Solid project lifecycle planning with IEC 61508 software development

By Richard Barry, Wittenstein high integrity systems

IEC 61508 is an international standard covering the development and usage of electrical, electronic and programmable electronic, safety-related systems. In this context, a safety related system is a system that performs one or more safety functions. IEC 61508 covers both hardware and software development, so, with due consideration, safety functions can be performed by software. IEC 61508 is widely used in the industrial automation and control fields, and also forms the basis for a growing selection of safety standards that have been tailored to meet the needs of specific industry sectors. Examples of such tailored standards include IEC 61513 for the nuclear power industry, IEC 61800-5-2 for power drive systems, and IEC 61511 for the process industry.

The standard defines the analysis, design, implementation, production and test requirements for safety-related systems in accordance to the Safety Integrity Level (SIL) assigned to the system. The SIL is assigned according to the risks associated with the use of the system under development, with a maximum SIL of 4 being assigned to systems with the highest perceived risk. The higher the assigned SIL number the lower the rate of failure must be for all identified unsafe failure modes. IEC 61508 comprise seven parts. Figure 1 illustrates the relationship between these parts. Parts 1 to 3 of the standard contain the primary information, with supplementary material provided by parts 4 to 7. As a whole, the seven parts define a system development Safety Lifecycle. The first three parts can be briefly summarised as:

- Part 1 defines the necessary development management system, including how the system safety requirements are to be calculated, elicited and defined.
- Part 2 relates to the hardware aspects of the system development. It contains the techniques required to reduce both systematic and random hardware failures.
- Part 3 relates to the software aspects of the system development. Software cannot suffer random failures, and therefore part 3 contains the techniques required to guard against systematic failures.

Systematic failures are generally not quantifiable, thus part 3 compliance provides a confidence level rather than a calculable failure rate. So who is to say whether or not the confidence level has been reached? To credibly claim compliance with IEC 61508 it is necessary to have your development processes and compliance evidence assessed by a third party who must be accepted as an expert in the field. While there is no body with a legal mandate to issue a de facto compliance certificate, there are several companies which can, as a commercial service, provide an audit and then offer a statement of your compliance (or not as the case may be). A functional safety assessment, and the associated SIL assignment, is applied to a system as a whole. Certification at the component level, whether software or hardware, to a specified SIL does not indicate that the component can be used without regard in a system that has been assigned a SIL of the same value or lower. Any certification evidence supplied by the component provider can only be used as evidence in the context of risks highlighted during the safety assessment of the entire system. The use of 'certified' components does not avoid the requirement to perform a safety assessment on the entire system. Certified components should, however, provide the system builder with a lower cost, faster, and much lower risk route to IEC 61508 compliance.

The techniques and measures described by part 2 with reference to the calculation and reduction of the hardware failure rate are mature and largely well understood. The same is not necessarily true of the part 3 software-related requirements. IEC 61508 is orientated around the ‘V’ development model, a simplistic version of which is depicted in figure 2. While essentially a top-down model, there are plenty of feedback paths permitting a more pragmatic iterative approach, provided the approach taken follows...
the development plans submitted to the third party assessor. The primary path through the V is to specify and develop the design along the downward path of the V whilst simultaneously locking in the verification definition used during the upward leg of the V. The output of the engineering process is documentation and the V dictates the sequence and synchronisation of the documents.

Annex A to part 3 provides ten tables to assist in the selection of techniques and measures to be undertaken during the phases of the V development process. Annex B provides additional and finer detailed information, also in tabular form. The table headings are listed in the sidebar. As can be seen, tables identify items that would typically be expected for any safety-related software development, although the table entry ‘Support tools and programming language’ is noteworthy and given special consideration later in this article. Both the Annex A and Annex B tables contain the six columns depicted in figure 3.

The ‘Technique/Measure’ column is self-explanatory, other than to note that letters following a row number are used to indicate an option. For example, if there is a row 3a and a row 3b, then the developer need only demonstrate compliance with the content of either row rather than with both. The ‘Ref’ column simply provides information on where further information or clarification on the technique or measure can be sourced within the IEC 61508 document itself. The ‘SIL1’, ‘SIL2’, ‘SIL3’ and ‘SIL4’ columns are used to indicate what is required for each of the respective levels. Entered in each of these columns will be one of the four possible values described in the table below.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR</td>
<td>The described technique or measure is Not Recommended for the given SIL. Its use at that SIL is actually discouraged.</td>
</tr>
<tr>
<td>---</td>
<td>There is neither a recommendation for or against the use of the technique or measure.</td>
</tr>
<tr>
<td>R</td>
<td>The use of the technique or measure is Recommended.</td>
</tr>
<tr>
<td>HR</td>
<td>The use of the technique or measure is Highly Recommended. Not to use the technique or measure therefore requires the formulation of a rational for its omission to be submitted to, and accepted by, the conformance assessor during the safety planning process.</td>
</tr>
</tbody>
</table>
As would be expected, the higher the SIL, the higher the number of HR entries encountered. IEC 61508 is not a lightweight standard and a systematic approach is required in order to form a cost-effective compliance argument. Compliance with the items denoted as Highly Recommended (HR) within the annex A and B tables can be unambiguously achieved by extending the tabular format to include a column that cross references each Highly Recommended technique or measure to the appropriate paragraph within the system development planning documentation. Unfortunately not all the IEC 61508 requirements are contained within tables. Sentences within the text that contain the word ‘shall’ also require compliance consideration.

An easy if rather laborious method we have successfully used to demonstrate compliance to these textual requirements is to extract each ‘shall statement’ into a separate compliance matrix, which is then completed during the project planning stages. The need for solid project lifecycle planning has already been highlighted several times. It is not only essential to minimise the commercial risk of the project, but is in fact a requirement of the standard itself. When seeking third party assessment it is essential to submit the development plans to the assessor prior to the commencement of any formal development activities.

Safety-related developments are by their nature costly so pre-approval of the processes to be undertaken prior to commencement can prevent expensive mistakes. It would be expected for a software project development to a high SIL level to include as a minimum a planning documentation (system/safety development plan, software development plan, requirements management plan, configuration management plan, software test plan, standards compliance statement/compliance matrix). IEC 61508 places requirements on the programming language and compiler used. For example, for projects with a high SIL value, the programming language is Highly Recommended to be strongly typed, and the compiler Highly Recommended to either be certified itself, or alternatively have ‘proof in use’ evidence of its correctness available.

Fundamentally we need to demonstrate that our system is safe, so any evidence chosen to demonstrate the correctness of the compiler through proof in use needs to be bullet-proof. It can be argued that the assembly of such evidence requires information on: the number of

---

**Sidebar – Names of the relevant Part 3 tables**

**The 10 Annex A tables are titled:**
1. Software safety requirements specification,
2. Software design and development: Software architecture,
3. Software design and development: Support tools and programming language,
4. Software design and development: Detailed design,
5. Software design and development: Software module testing and integration,
6. Programmable electronics integration (hardware and software),
7. Software safety validation,
8. Modification,
9. Software verification, and

**Annex B tables are titled:**
1. Design and coding standards,
2. Dynamic analysis and testing,
3. Functional and black box testing,
4. Failure analysis,
5. Modelling,
6. Performance testing,
7. Semi-formal methods,
8. Static analysis.
users of the compiler; what these users are compiling and how often and over what period (with exactly the same compiler version); what bugs have been discovered in the compiler; and assurance that all discovered bugs have been reported. To add to the difficulty, IEC 61508 also requires the utilised tools to be supportable for the lifetime of the product (note product, not project).

Credible conformance to these requirements will not be easy, necessitating special consideration during the project planning phase to ensure any alternative approach undertaken is acceptable to the assessor. An approach we have successfully used to demonstrate the reliance of the compiler (while also not using a strongly typed language) relies on a code coverage metric called unique cause modified condition/decision coverage (MCDC). The premise is that we test the compiler output directly against the system requirements in a manner such that it can be argued that all the compiler output is tested and therefore no reliance is being placed on the compiler itself.

The testing should demonstrate that the compiled code performs all the required functions while not performing any unwanted functions. This method tests the projects output (compiled code) directly against the projects input (requirements), demonstrating that the code output by the compiler conforms to the intention of the requirements themselves, without assuming anything about the behaviour of the compiler itself. This approach may sound laborious, but when working with a formalised requirements and testing process anyway, the extra effort required to demonstrate MCDC need not be great, and has the added benefit of being independent of the compiler used.

<table>
<thead>
<tr>
<th>Technique/Measure*</th>
<th>Ref.</th>
<th>SIL1</th>
<th>SIL2</th>
<th>SIL3</th>
<th>SIL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B.2.4</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>2a</td>
<td>Table B.7</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>2b</td>
<td>C.2.4</td>
<td>---</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
</tbody>
</table>

NOTE 1 – The software safety requirements specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.

NOTE 2 – The table reflects additional requirements for specifying the software safety requirements clearly and precisely.

* Appropriate techniques/strategies shall be selected according to the safety integrity level. Alternate or equivalent techniques/strategies are indicated by a letter following the number. Only one of the alternate or equivalent techniques/strategies has to be satisfied.

Figure 3. The six columns illustrated in the Annex A and Annex B table

July 2007