nonoperational}}{\text{total time}} \\
&= \frac{\text{MTTR}}{\text{MTBF}} \\
&= \frac{\text{MTTR}}{\text{MTTF+MTTR}}
\end{aligned}$$

Maintainability vs. Availability

So for our turbine example (every 3 years needs to be repaired for 1 month) we have:

$$\begin{aligned}
\text{availability} &= \frac{3 \text{ years}}{3 \text{ years} + 1 \text{ month}} \\
&= \frac{36 \text{ months}}{37 \text{ months}} \\
&= 97.3\%. \\
\text{Unavailability} &= \frac{1 \text{ month}}{3 \text{ years} + 1 \text{ month}} \\
&= \frac{1 \text{ month}}{37 \text{ months}} \\
&= 2.7\%.
\end{aligned}$$

Simplification

- If the MTTR is much smaller than MTTF, one can simplify unavailability, omit MTTR in the denominator of the formula expressing unavailability, and obtain

$$\text{Unavailability} \approx \frac{\text{MTTR}}{\text{MTTF}} \ .$$

- Note that, if we omit in the expression for availability MTTR in the denominator, we obtain 1, so we can't simplify the expression for availability in the same way.

Achieving Maintainability

- Maintenance of critical systems differs.
  - There are systems, which can be **temporarily shut down completely** for maintenance completely.
  - Other systems have to run continuously, and need to be **maintained while in service**.
    - Requires some form of **redundancy** in the system, so that some systems can take over the tasks of other modules, which are about to maintained.
    - Results in a system with **higher complexity**.
    - Further, the **safety level** of the system is **reduced** during maintenance.

Preventive Maintenance

- **Preventive maintenance** tries to keep a system in good order and remove the effects of wear and ageing before they result in a system failure.
  - Can be performed at a time determined by a **predetermined schedule**,
  - or might be performed as a **result of monitoring** the condition of the system.
  - It can be in both cases performed at a convenient time.

Corrective Maintenance

- **Corrective maintenance** aims to restore the system to its designed state following some form of failure.
  - This has to be usually carried out **as soon as a failure is detected**.
  - If there is **fault tolerance**, it might be possible to **wait with corrective maintenance**, until it is more convenient to carry it out.
    - Only possible, if not too many faults occur simultaneously.

Maintainability

- Modern aircraft are designed in such a way that **repair of failed modules** can in most cases be **delayed until** the aircraft returns to its **home base** while still maintaining acceptable levels of safety.
- In critical systems, **maintenance-induced failures** are a common problem.
  - It might be that maintenance doesn't fully correct the problem,
  - or that it introduces new problems.

Maintainability

- Attempts to improve maintainability of a system might reduce its safety.
  - Example: **Built-in test equipment (BITE)**.
    - Speeds maintenance by simplifying the task of locating faults.
    - Requires additional hardware/software, which increases the complexity of the system, and reduces reliability and therefore often safety.

(vi) System Integrity

- **Integrity of a system** = ability of a system to detect faults in its own operation and to inform a human operator.
- **Examples of systems** with **high integrity requirements**:
  - Railway signalling system
  - Aircraft autopilot
- Integrity can be **measured** by the probability a system operates without security penetration for a specific time given a specific thread profile and a specific rate of arrival of threads.

System Integrity

- The notion of "**system integrity**" has **not to be mixed up** with "integrity" in "**high-integrity systems**".
  - "High integrity systems" stands for "**dependable systems**".
  - Shift of meaning of the word integrity has taken place.
  - Maybe one needs a find a new word for "system integrity".

(vii) System Recovery

- **System recovery** is the ability of a system to restart itself quickly after a failure was detected.
  - Important, since it **cannot be guaranteed that a system will not fail** (e.g. transient faults like noise spikes in the power supply).
  - Required e.g. for satellites.
  - Required as well for safety-critical systems **with no failsafe state.** (see next slide).
- System recovery or **recoverability** can be measured by the average time to recover from failure, including data cleanup or reinitialisation. This is the software analogue of **MTTR**.

(viii) Failsafe Operation

- If a system has a **failsafe state**, a lower degree of reliability might be acceptable.
- **Failsafe state** = output state of a critical system, which is absolutely safe, and the safety of which is maintained by minimal requirements of the system.
- **Examples:**
  - For a railway signalling system, a failsafe state is obtained if **all lights are red**.
  - Failsafe state of **fire doors** is obtained if **they are closed** (assuming that in an emergency they can can be opened by hand).

Failsafe State (Cont.)

- If there is a failsafe state, a system should reach this, if something not covered by the routines of the system happens.
  - **Example:** If a **train enters a blocked section**, or the power goes down, all signals should turn red.
  - **Example:** If the **electric power in the department goes down**, because of its design, the **fire doors will close**.
    - The fire doors are kept opened by electromagnets. If the power goes down they close automatically.
- **Some systems don't have a failsafe state**.
  Example: An aircraft has no failsafe state while it is flying.
  - Reason, why some people are afraid of flying?

(ix) Data Integrity

- **Data integrity** = ability of a system to prevent damage to its own database and to detect and possibly correct errors that occur.
- Important esp. in certain
  non safety-critical dependable systems.
  - Computer systems in banks.
  - University data base of student marks.

(d) Identification of Safety Requirements

- Main stages for determining the safety related system requirements are:
  - **Identification of hazards** associated with a system (see Section 4).
  - **Classification** of these hazards.
  - Determination of **methods for risk reduction**.
  - Assignment of appropriate **reliability and availability requirements.**
  - Determination of an **appropriate safety integrity level.**
  - Specification of **development methods** appropriate to this safety integrity level.

Classification of Hazards

Hazards can be classified w.r.t.

- **Severity**, i.e. consequences of the hazard.
- **Risk** of the hazard.
- **Nature**:
  - **Can be controlled** by a system,
    - **Example:** A laser can be switched on and off by the system.
  - **Cannot be controlled** by a system
    - **Example:** Behaviour of pedestrians cannot be controlled by a road traffic control system.
  - **Can be controlled with some delay**.
    - **Example:** high voltage source might still be charged for some time after voltage is switched off.

Risk Reduction

- Some methods of risk reduction are
  - **To "design out" the hazard.**
    - Design the system, so that hazards are avoided principally.
    - **Example:** Choose a different chemical process in a chemical plant.
    - Usually most effective and as well most cost effective.
  - **Safety devices**, which control the equipment.
    - E.g. add a pressure relief valve.

Risk Reduction (Cont.)

- **Interlocks** = mechanism which ensure that hazardous actions are only performed at times when they are safe.
  - Might control equipment directly.
  - Might control **guards** = mechanisms which keep people away from dangerous parts of a system until it is safe.

Risk Reduction (Cont.)

- Interlocks
  - use **sensors** in order to determine the state of the system,
  - and activate **actuators** in order to control the equipment and guards.
- **Warning signs and signals.**
- **Management techniques** (e.g. appointment of safety officers, introduction of safety standards – good practises – and regulations).

(e) The Safety Case

- **Safety case** = document that sets out the safety justification of a system.
  - Describes design and assessment techniques used in the development of the system.
  - Provides evidence that risks associated with the system have been carefully considered, and steps have been taken to deal with them appropriately.
  - Safety case sometimes called **safety argument**, **safety justification** or **safety assessment report**.

The Role of Standards

- Documents which codify requirements in order to achieve **uniformly high levels of quality and safety**.
- Issued by different groups.
  - International organisations.
  - Governmental organisations.
  - Industrial groups.
  - Engineering societies (e.g. IEEE).
- Additionally there are **guide lines** and **codes of practice**.
- Standards might be
  - an **absolute requirement**,
  - **negotiable** (require alternative ways of achieving safety).

The Role of Standards

- Standards are important since
  - in case certification is needed it is usually vital that one has followed certain standards,
  - as a legal protection in case of an accident.
    - Even if an accident is due to a bug in the software, liability might be reduced if one has followed high
    - It can only demanded that a product is developed up to the current state of technology as set out by standards – for instance to demand full verification is considered in most cases currently not feasible.
    - That might change in the future.