Safety-Critical Systems:
Processes, Standards and Certification

for the Seminar
“Analysis, Design and Implementation of Reliable Software”

Author:
Robert Traussnig
Bonhoefferstrasse 6
D-33332 Gütersloh
count@upb.de

Advisor:
Dr. Holger Giese

Paderborn, January 2004
# Table of Content

1. MOTIVATION ........................................................................................................... 3  
2. INTRODUCTION ........................................................................................................ 3  
3. PROCESSES ............................................................................................................. 4  
   3.1. SOFTWARE CAPABILITY MATURITY MODEL (CMM) ........................................ 5  
4. STANDARDS ............................................................................................................. 7  
   4.1. ISO 9000 ........................................................................................................... 7  
   4.2. IEC 61508 ......................................................................................................... 9  
   4.3. BS EN 50128 RAILWAY INDUSTRY ............................................................... 11  
   4.4. RTCA/DO 178B .............................................................................................. 12  
5. CERTIFICATION ..................................................................................................... 14  
   5.1. DEFINITION .................................................................................................... 14  
   5.2. FORMS OF CERTIFICATION ......................................................................... 14  
   5.3. SYSTEM CERTIFICATION ............................................................................. 15  
   5.4. ASSESSMENT: ISO 15504 SPICE ................................................................. 15  
6. APPLICATION .......................................................................................................... 16  
7. CONCLUSION .......................................................................................................... 16  
8. REFERENCES .......................................................................................................... 17
1. Motivation

Computers are everywhere. However, the vast majority of computers are not the well known desktop or mainframe systems but tiny microprocessors embedded in everyday goods like microwave ovens or washing machines. Computers control the anti-locking brakes in our cars and the flight control systems of aircraft.

The main difference between everyday desktop computers and embedded systems are the consequences of incorrect operation. It might be time-consuming to re-type text after the wordprocessor has crashed, but imagine a crash of the computer that controls the engines of an aircraft. The operation of such a system has direct influence of the safety of its users or the environment and are called safety-critical systems.

This paper discusses the current methods for ensuring that software for safety-critical systems is developed using appropriate processes and standards and can be certified accordingly.

2. Introduction

A safety-critical computer system has to be designed with safety in mind. Identifying and assessing hazards is not enough to make a system safe.

According to Nancy Leveson [Leveson1995] “most accidents are not the result of lack of knowledge about hazards and their causes but of the lack of effective use of that knowledge by the organisation”.

For hardware, general safety design principles have been incorporated into standards, code of practice, and checklists in order to pass on lessons learned from accidents. For the development of safe software, standards are only beginning to emerge. The system hazard analysis identifies software-related safety requirements and constraints, which are used to validate the software requirements.

A safety-critical product can only be certified by an authority, if

- the development process is built on a structured model (e.g. Capability Maturity Model CMM)
- there are standards which have been applied when developing the product

The following sections describe and name standard process models, such as the V-Model and the CMM, a selection of Industry standards and focus and aspects of certification. An example of a civil aviation certification is briefly discussed.
3. Processes

A software engineering process is a framework for the development of a software product. A possible process model is the V-model as shown in Figure 2.

![V Development Model](Giese-Slides II-26)

The model describes the development of a system from the initial requirements analysis and specification phase, followed by design and implementation up to test and certification until servicing of the final product. Frequent iterations between phases of the development (shown in the model as arrows) are necessary to achieve high quality and to build a product that can be verified and certified.
3.1. **Software Capability Maturity Model (CMM)**

The initial Capability Maturity Model (CMM v1.0) was developed by the Software Engineering Institute and specifically addressed software process maturity. It was first released in 1990.

The CMM describes five distinct levels of maturity [CMM] in a staged representation:

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Staged Representation Maturity Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial</td>
</tr>
<tr>
<td>2</td>
<td>Managed</td>
</tr>
<tr>
<td>3</td>
<td>Defined</td>
</tr>
<tr>
<td>4</td>
<td>Quantitatively Managed</td>
</tr>
<tr>
<td>5</td>
<td>Optimizing</td>
</tr>
</tbody>
</table>

![Figure 3. CMM Levels](CMMI, page 33)

1. **Level 1 (initial)** represents a process maturity characterized by unpredictable results. Ad hoc approaches, methods, notations, tools, and reactive management translate into a process dependent predominantly on the skills of the team to succeed.

2. **Level 2 (managed)** represents a process maturity characterized by repeatable project performance. The organization uses foundation disciplines for requirements management; project planning; project monitoring and control; supplier agreement management; product and process quality assurance; configuration management and measurement/analysis. For Level 2, the key process focus is on project-level activities and practices.

3. **Level 3 (defined)** represents a process maturity characterized by improving project performance within an organization. Consistent, cross-project disciplines for Level 2 key process areas are emphasized to establish organization-level activities and practices. Additional organizational process areas include:

   - **Requirements development**: multi-stakeholder requirements evolution.
   - **Technical solution**: evolutionary design and quality engineering.
   - **Product integration**: continuous integration, interface control, change management.
• **Verification**: assessment techniques to ensure that the product is built correctly.

• **Validation**: assessment techniques to ensure that the right product is built.

• **Risk management**: detection, prioritization, and resolution of relevant issues and contingencies.

• **Organizational training**: establishing mechanisms for developing more proficient people.

• **Organizational process focus**: establishing an organizational framework for project process definition.

• **Decision analysis and resolution**: systematic alternative assessment.

• **Organizational process definition**: treatment of process as a persistent, evolving asset of an organization.

• **Integrated project management**: methods for unifying the various teams and stakeholders within a project.

4. **Level 4 (quantitatively managed)** represents a process maturity characterized by improving organizational performance. Historical results for Level 3 projects can be exploited to make trade-offs, with predictable results, among competing dimensions of business performance (cost, quality, timeliness). Additional Level 4 process areas include:

• **Organizational process performance**: setting norms and benchmarks for process performance.

• **Quantitative project management**: executing projects based on statistical quality-control methods.

5. **Level 5 (optimized)** represents a process maturity characterized by rapidly reconfigurable organizational performance as well as quantitative, continuous process improvement. Additional Level 5 process areas include:

• **Causal analysis and resolution**: proactive fault avoidance and best practice reinforcement.

• **Organizational innovation and deployment**: establishing a learning organization that organically adapts and improves.
4. Standards

Standards are required by law or contract during the development and certification of safety critical systems.

In the European Union, the European Committee for Electrotechnical Standardisation (CENELEC) has adopted many standards as European norms (EN)

Issuing bodies for standards are - among others - :

- International Organisation for Standardization (ISO)
- International Electrotechnical Commission (IEC)
- Institute of Electrical and Electronics Engineers (IEEE)
- European Committee for Electrotechnical Standardisation (CENELEC)
- UK British Standards Institute (BS)

The following sections describe four different standards, from the general ISO 9000 standard and the generic IEC 61508 to the industry specific standards EN 50128 (Railway Industry) and DO 178B (Civil Aviation).

4.1. ISO 9000

The International Standardization Organization (ISO) has released the ISO 9000 series of standards that are related to Quality Assurance and Quality Management in general. ISO 9000 requires the organization to “say what it does, do what it says, and be able to demonstrate it”.

The main standards are described in Fig. 4.:

<table>
<thead>
<tr>
<th>ISO Standard</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 9001</td>
<td>Model for quality assurance in design,</td>
</tr>
<tr>
<td></td>
<td>development, production, installation,</td>
</tr>
<tr>
<td></td>
<td>servicing</td>
</tr>
<tr>
<td>ISO 9002</td>
<td>Model for quality assurance in production,</td>
</tr>
<tr>
<td></td>
<td>installation and servicing</td>
</tr>
<tr>
<td>ISO 9003</td>
<td>Model for quality assurance in final</td>
</tr>
<tr>
<td></td>
<td>inspection and test</td>
</tr>
</tbody>
</table>

**Figure 4. ISO 9000 Family of Standards**

**ISO 9001** focuses on product conformity to international standards throughout the product lifecycle. Emphasis is put on the design element and performance factors. This standard is the most stringent of the ISO 9000 series.

**ISO 9002** does not include a design or R&D component. It is focused on the production, installation and servicing of products.
ISO 9003 covers product inspection and testing of ready-made components

For the development of software important is the ISO 9000-3 „Guide to the application of ISO 9001 to the development, supply and maintenance of software“. The standard was released in 1991 with a second edition published in 1997. Three main areas are covered, as described by Debra Hermann [Hermann1999]:

- **The quality framework:**
  - Management responsibilities
  - the quality system
  - audits
  - corrective action

- **Lifecycle activities:** Actions that are required during the various lifecycle phases, namely
  - contract review
  - requirements
  - development planning
  - quality planning
  - desing and implementation
  - testing and validation
  - acceptance
  - replication
  - delivery and installation

- **Supporting activities:** Actions that are not related to specific lifecycle phases, such as:
  - configuration management
  - document control
  - measurement
  - tools support
  - purchasing
  - training
4.2. IEC 61508

IEC 61508 is a standard developed by the IEC (International Electrotechnical Commission), a worldwide organization consisting of IEC National Committees in more than 60 countries of the world.

IEC prepares and publishes international standards for all electrical, electronic and related technologies. These serve as a basis for national standardization and as reference when drafting international contracts.

IEC 61508, titled "Functional safety of electrical/electronic/programmable electronic safety-related (E/E/PE) systems" is a generic standard and consists of 7 parts [IEC 61508]:

- **IEC 61508-1**: General Requirements
- **IEC 61508-2**: Requirements for electrical / electronic / programmable electronic safety-related systems
- **IEC 61508-3**: Software Requirements
- **IEC 61508-4**: Definitions and abbreviations
- **IEC 61508-5**: Examples of methods for the determination of safety integrity levels
- **IEC 61508-6**: Guidelines on the application of IEC 61508-2 and IEC 61508-3
- **IEC 61508-7**: Overview of techniques and measures

The standard aims to:

- release the potential of E/E/PE technology to improve both safety and economic performance
- enable technological developments to take place within an overall safety framework
- provide a technically sound, system based approach, with sufficient flexibility for the future
- provide a risk-based approach for determining the required performance of safety-related systems
- provide a generic standard that can be used directly by industry but can also help with developing sector standards (e.g. chemical plants, medical, or railway) or product standards (e.g. drive-by-wire)
• provide a means for users and regulators to gain confidence when using computer-based technology

• provide requirements based on common underlying principles to facilitate:
  o improved efficiencies in the supply chain for suppliers of subsystems and components to various sectors
  o improvements in communication and requirements (i.e. to increase clarity of what needs to be specified)
  o the development of conformity assessment services

**IEC 1508 Safety Life Cycle**

**Fig. 5. Overall Safety Lifecycle IEC 61508-1**

Figure 5. shows the overall Safety Lifecycle as described in IEC 61508-1 (graphical representation from [Giese-Slides II-28]).

The standard requires specific tasks to perform during the product lifecycle of a safety critical system:

**Step 3: Hazard and risk analysis**

The objective is to determine hazards of the EUC (Equipment Under Control) in operation, including misuse of the system.

**Step 5: Safety requirements allocation**

The allocation of safety functions to the E/E/PE system, including the safety integrity levels.
Step 13: Overall safety validation

The objective is to validate that the E/E/PE safety-related system meets the specification for the overall safety requirements.

Step 16: Decommissioning

Must ensure that the functional safety of the system is guaranteed during the decommissioning or disposal.

IEC 61508 does not cover the precautions that may be necessary to prevent unauthorized persons damaging or negatively affecting the functional safety of E/E/PE safety-related systems.

4.3. BS EN 50128 Railway Industry

EN 50128 was developed to identify “methods which need to be used in order to provide software which meets the demands of safety and integrity” [Herman]

The standards were adopted for the following reasons:

- to define a process for the specification and demonstration of dependability requirements for the railway industry
- to promote a common understanding and approach to the management of dependability
- to provide Railway Authorities and the railway support industry, throughout the European Community, with a process which will enable the implementation of a consistent approach to the management of Reliability, Availability, Maintainability and Safety (RAMS)

EN 50128 introduces software integrity levels (SILs), where each level is associated with a degree of risk when using the software system [EN 50128].

<table>
<thead>
<tr>
<th>SIL Level</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nonsafety related</td>
</tr>
<tr>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Medium</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Very High</td>
</tr>
</tbody>
</table>

Figure 6. SIL Levels [CMM]

EN 50126 defines a detailed risk classification scheme which utilizes a combination of qualitative and quantitative measures. It consists of six probability levels:

Incredible: extremely unlikely to occur
Improbable: unlikely to occur but possible
Remote: likely to occur at some time
Occasional: likely to occur several times
Probable: will occur several times
Frequent: likely to occur frequently

4.4. RTCA/DO 178B

The Requirements and Technical Concepts for Aviation (RTCA) DO 178B is officially a “guidance document” but widely accepted as an international standard. It was first issued in the United States in the 1980s and relates to civil aircraft. The last released version is a major revision from 1992. The standard is compatible with the European Organization for Civil Aviation Electronics (EUROCAE) standard ED-12B.

The intent of DO-178B is to describe the objectives of software life-cycle processes, describe the process activities, and to describe the evidence of compliance required at different software levels. The software levels are chosen by determining the severity of failure conditions on the aircraft and its occupants [DO-178B].

The verification objectives for DO-178B are set in place to detect and report errors that may have been introduced during the software development processes. Software verification objectives are satisfied through a combination of reviews and analyses, the development of test cases and procedures, and the subsequent execution of those test procedures.

Reviews and analysis provide an assessment of the accuracy, completeness, and verifiability of the software requirements, software architecture, and source code.

Most software code is written in a high level language such as C, C++ or Ada, and the coverage achieved by any given test is usually measured against high-level source code (also referred to as Structural Coverage). The development of test cases may provide further assessment of the internal consistency and completeness of the requirements. The execution of the test procedures provides a demonstration of compliance with the requirements. Software test cases should be based primarily on the software requirements and developed to reveal potential errors.

Software coverage analysis* is used to determine which requirements were not tested. This is supported by the structural coverage analysis objectives required by DO-178B that are intended to determine what software structures (e.g. statements or decisions) were not exercised as a result of these verification activities. This, in turn, reveals requirements that may have been in error, tests that were lacking adequate coverage for these structures, or dead code. Structural coverage analysis is performed to the degree required by the criticality of the software.

Structural coverage analysis may be performed on the source code; unless the software level is A and the compiler generates object code that is not directly traceable to source code statements.

Additional verification should then be performed on the object code to establish the correctness of such generated code sequences.
Modified Condition/Decision Coverage (MC/DC) - Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken on all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision’s outcome.

Decision Coverage (DC) - Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.

Statement Coverage (SC) - Every statement in the program has been invoked at least once.

The failure conditions are named, and have corresponding levels which are identified by letters. For each level there is a set of process objectives which must be satisfied. An example of a process objective is A-7.3 “Test Coverage of low-level requirements is achieved.”

It identifies six processes: software planning, software development, software verification, software configuration management, software quality assurance and software certification.
5. Certification

5.1. Definition

“Certification is the process of issuing a certificate to indicate conformance with a standard, a set of guidelines, or some similar document.”

[Storey1996]

Storey points out that “any organization or individual may issue a certificate, and its importance will clearly vary with its nature and its issuing body”. In many cases such a certificate will be issued in the form of a license from some authority. Such authorities are often government bodies, e.g. the U.S. Federal Aviation Administration (FAA) which approves all civil aircraft systems in the USA.

According to [Giese] the aims of certification are:

- To improve the safety of critical systems
- To increase the awareness of the implications of system performance on safety
- To enforce minimum standards of design and manufacture within the relevant industry
- to encourage a structure of professional responsibility

5.2. Forms of Certification

Certificates can be issued for

- the appropriate management structure in the organization
- the processes and procedures used to develop products
- standards
- the final product
- individuals like user or operator of a system
- experts, who themselves can issue certificates e.g. Designated Engineering Representative (DER) who perform certification tasks for the FAA

The certification process can vary greatly in different industries and countries. Storey compares the required compulsory certification of medical electronics in the U.K. with the voluntary certification of the same equipment in Germany or the U.S.
5.3. System Certification

For safety-critical systems, the certification process starts with the initial requirements analysis. Usually, a certification liaison [Storey] person acts as the main communicator between the manufacturer of the system and the certifying authority.

Where specific standards are mandatory, the liaison will make sure that these are incorporated and documented from the very beginning of the project. A verification plan will be produced for approval by the authority. During the project phases, appropriate documentation and supporting data will be produced and submitted to the authority. Usually, a series of reviews will be conducted and if all terms of the verification plan are met, the certificate or license issued.

5.4. Assessment: ISO 15504 SPICE

ISO 15504 SPICE (Software Process Improvement and Capability Determination) combines these two approaches (CMM and ISO 9000) into a single mechanism for developing a quality system in software. It embodies the reference framework that is a part of the ISO 9000 approach with the capability assessment and process maturity features of the CMM. In addition it establishes a migration path for existing assessment models and methods.

The aim of the ISO 15504 standard is to perform process assessment, process improvement, and capability determinations. The overall aim of the 15504 Standard is to… "Improve product quality through proven, consistent and reliable assessment of the state of an organization' software process and the employment of the results of these assessments as part of coherent improvement programs" (ISO 15504, 1998).

The intent of the overall ISO 15504 initiative is to embrace and encourage principles and practices that have been proven to be effective in software organizations.

The architecture of the standard is carefully designed to permit and encourage the development of methods and models, which serve specific domains or markets. Software process domains assessed by ISO 15504 are, Acquisition, Supply, Development, Operations, Maintenance, Supporting processes and Service support.

But, ISO 15504 intentionally does not specify a particular assessment methodology. It does impose certain requirements for any assessment process that would claim to be conformant with the Standard. It accomplishes this through an example assessment model (ISO 15504-5). This model is intended to demonstrate how an assessment process that satisfies the requirements of the Standard could be constructed.
6. Application

Example: FAA Certification of On-Board Computer:

Required sequence of action for a STC (Supplemental Type Certification) of a safety-critical aircraft computer system using Level B software by the U.S. FAA (Federal Aviation Administration) based on DO-178:

- Familiarization Meeting
- Formal Application
- DER Designation
- Preliminary Type Certification Board
- Certification Program Plan (CPP)
- Technical Meeting
- Pre-Flight Supplemental Type Certification (STC) Board
- Type Inspection Authorization (TIA)
- Conformity Inspections and Certification Flight Tests
- Aircraft Evaluation Group (AEG)
- Final Type Certification Board
- Supplemental Type Certificate (STC)
- Post Certification Activities

This lengthy process ensures coverage of all safety-critical aspects of the system. DERs (Designated Engineering Representatives) provide expertise and guidance through the process and work closely with the developer and the FAA.

7. Conclusion

The development of safety-critical software systems requires the introduction of mature development process into the organization as well as the use of acknowledged standards. Certification of such a system relies heavily on the ability of the organization to demonstrate and document proof of the correct application of these processes and standards.

As software development matures over time, we will see the propagation of these processes and standards into the development of non-safety critical applications in the near future.
8. References


