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 responsibe for harm via the EUC.
- Here we concentrate on Part 3: realization phase for safety-related software.

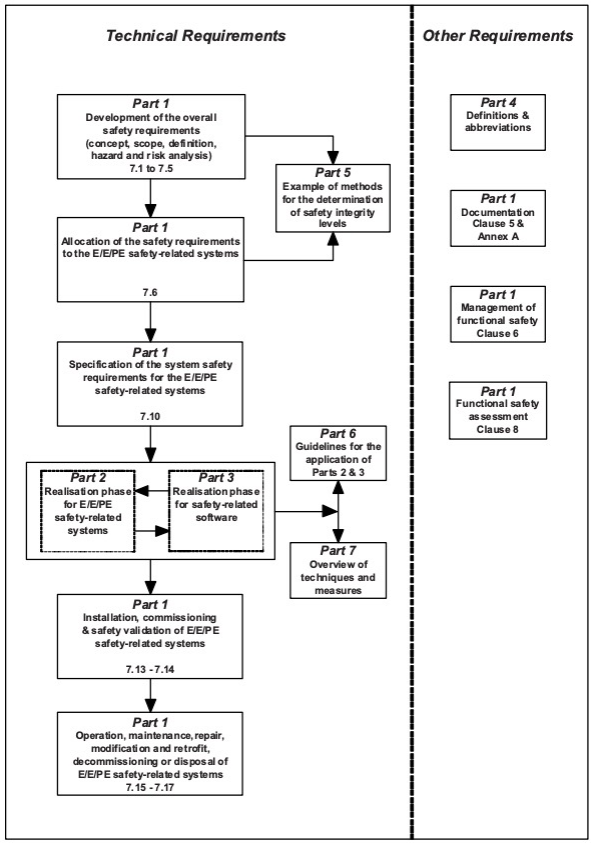
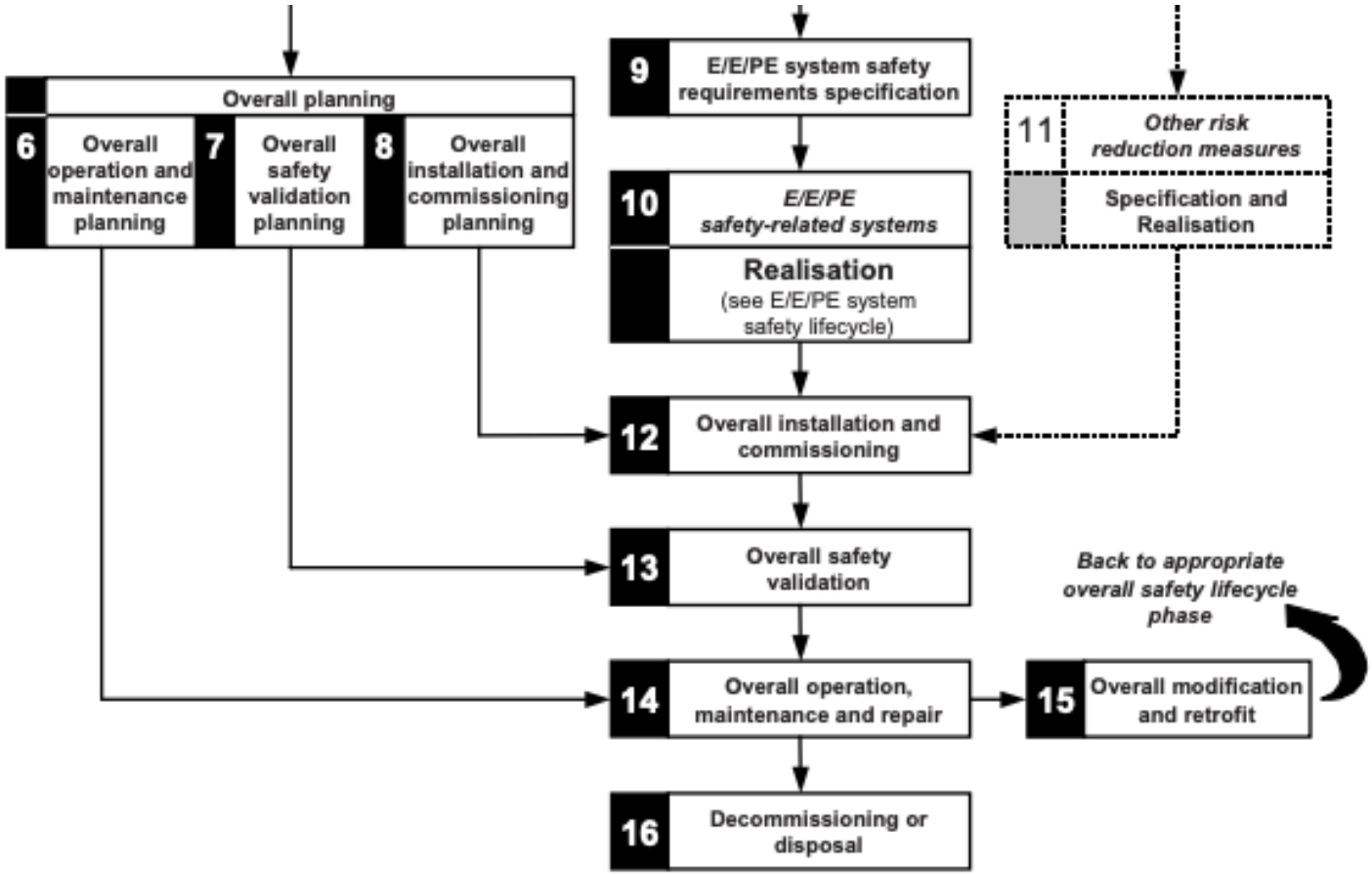


Figure 1 – Overall framework of the IEC 61508 series

IEC 61509 Lifecycle

IEC 61509 Lifecycle



Hardware Realisation

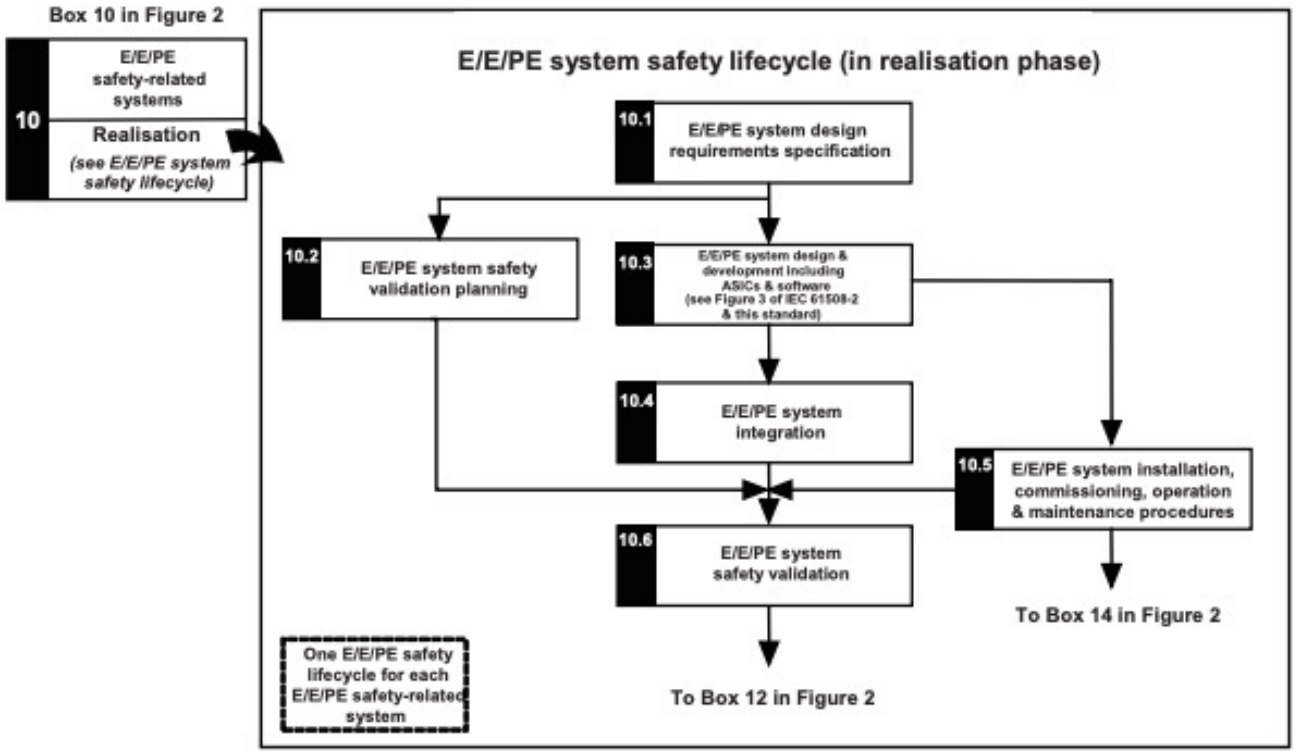


Figure 3 – E/E/PE system safety lifecycle (in realisation phase)

Software Realisation

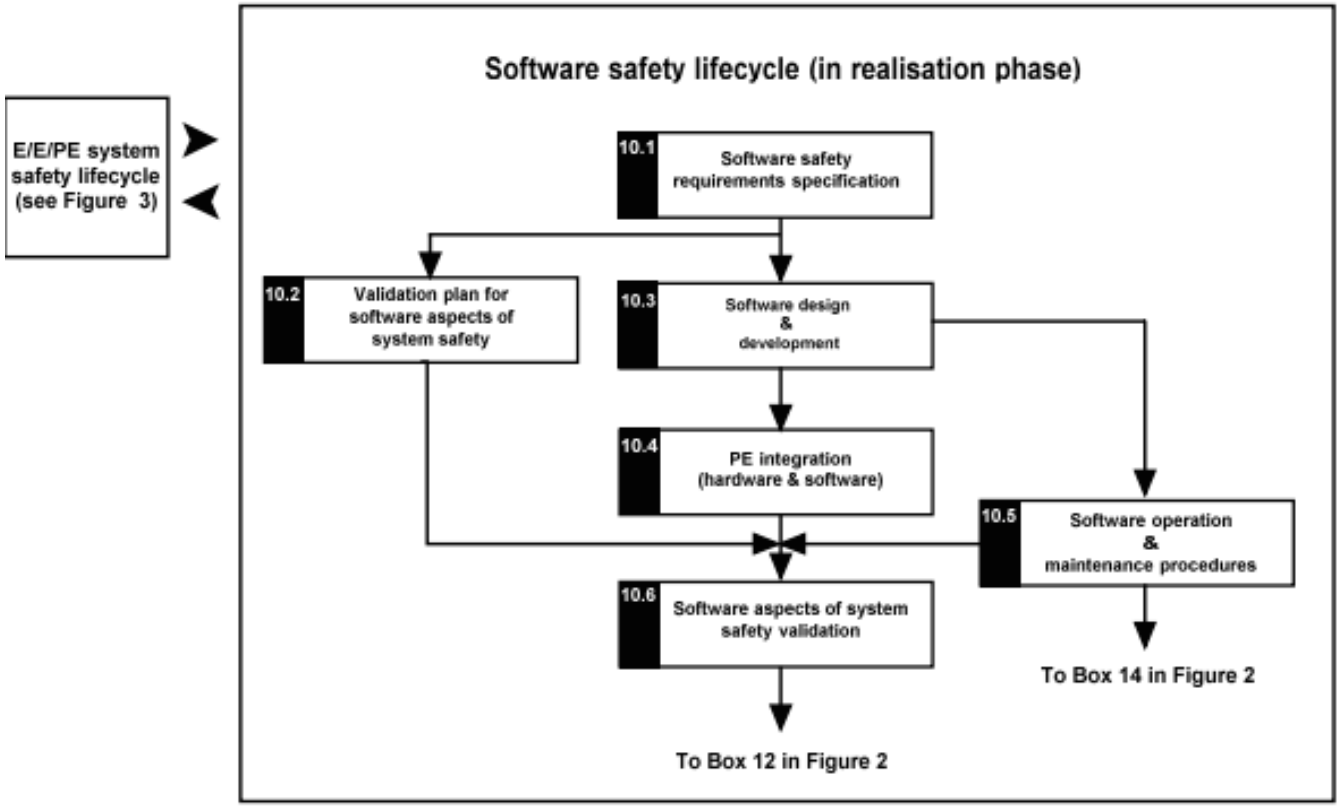


Figure 4 – Software safety lifecycle (in realisation phase)

IEC 61508 Hardware/Software relationship

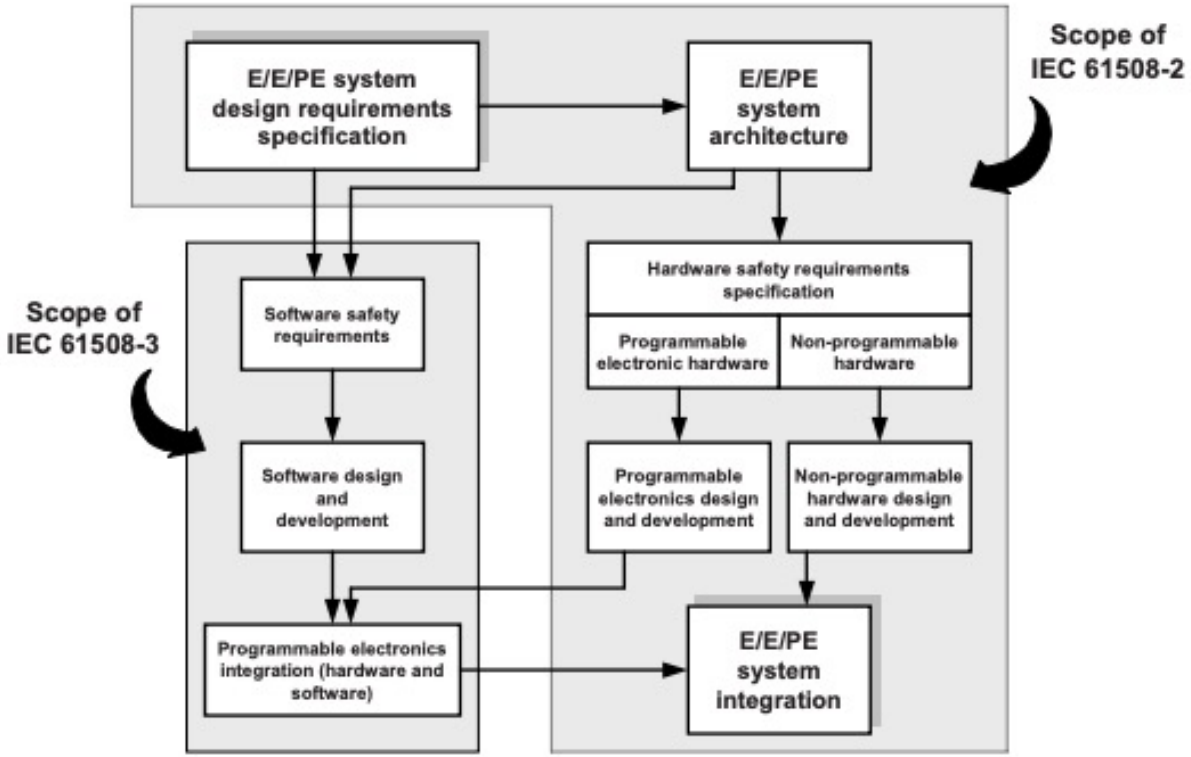


Figure 5 – Relationship and scope for IEC 61508-2 and IEC 61508-3

IEC 61508 Software Development Lifecycle

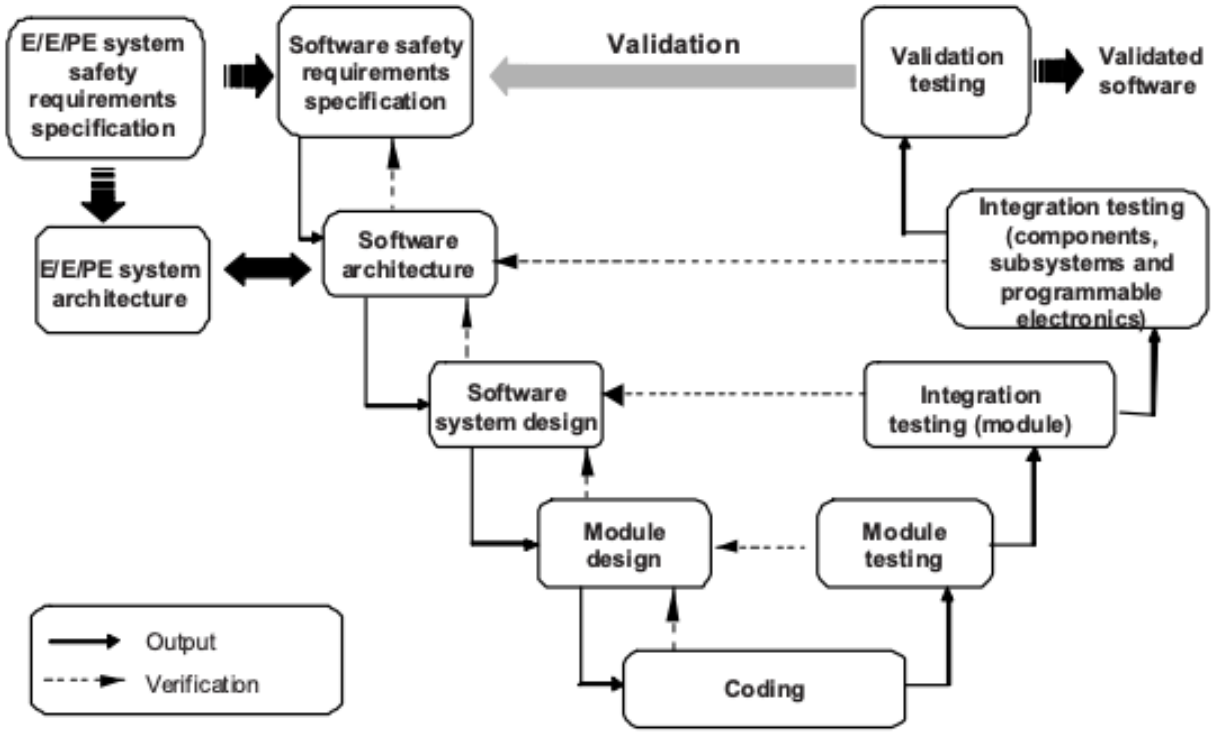


Figure 6 – Software systematic capability and the development lifecycle (the V-model)

Software Lifecycle Detailed Overview

Table 1 – Software safety lifecycle – overview

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
10.1	Software safety requirements specification	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	PE system; software system	7.2.2	E/E/PE safety requirements specification as developed during allocation (see IEC 61508-1) E/E/PE system safety requirements specification (from IEC 61508-2)	software safety requirements specification
10.2	Validation plan for software aspects of system safety	To develop a plan for validating the software aspects of system safety	PE system; software system	7.3.2	software safety requirements specification	validation plan for software aspects of system safety

Software Lifecycle

10.3	Software design and development	Architecture: To create a software architecture that fulfils 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)	software architecture design; software architecture integration test specification; software/ PE integration test specification (also required by IEC 61508-2)
10.3	Software design and development	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

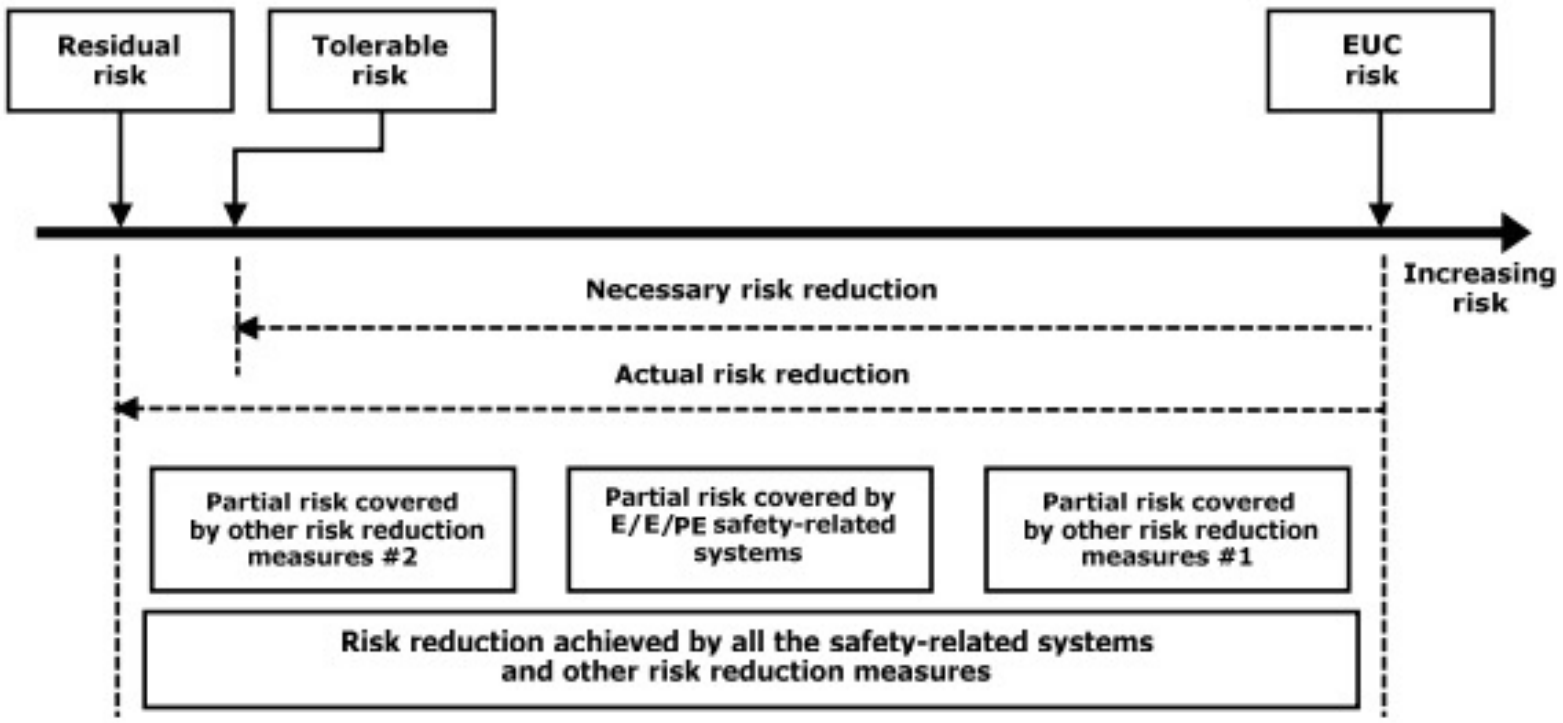


Figure A.1 – Risk reduction – general concepts (low demand mode of operation)

Risk reduced to tolerable levels

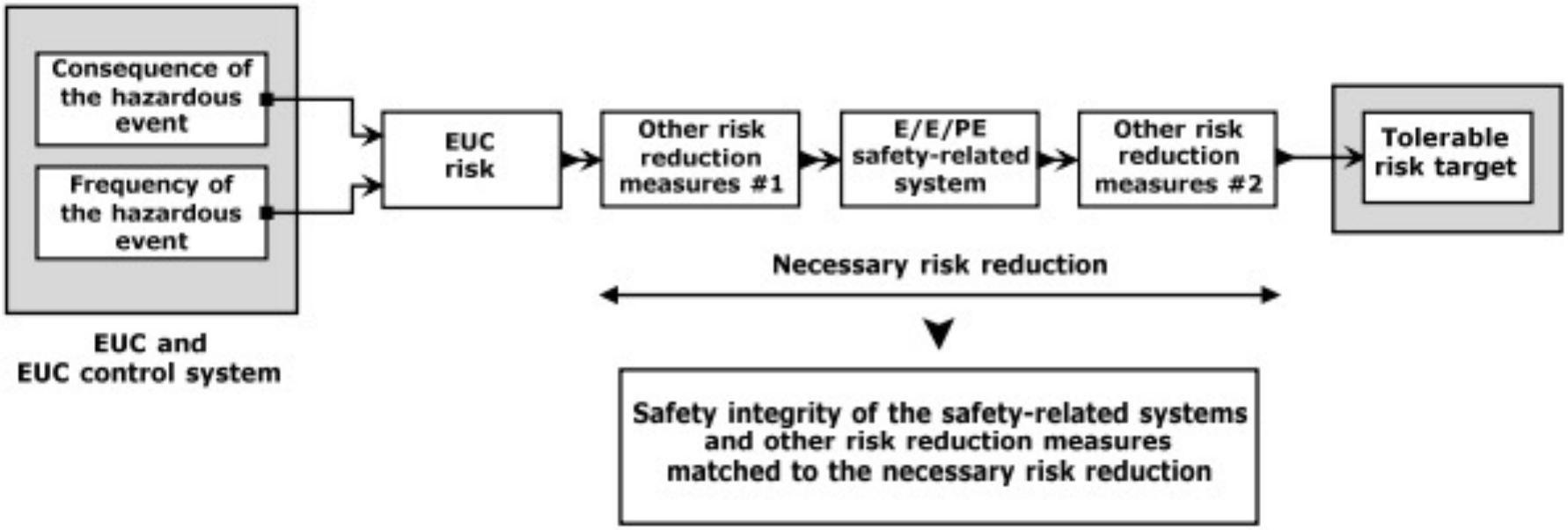


Figure A.2 – Risk and safety integrity concept

ALARP: One approach to achieving tolerable risk

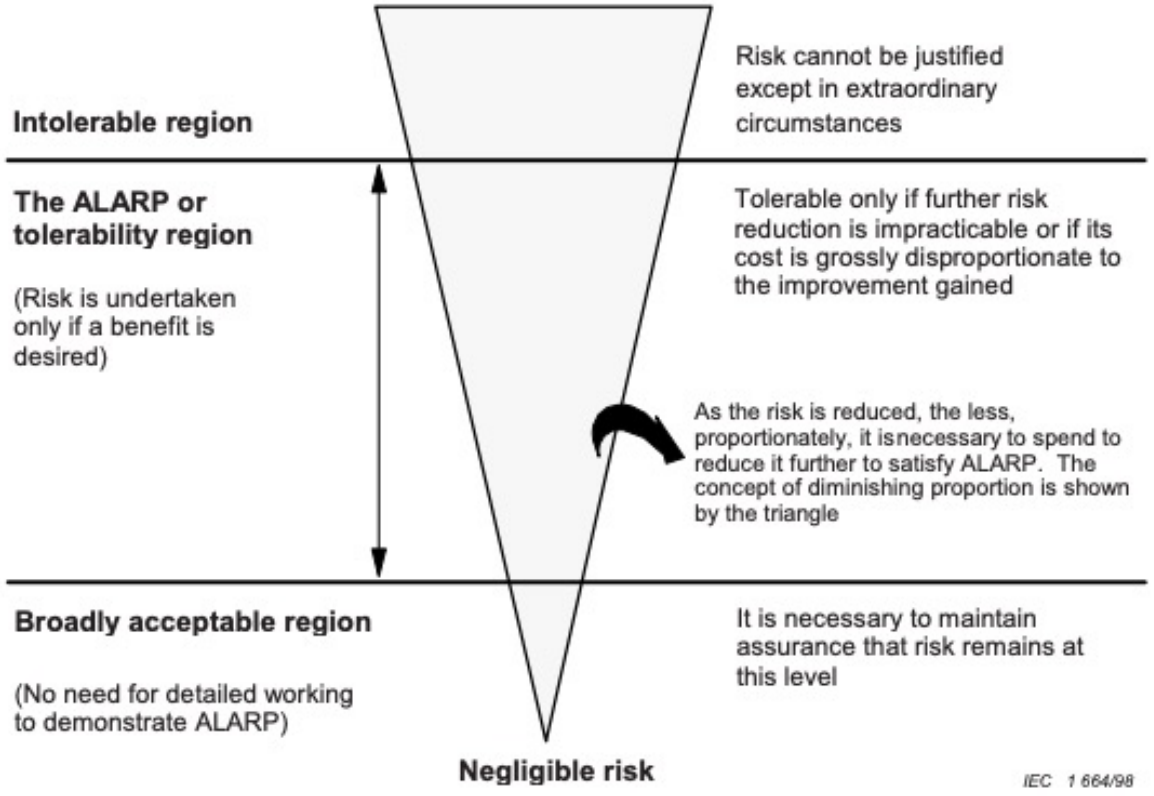


Figure C.1 – Tolerable risk and ALARP

Risk Classes

Table C.1 – Example of risk classification of accidents

Frequency	Consequence			
	Catastrophic	Critical	Marginal	Negligible
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

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 C.2 – Interpretation of risk classes

Risk class	Interpretation
Class I	Intolerable risk
Class II	Undesirable risk, and tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained
Class III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
Class IV	Negligible risk

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