5 (a) Life Cycle Models

Model from IEC 1508

- This is a model from the International Electrotechnical Commission (IEC).
- The IEC 1508 standard is intended to be a generic basis for standards in all industrial sectors.
- It has three phases:
  - Design,
  - planning/realisation
  - installation/operation
- Standard describes in detail:
  - the activities to be performed during each phase of the life cycle,
  - the expected inputs and outputs of each phase.
Other Life Cycle Models

- Probably more modern iterative models like the spiral model are used nowadays in safety critical systems as well.
  - They need some adaption since they might be as they stand too loose about requirements analysis and specification.
The safety life cycle is the process of safety specification and assurance.

The next slide describes the safety life cycle according to IEC 1508.

- It is very similar to the life cycle model of the IEC 1508.
- Differences are marked in red italic.
Main elements of the development process of safety critical systems are similar to that of less critical units.

However, the need to produce and demonstrate dependability requires that each phase is carefully structured and documented.

Was already discussed.

Problem of validation of specifications.

Cannot be done formally.

Two approaches of tackling that problem are software animation and prototyping.

Software prototyping means that one builds a prototype based on the specification in order to test it.

Problem is that the prototype is removed from the specification.

The prototype is only one example of implementing the specification.

Further, when building the prototype, one might not interpret the specification correctly. Therefore the prototype might not fully fulfil the specification.
**Animation and Prototyping**

- **Software animation** means that one directly executes or tests the specification.
  - For instance, one can check what happens if one specialises the specification to some instance, and if one executes an operation specified.
  - **Advantage** is that one works directly with the specification.
  - The **disadvantage** is that software animation might not be as concrete as prototyping.
    - One usually doesn’t obtain a real system.

**Animation vs. Simulation**

- Animation/prototyping are different from **simulation**.
  - **Simulation** is the development of trial software based on the design of the software, in order to test it.
  - Animation is a test of the specification.
  - Simulation is a test of the design.

---

**Top-Level Design**

- In top-level design, the following steps are carried out:
  - **Partitioning** of system functions into those carried out by hardware and those carried out by software.
  - **Decomposition** of hardware and software architecture into manageable modules.
    - Determination of interfaces between modules.
    - Specification of functions and safety features of each module.
  - **Identification of major data structures** within the software.

**Detailed Design**

- The detailed design precedes the actual implementation of the software.
  - First one gathers information about the module and makes initial decisions about it form.
    - One concentrates in this period on what the module has to do rather than how this is achieved.
Detailed Design

- The data gathered is as follows:
  - **Purpose** of the module, including primary and secondary requirements.
    - Includes behaviour in exceptional situations.
  - **Data use.**
    - Input.
    - Output.
    - External data shared with other modules.
    - Internal data shared by the functions of the module.
  - **Performance**
    - Time and space constraints of the module.
    - Often overlooked.

- **Fault conditions**
  - Failures which might occur within the module and its interfaces.
  - Conditions under which a failure mode can occur have to be identified.

- **Integrity level**
  - Which integrity level is to be assigned to that module.

- **Testing.**
  - Identification of, what conditions have to be verified by testing, and what level of correctness is to be achieved.

Once the above data has been collected, the units are decomposed into smaller and smaller modules, until modules which can be implemented directly have been identified.

Once the smallest units have been identified, they will be implemented.

Special techniques used in implementing, e.g. **defensive programming**.

- Defensive programming means that one adds extra tests, even if one assumes that they are not necessary, in order to catch programming errors as early as possible.
  - E.g. even if one assumes that an input parameter is is always \( \geq 0 \), one still deals with the case that it might be \(< 0\).

Then modules will be tested, see Section 8.
System Integration

- System integration means to assemble the complete system from its component modules.
- Then initial testing is performed, before progressing to full system testing and validation.
- There are two main techniques for system integration:
  - Progressive integration.
  - The big bang method.

Progressive Integration

- **In progressive integration**
  - One starts with combining a small number of modules, which are then tested, and problems are removed.
  - Then additional modules are added successively, which are tested, and problems are removed.
  - This is done until the complete system is assembled.
- **Advantage:**
  - Problems observed in testing only involve the few modules added in the last incrementation step.
  - Therefore the reasons for those problems can be more easily identified.

- **Disadvantage:**
  - Problems, associated with the overall system are not visible, until the complete system is assembled.
  - Then it is very expensive to deal with these problems.

Big Bang Method

- **In the big bang method**, all modules are combined immediately, and the complete system is tested.
  - Assumption is that the design of the overall system was done carefully and that each unit was tested thoroughly.
  - Then integration shouldn’t cause big problems.
- Advantages and disadvantages of the big bang method are the opposite of those for progressive integration.
- Big bang approach less common than progressive integration.
System Test and Certification

- System testing will be treated later.
- In the certification, the development process of the system is carefully investigated
  - Usually based on some form of standard.
  - The developer has to convince the regulators that all relevant hazards have been identified and dealt with.
  - The certification process **does not**
    - prove the correctness of the system,
    - remove any of the developer’s legal or moral obligations.
- However the certification process promotes the use of approved techniques and will usually lead to safer products.

(d) Designing for Safety

- We consider the design process with respect to safety of the overall system.
- There are four activities in the design process:
  - **Abstraction**
    - The operation of generalising and identifying the essentials.
  - **Decomposition**
    - The process of reducing an object into a number of simpler, smaller parts.
    - The analysis of interactions, interfaces and structures.
    - Modularisation.
  - **Elaboration**
    - The operation of detailing, adding features.
  - **Decision making**
    - Identification and selection of alternative strategies.
Programming languages might or might not support isolation of modules:
- **Assembler** and **C** don't support isolation of modules— all routines are allowed to write to all memory locations.
- More **modern languages** like Pascal, Ada or Java support isolation of modules in a better way.

Isolation of modules requires as well that **no module can block** other modules by getting hung up.
- By getting hung up one module with low criticality
  - e.g. a module responsible for communications in a space craft, could block another highly critical module, which operates in parallel, but is not related with the other module.

Problems with parallelism can be controlled by **operating systems**, which allocate time to different **threads** and therefore make sure that the system doesn't get hung up with one thread.
- **Problematic** to use operating systems, since they are usually too complex in order to be verifiable.
- Instead one can use **runtime kernels** with a **task scheduler**, which are much smaller than usual operating systems, and have therefore higher level of integrity.
  - Acceptable for **systems of lower criticality**.
Support of Software Isolation

For highest critical functions, use of runtime kernels is still not acceptable, because they are too complex.

- Instead one has to program the critical functions including scheduling completely by hand.

No Additional Material

For this subsection no additional material has been added yet.
(f) Safety Analysis

- Safety analysis is the process of assessing the safety of a system by looking at the associated hazards and the methods used by the system to cope with them.
  - Also called overall safety validation.
- The next slide gives the steps in safety analysis according to the Health and Safety Executive (HSE) in UK.

Steps in Safety Analysis (HSE)

1. Analyse the hazards.
   a. identify the potential hazards,
   b. evaluate the events leading to these hazards.
2. Identify the safety-related systems within the plant.
3. Design the safety-related system using the safety integrity criteria appropriate for the specific application.
4. Carry out a safety integrity analysis to assess the level of safety integrity achieved by the safety-related systems.
5. Ensure, from the analysis of (4), that the integrity levels of (3) have been achieved.

(g) Safety Management

- Safety management is the planning, organisation, monitoring and evaluation of safety aspects of a project.
In order to achieve a high degree of safety it is important that
- a safety culture is encouraged from the top;
- a well-defined safety policy exists in order to establish the working practices of the organisation and to ensure that these are followed;
- safety performance is monitored
  - both in-house and by using independent assessments of safety systems and procedures.

The next slide shows the management structure recommended by Interim Defence Standard 00-55 (MoD, UK, 1991).

Project Manager
- bears ultimate responsibility for development of the system and its safety.
- The project manager delegates much of the task of developing the system to the project management team.
- This team
  - is the design authority;
  - will at the end sign off the system on behalf of the project manager;
  - is primarily responsible for administrating the safety aspects of the project.

As part of dealing with the safety aspects, the project management team
- will perform hazard and risk analysis;
- will ensure that components are developed according to an appropriate integrity level;
- will perform verification at each phase of the design;
- this includes
  - preparation of a safety plan;
  - maintenance of the safety log (containing the safety activities and results of hazard analyses).
Safety Management Structure

- The **design team** carries out the actual specification, design and implementation of the system.
- The **independent safety assessor**
  - reviews and audits all activities and documents relevant to safety.
  - provides an independent check on the activities of the project management team.
- The **independent verification and validation team (IV&V team)**
  - looks at the design at each stage;
  - provides separate check on its correctness.
  - Often, the IV&V team is from an external company, specialised in that work.

### Essentials of Safety Management

- **Safety management** is the
  - planning,
  - organisation,
  - monitoring and
  - evaluation
  of safety aspects of a project.

- In order to achieve a high degree of safety it is important that
  - a **safety culture** is encouraged from the top;
  - a well-defined **safety policy** exists in order to establish the working practices of the organisation and to ensure that these are followed;
  - **safety performance** is monitored
    - both in-house and by using **independent assessments** of safety systems and procedures.
- The next slide shows the management structure recommended by Interim Defence Standard 00-55 (MoD, UK, 1991).
(h) Safety Management Structure (Interim Defence Standard 00-55)

Safety Management Structure

- **Project Manager** bears ultimate responsibility for development of the system and its safety.
- The project manager delegates much of the task of developing the system to the **project management team**.
- This team
  - is the **design authority**;
  - will at the end **sign off** the system on behalf of the project manager;
  - is primarily responsible for administrating the **safety aspects** of the project.

As part of dealing with the **safety aspects**, the project management team
- will perform **hazard** and **risk analysis**;
- will ensure that components are developed according to an appropriate **integrity level**;
- will perform **verification** at each phase of the design;
- this includes
  - preparation of a **safety plan**;
  - maintenance of the **safety log** (containing the safety activities and results of hazard analyses).

The **design team** carries out the actual specification, design and implementation of the system.
- The **independent safety assessor**
  - reviews and audits all activities and documents relevant to safety.
  - provides an independent check on the activities of the project management team.
- The **independent verification and validation team (IV&V team)**
  - looks at the design at each stage;
  - provides separate check on its correctness.
  - Often, the IV&V team is from an external company, specialised in that work.