

Project Experience with IEC 61508 and its Consequences

Abstract

The paper reports on the experiences with implementation of IEC 61508 in recent projects with European, North American and Japanese system vendors. The paper describes problems identified in implementing the standard and proposes a knowledge tool and a combination of software verification methods to mitigate these issues.

1 Introduction

In the past two decades only a small group of system vendors serving the nuclear, avionics, medical, railroad and process industry came in contact with requirements for Functional Safety of computerized systems. Now within the relatively short period of three years, the user requirements sections of many request for bids require engineering contractors and system suppliers world-wide to comply with the Functional Safety requirements of the international standard IEC 61508.

IEC 61508 has requirements for systems using complex electronics and programmable electronics whose failure could have an impact on the safety of persons and/or the environment. It describes methods to classify risk and specifies requirements on how to avoid, detect and control systematic design faults, particularly in software development, random hardware faults and common cause failures, and to a lesser extent operating and maintenance errors.

2 Success, Strength and Weaknesses of IEC 61508

Even though IEC 61508 appears to many people as a totally new set of requirements, the standard is just one in a long chain of standards on Functional Safety of computerized systems and software. New is its positioning as an International Basic Safety Publication outside of a particular industry sector as compared to other standards for the nuclear and avionics industry.

IEC 61508 also stands out in its system approach to address the complete safety installation from sensor to actuator with its technical as well as management issues. This system approach and the worldwide publicity make Buyers of Programmable Electronic Systems (PES) and Authorities see it as a major source of reference to reduce their uncertainty on complex systems in safety applications. The respect IEC 61508 receives at Authorities also drives the hope of decision makers in the plant operator community as well as with the equipment vendors that it will replace national regulations within a few years. This hope is supported by its recent ratification as European Norm EN 61508. In Europe, contradicting national standards need to be withdrawn by August 2004.

2.1 Market Analysis and Trends

How does the awareness for functional safety materialize in the market? The following is an excerpt from the 2001 edition of the exida.com "Safety System and Critical Control Market Report – *Growth Market with Strong Change*". Safety Logic Controllers established a solid market since mid of the 1980s. For 2001, the market is estimated to app. \$ 300,000,000 US dollars of equipment sales and app. \$ 380,000,000 US dollars of related service contracts. Services include logic solver programming and integration services, field services including installation, commissioning and periodic inspection and safety analysis services.

As the market for Safety Logic Controllers matured, a strong consolidation took place. During the 1990s, many of the leading automation equipment suppliers established safety divisions primarily by buying specialized Safety Logic Controller vendors. Others have brand labeling contracts.

ABB	bought	August Systems and ICI Eutech
Honeywell	bought	Pepperl & Fuchs Safety Systems
Siebe (Invensys)	bought	Triconex
Siemens	bought	Moore Products
Rockwell	brand labels	HIMA products
Yokogawa	brand labels	Siemens Moore products

It is expected that the number of suppliers will decrease further in the future as the trend toward Total Integration of Control and Safety within one product line continues. DOW Chemical in process industry and Siemens in machinery industry have first realized the advantages of Total Integration for application design. At this time, ten major suppliers comprise much of the market. Their current market estimated share is shown in figure 1.

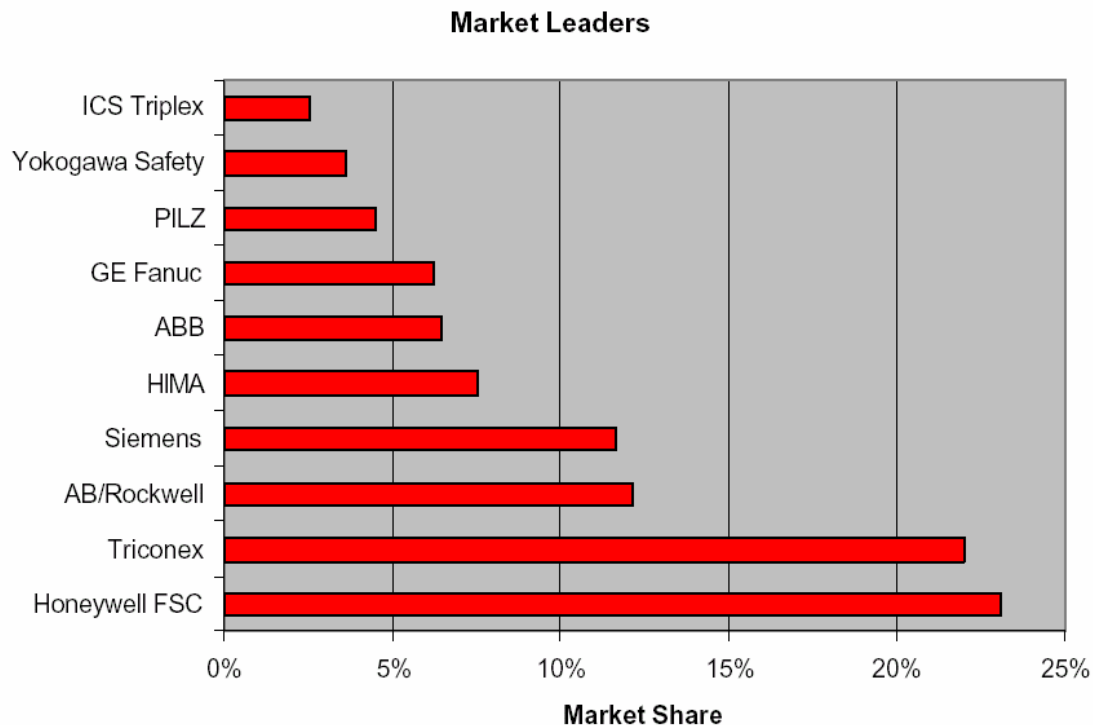


Fig. 1: Safety Logic Solver Market Leaders for 2001

With IEC 61508 and IEC 61511, the use of standard Programmable Logic Controllers (PLC) becomes hardly justifiable. These standards require that the application risk is determined and then reduced to a tolerable limit. The required risk reduction is classified by Safety Integrity Levels. Due to the requirement for demonstrable safety integrity, the use of standard PLCs will drop considerably outside Europe in favor of Safety Logic Controllers suitable for SIL2 and SIL3. Europe experienced this drop already due to more rigorous application standards and a strong influence of the TÜV assessment companies.

In 2000, SIL 2 emergency shutdown (ESD) was the strongest application area with 27% of sales (Figure 2). SIL 3 ESD had the next largest percentage with 23% while both SIL 1 ESD and burner management systems (BMS) applications had 13%. Machine Safety and Fire & Gas applications both had 8% of total sales. Critical Control applications and High Integrity Pressure Protection (HIPPS) applications in oil industry had 4% of sales each.

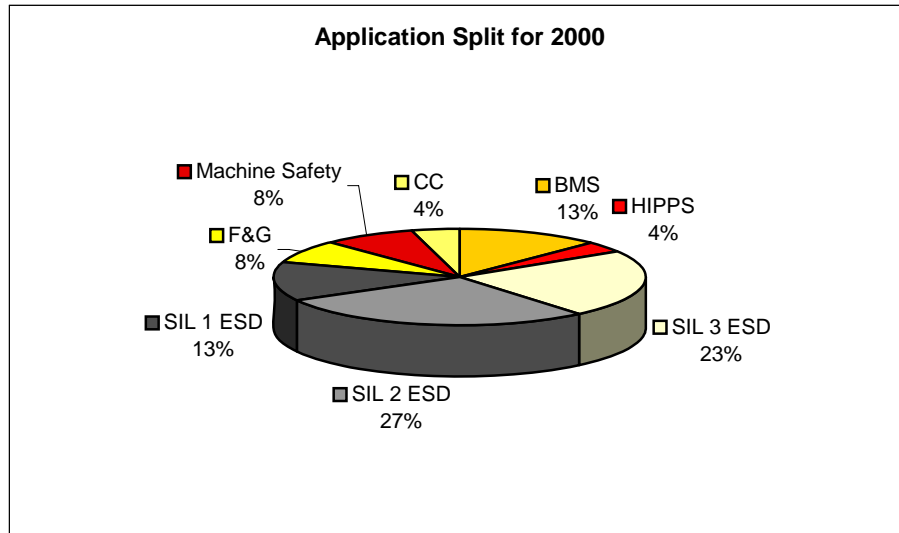


Fig. 2: Overall Application Splits for 2000

As a consequence of IEC 61508, more thorough risk classification and SIL verification is done at the major Operating Companies. This is expected to lead to a decline of the SIL3 ESD market segment whereas other market segments including the machine safety segment (Figure 3) are expected to grow. This is supported by the increased acceptance of Functional Safety and IEC 61508 in the Automotive industry world-wide incl. North America.

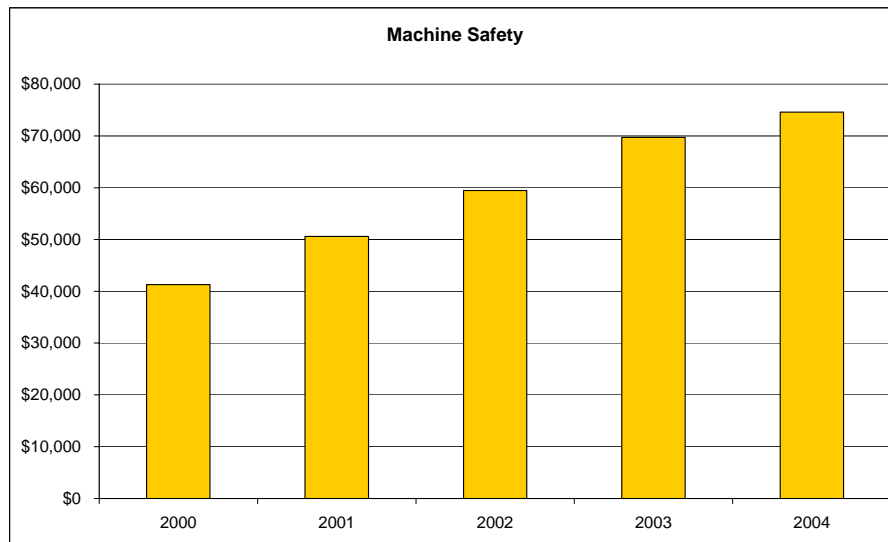


Fig. 3: Machine Safety Application Growth Worldwide

IEC 61508 and ISA S84.01 have recently resulted in end-user requests for field devices suitable for safety applications. Now pressure, temperature and level transmitters and valve positioners, solenoid valves are being enhanced to comply with IEC 61508. End-users would like to have their standard field devices enhanced for safety applications. They do not want field devices particularly designed and sold for safety. These enhanced products will take a strong and growing share of the safety automation market.

2.2 Strengths

IEC 61508 introduces a rigorous requirement driven approach to development of Functional Safety of applications and equipment alike. Again this is not new but it took time to implement

such a rigorous approach. As different projects have now been executed at operating companies and equipment vendors, one can see the benefit this approach provides:

Operating companies: More thorough risk analysis leads to reduced costs. It has been reported by SHELL [Wiegerinck 1999] that more than 70% of their Safety Functions in process applications are less than SIL3. Thus many were over-engineered in the past. On the other hand the experience based safety instrumentation concepts led to a small percentage of Safety Functions that were classified too low.

User – Vendor relation: More thorough risk analysis leads to more precise Specifications of safety functions, timing and safety integrity requirements. This makes it easier for vendors to understand the problem and propose adequate and cost-effective solutions.

Vendors: First examples from the development of PES show that the Safety Lifecycle with its rigorous requirements driven development approach leads to hardware and software projects being on time and meeting user expectations more accurately. The author was happy to participate in a project where on the same platform a safety system and a standard system were developed in the same time frame. The development team of the safety system could proudly claim that they delivered on time whereas the standard system was months overdue. The safety system development also met the specifications more accurately such that the customer later considered using safety system modules instead of standard system modules, at least for an interim period. It is now recognized that a rigorous requirements driven development approach enhances accountability of software project schedules.

Probabilistic approach: Due to the strong technical influence of well proven safety concepts there is a strong emphasis on random Hardware fault investigations in Europe. An example is the standard EN 954-1 for safety-related parts of machine control systems. Whereas this was justified in the past by unreliable electronic components and manufacturing techniques, IEC 61508 puts it in balance with other factors as the Common Cause failure by introducing probabilistic evaluation. One can demonstrate that the Probability of Failure on Demand of a redundant system configuration is better than a single channel system only if its Common Cause factor β is better than:

$$\beta < (1 - \text{SFF}_{1001D}) / (1 - \text{SFF}_{100ND})$$

SFF stands for Safe Failure Fraction, see tables 2 and 3 of IEC 61508-2;

1001D stands for single channel architecture with diagnostics;

100ND stands for redundant architecture with diagnostics where 1 out of N channel is sufficient to perform the safety function;

An example from the Architectural Constraint table for SIL 3 might help to understand:

$$\text{SFF}_{1001D} \geq 99\%$$

$$\text{SFF}_{1003D} \geq 60\%$$

$$\beta < 2,5\%$$

Such low Common Cause factors are not easy to achieve for a homogenous redundant system, it shows the crucial importance of the Common Cause investigation and the Safe Failure Fraction (diagnostic coverage) in a single channel of the redundant system configuration.

2.3 Weaknesses

The success of IEC 61508 is not something one could have easily predicted. IEC 61508 and derived standards are voluminous and quite difficult to read and interpret. Many requirements are not allocated to a certain range of Safety Integrity Levels or to the complexity of the design. This makes it difficult to tailor for smaller projects and makes Management of Functional Safety (too) expensive for Small and Medium Enterprises upfront. A way to mitigate this weakness will be discussed in section 3.1 of this document.

IEC 61508 defines Safety Integrity as a property of the complete safety installation from sensor to actuator including the operator. In addition, parts 2 and 3 of the standard go in great detail into design and verification and validation (V&V) of programmable electronic systems (PES) hardware and software. This leads often to confusion at vendors who market their "SIL 3" products. The terms SIL Capability and Safety Criticality, discussed in Section 3.2.2, may help to resolve the confusion.

The publicity of IEC 61508 makes people forget that Application Standards and European Harmonized Standards have absolute dominance over any generic standard or Basis Safety Publication. This leads frequently to situations where programmable electronic systems (PES) were designed and implemented following IEC 61508 but the safety assessor for the application or appliance refuses the approval as, e.g., specific requirements of European industry sector standards like EN 954-1 are not literally met.

The possible ambiguity in the interpretation encourages many to use the standard as a toolbox where they take out and require or implement what they understand or like. Whereas in North America, users seem to be mainly concerned about the hardware safety integrity (dangerous failure rates and safe failure fraction), in Europe many users seem to be more concerned about the software safety integrity. The standard also leaves (too) much room for interpretation. In Europe and Germany, most experts interpret the architectural Hardware constraints and diagnostic requirements of part 2 of the standard different than American experts do, leading to situations described in the above paragraph. European industry sector standards such as EN 954-1 and EN 298 do not accept systems where the failure mode of a single component could lead to an unsafe state whereas IEC 61508 does.

Another area where the probabilistic approach of the standard leads to a huge difference in requirements is on pre-existing software and products in low demand mode versus high demand mode application. For the same software the acceptance criteria for the quantitative evaluation of collected and accepted operating data or statistical tests come out dramatically different for high demand mode versus low demand mode. The required number of successfully treated demands for low demand mode of operation can be calculated for a given Probability of Failure on Demand (PFD) and Confidence Level (C) to:

$$n \geq - \ln(1-C) / \text{PFD}$$

The required number of successful operating hours for high demand or continuous mode of operation can be calculated for a given Probability of dangerous Failure per hour (PdF) and Confidence Level (C) to:

$$\text{hours} \geq - \ln(1-C) / \text{PdF}$$

Table 1 shows the number of demands resp. operating hours to be executed without showing errors:

	SIL 1	SIL 2	SIL 3
IEC 61508-2 (Table B.6) IEC 61508-7 (Table D.1)	at least 1 year different applications C = 95%	at least 1 year different applications C = 95%	at least 1 year different applications C = 99%

	SIL 1	SIL 2	SIL 3
Low demand mode of operation	PFD $\geq 10^{-2}$ results in	PFD $\geq 10^{-3}$ results in	PFD $\geq 10^{-4}$ results in
Required demands executed without errors	n ≥ 300	n $\geq 3,000$	n $\geq 46,000$
High demand or continuous mode of operation	PdF $\geq 10^{-6}$ results in	PdF $\geq 10^{-7}$ results in	PdF $\geq 10^{-8}$ results in
Required number of operating hours executed without errors	hours $\geq 3 \cdot 10^6$ years ≥ 342	hours $\geq 3 \cdot 10^7$ years ≥ 3420	hours $\geq 4.6 \cdot 10^8$ years $\geq 52.6 \cdot 10^3$

Table 1: Statistical SIL Demonstration for pre-existing Software¹

It is obvious from these figures, that it is very difficult to demonstrate proven-in-use for systems that operate in high SIL and high demand or continuous mode of operation. For the same system when it operates in low demand mode, the requirements are very reasonable, however. The huge discrepancy in feasibility between low demand mode application and high demand mode application for the same system, while mathematically clear, leads to considerable practical and financial implications.

A way to mitigate this weakness for using existing software in high demand applications will be discussed in section 3.2.3 of this paper.

¹ Constraints: (1) Precisely identifiable unit with a clearly restricted functionality and (2) Demands must cover the full range of normal and abnormal inputs and modes.

3 Approaches to overcome important weaknesses

3.1 Enhance Readability and Usability

IEC 61508 is difficult to understand and to tailor for smaller projects. Hence its implementation tends to be expensive for small and medium enterprises. Assessors of TÜV Nord and the company *exida.com* developed independently tools to make the standard(s) and its interpretation more understandable. The tools focus, however, on different objectives. Whereas the tool from TÜV Nord allocates each requirement to a SIL range, the *exida.com* tool SafetyCaseDB merges requirements tracking and IEC 61508 safety lifecycle support (Fig. 4) with the safety case approach as described by DStan 00-55.

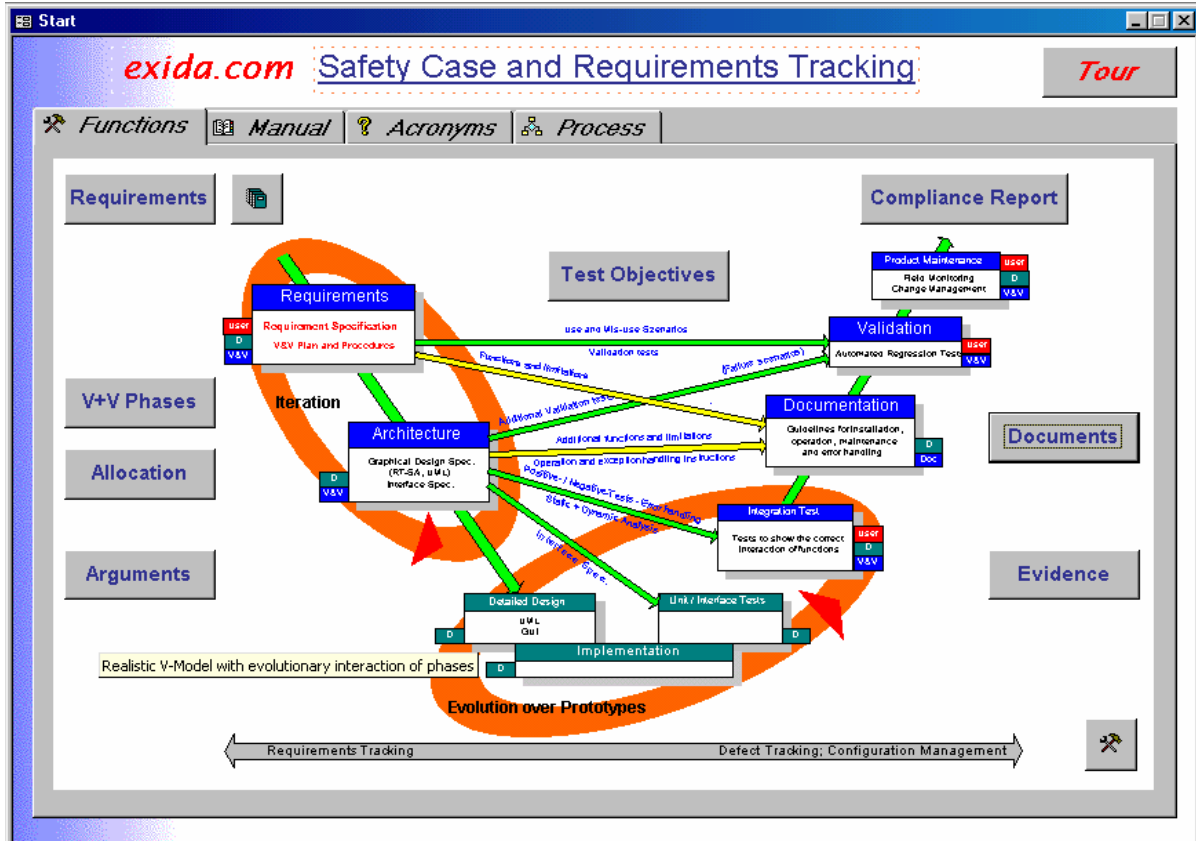


Fig. 4: Safety Lifecycle Support Functions of SafetyCaseDB

The *exida.com* knowledgebase tool helps development engineers and those with V&V responsibilities to understand and implement each requirement through a typical generic argument on how to meet the requirement and extensive document templates for evidence documents (Fig. 5).

As a characteristic inherited from the Safety Case methodology the tool generates one justification document for many Authorities and opens up the possibility to offload the design teams from compliance work, which can be done by safety specialists.

The knowledgebase has been used in different PES development projects and is currently being extended to the requirements of other standards such as EN 954-1, draft IEC 61511-1 and IEC 880 supplement 1. Outside the safety community, SafetyCaseDB and its underlying V&V-Case approach was enhanced and used in the EU Project 11956 „Broadband Access Services Solutions (BASS)“. The EU project with Lucent and the Italian Telecom company InfoStrada showed the advantages of a knowledge tool based V&V process for not-safety-related development.

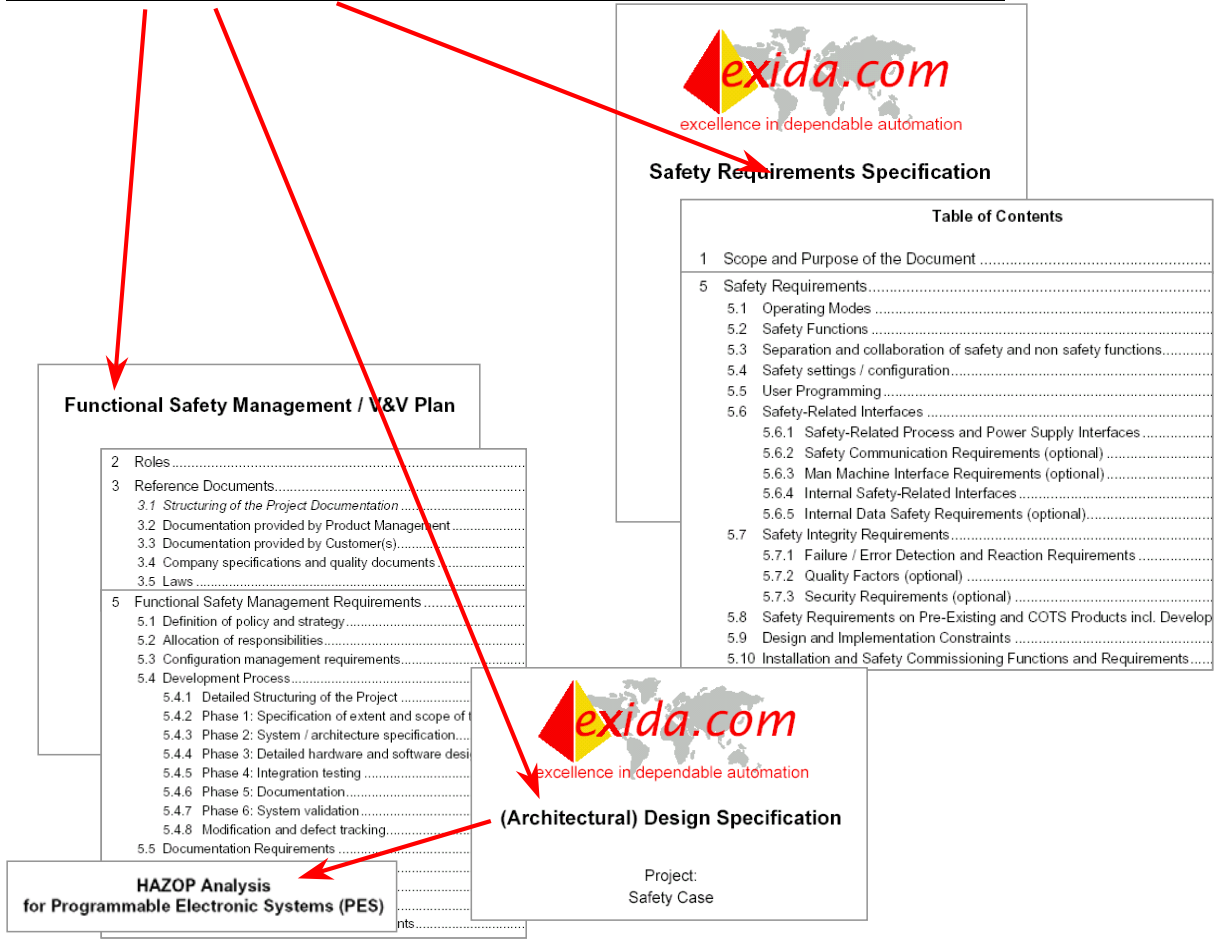
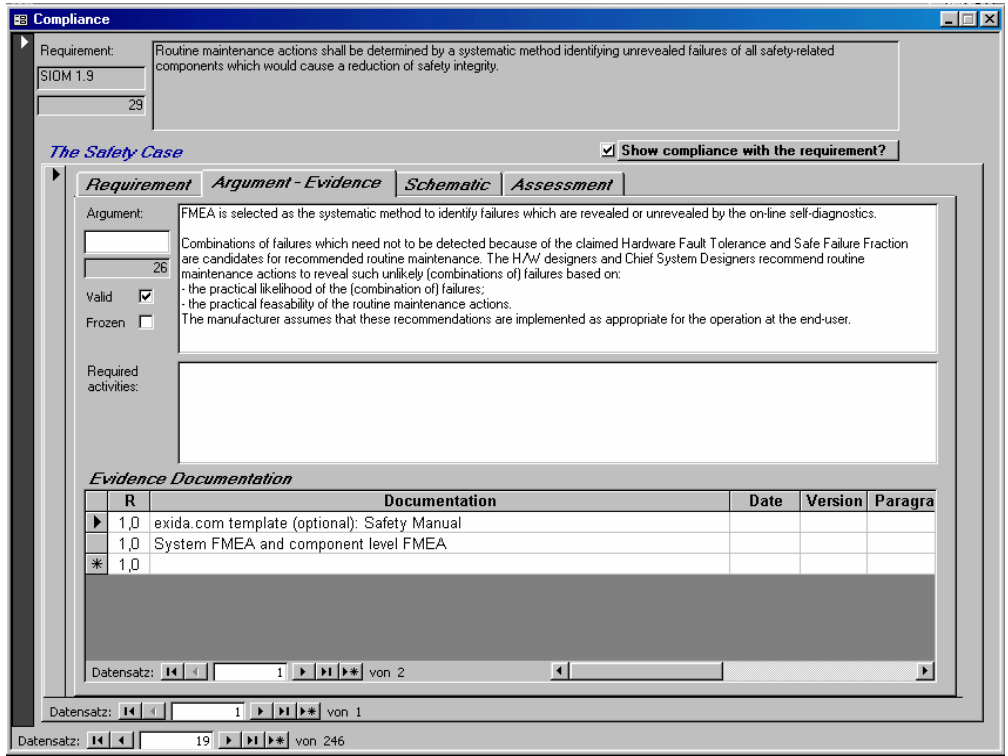


Fig. 5: Relation of Requirements, Arguments and Evidence Documents

3.2 Software Engineering

The following statement based on project experience might seem astonishing considering how long Software Design methods are being described in literature and supported by CASE tools, but “**Semi-formal Software Design and well-documented Module Tests are not yet State of the Art in Automation industry**”, not in Europe, North America and Japan. RT-SA and UML diagram techniques are used but mostly to enhance textual specifications than in a consistent manner to replace textual specifications. Even if it is accepted that they should be applied to safety projects, development teams do not have experience with these methods from their typical, non safety projects. The following sections will describe proposals to mitigate the discrepancy to the requirements of IEC 61508-3:

- The careful use of pre-existing software;
- The use of Software Criticality Analysis to achieve a better SIL allocation to Software;
- Combine overlapping methods and measures, not described by IEC61508-3.

3.2.1 Safety Dedication Process for Pre-existing Software

Today the responsible product managers typically define the next generation safety systems as a branch development of an existing product family. This allows and requires the development team to re-use the Software platform of the existing products, which leads to a controversial situation. The product manager expects a considerable reduction in development time and costs. The safety development team fears the lack of demonstrable quality of the existing software. The emphasize lays on the word “demonstrable”, as one cannot demonstrate that the development of the standard Software platform followed the requirements of IEC 61508-3.

To solve the issue, a Safety Dedication Process for Pre-existing Software is proposed (Fig. 6). The following scheme gives an overview of the relationship between development of the safety system and the safety investigation for the pre-existing software. The process was developed in a nuclear project with ABB / Westinghouse and TÜV and has shown to be successful.

The safety dedication process is applicable to pre-existing hardware and software products, whereas figure 6 emphasizes pre-existing software. The safety dedication process starts in **step one** with a collection and evaluation of the safety requirements imposed on the pre-existing (software) product by the future application of the safety system and the standards it shall meet. The requirements split into functional safety requirements resulting from how the pre-existing (software) product is used in the new safety system and safety integrity requirements originating from the safety criticality of the pre-existing product's use in the new safety system. The requirements are collected in a requirements tracking tool such as the *exida* SafetyCaseDB tool or another commercial tool such as DOORS™.

In **step two**, the suitability of the pre-existing software product to meet the safety function and safety integrity requirements is evaluated. This is done in three sub-steps: (1) as a paper study comparing the requirements and the pre-existing software product specification; (2) as practical validation tests to demonstrate that the safety function requirements are met by the pre-existing software product, (3) evaluation of the available development and V&V documentation of the pre-existing software product to identify weaknesses in meeting the safety integrity requirements (Gap analysis). The validation tests should be defined during the paper study.

In **step three**, a safety assessment of the pre-existing software is executed. The intention of the safety assessment is to evaluate the available safety measures:

- Proven operational experience of the pre-existing software product in other similar, although not safety-related applications;
- (Previous) Validation efforts demonstrated for the pre-existing software product;
- Test and analysis efforts during the Safety System V&V which cover the pre-existing software product.

To keep the operating experience of the pre-existing software product out of question, any requests for modification of the pre-existing software product or its usage should be avoided.

In **step four**, one specifies and executes or implements additional safety provisions to achieve the required safety integrity:

- Additional validation efforts to be executed by the safety system development team on the pre-existing software product, possibly during their safety system integration tests;
- Safety measures to be implemented by the safety system designer around the pre-existing software product to mitigate safety weaknesses which remained even after the execution of the safety activities listed above. Such safety measures might be implemented by means of a separate safety layer – see below.

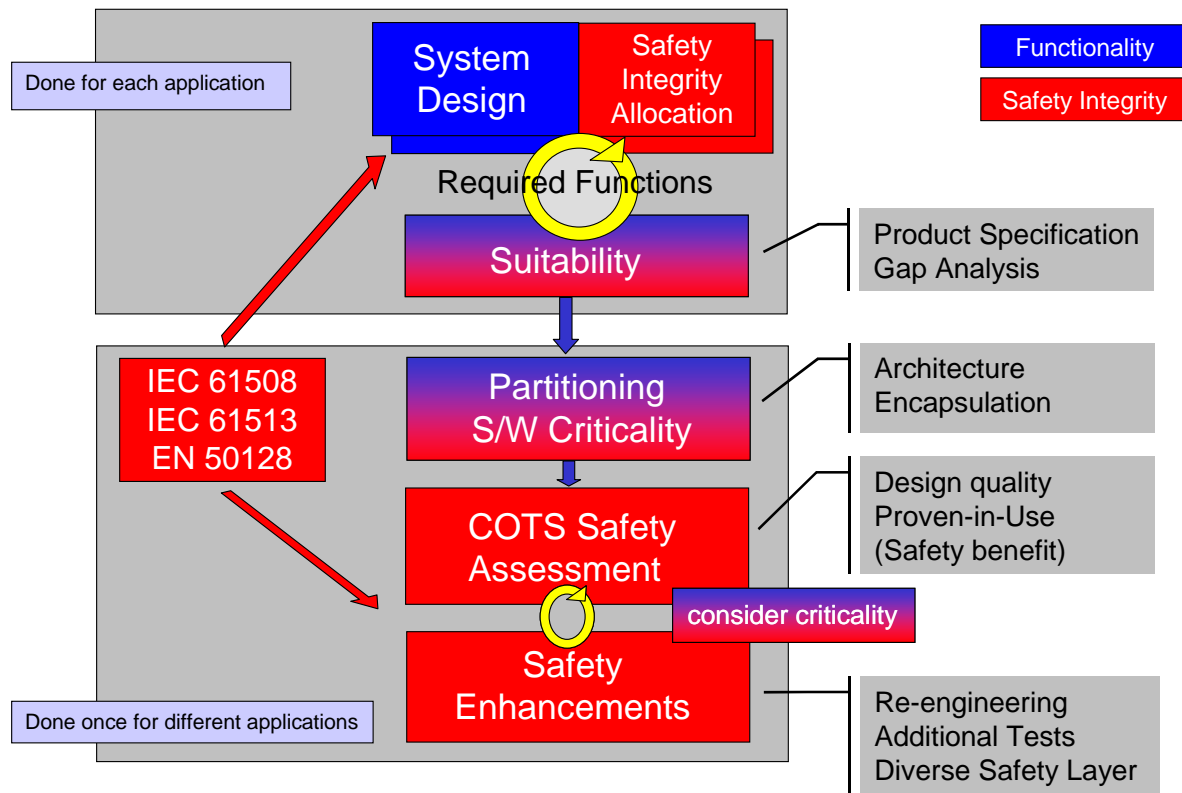


Fig. 6: Safety Dedication Process for Pre-existing Software

The key question for the third step, the safety assessment of the pre-existing software product is “How much is enough for a given criticality of use of the pre-existing software product?”. This might be formally answered by referring to the applicable safety standards (e.g. EN 50128, IEC 880, IEC 61508), which may, however, turn out to be an elimination scenario for most pre-existing (commercial of the shelf - COTS) software products, as their vendors might not be willing to meet the requirements of these standards on verification & validation and proven-in-use demonstration. Hence it is proposed to take into account not only the allocated safety integrity (SIL), but also the criticality of malfunctions of the pre-existing software product in the given application. This can be done by the method of Software Criticality Analysis (SCA), described below, using worst-case failure modes of the pre-existing software product irrespective of the previous development and V&V efforts. This approach minimizes the need for a detailed investigation of the pre-existing software product itself and is accepted by prEN50128.

3.2.2 Software Criticality Analysis

Currently IEC 61508-3 specifies which design and V&V measures shall be applied to Software implementing safety functions. It does not distinguish, however, between Software implementing safety functions and Software implementing safety support functions such as self-tests and code generation tools. This section describes a method successfully used since many years to evaluate the criticality of software in addition to the SIL of the safety function. The criticality evaluation complements the SIL allocation.

The method is proposed for the next IEC 61508-3 release as a way to do a **Tailoring of the IEC 61508-3 requirements** based on a detailed evaluation of the potential safety criticality of the software system.

3.2.2.1 Safety Criticality Categories

Safety Criticality denotes the potential of an encapsulated (software) unit to create an unsafe situation of the safety-related system, in case of a deviation from its specified functionality. The following **Criticality Categories** are distinguished:

- C3: Safety Critical** denotes a function, where a single deviation from the specified function may cause an unsafe situation.
- C2: Safety Relevant** denotes a function, where a single deviation from the specified function cannot cause an unsafe situation, but the combination with a second failure may cause an unsafe situation.
- C1: Interference Free** denotes a function, which is not safety critical (C3) or safety relevant (C2), but has interfaces with such functions.
- C0: Not-Safety-Related** denotes a function which is not safety critical (C3) or safety relevant (C2) and is unable to interact with such functions by architectural design.

The architecture and detailed design of the software sub-system should be structured into units of different Criticality. The Criticality allocation shall be limited to software units which meet the following constraints:

- clearly restricted functionality;
- encapsulated with a well defined interface;
- the timely execution of all C3 functions is monitored by the logical and timely program flow monitoring as required by IEC 61508-2.

The encapsulation might be achieved by hardware memory protection or software means, depending on the required SIL of the implemented safety function.

3.2.2.2 Tailoring of the IEC 61508-3 Requirements

The benefit of the Software Criticality Analysis is that it allows a justified reduction of the required **SIL Capability** and hence of the compliance effort for development, V&V and modification of the individual software unit. Table 2 shows how a lower criticality of the software unit may be used to lower the required SIL Capability.

SIL Capability		Criticality of the (software) unit			
		C0	C1	C2	C3
SIL of the safety function or safety-related	SIL1	No safety integrity requirements	No safety integrity requirements	Safety Demonstration following SIL1 (recommended)	Safety Demonstration following SIL1
	SIL2	No safety integrity requirements	See remark	Safety Demonstration following SIL1	Safety Demonstration following SIL2

SIL Capability	Criticality of the (software) unit			
	C0	C1	C2	C3
SIL3	No safety integrity requirements	See remark	Safety Demonstration following SIL2	Safety Demonstration following SIL3
SIL4	No safety integrity requirements	See remark	Safety Demonstration following SIL3	Safety Demonstration following SIL4

Table 2: Relation of SIL, Criticality and required Software Safety Integrity (SIL Capability)

Remark: For Interference free software functions four options exist:

- 1. No safety integrity requirements**, if the interference freeness can be demonstrated, e.g., by the use of hardware memory protection (e.g. MMU);
- 2. No safety integrity requirements**, if the implementation languages for the software unit enforces encapsulation like Modula, ADA, JAVA, C#;
- 3. Pointer analysis** for the pre-existing software product for other languages like C, C++ and Assembler.
- 4. Safety integrity requirements as of C3**, if option 1. or 2. are not applicable.

The Criticality Analysis results in much better understanding of any (software) system and good justification for much less investment in activities which are required by the standard but often not done, like semi-formal design methods, use of CASE tools and documented coverage tests for software modules. Table 3 shows as example **differences** as of IEC 61508-3 Annex A in highly recommended software design and V&V techniques for software units of SIL3, C2 and SIL3, C3.

SIL3	C2	C3 – additional effort
Architecture	<ul style="list-style-type: none"> ▪ Separation of safety and non-safety software ▪ Structured methods ▪ Failure detection and diagnosis (required by IEC 61508-2) ▪ Use of trusted modules 	<ul style="list-style-type: none"> ▪ Semi-formal methods using Computer aided tools ▪ Graceful degradation
Detailed design	<ul style="list-style-type: none"> ▪ Semi-formal methods ▪ Design and Coding standards ▪ Suitable strongly typed language ▪ Program sequence monitoring ▪ Tools with increased confidence from use 	<ul style="list-style-type: none"> ▪ Computer aided tools ▪ Defensive programming (always recommend by the author) ▪ Language subset
Module Testing	<ul style="list-style-type: none"> ▪ Dynamic analysis and testing ▪ Functional and black box testing <ul style="list-style-type: none"> ▪ Boundary value analysis ▪ Equivalence classes and input partition testing 	<ul style="list-style-type: none"> ▪ Performance testing <ul style="list-style-type: none"> ▪ Stress testing, Response timings and memory constraints ▪ Interface (coverage) testing
Integration Testing	<ul style="list-style-type: none"> ▪ Functional and black box testing ▪ Configuration management and Error tracking 	<ul style="list-style-type: none"> ▪ Performance testing (always recommend by the author)

SIL3	C2	C3 – additional effort
Modification	<ul style="list-style-type: none"> ▪ Impact analysis ▪ Re-verify changed and affected modules ▪ Configuration management ▪ Error tracking 	<ul style="list-style-type: none"> ▪ Re-validate complete system
Validation	<ul style="list-style-type: none"> ▪ Functional and black box testing ▪ Simulation / modeling 	
Verification	<ul style="list-style-type: none"> ▪ Static analysis <ul style="list-style-type: none"> ▪ Control and Data flow analysis ▪ Design and code reviews ▪ Dynamic analysis and testing <ul style="list-style-type: none"> ▪ Boundary value analysis 	<ul style="list-style-type: none"> ▪ Structure based testing (coverage testing)

Table 3: Difference of Safety Demonstration Effort between Criticality C2 and C3 in SIL3

The Software Criticality Analysis may be performed and formalized using techniques such as Software HAZOP (Hazard and Operability Analysis) [DStan 00-58] which is a systematic design examination to identify what variations from the design intent could occur in the functions, parameters and attributes.

3.2.2.3 Tailoring of the IEC 61508-3 Requirements on Support Tools

IEC 61508-3 requires that the Safety Integrity of software development support tools is demonstrated by either Certification **or** a demonstration for “Increased-Confidence-from-Use”. More specific requirements could be defined using Safety Criticality.

Grouping of Support Tools		Requirement
Tools generating Source Code and Executable Examples are: Compiler incl. libraries, Linker, CASE tools with code generation	C2	As for C2, the tool shall meet the IEC 61508-3 requirements one SIL lower than the SIL of the executable code being generated. Reasoning: The executable code generated by the tool is subject to extensive testing as specified for the SIL. Hence, the requirements on the tools can be less than on the executable code itself.
Tools which support code generation Examples are: Version Control, Make Utility	C2 or C1	As for C2, the tool shall meet the IEC 61508-3 requirements one SIL lower than the SIL of the executable code being generated or the output of the tool is verified (e.g. by diff tool). Reasoning: The source code delivered by the tool or the code generation controlled by the tool can be subject to verification and testing.
Design and Test support tools which do not generate code but enhance the safety integrity of the software Examples are: CASE tool without code generation, Debugger, Emulator, Profiler, Unit Test tool	C1	No safety integrity requirements on the tool exist, if its output is subject to (data) verification (and testing) as specified for the SIL. If the output of the tool cannot independently be verified and is an important contribution to the Safety Case then the tool shall be classified C2. Reasoning: The output of the tool cannot introduce additional software faults but can cause a non-detection of software faults during V&V. Hence the output of the tool should be checked.

Grouping of Support Tools		Requirement
Support tools which do not generate code nor enhance the integrity of the safety-related software Examples are: Editor, IDE	C1	No safety integrity requirements Reasoning: Any output of the tool is subject to V&V as specified for the SIL.

Table 4: Safety Integrity of Software Development (Support) Tools

3.2.3 Overlap of Safety Measures for pre-existing Software products

A fundamental concept of the safety dedication process for pre-existing software products described in section 3.2.1 is the combination and overlapping of safety measures. As said before, it is usually impossible to demonstrate that the software development and V&V process of a pre-existing (software) product complies with IEC 61508(-3) literally. This must not mean, however, that the pre-existing product is inadequate for the considered safety application.

The standard puts significant emphasis on fault avoidance by excellent development and V&V. The concept presented herein, builds however on the combination of three aspects to achieve the required Safety Integrity for the pre-existing software product:

1. Lowering of the safety criticality of the pre-existing software product as used in the safety system;
2. Fault avoidance before commissioning;
3. Fault control at runtime.

Lowering of the safety criticality and fault control for the pre-existing software product can be achieved by (1) diversity and (2) safety measures around the pre-existing software product, called safety layer. Additional fault control can be achieved by extensive online test. Fault avoidance for a pre-existing software product can be best achieved by demonstrated operating experience and extensive testing during the development of the safety target system.

A very simple example may demonstrate the benefits of the approach:

- Example: A robot axis drive controller is to be developed. The controller software is a pre-existing product.
- Risk: The hazard analysis of the robot shows that run-away is the critical failure mode. The risk classification shows that the axis control must meet SIL2.
- Issue: Without other measures the controller software would be SIL2. Meeting SIL2 is commercially not viable for the controller vendor.
- Solution: Add an independent run-away monitor as a Safety Bag. The controller software and the run-away monitor shall now meet SIL1 (SIL2, C2) or no safety integrity is shown for the controller software (SIL0) and the run-away monitor meets SIL2.
- Limitation: As both software run on the same system, the controller software can not be SIL0 even if the monitor meets SIL2.
- Implementation: Show extensive operating experience with the axis controller software to meet SIL1 or execute the necessary number of tests. Develop the run-away monitor according to SIL1 requirements.

The safety suitability evaluation (section 3.2.1, step 2) should also give some indication on the safety benefits achieved by using the pre-existing software product. The term "Safety Benefit" denotes the benefits an application gains from the use of the pre-existing software product and which would be difficult to achieve if the pre-existing software product would not have been used. A good example for the benefit of RT-OS is their support of Hardware memory protection

and hence less systematic software errors by better encapsulation of RT-OS processes. Encapsulation is particularly important for avoiding errors during software modification.

The overlapping of safety measures is summarized in Fig. 7.

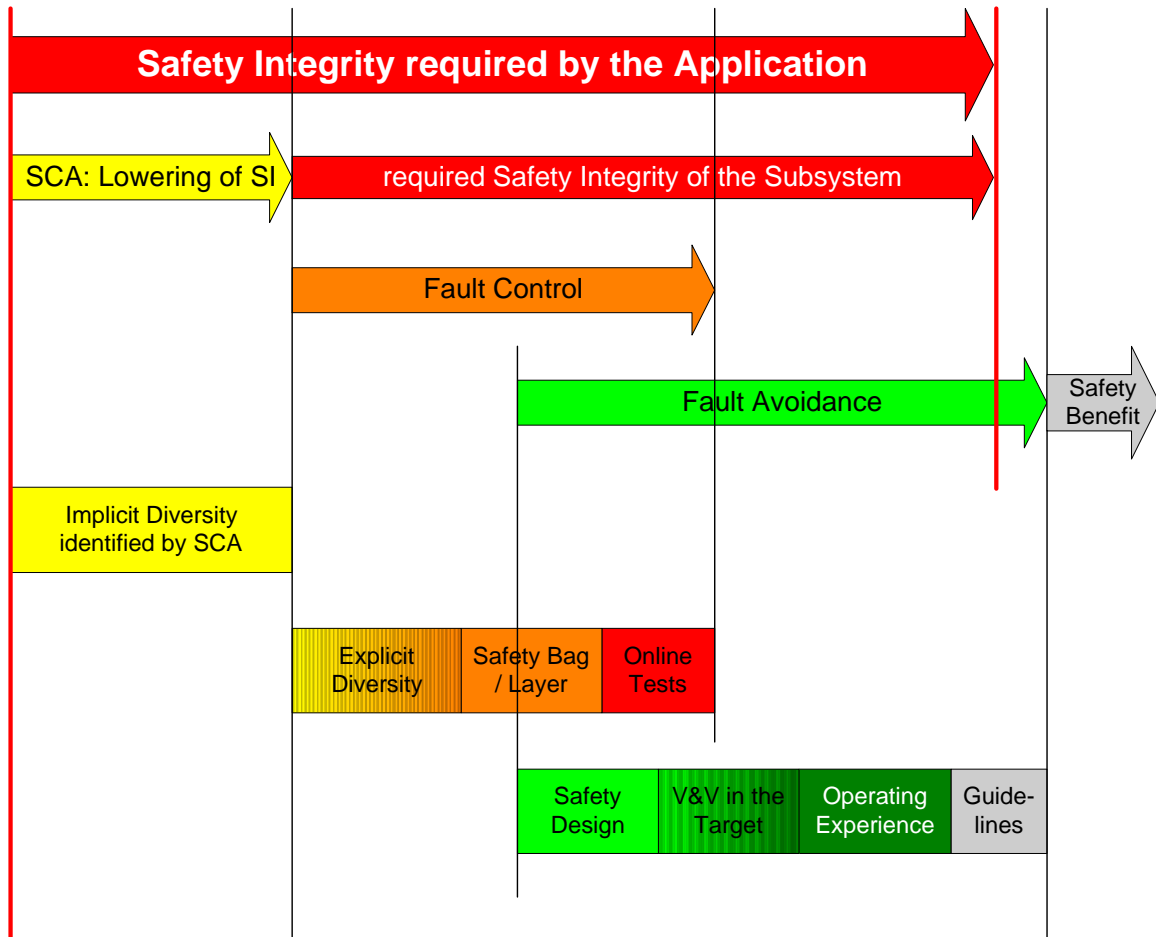


Fig. 7: Overlapping of Safety Measures

4 Conclusion

IEC 61508 is here and is a big success. Buyers of Programmable Electronic Systems and Authorities see it as a major reference to reduce their uncertainty on complex systems in their safety applications, but the learning curve is steep. The paper introduced a knowledge tool to smoothen the process of identifying and understanding the IEC 61508 requirements and introducing the appropriate safety techniques.

As with any new specification, the standard leaves room for improvement. The paper described a safety demonstration process for pre-existing software and a criticality analysis technique to support tailoring of the IEC 61508 requirements. Both are not specified by the standard but have great benefit in demonstrating compliance with the standard while meeting the today needs for shorter time to market.

Literature

- [DOORS] <http://www.telelogic.com/products/doorsers/index.cfm>
- [DStan 00-55] DStan 00-55; Requirements for Safety Related Software in Defence Equipment
- [DStan 00-58] DStan 00-58; HAZOP Studies on Systems Containing Programmable Electronics
- [EN 298] EN 298; Automatic burner control systems for gas burners and gas burning appliances with or without fans
- [EN 50128] EN 50128; Railway Applications: Software for Railway Control and Protection Systems
- [EN 954] EN 954-1:1996; Safety-related parts of control systems, Part 1: Generic principles for design
- [ISO 13849] ISO/DIS 13849-1:2002 replacement for EN 954-1:1996
- [exida 2001] Safety System and Critical Control Market Report – *Growth Market with Strong Change* (www.exida.com)
- [IEC 880] IEC 60880:1986; Software for computers in the safety system of nuclear power stations
- [IEC 61508] IEC 61508:1998 and 2000, part 1 to 7; Functional Safety of Electrical, Electronic and Programmable Electronic Systems
- [IEC 61511] IEC 61511: 2002, part 1 to 3; Functional safety - Safety instrumented systems for the process industry sector
- [ISA S84] ANSI/ISA S84.01:1996; Application of Safety Instrumented Systems for the Process Industries
- [Wiegerinck 1999] Interkama / ISATech 1999; Compliance to IEC 61508 (/ IEC 61511) Classification and Implementation of Instrumented Protective Functions – The SHELL approach; Jan Wiegerinck; SHELL Global Solutions