Safety Systems

SCSD Lecture 14 Mar 2024
IEC 61508

- 61508 is the basic standard for the safety of software-controlled equipment.
- Over the next two lectures we will consider this in some detail because it introduces the basic ideas that have been incorporated into many standards for software products.
- This will allow us to consider some specific techniques and how they are deployed.
- 61508 Considers the risks arising from the operation of the Equipment under control (EUC)
- It is comprehensive but gives particular consideration to the realization of software.
APPLICATION OF IEC 61508

• There is equipment under control (EUC) which, with its control system, poses risks to its surroundings
• The risks will be reduced to tolerable levels by safety functions
• Safety functions are performed by E/E/PE systems

Ack: Felix Redmill
Overall Structure of 61508 series

- We are mainly concerned with software.
- However, software can only be responsible for harm via the EUC.
- Here we concentrate on Part 3: realization phase for safety-related software.
IEC 61509 Lifecycle

1. Concept
2. Overall scope definition
3. Hazard and risk analysis
4. Overall safety requirements
5. Overall safety requirements allocation
IEC 61509 Lifecycle

```
6 Overall operation and maintenance planning
7 Overall safety validation planning
8 Overall installation and commissioning planning

9 E/E/PE system safety requirements specification

10 E/E/PE safety-related systems
   Realisation
   (see E/E/PE system safety lifecycle)

11 Other risk reduction measures
   Specification and Realisation

12 Overall installation and commissioning

13 Overall safety validation

14 Overall operation, maintenance and repair

15 Overall modification and retrofit

16 Decommissioning or disposal

Back to appropriate overall safety lifecycle phase
```
Hardware Realisation

Figure 3 – E/E/PE system safety lifecycle (in realisation phase)
Software Realisation

Figure 4 – Software safety lifecycle (in realisation phase)
Figure 5 – Relationship and scope for IEC 61508-2 and IEC 61508-3
IEC 61508 Software Development Lifecycle

![Diagram of IEC 61508 software development lifecycle]

Figure 6 – Software systematic capability and the development lifecycle (the V-model)
# Software Lifecycle Detailed Overview

## Table 1 - Software safety lifecycle – overview

<table>
<thead>
<tr>
<th>Safety lifecycle phase</th>
<th>Objectives</th>
<th>Scope</th>
<th>Requirements subclause</th>
<th>Inputs (information required)</th>
<th>Outputs (information produced)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Software safety requirements specification</td>
<td>To specify the requirements for safety-related software in terms of the requirements for software safety functions and the requirements for software systematic capability; To specify the requirements for the software safety functions for each E/E/PE safety-related system necessary to implement the required safety functions; To specify the requirements for software systematic capability for each E/E/PE safety-related system necessary to achieve the safety integrity level specified for each safety function allocated to that E/E/PE safety-related system</td>
<td>PE system; software system</td>
<td>7.2.2</td>
<td>E/E/PE safety requirements specification as developed during allocation (see IEC 61508-1)</td>
<td>E/E/PE system safety requirements specification (from IEC 61508-2)</td>
</tr>
<tr>
<td>10.2 Validation plan for software aspects of system safety</td>
<td>To develop a plan for validating the software aspects of system safety</td>
<td>PE system; software system</td>
<td>7.3.2</td>
<td>software safety requirements specification</td>
<td>validation plan for software aspects of system safety</td>
</tr>
</tbody>
</table>
### Software Lifecycle

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3</td>
<td>Software design and development</td>
</tr>
</tbody>
</table>

**10.3** Software design and development

| Architecture: |
| To create a software architecture that fulfills the specified requirements for safety-related software with respect to the required safety integrity level; |
| To evaluate the requirements placed on the software by the hardware architecture of the E/E/PE safety-related system, including the significance of E/E/PE hardware/software interactions for safety of the equipment under control |

| PE system; software system |

| 7.4.3 | software safety requirements specification; E/E/PE system hardware architecture design (from IEC 61508-2) |

| 7.4.4 | software safety requirements specification; software architecture design |

| Support tools and programming languages: |
| To select a suitable set of tools, including languages and compilers, run-time system interfaces, user interfaces, and data formats and representations for the required safety integrity level, over the whole safety lifecycle of the software which assists verification, validation, assessment and modification |

| PE system; software system; support tools; programming language |

| 7.4.4 | software safety requirements specification; software architecture design |

| Support tools and coding standards; selection of development tools |
Risk guides the level of rigour in development

Figure A.1 – Risk reduction – general concepts (low demand mode of operation)
Risk reduced to tolerable levels
ALARP: One approach to achieving tolerable risk

Intolerable region

The ALARP or tolerability region
(Risk is undertaken only if a benefit is desired)

Tolerable only if further risk reduction is impracticable or if its cost is grossly disproportionate to the improvement gained

As the risk is reduced, the less, proportionately, it is necessary to spend to reduce it further to satisfy ALARP. The concept of diminishing proportion is shown by the triangle

Broadly acceptable region
(No need for detailed working to demonstrate ALARP)

It is necessary to maintain assurance that risk remains at this level

Risk cannot be justified except in extraordinary circumstances

Negligible risk

Figure C.1 – Tolerable risk and ALARP
## Risk Classes

Table C.1 – Example of risk classification of accidents

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Catastrophic</td>
</tr>
<tr>
<td>Frequent</td>
<td>I</td>
</tr>
<tr>
<td>Probable</td>
<td>I</td>
</tr>
<tr>
<td>Occasional</td>
<td>I</td>
</tr>
<tr>
<td>Remote</td>
<td>II</td>
</tr>
<tr>
<td>Improbable</td>
<td>III</td>
</tr>
<tr>
<td>Incredible</td>
<td>IV</td>
</tr>
</tbody>
</table>

**NOTE 1** The actual population with risk classes I, II, III and IV will be sector dependent and will also depend upon what the actual frequencies are for frequent, probable, etc. Therefore, this table should be seen as an example of how such a table could be populated, rather than as a specification for future use.

**NOTE 2** Determination of the safety integrity level from the frequencies in this table is outlined in Annex D.
## Risk Classes

<table>
<thead>
<tr>
<th>Risk class</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I</td>
<td>Intolerable risk</td>
</tr>
<tr>
<td>Class II</td>
<td>Undesirable risk, and tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained</td>
</tr>
<tr>
<td>Class III</td>
<td>Tolerable risk if the cost of risk reduction would exceed the improvement gained</td>
</tr>
<tr>
<td>Class IV</td>
<td>Negligible risk</td>
</tr>
</tbody>
</table>
Summary

- Introduced IEC 61508
- The overall lifecycle
- The software lifecycle
- Risk
- How risk informs software development
- Next lecture we will consider how this influences development practice