The Application of IEC 61508 in the Automotive Sector

Günter Glöe, Folkert Jürgens, Gerhard Rabe
TÜV Nord Gruppe, Hamburg / Hannover

Abstract

With the IEC 61508 /IEC 61508/ there is an international standard available that provides profound guidelines for the development and use high quality embedded systems. On a first glance it looks as though the standard might be fit for practical application.

However, going into very detail of the standard and dealing with the requirements demanded there leads to severe problems which are based mainly on the size of the standard and the even high number of requirements. In addition a majority of requirements is 'hidden' somewhere in the text and not to be identified obviously.

To solve this problem it is recommended to use support tools. In the following presentation it will be demonstrated how IEC 61508 is being applied in a concrete project and which procedures have been developed to make work efficient and to meet customer's time and cost requirements.

The Application

A large developer of automotive systems from abroad developed an electronic steering system which was to be verified and validated for the sake of getting official licence for the use in Europe. The electronic system consists of sensors adjusted to a conventional steering wheel which transfer the angle information to a microprocessor based control system. This system itself is – via a process interface – connected to a series of final elements (valves) that manage the steering process.

The software we are talking about has not been developed being aware of IEC 61508 and its requirements. In addition, from its starting point it has not been for safety purpose. Relevant parts have been designed to provide different functions needed to run certain types of cars in a reliable manner. So, from a certain point of view mayor parts of the software may be considered to be COTS.

Language for application software is „C". Size of the executable code is about 100kB.

The amount of documented development information available is of the size known from usual projects. Thus the amount of documentation is less than the size demanded by IEC 61508.

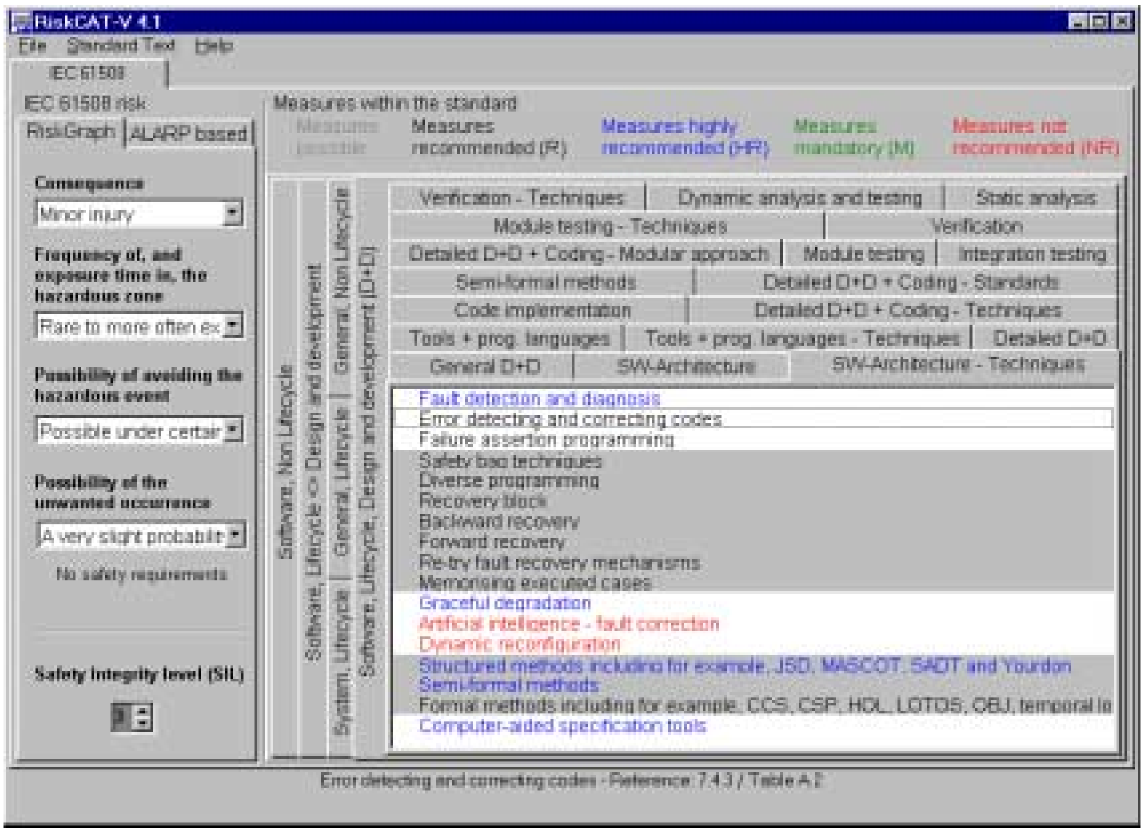
The safety aspects of this application are as follows: The electronic steering system will be implemented into vehicles which themselves will be moving for a certain limited time on public roads. In so far erroneous conditions may cause malfunctions of the steering system and hence endanger persons, either in the vehicle itself, on the road or on pavements.

A risk calculation took into account that in addition to the electronic system there is a mechanical back-up steering system which allows further operation of the vehicle in case of electronic break down. Due to that reason the requirements for functionality, safety, reliability and other quality characteristics have been imposed by experts from the automotive sector to SIL 2.

In the course of the licensing procedure it had to be proved that all SIL 2-requirements of IEC 61508 have been met in the electronic steering system.

Handling of IEC 61508

To handle IEC 61508 /IEC 61508/ in an efficient and reproducible manner we used the RiskCAT tool /CATS 2001/.


RiskCAT-V 4.1
File
Standard Text
Help
IEC 61508
IEC 61508 risk
RiskGraph
ALARP based
Measures within the standard
Measures possible
Measures recommended (R)
Measures highly recommended (HR)
Measures mandatory (M)
Measures not recommended (NR)
Verification - Techniques
Dynamic analysis and testing
Static analysis
Module testing - Techniques
Verification
Detailed D+D + Coding - Modular approach
Module testing
Integration testing
Semi-formal methods
Detailed D+D + Coding - Standards
Code implementation
Detailed D+D + Coding - Techniques
Tools + prog. languages
Tools + prog. languages - Techniques
Detailed D+D
General D+D
SW-Architecture
SW-Architecture - Techniques
Software, Non Lifecycle
Software, Lifecycle <> Design and development
General, Non Lifecycle
General, Lifecycle
System, Lifecycle
Software, Lifecycle, Design and development (D+D)

Consequence: Minor injury

Frequency of, and exposure time in, the hazardous zone: Rare to more often ex

Possibility of avoiding the hazardous event: Possible under certai

Possibility of the unwanted occurrence: A very slight probabili

No safety requirements

Safety integrity level (SIL): 1

Fault detection and diagnosis
Error detecting and correcting codes
Failure assertion programming
Safety bag techniques
Diverse programming
Recovery block
Backward recovery
Forward recovery
Re-try fault recovery mechanisms
Memorising executed cases
Graceful degradation
Artificial intelligence - fault correction
Dynamic reconfiguration
Structured methods including for example, JSD, MASCOT, SADT and Yourdon
Semi-formal methods
Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal lo
Computer-aided specification tools

Error detecting and correcting codes - Reference: 7.4.3 / Table A.2

Figure 1: Screen of the IEC 61508 support tool RiskCAT

First step was to create a **list of all requirements** from IEC 61508 applicable

- for SIL2
- hardware and / or software
- for the life cycle phases before „Overall installation and commissioning“.

Second step was to go through the information suggested by IEC 61508-1 in tables A.1, A.2 and A3 to select those ‘information’ to be subject of assessment. Furthermore we went through the ‘documents’ provided and identified those containing the information to be subject to assessment. This resulted in the list of **documents to be subject of**

assessment. It contains seven documents. Because of certain reasons hardware was not in the focus.

In the third step we went through the list of all requirements (about 450) and selected those **requirements applicable to the information under assessment**. Result of this step has been a set of seven checklists assigning the selected requirements (about 100) to the seven documents. Two examples of the checklists are shown below. (The formatting has been changed for this presentation. In the original checklists there is more space to give arguments on conformance.)

Requirements from part 3, clauses 7.2.2 and 7.2/Table A.1	**degree of obligation for SIL 2**	**Conformance**
Derivation of specification from safety requirements and planning	mandatory	
Availability of requirements specification to SW developers	mandatory	
Sufficiently detailed specification to achieve required safety integrity	mandatory	
Review of the specification by developer	mandatory	
Developer shall consider during review safety functions, architecture, ...	mandatory	
Procedures for resolving disagreements over assignment of SW SIL	mandatory	
Clear, precise, verifiable, maintainable, ... specification	mandatory	
If not already defined: Detailed modes of operation of the EUC	mandatory	
Specification of constraints between HW and SW	mandatory	
Consideration of SW self-monitoring, monitoring of HW, ...	mandatory	
Clear identification of non-safety functions performed	mandatory	
Expression of required safety properties of the product	mandatory	
Computer-aided specification tools	recommended	
Semi-formal methods	recommended	
Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	recommended	

Figure 2: Checklist for Software Requirements Specification

Requirements from part 3	**degree of obligation for SIL 2**	**Confor-mance**
Determination of division of responsibility between supplier and user	mandatory	
Selection of a suitable set of development tools	mandatory	
Consideration of tools supplying services over the whole E/E/PE-lifetime	highly recommended	
Selection of suitable programming language	mandatory	
If no suitable language is available: detailed justification for actual choice	mandatory	
Review and usage of coding standards	mandatory	
Coding standards specifying good programming practice, documentation, ...	mandatory	
Contents of source code documentation are description, inputs, ...	highly recommended	
Suitable programming language	highly recommended	
Strongly typed programming language	highly recommended	
Certified tools	highly recommended	
Tools: increased confidence from use	highly recommended	
Certified translator	highly recommended	
Translator: increased confidence from use	highly recommended	
Library of trusted/verified software modules and components	highly recommended	
Readable, understandable, testable code satisfying requirements	mandatory	
Review of code of each module	highly recommended	
Structured methods including for example, JSD, MASCOT, SADT and Yourdon	highly recommended	
Semi-formal methods	highly recommended	
Modular approach	highly recommended	
Design and coding standards	highly recommended	
Structured programming	highly recommended	
Use of trusted/verified software modules and components (if available)	highly recommended	
No dynamic objects	highly recommended	
No unconditional jumps in programs in higher level languages	highly recommended	
Software module size limit	highly recommended	
Information hiding/encapsulation	highly recommended	
One entry/one exit point in subroutines and functions	highly recommended	
Fully defined interface	highly recommended	

Figure 3: Checklist for Code

Because quality is achieved during design and development the most important use of these checklists would be to give them to the design and development team at the start of the project and ask for compliance with these requirements. By this final testing and certification should be without any problems.

However – as usual today – in the project we report about, these requirements as well as the checklists were created by the assessors after design and development. So the fourth step handling IEC 61508 was to ask for the **producer's arguments for compliance** with the subset of requirements given in the checklists. Authors feel that this may be the most important step in an assessment:

- to assist designers and developers in recognising the requirements imposed on embedded controllers and their software according to the state of the art (IEC 61508) and

- to assist designers and developers in recognising themselves the extent of compliance of their products with state of the art.

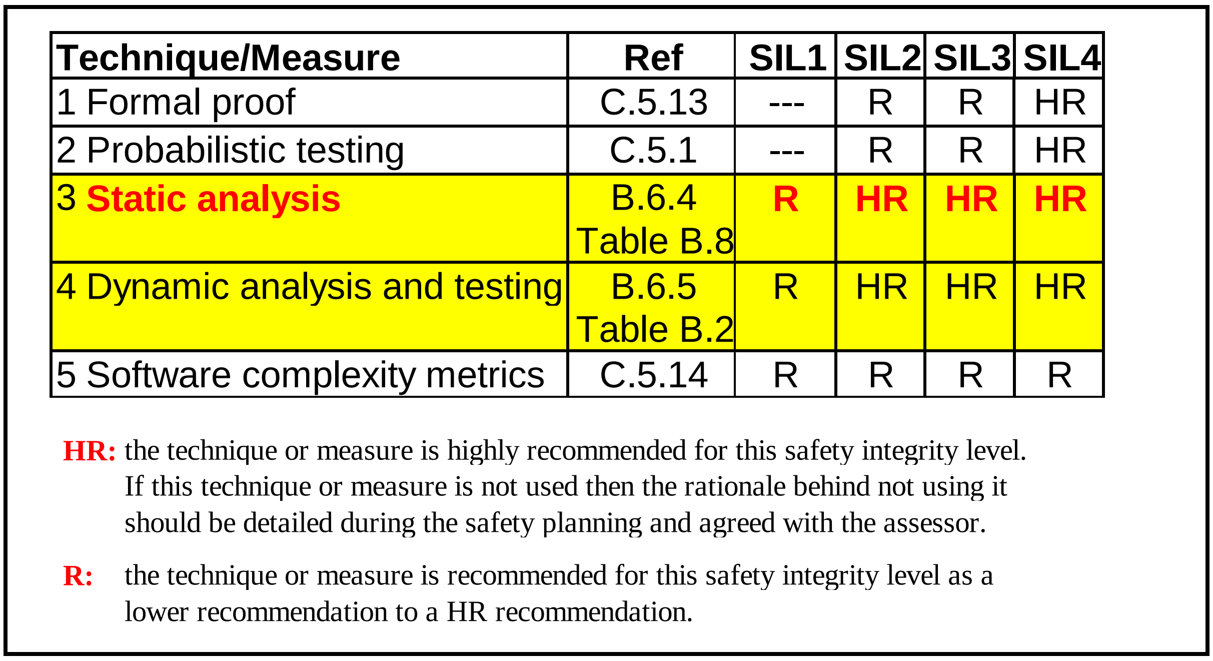
Last step will be assessors evaluation of the arguments given with respect to standards conformance.

Results

The results presented here are not the specific results of the project discussed until now but some general observations.

Reflecting the results from IEC 61508 conformance checklists it may be recognised that products / the producing companies meet about one third to about half of the IEC 61508 requirements without any special preparation. To meet the second half needs additional effort. Therefore it is useful or even necessary to decide about IEC 61508 compliance needs at the beginning of development. To require IEC 61508 compliance after having finished the development in most cases will not lead to good success. Consequence for customers – of control systems as well as of libraries or tools – is that it is not very promising just to buy a component and assume IEC 61508 compliance if no special precautions were taken.

IEC 61508 puts the same emphasis on static analysis as on dynamic testing as may be seen in the following table:



Technique/Measure	**Ref**	**SIL1**	**SIL2**	**SIL3**	**SIL4**
1 Formal proof	C.5.13	---	R	R	HR
2 Probabilistic testing	C.5.1	---	R	R	HR
3 **Static analysis**	B.6.4 Table B.8	**R**	**HR**	**HR**	**HR**
4 Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
5 Software complexity metrics	C.5.14	R	R	R	R

HR: the technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it should be detailed during the safety planning and agreed with the assessor.

R: the technique or measure is recommended for this safety integrity level as a lower recommendation to a HR recommendation.

Figure 4: Table A.9 from IEC 61508-3

Because no static analysis results have been provided by the producers the assessors made their own mind on the code quality by static analysis. As may be well known strength of dynamic testing is

- to assure that required functions – especially frequently needed ones – are well performed.

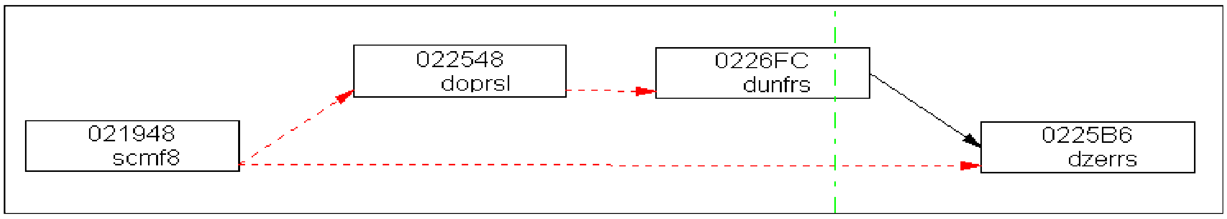
Strength of static analysis is

- to identify code properties which may cause trouble – especially in unusual situations.

Typical situations were safety related systems are really needed are seldom encountered unusual situations. So static analysis is important to assure reliable functioning and avoid crashes just in unusual situations. What has been found by analysis – not just in the project we are talking about here – is that

- routines are not only interconnected via CALLs and RETURNs but as well via JUMPs
- unused interrupts are not served with an error handling routine but may enter just somewhere into some routine
- unused memory is without defined contents instead of well defined statements leading to error handling

Looking at actual code it may be recognised that quality of application software in a lot of cases is better than quality of libraries or other pre-developed components.



- Each box presents a routine.
- Dotted red lines are JUMPs.
- The black line is a CALL.
- Problem: Stack-Balance?

Figure 5: (Typical) part of the calling hierarchy of a library

Conclusions

The project and procedures we presented here – and even a lot of other similar projects – demonstrated that IEC 61508 can practically be applied only with the support of electronic tools, like RiskCAT. This is mainly caused by the immense size of the standard and the high number of requirements. And even the identification of requirements in the text (shall, should, may) is difficult and time consuming.

The use of a tool enables the extraction of requirements (out of the huge number of requirements) that are relevant for the application and directly related to the appropriate SIL. By this the amount of effort can be calculated much more efficient.

Another result of this project is the evidence that a ‚post-development‘ assessment based on IEC 61508 can be conducted with reasonable effort and within predictable time. Real ‚life‘ systems can be developed such that they comply with the requirements of this standard.

References

/CATS 2001/	Code Analyzer Tool Set (CATS); RiskCAT: Requirements Derivation V4.1 from IEC 61508, User's Manual; August 27, 2001
/IEC 61508/	IEC 61508; Functional safety of electrica l/ electronic / programmable electronic safety-related systems; 2000