

Matti Vuori, Heikki Virtanen, Johannes Koskinen
& Mika Katara

Safety Process Patterns in the Context of IEC 61508-3



Tampereen teknillinen yliopisto. Ohjelmistotekniikan laitos. Raportti 15
Tampere University of Technology. Department of Software Systems. Report 15

Matti Vuori, Heikki Virtanen, Johannes Koskinen & Mika Katara

Safety Process Patterns in the Context of IEC 61508-3

Tampere University of Technology. Department of Software Systems
Tampere 2011

ISBN 978-952-15-2596-4

ISSN 1797-836X

Contents

Foreword	6
Part I: Description of the pattern collection	7
1 Some background on patterns in software development	7
2 The purpose of safety process pattern collection	8
3 Qualities of a good pattern collection	8
4 Context for the patterns	10
4.1 Process context	10
4.2 Knowledge and culture related context	13
5 Structure and contents of a safety process pattern	14
6 Structure of the pattern collection	15
7 Patterns included in the collection	16
7.1 List of included patterns	16
7.2 Visual views to the pattern collection	19
7.2.1 Generic organisational patterns	20
7.2.2 Generic process and product control patterns	20
7.2.3 Software Safety Requirements Specification.....	21
7.2.4 Software Design & Development	21
7.2.5 Software Aspects of System Safety Validation	22
7.2.6 Software Modification.....	22
7.2.7 Functional Safety Assessment.....	22
7.2.8 Software Operation & Maintenance Procedures.....	22
7.3 Examples.....	23
7.3.1 Phase Workflow	23
7.3.2 Assign Roles and Responsibilities	25
7.3.3 Software Validation Planning	27
8 References	30
Part II: The Safety Process Pattern Collection	31
DISCLAIMER	31
1 Generic organisational patterns	32
1.1.1 Multiple Viewpoints	32
1.1.2 Understand Cultures in Co-operation	33
1.1.3 Assign Roles and Responsibilities	35
1.1.4 Diversity in Team Practices.....	37
1.1.5 Competence Management.....	39

1.1.6	Continuous Communication	40
1.1.7	Transparency of Action and Information.....	42
1.1.8	Anti-pattern: Information Hiding.....	43
2	Generic process and product control patterns.....	45
2.1.1	Phase Workflow	45
2.1.2	Verification of a Work Product.....	47
2.1.3	Split and Manage Details	49
2.1.4	Single Development Task Control Workflow	50
2.1.5	Acceptance of Phases and Tasks	51
2.1.6	Configuration Management.....	53
2.1.7	Forward Tracing.....	55
2.1.8	Backward Tracing	57
2.1.9	Suspect and Prohibit.....	59
2.1.10	Escalation of Issues	61
2.1.11	Use of Checklists	62
2.1.12	Continuous Improvement	65
3	Development approaches and technologies.....	67
3.1.1	Flow Between Design Levels and Tests.....	67
3.1.2	Selection of Methods / Techniques	69
3.1.3	Use of Formal Methods.....	71
3.1.4	Selection of Support Tools and Development Languages	73
4	Software Safety Requirements Specification.....	75
4.1.1	Software Safety Requirements Specification.....	75
5	Software Design & Development	78
5.1	General.....	78
5.1.1	Software Development.....	78
5.2	Software Architecture Design & Verification.....	80
5.2.1	Software Architecture Design.....	80
5.2.2	Software Architecture Verification	82
5.2.3	Technical Diversity.....	84
5.2.4	Formal Methods Aided Design and Verification of Joint Behaviour	86
5.3	Software System Design.....	87
5.3.1	Software System Design – general	87
5.3.2	Software System Design Verification	89
5.3.3	Generic Glue.....	91
5.4	Module Design and Implementation.....	93
5.4.1	Detailed Module Design.....	93

5.4.2	Glue Design and Implementation	95
5.4.3	Coding	96
5.4.4	Analytic Design and Code Quality Assessment.....	98
5.5	Verification Testing	100
5.5.1	Verification Testing	100
5.5.2	Module Testing and Simulation	102
5.5.3	Module Integration Testing.....	104
5.5.4	PE Integration Testing.....	106
5.5.5	Regression Testing	109
5.5.6	Model-Based Testing	111
6	Software Aspects of System Safety Validation.....	112
6.1.1	Software Validation Planning	112
6.1.2	Software Validation	114
6.1.3	Configuration Auditing.....	116
7	Software Modification.....	118
7.1.1	Software Modification Planning	118
7.1.2	Software Modification.....	119
7.1.3	Impact Analysis.....	121
8	Functional Safety Assessment	123
8.1.1	Functional Safety Assessment.....	123
8.1.2	Failure Analysis.....	125
9	Software Operation & Maintenance Procedures.....	127
9.1.1	Writing of the Safety Manual	127

Foreword

Standards can be difficult to comprehend and to implement in practice. This is due to many factors, such as the generic nature of standards in using concepts and vocabulary of any particular context and also the specific nature of the standards, which makes them refer to and acknowledge only the issues that they have been authorised to tackle – the idea being that there are other standards for other issues.

Safety-related standards can thus be difficult to grasp and the IEC 61508 series is no exception. While one expert in a company may have the time and capability to fully understand the standard, it needs to be communicated to others so that it is practiced in projects and other day-to-day activities. Some external help is clearly required. Training is one route, and even it needs more understandable descriptions so as to communicate the issues.

A process pattern is a concept that aims to present important aspects of an activity with a modular expression that can become familiar to personnel. In fact, the pattern descriptions highly resemble the description used in many companies, such as:

- Process description cards used as instructions.
- Templates of use cases used in software development.

Therefore, in the Ohjelmaturva project we have done research on the use of safety process patterns to help in utilising the IEC 61508 standard series (2nd edition) and especially its third part (IEC 561508-3 2nd ed.) which concerns software development. This report presents a) some ideas behind the patterns aiming to give guidance to future pattern developers and b) a preliminary pattern collection.

The patterns presented in this reports do not form a complete collection of all necessary patters, nor do they cover all aspects of the standards, but present a view to the standards that in our opinion does not have conflicts with the standards and can greatly aid in their understanding and utilisation.

Note that this report mostly addresses issues of the traditional V-model based development. For an analysis of how the standards' requirements could be fulfilled in an agile development process, see the sister publication to this report, "Agile Development of Safety-Critical Software" (Vuori, 2011).

Finally, the authors would acknowledge partial funding from Tekes and the following companies participating in the Ohjelmaturva project: ABB, Bronto Skylift, EPEC, John Deere Forestry, Konecranes, Metso, Safety Advisor, Sandvik Mining and Construction, and Sundcon.

Part I: Description of the pattern collection

1 Some background on patterns in software development

Patterns are recurring structures or relationships between elements. The concept is used in trying to understand and share the understanding of complex phenomena both in humans' actions and in technological systems. They are developed by examining an existing or described activity and detecting the pattern by analysing. Patterns use a concise "pattern" language that describes the defining elements of the pattern in a generic form. The elements include things like name, context, solution, resulting context and other information.

Patterns have been developed for many purposes since the 1990's:

- Organisational patterns have been developed to make organisational structures and behaviour visible, also in software development organisations, including agile development (Organizational patterns. Wikipedia article).
- Software design patterns have advanced understanding of software design and architectures (Category: Software design patterns. Wikipedia article).
- Use cases are a very important usage of the pattern philosophy (Use case. Wikipedia article). A use case is a very recurring element in every software development project – many of those are identified and presented in a standard way.
- Process patterns capture among other things, software development issues (Ambler, 2011 has built a nice web site around those).
- Project patterns research has included studies of global software development projects (Välimäki, Kääriäinen & Koskimies, 2009).
- Communication and knowledge sharing in the context of software engineering (see Vesiluoma 2009 whose dissertation also contains plenty of information about patterns).

Thus, patterns have evolved into a proven tool to understanding a domain's activity and issues and to externalise and share knowledge.

Patterns use a concise presentation consisting of short descriptions of key elements. This same principle has been utilised in many organisations as a template process instruction, making instructions to have a generic, standardised form, short length (optimally one page) and thus better understandability than traditional, longer instructions. Thus, patterns have a lot of potential to be used in companies' processes.

2 The purpose of safety process pattern collection

The goal is to help organisations make better product development and safer systems, as, with the help of the patterns, they could 1) understand better the standards' requirements and 2) understand how the requirements show in practical software development work.

The patterns can be used for many purposes:

- Description of development processes & safety lifecycle processes, in an understandable way.
- Frequently Asked Questions – they should provide answers to common needs of understanding how a process or task should be carried out.
- Presentation of generic workflows to be used as design & tailoring of workflows in companies and in evaluating current practices.
- Explaining informal appendix to the official standards.
- A basis for an organisation's understanding of its own activities and a description of those, to enable communication, training and development of practices.

It should be noted that the patterns are just models and should *not* contain all the details of the safety standards, but an overview of them and links to the relevant parts of the standards. If the patterns aimed at completeness, organisations would trust too much on them and neglect understanding of the actual standards.

3 Qualities of a good pattern collection

We understand that a good pattern collection has these qualities:

- It is practical.
- It describes the whole, the context, gives orientation (metapatterns).
- It includes patterns that describe parts of the overall process.
- It includes patterns that show relevant principles and modes of action that are applied in all process phases. (Those compare with generic requirements in CMMI).
- It is simple and easy to understand for all process participants (developers, safety experts, managers) at many competence levels – not just university graduates. This means that the terminology needs to be general and charts should use a generic notation and understanding of them should not require understanding of for example UML diagrams.
- It is a simplified presentation of the reality, concentrating on key issues. The descriptions, including charts can be simplifications.
- It provides a shared view to the development, including things that have meaning for all participants, occupational groups and stakeholders.
- Thus it can be used in training – both in general training and in initiation into a project's practices.

- It is concise and short. (Even the standards emphasise short documentation in safety matters.)
- It is modular. Patterns should be as independent as possible.
- It is reasonably independent of development lifecycle used. The patterns can be implemented in any development process with minimal tailoring, providing a solid backing for process development.
- It presents itself in such a way that its informal nature is very clear to everyone.

The patterns can be of various types:

- Generic patterns penetrate the whole process and are applied in many process phases / sub-processes and tasks.
- Task patterns are used for a single purpose, showing the process flows.
- A viewpoint pattern can show a specific viewpoint – such as how modelling can be used in a task.
- An anti-pattern: a pattern that should not be used. If its use is detected, it should be stopped. An anti-pattern may be thought of as a good idea, but with some analysis will be understood to be harmful and should be replaced with a better pattern.

A good pattern collection should have variety – just as the life and activity in an organisation does. That is why our collection has patterns of various types.

4 Context for the patterns

4.1 Process context

The context for the patterns presented is formed by the two views that IEC 61508-3 presents into the software system development: a) the software system safety lifecycles for the overall system, the E/E/PE system and the software system and b) the V model that defines the work flow of the actual development work.

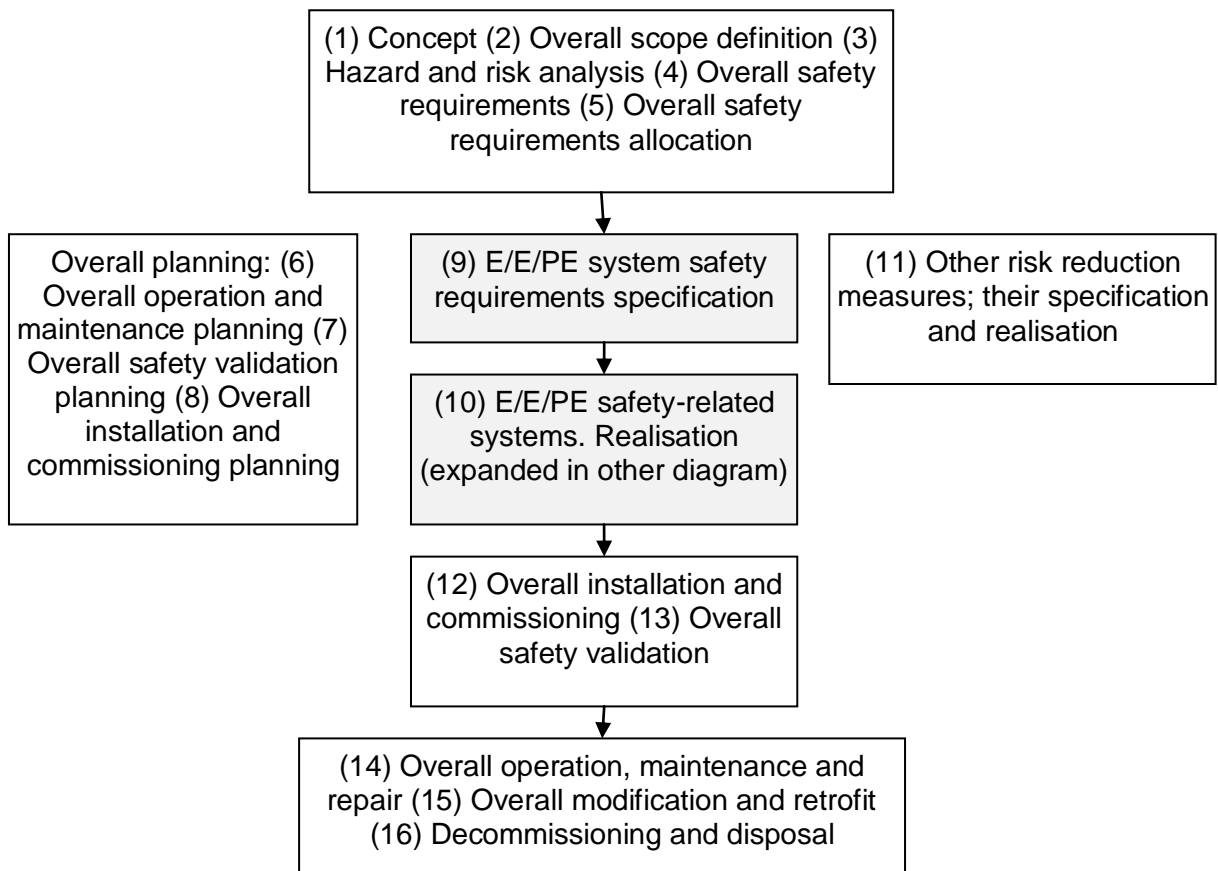


Figure 1. The overall safety lifecycle. A simplified version of figure 2 of IEC 61508-3 (2nd ed.) to show the role of software development.

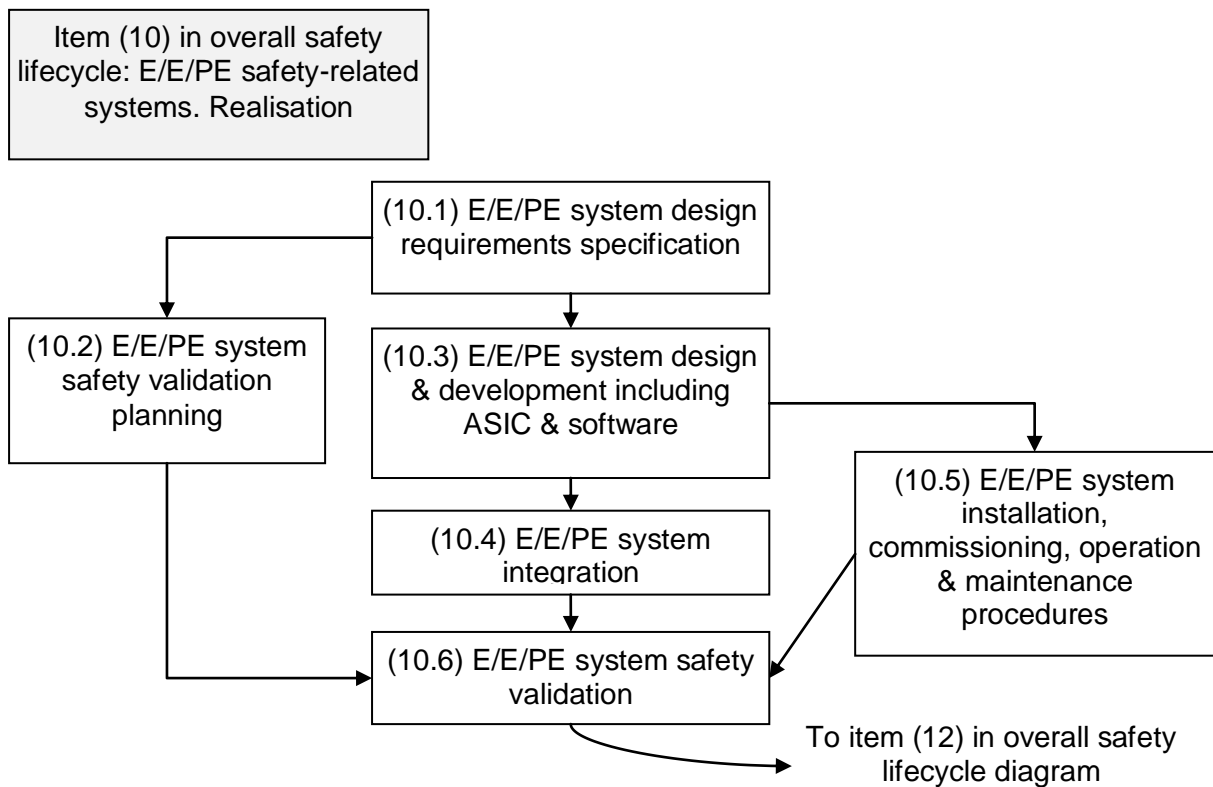


Figure 2. The E/E/PE system safety lifecycle. Expanded box 10 of the overall system safety lifecycle diagram. Redrawn from figure 3 of IEC 61508-3 (2nd ed.).

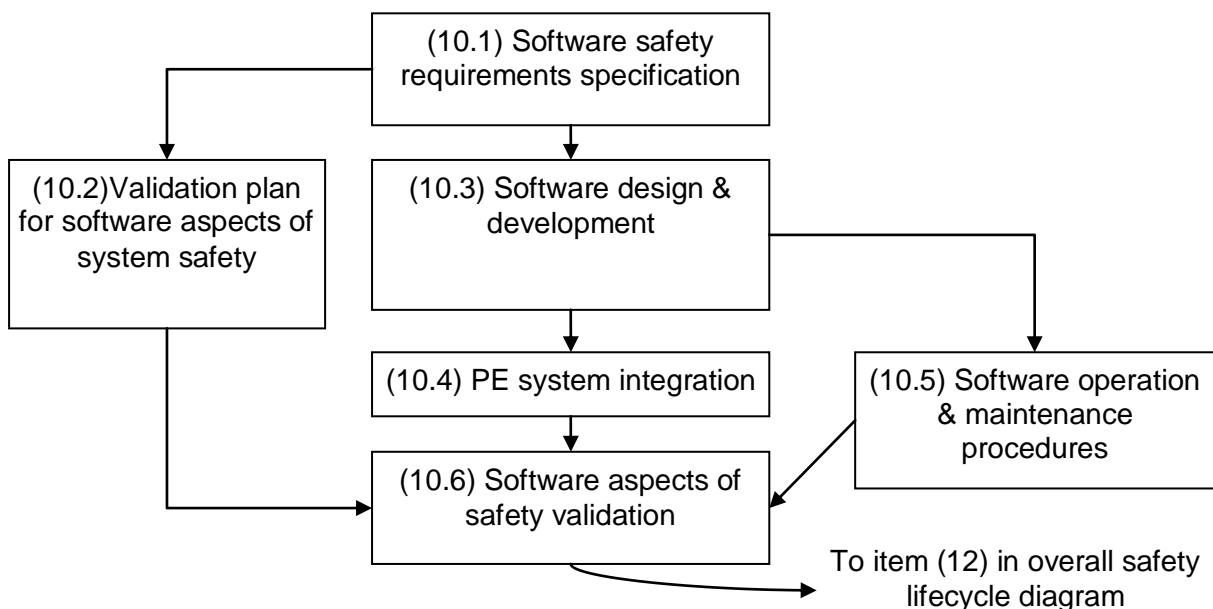


Figure 3. The software safety lifecycle. A part of the E/E/PE system safety lifecycle and thus also part of box 10 of the overall system safety lifecycle diagram. Redrawn from figure 4 of IEC 61508-3 (2nd ed.).

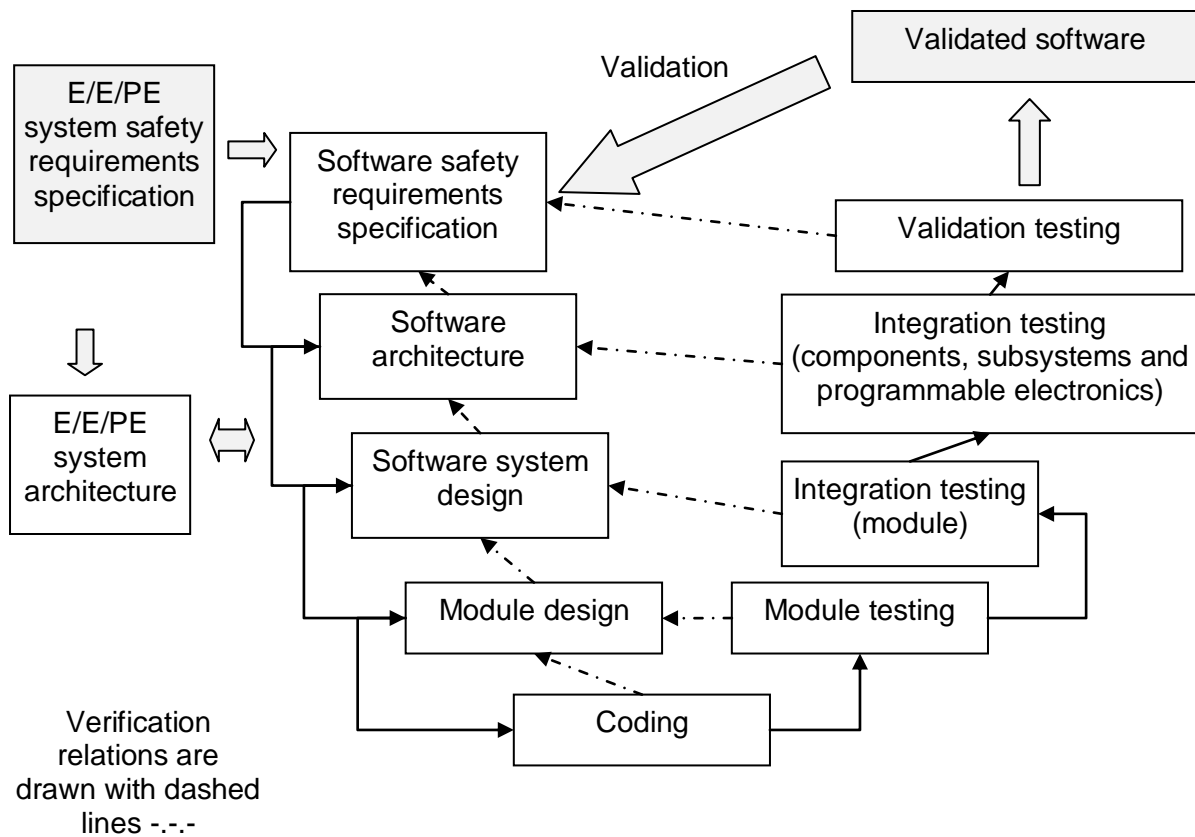


Figure 4. The V model of the system developed. Redrawn from figure 6 of IEC 61508-3 (2nd ed.).

As we aim to support practical software development, the main outline of patterns will consist of elements of the V model, supplemented with items from the safety lifecycle models.

For an analysis of how the standard's requirements could be fulfilled in an agile development process, see the sister publication to this report (Vuori et al. 2011)

4.2 Knowledge and culture related context

The following figure shows the context of this pattern work from the point of view of knowledge and culture. We aim to provide information on the safety standards so that it supports good software engineering principles, good safety management and quality management, in a form that suits product development practices and development cultures in companies

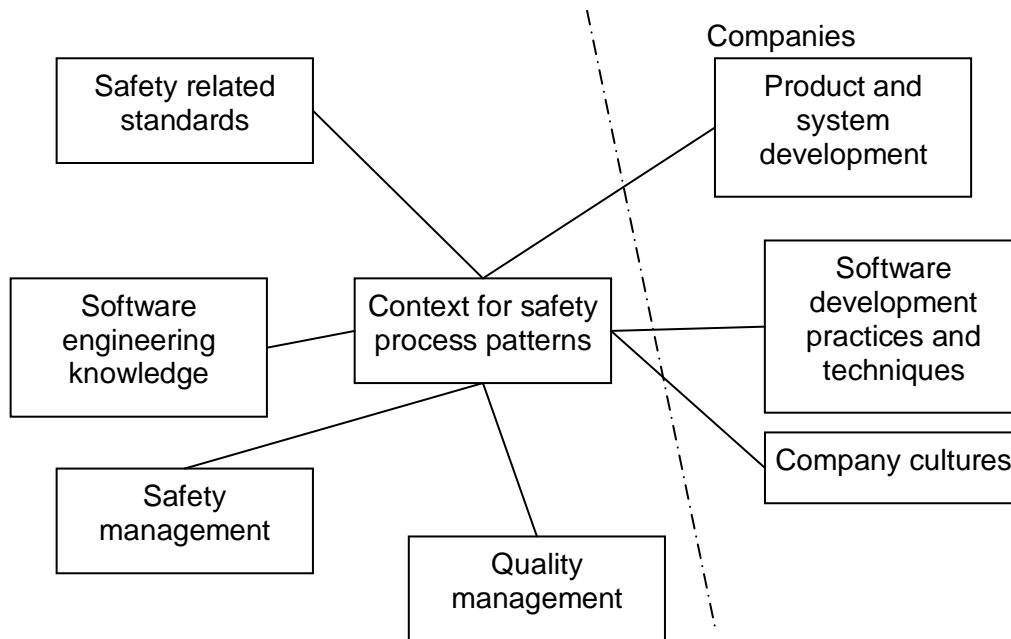


Figure 5. Knowledge and culture related context for the patterns.

5 Structure and contents of a safety process pattern

The patterns use a standardized structure, based on the one described by Koskinen & Katara (2011). It is presented in Table 1.

Table 1. Structure of a safety process pattern.

Element	Description
Name	An unique name for the pattern. Describes what the pattern is a about.
Context	The context describes the initial situation where the pattern is thought to be applied (i.e. where the steps should be performed). It may also have preconditions or requirements that have to be fulfilled before the pattern can be applied. Such a requirement could include, among others, a reference to some other pattern.
Problem	The patterns try to solve a problem, which is described in this section. In our patterns, the problems are usually given in the form “How to...”.
Forces	The forces section presents the reasons to apply the pattern. The forces do not discuss the solution, but usually after defining the forces, the solution is more or less obvious and easy to adopt.
Solution	The solution defines the steps that should be followed to solve the problem. The steps cover the requirements defined in the standard. For example, if the standard requires documentation to be written, the solution will have a step corresponding to that requirement. The solution should cover the forces defined earlier in the forces section. There is usually a picture – a diagram, picture of a process flow, a mind map or similar.
Resulting Context	Resulting context is the new context which is achieved after the pattern has been applied. It describes what has been achieved by applying the pattern.
Related Patterns	References to other patterns that are related to this pattern. Often patterns whose execution precedes this pattern or starts after this pattern; patterns that are similar to this one or ones that are more detailed implementations of a more abstract pattern.
Standard References	References to the clauses in IEC 61508 standard series that explain the elements of the pattern or give requirements for its execution.
Authors	Who has written the pattern.
Status	During development, acceptance status. During the rest of the pattern lifecycle, update information.
Notes	Freeform notes and links to more information, for example to Wikipedia pages.
Tags	Classification tags. Any number of tags that help in classifying the pattern.

Some notes on the elements of the pattern template:

The elements, or fields, "standard references", "authors", "notes" and "tags" are not usually found in patterns found in literature, but extensions important in practical long-term usage of the patterns.

The patterns found in literature often include an element called "rationale" or "justification". Sometimes the justification can be divided into parts, like for example Vesiluoma (2009) "basic idea", "positive instance" (when the pattern is especially valuable) and "negative instance" (when the pattern might be unsuitable). In our case the rationale results from requirements of the IEC 61508 standard series and thus there are not many alternatives. Still, if future research assesses the patterns in different contexts – like various software lifecycle models – these applicability issues should be revisited.

But the number of elements should be kept as small as possible. Adding more elements will at some point make the patterns worse, not better. Therefore, the information regarding applicability will be included in the "notes" field, as needed.

6 Structure of the pattern collection

The patterns can belong into the following main types:

- Generic organisational patterns – generic principles and practices in an organisation in projects and in its overall activities and processes.
- Generic process and product control patterns. Patterns that are repeated during a project many times and are implemented during the development lifecycle or the safety lifecycle many times. These are process issues that provide a solid context for development.
- Development approaches and technologies. These include technology choices, etc. that are applied in the context of the control patterns and during the lifecycle and process patterns.
- Patterns for individual phases of the safety lifecycle or the software development lifecycle. These may be carried out only once in a project.

In addition to this classification, patterns have “tags” assigned to them which can be used to create classification dynamically – for example, combining all the patterns into one view that have a tag “verification” assigned to them. Some things that tags can express:

- Development process phase and task.
- Type of task, like design or verification.
- An approach, like automation or formal methods.
- A specific tool or technique, like FMEA, test-driven design or Application Lifecycle Management System.
- Links to company-specific aspects.

Thus, the tags can be used to create dynamic views of the patterns for example in a company's information system.

The structures of the pattern collection are shown in two ways: the document's structure follows the primary classification and is reflected in the table of contents.

In addition to that there is a graphical view that is based on the most relevant tags. Drawing of various "maps" of the pattern collection is best done for a purpose in a context and complementing the patterns with more content (the organisation's own principles, processes, patterns) and thus we encourage the readers to think how the patterns would fit their own context.

7 Patterns included in the collection

7.1 List of included patterns

The following patterns are included in the collection introduced in this report. The subheadings starting from "Software Safety Requirements Specification" correspond with chapters of IEC 61508-3 (2nd ed.).

Table 2. The patterns introduced in this collection.

Name	Problem
• Generic organisational patterns	
Multiple Viewpoints	How to identify the viewpoints that project participants should represent in the project?
Understand Cultures in Co-operation	How can we co-operate in a multi-cultural project so that cultural differences are managed so that they will not endanger safety?
Assign Roles and Responsibilities	How can we select project participants and assign roles and responsibilities so that utilisation of expertise and required independence are in an optimal balance?
Diversity in Team Practices	How to apply the principle of diversity in all tasks?
Competence Management	How can we ensure that each member has the required competence?
Continuous Communication	How can we get rapid input from everyone in such a way that it doesn't make progress slower, but makes the project proceed more efficiently? How can we get busy professionals to participate in the process?
Transparency of Action and Information	How can we know what is actually being done and whether there are any problems?
Anti-pattern: Information Hiding	We know that we have problems and are ashamed of it. How can we hide the situation until a miracle happens and the problem is solved? How can we suppress information so that it will not be leaked to competitors or media?
• Generic process and product control patterns	
Phase Workflow	How is a process phase carried out, satisfying safety lifecycle process requirements?
Verification of a Work Product	How to verify that a work product meets its requirements.
Split and Manage Details	When a work product, like a module or a specification, is being developed, how can we ensure that work can be carried on in parallel, so that we can address issues independently and verify each small task and see the state of the whole?
Single Development Task Control Workflow	How to have a simple, generic workflow that allows tracking of the completion of single tasks and progress of a set of tasks?
Acceptance of Phases and Tasks	How are work products accepted in the development process so that they can be safely utilized?

Name	Problem
Configuration Management	How can we know what elements, in what configuration and in what version the software system assessed consists of and what has changed compared with a baseline?
Forward Tracing	How can we link activities and development items so that we can at any time find out what actions are planned and have been carried out regarding the items? How can we follow the chain from any requirement to its verification and testing in an easy way? If some aspect of the work product changes, how can we know what items in the following process phases are invalidated due to that change?
Backward Tracing	How can we know what requirements, plans or instructions our work is based on and thus must be verified against?
Suspect and Prohibit	How do support practical suspicion and convert it into practical action? How can we know what – if anything – should be changed based on our suspicion?
Escalation of Issues	How to raise the issue and have it handled at an appropriate level in the project or line organization?
Use of Checklists	How can we remember all issues that need to be checked? How can we be sure that others remember all issues that need to be checked?
Continuous Improvement	How to improve ways of action so that in future projects are more efficient and have fewer problems?
<ul style="list-style-type: none"> • Development approaches and technologies 	
Flow Between Design Levels and Tests	While utilising a controlled approach, how to support a constant flow of testing ideas?
Selection of Methods / Techniques	How to select methods and techniques so that the decision leads to such ways of action that lead to a safe system, fulfil the requirements of IEC 61508 standard series and can be justified before the project begins, and afterwards?
Use of Formal Methods	How to introduce formal methods into an organization?
Selection of Support Tools and Development Languages	How to select a set of support tools and languages that fulfil safety requirements and can be proven to produce reliable results.
<ul style="list-style-type: none"> • Software Safety Requirements Specification 	
Software Safety Requirements Specification	How to specify safety requirements for the software system.
<ul style="list-style-type: none"> • Software Design & Development 	
Software Development	How to create a software system that fulfils the specified requirements with respect to the required safety integrity level?
Software Architecture Design	How to create a software architecture that fulfils the specified requirements with respect to the required safety integrity level?
Software Architecture Verification	How to ensure that the software architecture design adequately fulfils the software safety requirements specification?
Technical Diversity	How to create such a system that not more than one element of it fails due to a disturbance? Or: how to avoid common cause failures?
Formal Methods Aided Design and Verification of Joint Behaviour	How should the parts (components) defined in the architecture behave in order to obtain the expected behaviour of the total (sub)system?
Software System Design – general	How to design the software so that it can be implemented, verified and validated.
Software System Design Verification	How to ensure that there are no incompatibilities between the software system design specification and software architecture design?

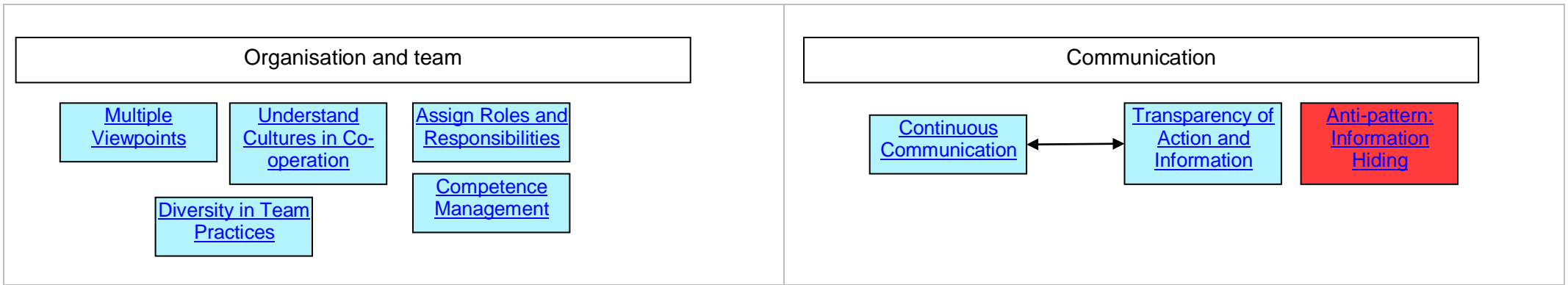
Name	Problem
Generic Glue	How can one be sure that the given requirements for the next phase are well defined i.e. complete and do not contain any contradiction?
Detailed Module Design	How to develop an individual software module so that it can be implemented safely and reliably.
Glue Design and Implementation	How can one be sure that design is correct, detailed enough, can be implemented with reasonable effort and without extra design decisions?
Coding	How to create reliable and safe program code, which is easy to modify when the need arises and which is also easy to audit – for purpose and for safety and security.
Analytic Design and Code Quality Assessment	How to create reliable and safe program code, which is easy to modify when the need arises and which is also easy to audit – for purpose and for safety and security?
Verification Testing	How to verify programs and their components at all abstraction levels?
Module Testing and Simulation	How to test the module to verify that it meets requirements?
Module Integration Testing	How to test the collection of modules in the architecture to verify that the system meets functional requirements?
PE Integration Testing	How to test the integrated system so that its functioning and functional safety can be verified?
Regression Testing	A change in software can lead to problems in other parts of the system. To identify those effects, regression testing is used.
Model-Based Testing	How to create tests that cover the specification and can be maintained with reasonable effort when the specification changes.
• Software Aspects of System Safety Validation	
Software Validation Planning	How to develop a plan for validating the safety-related software aspects of system safety?
Software Validation	How to ensure that the integrated system complies with the software safety requirements specification at the required safety integrity level?
Configuration Auditing	How can we assess how the system differs from a last validated baseline?
• Software Modification	
Software Modification Planning	How to plan modification of the software so that the modification activities can be performed safely and so that the resulting product is fully understood and can be validated?
Software Modification	How to ensure that the required software systematic capability is sustained when the validated software is modified?
Impact Analysis	How can we best assess how a proposed change impacts the system?
• Functional Safety Assessment	
Functional Safety Assessment	How to analytically assess the functional safety of software?
Failure Analysis	How can we analyse software errors and system failures in order to prevent them occurring again? How can we understand how the system handles failures and whether it does it properly? How can we understand how failures propagate through the system?
• Software Operation & Maintenance Procedures	
Writing of the Safety Manual	How can we communicate our knowledge of safe use to the users?

7.2 Visual views to the pattern collection

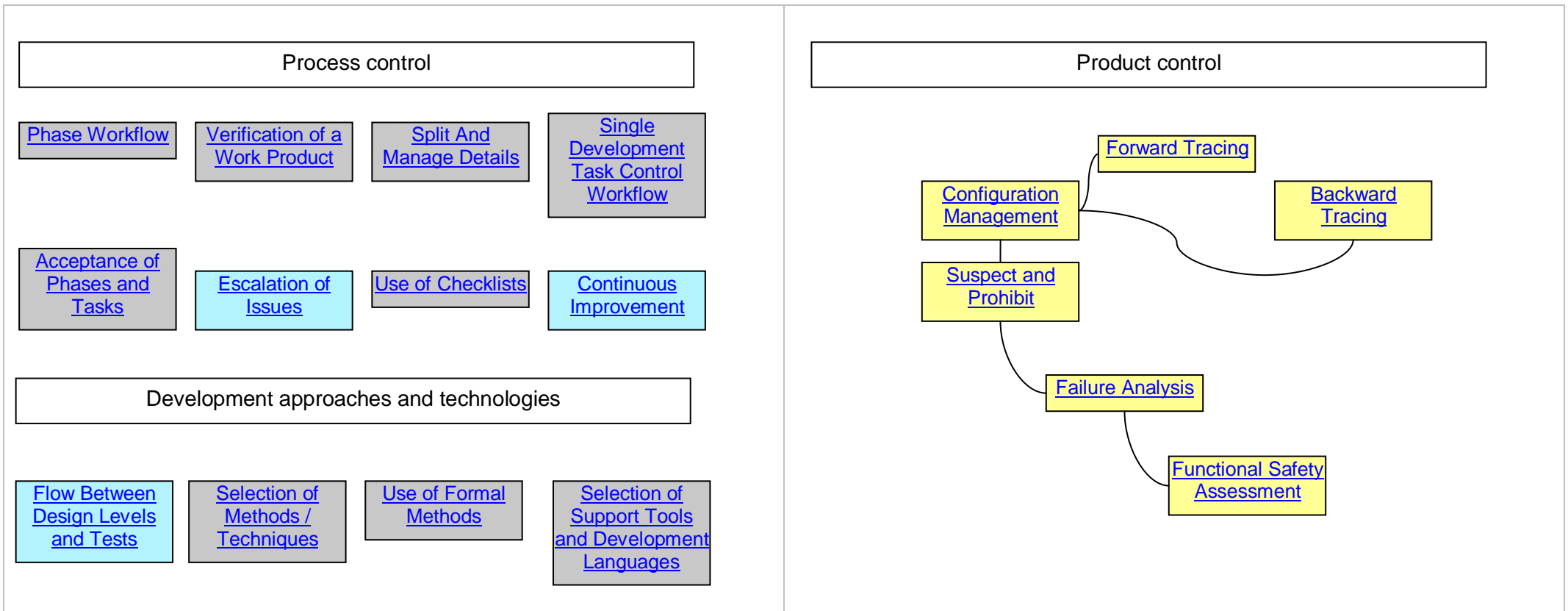
In the following pages we present the pattern collection in a visual form, collected into groups. The groupings are just examples of many possible ways to do such grouping. When the patterns are used in an organisation, the views here should only be used as a starting point and should be supplemented with other similar patterns that the organisation has created and used. Thus, they will be integrated in the operational context of an organisation and the relationships between patterns are defined in the particular context.

All the boxes contain links to the pattern descriptions so in the PDF version of this report it is possible to click and follow the links to study the connections and relations.

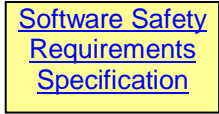
7.2.1 Generic organisational patterns



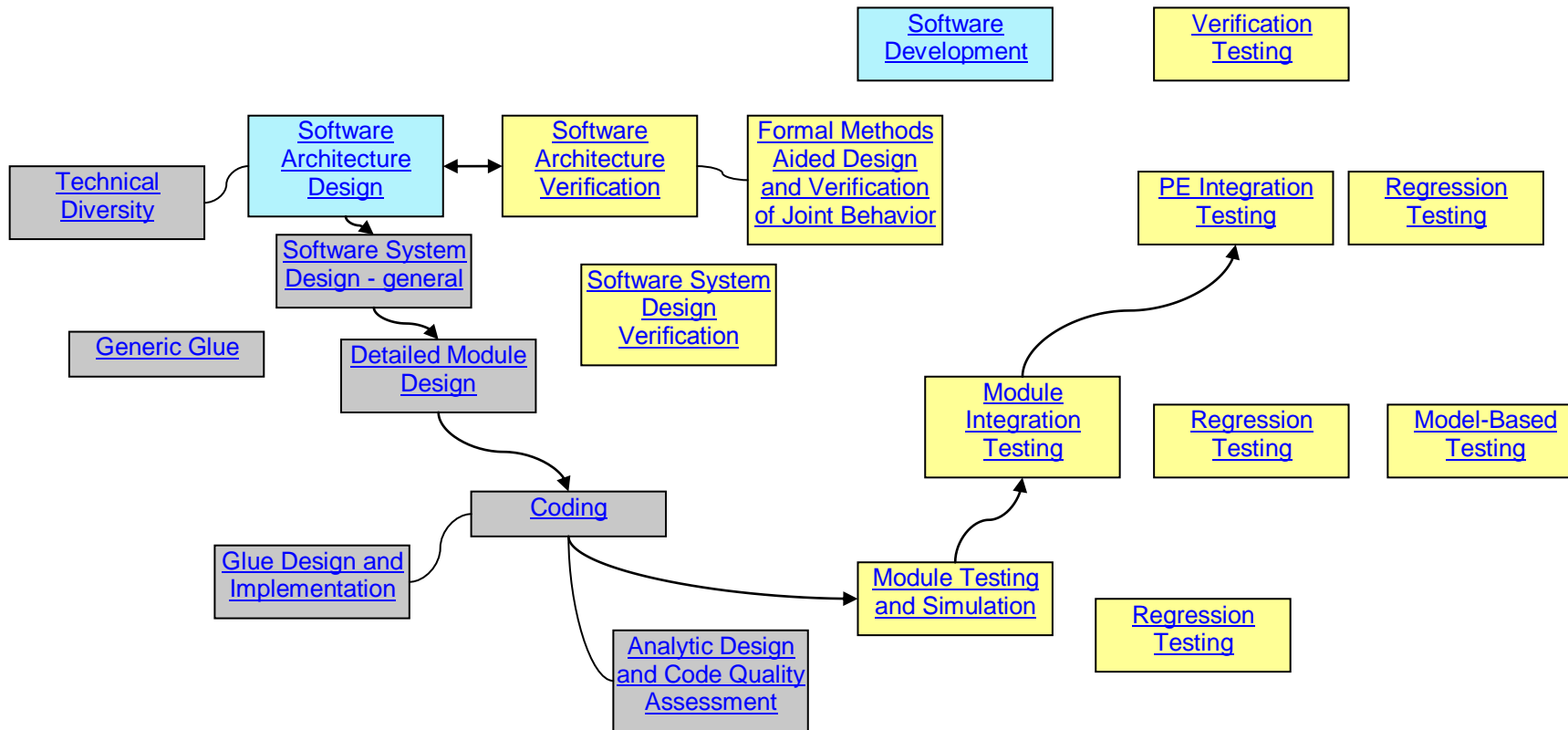
7.2.2 Generic process and product control patterns



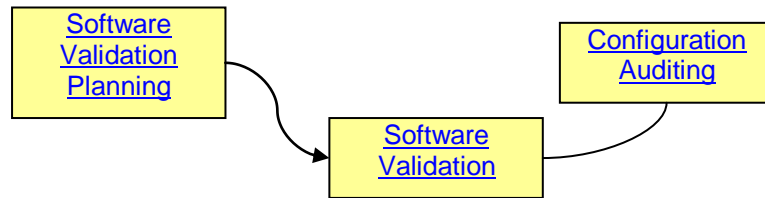
7.2.3 Software Safety Requirements Specification



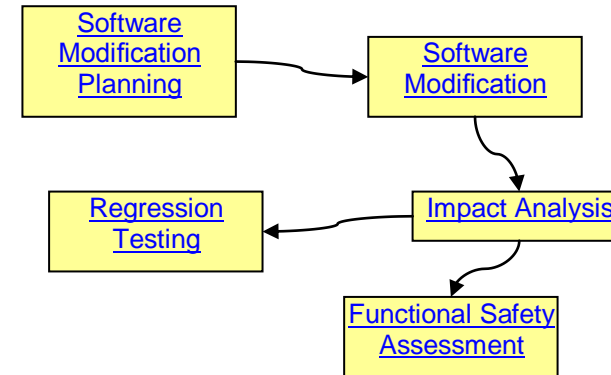
7.2.4 Software Design & Development



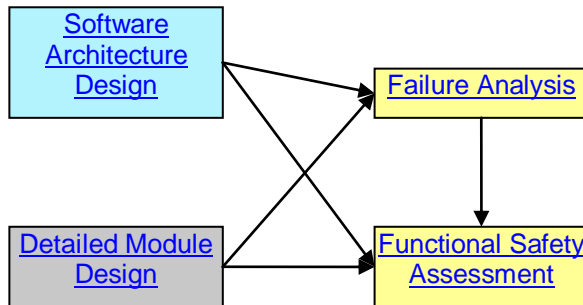
7.2.5 Software Aspects of System Safety Validation



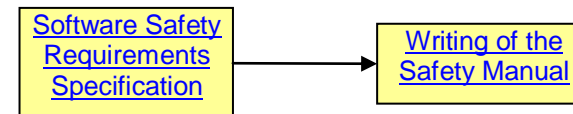
7.2.6 Software Modification



7.2.7 Functional Safety Assessment



7.2.8 Software Operation & Maintenance Procedures



7.3 Examples

The following are three examples of pattern types included in the collection:

- Phase Workflow is an example of a process workflow pattern.
- Assign Roles and Responsibilities presents an approach to a situation and contains a mind-map.
- Software Validation Planning. This is a pattern that combines a workflow mindset and a safety conscious understanding of issues and has many references in the IEC 61508 series.

7.3.1 Phase Workflow

A phase is an important building block of any software development lifecycle and also quite strictly influenced by the IEC 61508 requirements. Therefore, it is a natural application for process patterns.

The pattern contains a – as the name implies – process workflow, which is shown in a simplified form, suitable for explaining to various interest groups in training, auditing and other situations.

Name	Phase Workflow
Context	A development phase is started after a previous one has been completed.
Problem	How is a process phase carried out, satisfying safety lifecycle process requirements?
Forces	Each phase needs to implement the safety management principles and tasks that the IEC 61508 (2 nd ed.) series requires, as they are seen to be critical for the process to produce a safe system.
Solution	<p>A generic work flow:</p> <pre> graph TD A[Project plan and lifecycle model] --> D[Development work on output work products] B[Safety plan] --> D C[Instructions for phase tasks] --> D E[Inspection and review of inputs (previous phase)] --> D D --> F[Verification (at this phase; note the V-model)] F --> G[Phase review and acceptance] G --> H[Outputs to next phase] F -.-> D I[Safety assessment] -.-> D I -.-> J[Need for updated risk analysis?] J -.-> K[Return to prev. Phase if needed] K -.-> E D --> L[Development records] D --> M[Product documentation] D --> N[Acceptance records] </pre> <p>Critical elements of the process:</p> <ul style="list-style-type: none"> • Inputs need to be inspected and reviewed. By inspection we mean a thorough analysis of, for example, requirements and by review we mean reaching a consensus that inputs are flawless for the purpose of the phase. • Guidance is provided by project level plans and task specific instructions. Adherence to plans is checked in reviews.

Name	Phase Workflow
	<ul style="list-style-type: none"> • All safety related tasks are documented by records. • During the work, mostly analytic verification is carried out, but testing will happen later – note the V-model as a framework. • For most work products, safety is assessed and the work product corrected as required. • There is always a feedback loop to the previous process phases. • Because development produces new information about the use of the product, it needs to be assessed if hazard or risk analyses need to be updated. This may necessitate updating of many requirements. • All items under work and work products are configuration controlled. This includes documentation. • Before transferring outputs to the next phase they need to be reviewed and accepted. This internal acceptance must not be confused with validation.
Resulting Context	A successfully carried out process phase, providing solid output for the next phase and next development tasks.
Related Patterns	Verification of a Work Product Acceptance of Phases and Tasks Configuration Management
Standard References	IEC 61508-1 (2 nd ed.) presents the generic process requirements IEC 61508-3 (2 nd ed.) explains how this process is implemented in software development tasks
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	workflow, phase, process

7.3.2 Assign Roles and Responsibilities

Assigning roles and responsibilities does not happen that often in a project, but during a company's lifecycle it obviously happens tens or hundreds of times. It is also a critical task to understand and carry out, as achieving good safety requires good competencies. However, this pattern does not contain a workflow, but a mind map, describing the criteria and thinking to be applied in the assignment situation.

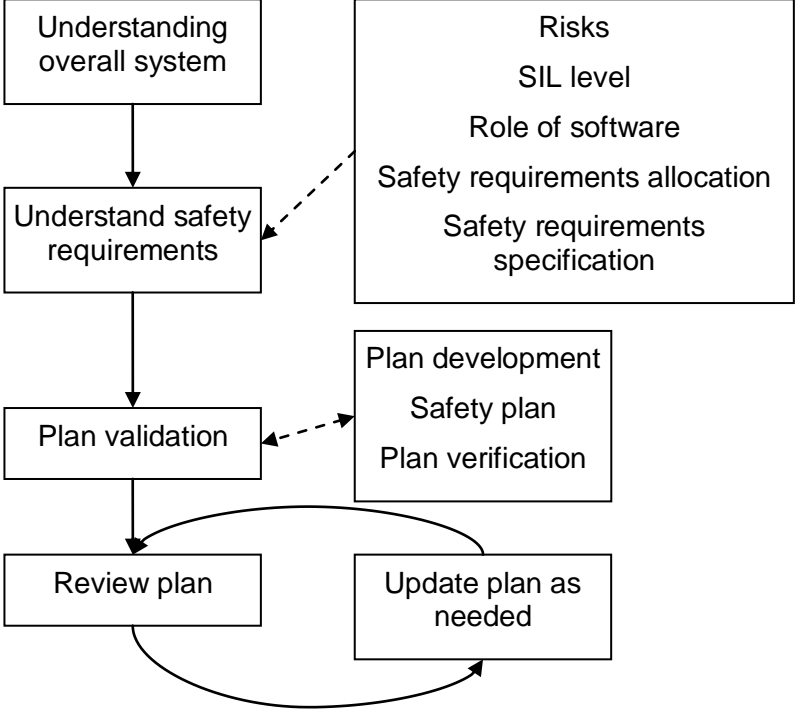
Name	Assign Roles and Responsibilities
Context	A safety-critical development project is being planned. Participants, roles and responsibilities need to be assigned to individuals.
Problem	How can we select project participants and assign roles and responsibilities so that utilisation of expertise and required independence are in an optimal balance?
Forces	When a project starts, a requirement for safety critical development is an explicit assignment of responsibilities and thus also roles.
Solution	<p>Key points in this process:</p> <ul style="list-style-type: none"> • Based on SIL level and other project requirements, understand the mandatory requirements for participants. • Identify the need for independence in verification and validation and select individuals for those tasks. They cannot have a role in development tasks. • Define who is responsible for safety and will accept project's products. • The challenging parts of the project require experience and skills. Assign the most capable people to the challenging tasks. • At high SIL levels, safety attitudes have a high importance in team selection. • Support organisational learning by combining various levels of expertise. • Consider knowledge transfer with other units and subcontractors when selecting team members. • Consider employing external experts for added competence even when independence is not a requirement. • If external validation (perhaps leading to certification) is required, plan a good way to include that party in the process from early on. • While all participants should have generic safety related knowledge and skills, project-based training should always be considered. <p>Defining roles and responsibilities does not mean that development needs to be bureaucratic, it just means that we know that someone will concentrate especially on the issues and who can help others to do their tasks better.</p>

Name	Assign Roles and Responsibilities
	<pre> graph TD A[Project participation selection mind map] --- B[SIL level – risks and requirements] A --- C[Safety knowledge, training] A --- D[Experience] A --- E[Attitude] A --- F[Tasks requiring independence] A --- G[Knowledge transfer] A --- H[Stakeholders] A --- I[Management and leadership] A --- J[Responsibility for safety] A --- K[Most critical design tasks] </pre>
Resulting Context	A formed project organisation where everyone understands what is expected from him/her and what he/she can expect from others. A good starting point for flexible collaboration.
Related Patterns	Multiple Viewpoints
Standard References	IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes competence requirements for project personnel IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes criteria for appropriateness of competence
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Wikipedia article Project governance describes project roles related issues http://en.wikipedia.org/wiki/Project_governance
Tags	project, planning, organisation, roles

7.3.3 Software Validation Planning

This is again something that happens only once in a projects, but requires careful thinking so that the projects can be effective and efficient. This is a pattern that combines a workflow mindset and a safety conscious understanding of issues and has many references in the IEC 61508 series.

Name	Software Validation Planning
Context	Software safety requirements specification has been finalised.
Problem	How to develop a plan for validating the safety-related software aspects of system safety?
Forces	Validation is a task that needs to be a planned activity so that the plans can be assessed to be sufficient for the requirements of IEC 61508 and so that the validation can afterwards be compared with the plan to see that it has been carried out properly. Validation of software is also done in the context of the overall system.
Solution	<p>The main process:</p> <ul style="list-style-type: none"> • Understand the overall context and the system and the software's role in it. • Understand the validation requirements, based on the project's SIL level. • Make a clear distinction in all plans between the validation of safety requirements and the validation of other product requirements. • Decide on the parties who make the validation, considering the required independence (for example, independent company unit, external validator) and a need for certification. • Create an overall safety plan that ensures that the development and safety assurance process is sufficient. • Plan all verification steps so that they ensure that the validation will proceed smoothly. • Plan the validation, leaving sufficient calendar time for its activities. This is usually in a form similar to a project plan. Do close collaboration in this with the party who will be doing the validation. • Consider in the plans that the validation process may not pass at the first time and thus changes may need to be made and validation repeated. • Plan some coordinated collaboration with the party doing the validation so that the development process can be guided into a positive direction (but maintaining independence of the validator). • Review the plan with all stakeholders and ensure that everyone understands the criticality of the validation – without it the product cannot be taken into use.

Name	Software Validation Planning
	 <pre> graph TD A[Understanding overall system] --> B[Understand safety requirements] B --> C[Plan validation] C --> D[Review plan] D --> E[Update plan as needed] E --> C F[Risks SIL level Role of software Safety requirements allocation Safety requirements specification] -.-> B G[Plan development Safety plan Plan verification] -.-> C </pre> <p>The flowchart illustrates the Software Validation Planning process. It begins with 'Understanding overall system', which leads to 'Understand safety requirements'. This step is influenced by a box containing 'Risks', 'SIL level', 'Role of software', 'Safety requirements allocation', and 'Safety requirements specification'. From 'Understand safety requirements', the process moves to 'Plan validation', which is influenced by a box containing 'Plan development', 'Safety plan', and 'Plan verification'. The process then proceeds to 'Review plan', which leads to 'Update plan as needed'. A feedback loop exists from 'Update plan as needed' back to 'Plan validation', and another from 'Review plan' back to 'Update plan as needed'.</p>
Resulting Context	A planned validation process which can be executed when the product is ready for validation.
Related Patterns	Software Validation
Standard References	<p>IEC 61508-1 (2nd ed.), clause 7.14 describes safety validation requirements</p> <p>IEC 61508-3 (2nd ed.), clause 7.7 defines the process for system validation.</p> <p>IEC 61508-3 (2nd ed.), table A.7 presents recommended techniques for software aspects and properties of system safety validation at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.7 describes the strictness of various ways of application of the software aspects and properties of system safety validation</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>While the validation plan should in an “ideal world” be based on stable requirements, things change and evolve and thus the validation plan needs to be updated as well during the development process.</p> <p>See Wikipedia article Verification and Validation http://en.wikipedia.org/wiki/Verification_and_validation</p>
Tags	validation, software system, software aspects, overall system

8 References

- Ohjelmaturva publications:

Koskinen, Johannes & Katara, Mika. 2011. Safety Process Patterns: Demystifying Safety Standards. Manuscript. Department of Software Systems, Tampere University of Technology. 14 pages.

Process Patterns for Safety Critical Software. [Referenced 2011-05-04] Available at <http://sites.google.com/site/safetypatterns>. (This is a TUT developed site for collecting safety process patterns based on IEC 61508)

Vuori, Matti. 2011. Agile Development of Safety-Critical Software. Tampere University of Technology. Department of Software Systems. Report 14. 112 p. Available at the Tampere University of Technology DPub system: <http://dspace.cc.tut.fi/dpub/>.

- Other references:

Ambler, Scott W. 2011. The Process Patterns Resource Page. Referenced 2011-05-04] Available at: <http://www.ambysoft.com/processPatternsPage.html>.

Vesiluoma, Sari. 2009. Understanding and Supporting Knowledge Sharing in Software Engineering. Tampere University of Technology, Publication 843. 158 p. Available at: <http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/6174/vesiluoma.pdf>

Wikipedia. Process patterns.[Referenced 2011-05-04] Available at: http://en.wikipedia.org/wiki/Process_patterns.

Wikipedia. Category: Software design patterns. [Referenced 2011-05-04] Available at: http://en.wikipedia.org/wiki/Category:Software_design_patterns.

Wikipedia. Organizational patterns. [Referenced 2011-05-04] Available at: http://en.wikipedia.org/wiki/Organizational_patterns.

Wikipedia. Use case. [Referenced 2011-05-04] Available at: http://en.wikipedia.org/wiki/Use_case.

Välimäki, Antti; Kääriäinen, Jukka & Koskimies, Kai. 2009. Global Software Development Patterns for Project Management. Proceedings of EuroSPI 2009, CCIS 42, pp. 137-148.

Part II: The Safety Process Pattern Collection

DISCLAIMER

The descriptions in this collection are designed to be informative only and are aimed to help skilled professionals to gain further understanding of how to apply the IEC 61508 standard series in practical software development and to share that understanding in their organisations.

For the requirements of and official information contained in the standards, readers shall study the official standard documents and make any process decisions based on that information.

The authors of this collection will give no guarantee, implied or otherwise, of the correctness or applicability of the information given herein.

1 Generic organisational patterns

Name	1.1.1 Multiple Viewpoints
Context	Safety of complex systems can only be reached by combining several points of view during development.
Problem	How to identify the viewpoints that project participants should represent in the project?
Forces	In order to be efficient and effective, a project needs to be lean and not have too many participants. Yet it needs to be assured that all important viewpoints can be presented during development.
Solution	<p>All necessary viewpoints (incl. stakeholder viewpoints) need to be identified and after that decided, which ones need inclusion in the project team, and for which others the collaboration and communication channels suffice.</p> <p>Through the analysis we can gain a collaboration network that will help build excellent safety that is not compromised through the unplanned actions of any party.</p> <pre> graph TD V[Viewpoints mind map] --- O[Overall system, high-level systems] V --- E[E/P/EP, hardware] V --- T[Technologies] V --- P[Project] V --- D[Drive and critique] V --- Q[Quality] V --- M[Marketing and sales] V --- PM[Product management] V --- S[Safety] V --- X[Experience] V --- C[Certification] V --- SC[Subcontractors] V --- CU[Customers] </pre>
Resulting Context	A harmonic project where all necessary viewpoint are present and thus everyone can work towards a safe and otherwise excellent product.
Related Patterns	Continuous Communication
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	Matti Vuori

Name	1.1.1 Multiple Viewpoints
Status	Version 2011-04-29
Notes	Sometimes, inclusion of too many viewpoints actively in a project may compromise safety, but when using a systematic professional development process there should not be a danger of that; instead the effect is positive on safety.
Tags	project, principles, organisation, viewpoints

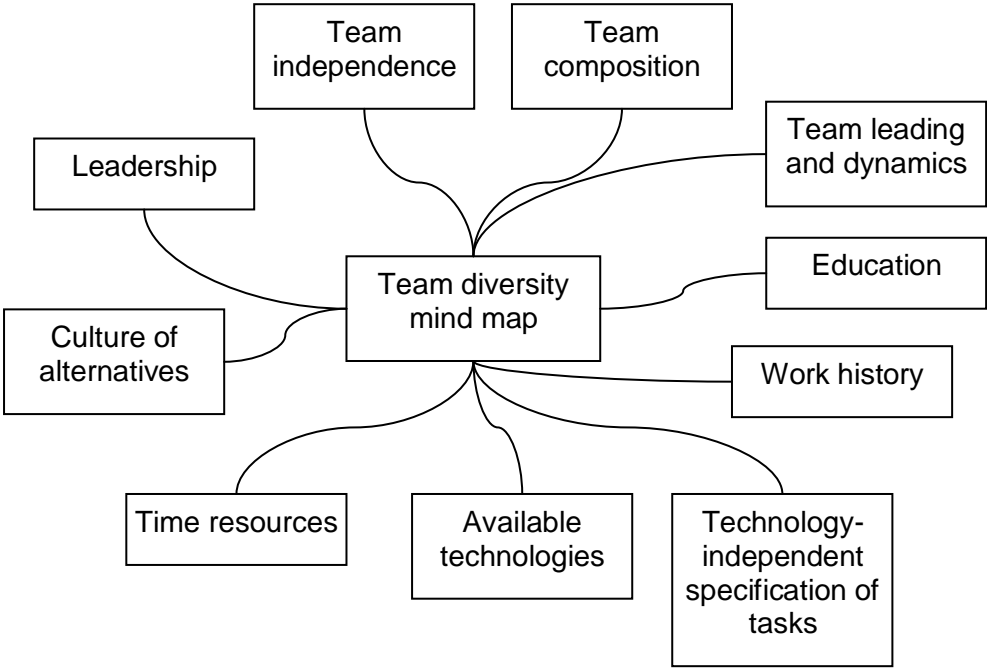
Name	1.1.2 Understand Cultures in Co-operation
Context	When planning development collaboration and co-operation in distributed mode, many cultures are integrated in the process.
Problem	How can we co-operate in a multi-cultural project so that cultural differences are managed so that they will not endanger safety?
Forces	Cultural patterns are so strong that they will always overpower any formal instructions and guidelines.
Solution	<p>Cultures of all participating organisations need to be identified and potential problems assessed. Working modes need to be planned accordingly. Some examples:</p> <ul style="list-style-type: none"> • If participating cultures emphasise oral communication and discussion, it needs to be practiced instead of just sending instructions in documents. • Conflicts of cultural status may prohibit testers from giving feedback to “superior” parties (this has happened in India due to the caste system, which still is in effect though informally). • Attitude towards time and punctuality of deadlines varies in cultures. • Very small details in expressions can make Finnish written communication seem hostile to other cultures. • There are many cultures, where “yes” is a common response even if the personnel does not have a clue of how a task should be done. • Some cultures may show more initiative and problems solving attitude. <p>Consultants should be used when starting co-operation with new national cultures. Teams need to be given intercultural training.</p>

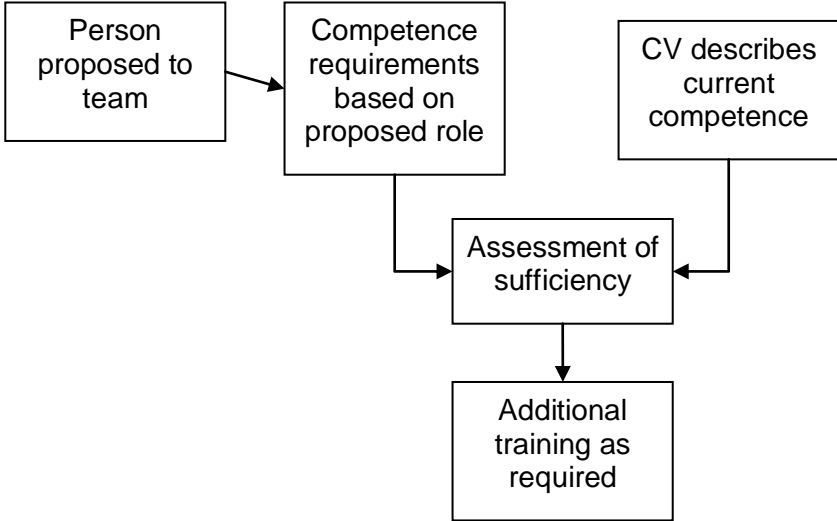
Name	1.1.2 Understand Cultures in Co-operation
	<pre> graph TD C[Communication patterns] --- M[Cultural issues mind map] S[Status of written information] --- M P[Personal communications] --- M R1[Relation to power] --- M R2[Relation to time and deadlines] --- M M --- PC[Project cultures] M --- MF[Motivational factors] M --- CS[Competence in safety issues] M --- TP[Trust in promises] M --- PSI[Problem solving and initiative] M --- IS[Importance of safety] </pre>
Resulting Context	An understood system of various cultures that can work together efficiently.
Related Patterns	Assign Roles and Responsibilities
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	See Wikipedia article Multiculturalism: http://en.wikipedia.org/wiki/Multiculturalism
Tags	project, principles, organisation, cultures

Name	1.1.3 Assign Roles and Responsibilities
Context	A safety-critical development project is being planned. Participants, roles and responsibilities need to be assigned to individuals.
Problem	How can we select project participants and assign roles and responsibilities so that utilisation of expertise and required independence are in an optimal balance?
Forces	When a project starts, a requirement for safety critical development is an explicit assignment of responsibilities and thus also roles.
Solution	<p>Key points in this process:</p> <ul style="list-style-type: none"> • Based on SIL level and other project requirements, understand the mandatory requirements for participants. • Identify the need for independence in verification and validation and select individuals for those tasks. They cannot have a role in development tasks. • Define who is responsible for safety and will accept project's products. • The challenging parts of the project require experience and skills. Assign the most capable people to the challenging tasks. • At high SIL levels, safety attitudes have a high importance in team selection. • Support organisational learning by combining various levels of expertise. • Consider knowledge transfer with other units and subcontractors when selecting team members. • Consider employing external experts for added competence even when independence is not a requirement. • If external validation (perhaps leading to certification) is required, plan a good way to include that party in the process from early on. • While all participants should have generic safety related knowledge and skills, project-based training should always be considered. <p>Defining roles and responsibilities does not mean that development needs to be bureaucratic, it just means that we know that someone concentrates especially on the issues and can help others to do their tasks better.</p>

Name	1.1.3 Assign Roles and Responsibilities
	<pre> graph TD A[Project participation selection mind map] --- B[SIL level – risks and requirements] A --- C[Safety knowledge, training] A --- D[Experience] A --- E[Attitude] A --- F[Tasks requiring independence] A --- G[Knowledge transfer] A --- H[Stakeholders] A --- I[Management and leadership] A --- J[Responsibility of safety] A --- K[Most critical design tasks] </pre>
Resulting Context	A formed project organisation where everyone understands what is expected from him/her and what he/she can expect from others. A good starting point for flexible collaboration.
Related Patterns	Multiple Viewpoints
Standard References	IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes competence requirements for project personnel IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes criteria for appropriateness of competence
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Wikipedia article Project governance describes project roles related issues http://en.wikipedia.org/wiki/Project_governance
Tags	project, planning, organisation, roles

Name	1.1.4 Diversity in Team Practices
Context	In all project and product planning phases, good heuristic principles are applied and understood by all participants.
Problem	How to apply the principle of diversity in all tasks?
Forces	Diversity means that elements of activity are based on varying underlining principles and approaches so that if one approach fails, another still succeeds. This principle is applied in every activity.
Solution	<p>Diversity is ultimately a technical thing but individuals reach it.</p> <p>For example:</p> <ul style="list-style-type: none"> • Designers with similar education are prone to use similar designs and neglect similar issues, making the system very vulnerable. • Testers who have had similar training or who have similar work experience will use similar methods. • If a company supports only one official technique in the specification, all designs will be prone to its problems. • If requirements are tied to implementation, it will lead to similar technical solutions. • If managers are too technologically oriented, they will see one “best practice” and suppress alternatives, reducing diversity. <p>Ways to improve diversity:</p> <ul style="list-style-type: none"> • Hiring of people with different backgrounds, from different schools and different industries. • Keeping teams that do redundant designs in independent teams and monitoring that they have alternative approaches. • Use testers in test teams so they can have more independent thinking and approach to systems. • Allocate enough time for development. If people are busy, they start to copy approaches and lessen diversity and make the system prone to common cause failures. • Diversity requires creativity. Team composition is critical here. • Support for diversity requires leadership. • There needs to be plenty of information on alternative approaches freely available. Systematic technology management and the creation of re-usable assets will help here.

Name	1.1.4 Diversity in Team Practices
	 <pre> graph TD A[Team diversity mind map] --- B[Team independence] A --- C[Team composition] A --- D[Team leading and dynamics] A --- E[Education] A --- F[Work history] A --- G[Technology-independent specification of tasks] A --- H[Available technologies] A --- I[Time resources] A --- J[Culture of alternatives] A --- K[Leadership] </pre>
Resulting Context	A project organisation capable of working in diverse ways and creating diverse designs, implementations and testing.
Related Patterns	Competence Management Multiple Viewpoints Transparency of Action and Information Technical Diversity
Standard References	IEC 61508-1 (2 nd ed.), sub-clauses 7.6.2.7 and 7.6.2.8 describe requirements for independence of design solutions considering common cause failures
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	project, planning, organisation, teams, diversity, principles

Name	1.1.5 Competence Management
Context	When selecting team members, their competence is an issue that needs to be managed systematically.
Problem	How can we ensure that each member has the required competence?
Forces	IEC 61508 (2 nd ed.) requires competence of all participants to be ensured.
Solution	<p>The basic workflow in the beginning of a project:</p>  <pre> graph TD A[Person proposed to team] --> B[Competence requirements based on proposed role] C[CV describes current competence] --> D[Assessment of sufficiency] B --> D D --> E[Additional training as required] </pre> <p>All proposed participants' competence requirements in the proposed role are defined and compared with the person's current competence. Additional training is given if the competence is not sufficient.</p> <p>This process requires active CV management so that all descriptions are up-to date, and internal training system with which training can be given in a flexible manner as needed.</p>
Resulting Context	A person is accepted to a project, with known competence.
Related Patterns	Assign Roles and Responsibilities
Standard References	IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes competence requirements for project personnel IEC 61508-1 (2 nd ed.), sub-clause 6.2.13 describes criteria for appropriateness of competence
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Work experience can of course not be improved by training.
Tags	project, planning, organisation, competence, CV, training

Name	1.1.6 Continuous Communication
Context	During development, things are developing fast, yet communication is needed to keep the developments on track and to address possible problems as soon as possible.
Problem	How can we get rapid input from everyone in such a way that it does not make progress slower, but makes the project proceed more efficiently? How can we get busy professional to participate in the process?
Forces	Complex systems can have complex problems if communication fails. If we get input too late, mistakes, problems and incompatibilities, thus potential hazards, are difficult and costly to correct.
Solution	<p>A style of project collaboration that combines rhythmic team input and continuous input from everyone present. Some main elements:</p> <ol style="list-style-type: none"> 1) Rhythmic flow of viewpoints – group review of input, expert development – group review of results, validation included in workflow. Splitting of tasks makes this flexible and allows for concentration on issues. 2) Continuous change of opinions through an Application Lifecycle Management (ALM) system commenting functions and other channels <p>Examples:</p> <ul style="list-style-type: none"> • Review of input in a team, using several viewpoints and various kinds of expertise. • The development underway is always kept open in the information system and comments are welcome from everyone. • Splitting of development into small, manageable parts so communication can be concentrated and be traced to issues. • All development and design information, especially safety related information is constantly accessible in a way that supports its linking into any development situation (like safety requirements and safety assessments linked to a subsystem design). • Review of task results both in meetings and via ALM system. • Input from validation and certification parties is always welcome and especially supported at certain steps (like concept, project plan, architecture, implementation, after verification). <ol style="list-style-type: none"> 3) A meeting rhythm for teams keeps things coordinated <ul style="list-style-type: none"> • For example, weekly on a certain day at a certain time, or more frequently if the process so requires (many agile teams have short daily meetings) 4) Key roles that aid in forwarding information between participating parties <ul style="list-style-type: none"> • Site coordinator, project managers etc.

Name	1.1.6 Continuous Communication		
	Type of item / issue	Internal collaboration and communication	External collaboration and communication
	<pre> graph TD A["Big steps"] --> B["Ongoing work"] B --> C["Split to items"] C --> D["The Big Picture"] </pre>	<pre> graph TD E["Team work Team analysis Team review"] --> F["Assign"] F --> G["Expert work"] G <--> H["Team collaboration Team assessment"] I["Team meetings kept in defined rhythm"] J["Shared real-time views"] </pre>	<pre> graph TD K["Validator / certification body input"] L["Continuous external comments and review"] </pre>
	<p>Obviously, communication does not just happen. The above approach just enables. Project and company cultures need to be addressed as well.</p>		
Resulting Context	A constant, natural, yet managed flow of information. Environment where knowledge and expertise can be utilized easily and as soon as needed. Fewer problems, fewer hazards.		
Related Patterns	Multiple Viewpoints Transparency of Action and Information		
Standard References	(No direct reference in the IEC 61508 standard series)		
Authors	Matti Vuori		
Status	Version 2011-04-29		
Notes	<p>For Application Lifecycle Management systems see Wikipedia article Application lifecycle management: http://en.wikipedia.org/wiki/Application_lifecycle_management</p>		

Name	1.1.6 Continuous Communication
Tags	project, planning, organisation, communication, ALM, principles

Name	1.1.7 Transparency of Action and Information
Context	Many tasks are being carried out, perhaps on many sites.
Problem	How can we know what is actually being done and whether there are any problems?
Forces	In safety critical development we need to get information on any problems immediately, when they can be solved efficiently and do not cause bigger problems.
Solution	<p>All information related to tasks, their progress and results needs to be transparently accessible by all, from team colleagues to top management. This can be accomplished with the use of shared information systems. That way, the management information is not delivered with periodical reports, but real-time views into the contents and logs of the information system, such as an Application Lifecycle Management system.</p> <pre> graph TD subgraph Level1 [Top management] B1[Business and portfolio view] end subgraph Level2 [Product management, unit management] B2[Coordination, portfolio view] end subgraph Level3 [Project coordination and management, interest groups] B3[Control and coordination and problem identification] end subgraph Level4 [Site level, Team level] B4[Control and coordination, correction of problems] end subgraph Level5 [Individuals] B5a[Task progress] B5b[Task results] end B5a --> B4 B5b --> B4 B4 --> B3 B3 --> B2 B2 --> B1 </pre>
Resulting Context	Everyone has access to progress information (within project / company confidentiality). Potential problems can be identified and corrected promptly.
Related Patterns	Continuous Communication
Standard References	(No direct references in IEC 61508 standard series)
Authors	Matti Vuori

Name	1.1.7 Transparency of Action and Information
Status	Version 2011-04-29
Notes	For Application Lifecycle Management systems see Wikipedia article Application lifecycle management: http://en.wikipedia.org/wiki/Application_lifecycle_management
Tags	project, planning, organisation, communication, transparency, ALM

Name	1.1.8 Anti-pattern: Information Hiding
Context	Many tasks are being carried out, perhaps on many sites.
Problem	We know that we have problems, and are ashamed of it. How can we hide the situation until a miracle happens and the problem is solved? How can we suppress information so that it will not be leaked to competitors or media?
Forces	Hiding information is strong practice based on technological positivism and belief in formal reporting – both of which are now understood not to be valid assumption anymore. Periodical reporting as the only tool of informing about progress simply does not work. Specifications based on “need-to-know” information will leave a lot lacking. Choosing communication patterns based on fear of leaks will hinder co-operation and innovation. All this compromises a project and safety. Information hiding simply needs to stop.
The bad pattern	Bad practices: <ul style="list-style-type: none"> • Project is controlled based on promises, not real information, causing big problems when there is no real progress or deliverables. • Reports are beautified, not showing real data, making them misleading. • Problems are not reported because of shame and the hope for a miracle to happen. • Early developments are not shared causing different teams to work in different directions, under different assumptions and goals. • Subcontractors give false promises of progress which cannot be validated as there is no access to real data. • Subcontractors cannot innovate because they are given minimal information because of fear of information leaks. • Etc...
Resulting Context	Due to lacking information problems are not seen until they cause very large and costly problems endangering the whole product and even the business.
Related Patterns	=> Use pattern Transparency of Action and Information instead Continuous Communication
Standard References	(No direct references in IEC 61508 standard series)
Authors	Matti Vuori

Name	1.1.8 Anti-pattern: Information Hiding
Status	Version 2011-04-29
Notes	What is an anti-pattern? Usually it is a seemingly good idea that ultimately produces bad results. See Wikipedia article: http://en.wikipedia.org/wiki/Anti-pattern . Information hiding is such an idea, that many managers have believed and still believe in.
Tags	anti-pattern, project, planning, organisation, communication, transparency, hiding, principles

2 Generic process and product control patterns

These are patterns that are utilized in almost all parts of the process, in ways that suit the particular situation. These patterns are used to control the product and process phases and tasks.

Name	2.1.1 Phase Workflow
Context	A development phase is started after a previous one has been completed.
Problem	How is a process phase carried out, satisfying safety lifecycle process requirements?
Forces	Each phase needs to implement the safety management principles and tasks that the IEC 61508 (2 nd ed.) series requires, as they are seen to be critical for the process to produce a safe system.
Solution	<p>A generic work flow:</p> <pre> graph TD A[Project plan and lifecycle model] --> B[Inspection and review of inputs (previous phase)] B --> C[Extracting, specifying and elaborating of phase requirements] C --> D[Development work on output work products] D --> E[Verification (at this phase; note the V-model)] E --> F[Phase review and acceptance] F --> G[Outputs to next phase] H[Development records] <--> D I[Product documentation] <--> D J[Verification records] <--> E K[Acceptance records] <--> F L[Safety assessment] <--> D L <--> E M[Need for updated risk analysis?] <--> L M <--> N[Return to prev. phase if needed] N -.-> B </pre>

Name	2.1.1 Phase Workflow
	<p>Critical elements of the process:</p> <ul style="list-style-type: none"> • Inputs need to be inspected and reviewed. By inspection we mean a thorough analysis of, for example, requirements and by review we mean reaching a consensus that inputs are flawless for the purpose of the phase. • Guidance is provided by project level plans and task specific instructions. Adherence to plans is checked in reviews. • All safety related tasks are documented by records. • During the work, mostly analytic verification is carried out, but testing will happen later – note the V-model as a framework. • For most work products, safety is assessed and the work product corrected as required. • There is always a feedback loop to the previous process phases. • Because development produces new information about the use of the product, it needs to be assessed if hazard or risk analyses need to be updated. This may necessitate the updating of many requirements. • All items under work and work products are configuration controlled. This includes documentation. • Before transferring outputs to the next phase they need to be reviewed and accepted. This internal acceptance must not be confused with validation.
Resulting Context	A successfully carried out process phase, providing solid output for the next phase and next development tasks.
Related Patterns	Verification of a Work Product Acceptance of Phases and Tasks Configuration Management
Standard References	IEC 61508-1 (2 nd ed.) presents the generic process requirements IEC 61508-3 (2 nd ed.) explains how this process is implemented in software development tasks
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	workflow, phase, process

Name	2.1.2 Verification of a Work Product
Context	A work product has been created. After its creation it needs to be verified that it meets any requirement.
Problem	How to verify that a work product meets its requirements.
Forces	The process of safety-critical development is very much requirement-centred and thus its success and the product's and project's acceptance are based on requirements being met.
Solution	<p>The requirements to any work product come from its previous phase; a phase in the safety lifecycle or in the V-model, or any process requirements, from the IEC 61508 (2nd ed.) series. Thus, the verification needs the following steps:</p> <ul style="list-style-type: none"> • Planning of the verification. The plans are assessed (reviewed) to meet all requirements – that the verification tackles all requirements and is carried out in such a way that it meets any process requirements. • Determining of the configuration of the work product being verified. • Inspecting that the work product' design is based on the requirements and fulfils all the mandatory requirements assigned to it. This inspection compares the work product and its "parent" work product. • Experimental verification, using simulation or testing that demonstrates fulfilling of the requirements. • Review and acceptance of the verification results, by the person assigned to be responsible for functional safety. • Storing of all verification documents as evidence of verification <div style="text-align: center;"> <pre> graph TD Parent[Parent work product] -- Requirements --> Child[Child work product] Child -.- Verification -.-> Parent </pre> </div> <p>If the work product does not pass verification, it will need to be modified and the verification repeated until it passes. Criteria for passing are expressed in the verification plan(s).</p> <p>The most relevant means of verification can depends on the progress of the development. For example, program code is mostly verified analytically in the first steps of its development, but later the verification is mostly based on testing.</p>

Name	2.1.2 Verification of a Work Product
	When the work product is changed, for example requirements change, the verification is invalidated, as it only applies to the previous version of the work product.
Resulting Context	A verified work product, which can act as input to the next development phase, finally leading to readiness for validation.
Related Patterns	Configuration Management explains the configuration related issues. Software Validation explains the steps in validation, readiness to which verification build. Generic Glue pattern describes a helping technique.
Standard References	IEC 61508-1 (2 nd ed.), clause 7.18 IEC 61508-3 (2 nd ed.), clause 7.9.2
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	See Wikipedia article Verification and Validation http://en.wikipedia.org/wiki/Verification_and_validation
Notes	workflow, verification

Name	2.1.3 Split and Manage Details
Context	A work product is under development.
Problem	When a work product, such as a module or a specification, is being developed, how can we ensure that the work can be carried on in parallel, so that we can address issues independently and verify each small task and see the state of the whole?
Forces	In developing any non-trivial system, the work is divided into many parts which are then developed in parallel and in an interleaved manner. We need a mechanism with which we can handle the parts, concentrate on them and track the progress of each individually.
Solution	<p>By splitting the tasks into items, each of those can be handled individually from requirements to validation. An Application Lifecycle Management system is used to do the splitting and handling of the split items.</p> <pre> graph TD Req[Requirements] --> Sub[Subsystem, design or task] Req --> Dev[Development, design, implementation] Sub --> S1[Split 1] S1 --- S2[Split 2] S2 --- SN["Split N - status, responsibility, version, activities"] Dev --> Ver[Verification] Ver --> RA[Review & acceptance] RA --> Val[Validation] RA -.-> Tracking of progress SN Ver -.-> SN </pre>
Resulting Context	A complex subsystem or task split into manageable items, allowing an efficient development process and effective task performance.
Related Patterns	This pattern is utilised in practically all patterns, even though their description might, for the sake of generality, imply a larger grained approach.
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	Matti Vuori

Name	2.1.3 Split and Manage Details
Status	Version 2011-04-29
Notes	
Tags	workflow, phase, process, split

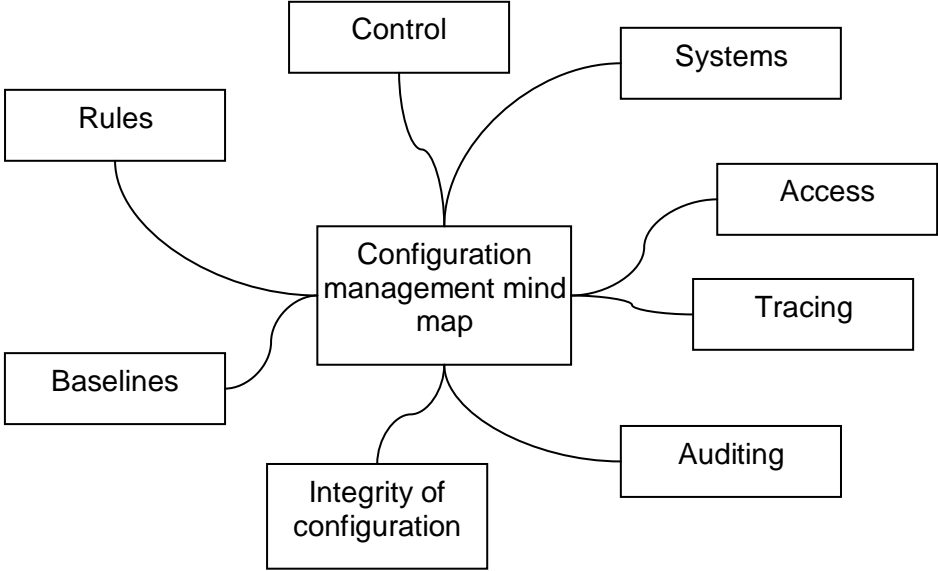
Name	2.1.4 Single Development Task Control Workflow
Context	A single design task is started and its progress is tracked from draft to closure.
Problem	How to have a simple, generic workflow that allows tracking of the completion of single tasks and progress of a set of tasks?
Forces	Tracking of progress with defined workflows is essential in any kind of development.
Solution	<p>A generic single task workflow from draft to acceptance:</p> <pre> graph TD Draft[Draft of a work item] --> Ready[Ready] Ready --> Approved[Approved] Approved --> Closed[Closed] Ready -.-> Draft </pre> <p>Often for a set of tasks so that the general progress can be assessed</p> <p>Closed statuses: done, rejected, duplicate, trash</p> <p>An approved version is always frozen so that its configuration, design and implementation can be controlled.</p>
Resulting Context	A closed work item
Related Patterns	Phase Workflow Split and Manage Details Acceptance of Phases and Tasks
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	
Status	Version 2011-04-29

Name	2.1.4 Single Development Task Control Workflow
Notes	This kind of workflows can be implemented in many kinds of design management tools: task based tools, Application Lifecycle Management systems, physical task boards, or shared Excel sheets.
Tags	workflow, phase, task, process, acceptance

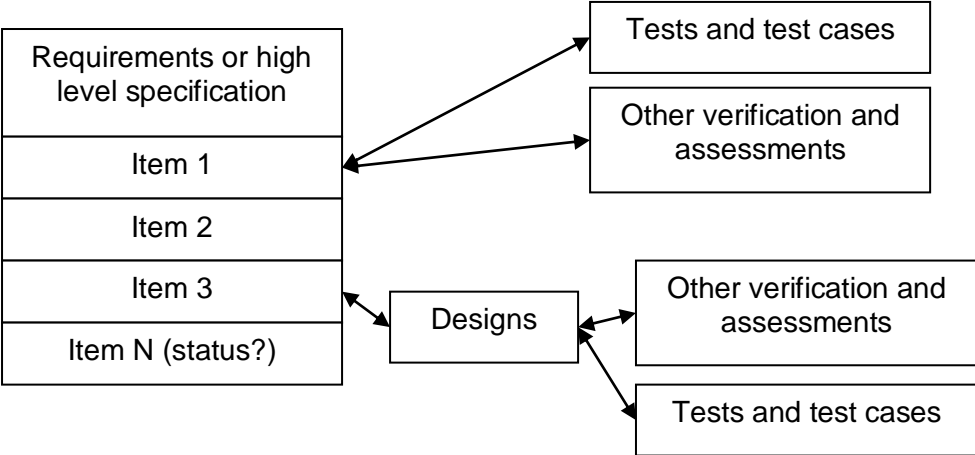
Name	2.1.5 Acceptance of Phases and Tasks
Context	When a work product has been created and verified, it still needs to be reviewed and accepted by the person who is appointed to be responsible for safety.
Problem	How are work products accepted in the development process so that they can be safely utilized?
Forces	In safety-critical development, acceptance of work products means taking responsibility for safety. Therefore, it needs to be based on clear evidence that all activities have been carried out properly and that the work product meets its requirement.

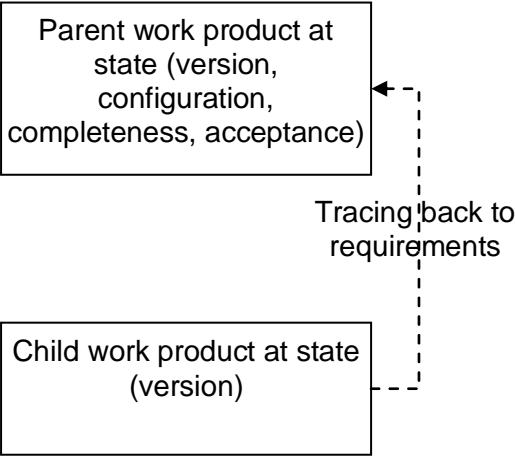
Name	2.1.5 Acceptance of Phases and Tasks
Solution	<p>The acceptance is based on the following tasks:</p> <ul style="list-style-type: none"> • Getting an impulse that a process phase and its related work products could be accepted. • Gathering evidence that the execution of that phase meets all requirements (correct plans, correct execution, proper verification, complete documentation, thorough safety assessments etc...). • A project review of the phase. • Based on evidence and review, formal acceptance. <p>If the phase cannot be accepted, its tasks need to be repeated and acceptance procedures re-executed until acceptance can be given.</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 20px;"> <p>Meets process requirements (safety processes / tasks)?</p> <p>Meets product requirements (safety)? (Analyses, verification, documentation, records, inspection?)</p> </div> <div style="text-align: center;"> <pre> graph TD A[Work product or process phase proposed for acceptance] --> B[Review] B --> C[Acceptance] C --> D[Next phase] C -.-> A </pre> </div> </div>
Resulting Context	A process phase and its associated work products have been accepted and thus the work can move to the next phase.
Related Patterns	(All)
Standard References	(All IEC 61508 (2 nd ed.) process requirements)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Acceptance is a project and product management term and not widely used in IEC 61508 series, because it is always informal and only validation and certification will produce real acceptance for safety matters.
Tags	workflow, phase, task, acceptance

Name	2.1.6 Configuration Management
Context	During the process, elements of the software system are developed and changed during their design, implementation, integration and so on. Thus there may be developments at various stages in various phases.
Problem	How can we know what elements, in what configuration and in what version the software system assessed consists of and what has changed compared with a baseline?
Forces	Verification and validation of software are only relevant if we know what exact components the software system consists of. For validation, we need information about what has changed. Thus, all individual items and configurations need to be carefully controlled.
Solution	<p>Each individual item is determined carefully by (not a complete list):</p> <ul style="list-style-type: none"> • Its identification • Its version • Rules for using it in the configuration • Knowledge of its compatibility with other item <p>Each software system configuration is tracked for:</p> <ul style="list-style-type: none"> • Inclusion of new items or versions of existing items • Removal or change of items <p>Thus, any changed configuration will form a new configuration, which needs to be assessed.</p> <p>All documentation, assessments and other activities need to be associated with the exact configuration they were done for.</p> <p>A concept of baseline is used, to determine a stable “starting point” for development. Baselines are used at given times to freeze the configuration after it has, by architecture design, evolved. After that time, strict controls of configuration are applied.</p> <p>Configuration management needs to be based on a system that allows easy carrying of configuration related tasks, and also auditing of those tasks and any configuration.</p> <p>Changes of configuration and configuration information need to be authorized.</p> <p>Also, it needs to be possible to determine the configuration of any system released to production or marketed.</p>

Name	2.1.6 Configuration Management
	 <pre> graph TD CM[Configuration management mind map] --- Control CM --- Systems CM --- Access CM --- Tracing CM --- Auditing CM --- Integrity[Integrity of configuration] CM --- Baselines Rules --- CM </pre>
Resulting Context	Continuous knowledge of configurations and their item. Ability to construct any previous configuration if required.
Related Patterns	Configuration Auditing
Standard References	IEC 61508-1 (2 nd ed.), sub-clause 6.2.10 IEC 61508-1 (2 nd ed.), sub-clause 6.2.3 describes the configuration management principles
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Modern configuration management systems support the requirements of safety-critical development. See Wikipedia article Configuration Management: http://en.wikipedia.org/wiki/Configuration_management
Tags	product, configuration, management, control, tracing

Name	2.1.7 Forward Tracing
Context	A work product – most often a specification – has been created. Now we need to track that all its items are handled properly at the next development process phases, especially on the route from requirements to testing.
Problem	<p>How can we link activities and development items so that we can at any time find out what actions are planned and have been carried out regarding the items?</p> <p>How can we follow the chain from any requirement to its verification and testing in an easy way?</p> <p>If some aspect of the work product changes, how can we know what items in the following process phases are invalidated due to that change?</p>
Forces	In safety-critical development, every design and implementation needs to be based on previous, accepted work and verified and validated accordingly. Therefore, it is critical to know what high level items have been verified and validated and if they change, what needs to be verified and validated again. And if the requirements change, what designs need to be revised.
Solution	<p>The following principles need to be used:</p> <ul style="list-style-type: none"> • The work products are identified (and version controlled) • The work products are structured so that each item (e.g. each requirement or each design element) can be identified and addressed individually. This is achieved by allocating them individual identification codes in configuration and documentation systems. • In lower level work products, their elements are linked to elements of higher level work products and thus can be shown to have a connection. • Thus, there is a systematic linkage from requirements to verification, including all testing and various analyses and safety assessments. • A development management tool is used, which can report the connections present, and also items that have no connections. <p>This technique is used to assess:</p> <ul style="list-style-type: none"> • Coverage of designs and development tasks (e.g. what requirements have test cases for them). • The consequences of proposed changes. If for example a requirement were changed, how would the change propagate through the design – to functional specifications, implementations, to test plans and test cases? • Consequences of the changes made.

Name	2.1.7 Forward Tracing
	 <p data-bbox="443 801 719 898">Links used to identify inputs, requirements, verification basis</p> <p data-bbox="392 1025 1390 1122">The techniques are used in everyday project management and control of development process, but also in verification and validation as proof that the system is integrated.</p>
Resulting Context	All design elements have been connected with links to lower level elements that make it possible to do the tracing for any purpose.
Related Patterns	Backward Tracing implements the tracing to the other direction.
Standard References	
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	product, configuration, management, control, tracing

Name	2.1.8 Backward Tracing
Context	A work product, a detailed design or implementation, has been created. Now we need to track that it has properly handled all higher level issues that affect it or are assigned it. This applies to the whole work product (like a test plan) and its individual elements (like test cases).
Problem	How can we know what requirements, plans or instructions our work is based on and thus must be verified against?
Forces	When doing systematic development work, work items depend on items on a higher abstraction level and on the previous development phase. To be able to assess conformance with the “parent” items and to be able to detect changes in them that affect the current item, a mechanism is needed.
Solution	<p>The following principles need to be used:</p> <ul style="list-style-type: none"> • The work products are identified (and version controlled) • The work products are structured so that each item (e.g. each requirement or each design element) can be identified and addressed individually. This is achieved by allocating them individual identification codes in configuration and documentation systems. • All items are linked to appropriate higher level (input) work products, and thus can be shown to have a connection. • A development management tool is used, which can report the connections present, and also items that have no connections. <p>This technique is used to:</p> <ul style="list-style-type: none"> • Gain easy access to requirements and other input information on the current development item. • Assess the consequences of proposed changes. If an implementation is proposed to be changed, for example, it needs to be assessed that the new implementation meets all requirements. • Audit the current development work product. <p>The techniques are used in everyday project management and control of development process, but also in verification and validation as proof that the system is integrated and valid by structure.</p> <div style="text-align: center; margin-top: 20px;">  <pre> graph TD A["Child work product at state (version)"] -.-> B["Parent work product at state (version, configuration, completeness, acceptance)"] style A stroke-dasharray: 5 5 style B stroke-dasharray: 5 5 </pre> <p>The diagram shows two rectangular boxes. The top box is labeled "Parent work product at state (version, configuration, completeness, acceptance)". The bottom box is labeled "Child work product at state (version)". A dashed line with an arrowhead at the top points from the bottom box to the top box. To the right of this line, the text "Tracing back to requirements" is written.</p> </div>

Name	2.1.8 Backward Tracing
Resulting Context	All design elements have been connected with links to higher level (input) elements that make it possible to do the tracing for any purpose.
Related Patterns	Forward Tracing implements the tracing to the other direction.
Standard References	
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	product, configuration, management, control, tracing

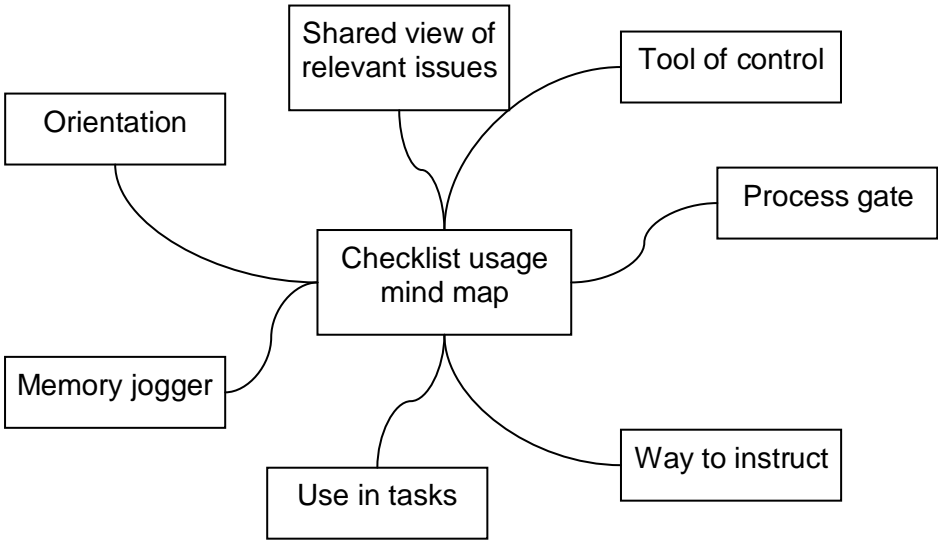
Name	2.1.9 Suspect and Prohibit
Context	Whenever something is proposed (like a design or a modification), it is important to have a suspicion that it might affect something else, might have problems, might be prone for external influences or just not be robust enough.
Problem	How do support practical suspicion and convert it into practical action? How can we know what – if anything – should be changed based on our suspicion?
Forces	Healthy suspicion is a critical element in especially safety-critical development. Without it, even formal assessments do not provide good results. It is an element of thinking that should be present in every development task.
Solution	<p>A generic suspect-action process:</p> <pre> graph TD A[Design or process artefact] <--> B[External environment] A <--> C[Internal environment] A --> D[Identify and analyse relationships] E[Use tools to show relationships Use analysis methods] --> D D --> F[Formulate suspicion hypothesis] F --> G[Do analysis to verify suspicion] G --> H[Make corrective action (change design, change environment)] G --> I[Make plans to verify suspicion in tests] </pre> <p>Some product development management tools show a view of “susceptible” relations. In some interactive tools the list is then manually edited, meaning that a conscious decision is made about what relationships to analyse, the decision can be traced later (for example during validation). These kinds of tools make the process efficient yet transparent and systematic.</p> <p>The hypotheses are analysed with techniques like FMEA or cause-consequence analysis. Corrective actions include changing plans or designs to be more tolerant and thus to prohibit any undesirable behaviours.</p> <p>This general pattern is used in every process phase, most clearly in analysing designs and modification.</p>

Name	2.1.9 Suspect and Prohibit
Resulting Context	New information based on suspicion, with which the impacts of design decisions can be formulated, designs can be changed and the robustness of designs can be improved.
Related Patterns	Impact Analysis is one manifestation of this.
Standard References	(No direct references in IEC 61508 standard series)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	workflow, analysis, impact, suspect

Name	2.1.10 Escalation of Issues
Context	A problem has been detected at some time during the lifecycle and the person detecting the problem does not possess the required authority to handle it.
Problem	How to raise the issue and have it handled at an appropriate level in the project or line organization?
Forces	There are many kinds on potential problems and if they require spending money or require technological changes, more authority may be needed to resolve the issue.
Solution	<p>An escalation procedure is utilised. It can be planned at company level instructions and as part of risk management processes, but it is implemented especially in projects and needs to be described in project plans.</p> <p>The basic flow is like this:</p> <ul style="list-style-type: none"> • An individual detects an issue that needs to be handled. If handling is in his/her power (like correcting a software defect), it is done immediately without any other action than following the correction procedure. • If, however, the individual does not have authority to resolve the issue, he/she reports the issue to his/her superior (like a team leader, or site coordinator). • The superior assesses his/her authority to solve the issue and if it is insufficient (like if the resolution requires more money than he/she is allowed to spend) in turn reports the issue “upstream” in the command chain to his/her superior. • This way the issue can and sometimes should reach the top management. • The issue may return back to the reporter with authority to proceed with action – make a large change of technology, spend money not budgeted, hire experts or other. • Or the issue is passed to another party to resolve (like sales to negotiate with customer, or a technology team to develop a new solution). • When the process is implemented in a formal workflow, it cannot be suspended based on opinion. • The process should be implemented in an information system where the issue can be forwarded to an appropriate party and all participants can have receipts / notifications of handling the issue and in general the process can be monitored.

Name	2.1.10 Escalation of Issues
	<pre> graph TD P[Potential problem] --> R1[Report & follow issue] R1 --> L1[Levels of organization with sufficient authority] L1 --> H1[Higher levels of organization or top management] H1 --> A1[Get authority and initiate action] A1 --> R2[Report & follow issue] R2 --> L2[Resolving authority? If yes, resolve, if not, pass upstream] L2 --> A2[Get authority and initiate action] A2 --> I[Individuals in team] I --> P </pre> <p>The flowchart illustrates the escalation of issues process. It starts with a 'Potential problem' at the bottom, which leads to a 'Report & follow issue' box. From there, it goes to 'Levels of organization with sufficient authority'. If not resolved, it moves to 'Higher levels of organization or top management', then to 'Get authority and initiate action'. This leads to another 'Report & follow issue' box, which then goes to a decision box: 'Resolving authority? If yes, resolve, if not, pass upstream'. If not resolved, it goes to 'Get authority and initiate action' again, which leads to 'Individuals in team', which then loops back to 'Potential problem'.</p>
Resulting Context	An issue has been resolved promptly.
Related Patterns	Transparency of Action and Information
Standard References	(No direct references in IEC 61508 standard series)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	project, organisation, risk management, escalation, process

Name	2.1.11 Use of Checklists
Context	Something needs to be checked, reviewed, remembered.
Problem	How can we remember all issues that need to be checked? How can we be sure that others remember all issues that need to be checked?
Forces	Technological systems can be complex and people in projects are sometimes busy. Various means are needed to ensure that all items are handled properly, especially when reviewing developments for their acceptance.
Solution	Checklists should be used in various situations. In general they provide a

Name	2.1.11 Use of Checklists
	<p>shared view of what is essential and thus help keep everyone's thoughts in alignment.</p> <p>Checklists can also be mandatory. This means that a project phase cannot be accepted until "all checkmarks are in place". This can be implemented in any tool, but often project management tools with such functionality are a good choice.</p> <p>Even a basic implementation workflow can be presented as a checklist: [x] Coding, [x] Analysis, [x] Testing, [] Documentation.</p> <p>Often, all reviews have tailored checklists to help ensure that all relevant issues are checked (like all quality factors of architecture or all things to present in a project plan).</p> <p>Checking the existence and readiness of all documentation is one important use for checklists in safety-critical development.</p> <p>Checklists are a traditional technique in risk analysis and safety assessment.</p> <p>Thus, checklists have plenty of uses and benefits. Note that checklists do not need to be lists. A mind map style of presentation can suit many purposes better.</p>  <pre> graph TD C[Checklist usage mind map] --- O[Orientation] C --- MJ[Memory jogger] C --- UT[Use in tasks] C --- WI[Way to instruct] C --- PG[Process gate] C --- TC[Tool of control] TC --- SV[Shared view of relevant issues] </pre>
Resulting Context	All things to check have been remembered and handled with the aid of checklists.
Related Patterns	Acceptance of Phases and Tasks
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	Matti Vuori
Status	Version 2011-04-29

Name	2.1.11 Use of Checklists
Notes	
Tags	process, tools

Name	2.1.12 Continuous Improvement																		
Context	Any activity.																		
Problem	How to improve ways of action so that in future projects are more efficient and have fewer problems?																		
Forces	Continuous improvement is a principle required by all quality management systems and standards. As safety-critical development can be challenging, improvement is an important issue and needs more attention.																		
Solution	<p>The core of continuous improvement is a rhythmic habit of reflecting on project's and teams' performance, success, problems and emerging possibilities of improvement.</p> <p>The picture shows many possible "reflection points".</p> <table border="1"> <thead> <tr> <th>Reflection point</th> <th>Viewpoint</th> </tr> </thead> <tbody> <tr> <td>Any review</td> <td>How could we improve ways to do this kind of thing?</td> </tr> <tr> <td>Risk analysis</td> <td>New risks to develop generic approaches for</td> </tr> <tr> <td>Design analysis</td> <td>New design patterns to handle problems</td> </tr> <tr> <td>Team meeting</td> <td>Any problems that require solving and thus creates improvement?</td> </tr> <tr> <td>Validation / certification</td> <td>How to make it more efficient next time?</td> </tr> <tr> <td>Phase retrospect / lessons learned meeting</td> <td>General assessment of performance. Opportunities for improvement?</td> </tr> <tr> <td>Project development audit</td> <td>General assessment of performance. Opportunities for improvement?</td> </tr> <tr> <td>Analysis of metrics</td> <td>What do any metrics say? Compared to others?</td> </tr> </tbody> </table>	Reflection point	Viewpoint	Any review	How could we improve ways to do this kind of thing?	Risk analysis	New risks to develop generic approaches for	Design analysis	New design patterns to handle problems	Team meeting	Any problems that require solving and thus creates improvement?	Validation / certification	How to make it more efficient next time?	Phase retrospect / lessons learned meeting	General assessment of performance. Opportunities for improvement?	Project development audit	General assessment of performance. Opportunities for improvement?	Analysis of metrics	What do any metrics say? Compared to others?
Reflection point	Viewpoint																		
Any review	How could we improve ways to do this kind of thing?																		
Risk analysis	New risks to develop generic approaches for																		
Design analysis	New design patterns to handle problems																		
Team meeting	Any problems that require solving and thus creates improvement?																		
Validation / certification	How to make it more efficient next time?																		
Phase retrospect / lessons learned meeting	General assessment of performance. Opportunities for improvement?																		
Project development audit	General assessment of performance. Opportunities for improvement?																		
Analysis of metrics	What do any metrics say? Compared to others?																		

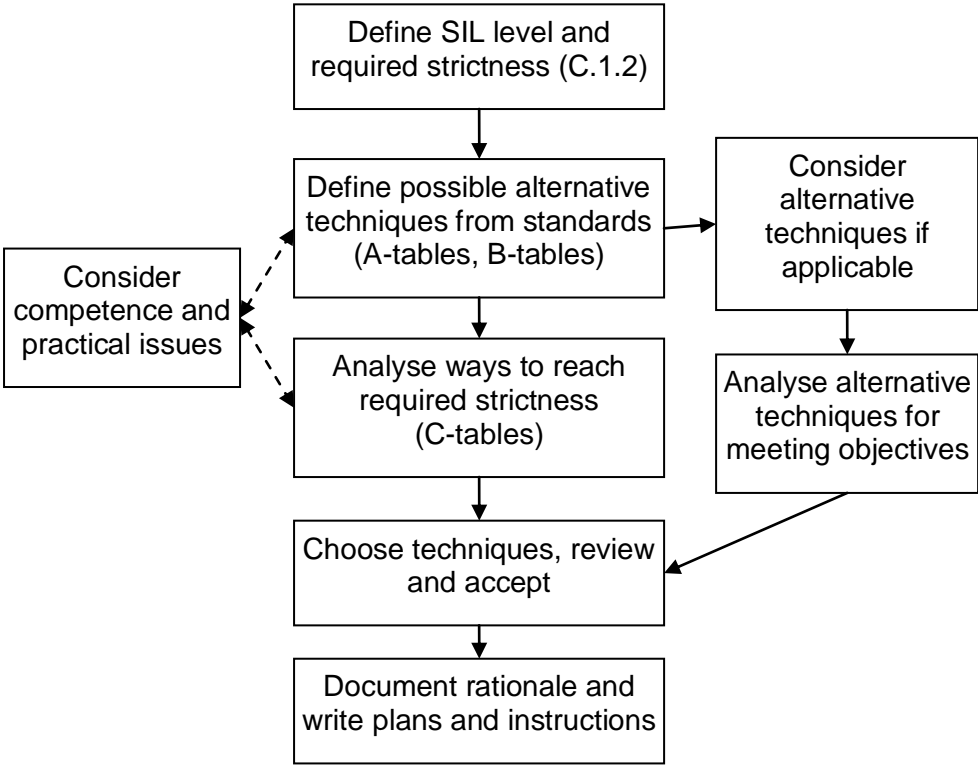
Name	2.1.12 Continuous Improvement
	<p>The analysis should lead to improvements during the same project, but also improvements in general project practices, which is why process developers should participate in these activities.</p> <p>At a larger scale, the same principles apply at company level, but that is not addressed here.</p>
Resulting Context	An improved way of action.
Related Patterns	
Standard References	(No direct reference in IEC 61508; this is more in the scope of ISO 9000 quality management standard series.)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	Continuous improvement is a core practice in most quality management system, but lately (2011) it has become more known in the agile movement by the concept of “Kaizen” (see http://en.wikipedia.org/wiki/Kaizen)
Tags	process improvement, continuous improvement, organisation, quality, ISO 9000

3 Development approaches and technologies

Name	3.1.1 Flow Between Design Levels and Tests
Context	During any phase of development before testing.
Problem	While utilising a controlled approach, how to support a constant flow of testing ideas?
Forces	Good testing is a key to a robust system. While the development needs to be a controlled activity, a holistic approach that supports holistic understanding of requirements and testing and constant evolution of testing ideas is essential for the quality of testing and for development efficiency.
Solution	<p>The development process provides smooth flows of test analysis based on many abstraction levels and techniques. Tests of any type or any test level are considered during any task.</p> <p>This is a thinking pattern that permeates all processes.</p> <p>Key elements</p> <ul style="list-style-type: none"> • Test analysis attached to any phase and task, either formally or informally. • Thinking of all test levels at any test level, if we have test ideas that might be useable. Thinking at system level even when doing detailed design. • Keeping in mind the total picture of testing – while management systems help, this is a mental pattern. • Understanding that work at different levels in design and testing is just a helping abstraction that shows us opportunities to think in different ways and to gradually cover all details. • Keeping information open and up-to-date so that dynamism between design levels can bring new ideas. • Realising that constant test design is a tool that helps in evolving the design. • Utilisation of many practices in gaining good test ideas. • Continuous flow of test case designs.

Name	3.1.1 Flow Between Design Levels and Tests
	<pre> graph TD A[Requirement or design activities at various abstraction levels (requirements, architecture, design, implementation)] --> B[Reviews] A --> C[Analyses] A --> D[Safety assessment] B --> E[Conscious test analysis to identify good tests and test cases] C --> F[Conscious collecting of test ideas and test cases] D --> F E --> F F --> G[A continuous flow of test case] G --> H[A test pool consisting of tests at any test level (module, integration, system, validation)] I[Constant evaluation and restructuring of test pool to maintain its quality and usability] <--> H H <--> J[Test management to keep track] </pre>
Resulting Context	A smooth development process without any gray areas.
Related Patterns	Constant Communication Transparency of Action and Information
Standard References	(No direct references in IEC 61508 standard series)
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	This kind of flow is essential in all development lifecycle models and supported especially in agile development.
Tags	principles, process, flow, verification, testing, test case

Name	3.1.2 Selection of Methods / Techniques
Context	When planning how a task – design, verification or other – should be carried out, one needs to select methods and techniques.
Problem	How to select methods and techniques so that the decision leads to such ways of action that lead to a safe system, fulfil the requirements of IEC 61508 standard series and can be justified before the project begins, and afterwards?
Forces	There are requirements in the standard series on how “strict” the methods used should be in any task. Therefore, the selection of methods needs to be carried out systematically.
Solution	<p>The basic selection process:</p> <ul style="list-style-type: none"> • Inputs the SIL level, based on hazard and risk analysis of the overall system. • Determination of the required “strictness” – see IEC 61508-3 (2nd ed.), chapter C.1.2. • For each development task, find a list of alternatives in corresponding tables in IEC 61508-3 (2nd ed.) Annex A and Annex B. • Analyse using IEC 61508-3 (2nd ed.), Annex C tables, which methods / techniques could give, in this particular situation, the required strictness. • Consider in the analysis the development team’s competence and tools for each method / technique and any other practical influencing factors. • Document the rationale for the decisions. • Present the selections and possible alternatives in the project plans (Project Plan, Safety Plan, Verification Plan, Validation Plan).

Name	3.1.2 Selection of Methods / Techniques
	 <pre> graph TD A[Define SIL level and required strictness (C.1.2)] --> B[Define possible alternative techniques from standards (A-tables, B-tables)] B --> C[Analyse ways to reach required strictness (C-tables)] C --> D[Choose techniques, review and accept] D --> E[Document rationale and write plans and instructions] B --> F[Consider alternative techniques if applicable] F --> G[Analyse alternative techniques for meeting objectives] G --> D H[Consider competence and practical issues] -.-> B H -.-> C </pre> <p>The flowchart illustrates the process of selecting methods and techniques. It begins with defining the SIL level and required strictness (C.1.2). This leads to defining possible alternative techniques from standards (A-tables, B-tables). From here, the process branches into two paths: one leading to 'Analyse ways to reach required strictness (C-tables)' and another to 'Consider alternative techniques if applicable'. The latter path leads to 'Analyse alternative techniques for meeting objectives', which then feeds into 'Choose techniques, review and accept'. A feedback loop 'Consider competence and practical issues' influences both the initial selection and the analysis of ways to reach required strictness. Finally, the chosen techniques are reviewed and accepted, leading to the documentation of rationale and plans.</p>
Resulting Context	A selection of methods / techniques to be used in the project has been decided.
Related Patterns	(All safety lifecycle phases) Selection of Methods / Techniques
Standard References	IEC 61508-3 (2 nd ed.), chapter C.1.2 shows how to define the required strictness of method usage IEC 61508-3 (2 nd ed.), Annex A presents possible methods / techniques IEC 61508-3 (2 nd ed.), Annex B presents in more detail possible methods / techniques
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	While the standard presents a given selection of methods / techniques, others can be used, as long as when using them, the requirements and objectives can be met.
Tags	development, techniques, tools

Name	3.1.3 Use of Formal Methods
Context	Selection of development techniques.
Problem	How to introduce formal methods into an organization?
Forces	Safety standards highly recommend use of formal methods at high SIL levels. Unfortunately formal methods are not so mature and generic as required to be used out of a box. Even worse than that, the use of these requires knowledge and skills which are not common among software professionals.
Solution	<p>Using formal methods is not that much more difficult than ordinary programming. It is only different and can be learned by practising. In safety context the required diversity gives natural possibility for experimenting.</p> <p>There are three common uses of formal methods in software.</p> <ol style="list-style-type: none"> 1. Formal specification of the requirements helps to understand them more deeply and make omissions and internal contradictions more visible. For example, a notation called Safecharts can be used to capture and inspect safety requirements independently but still along with the functional requirements. More traditional specification languages are for example LTL, TLA, Lotos, SDL and CSP. 2. With formal methods one can make abstract implementations with which various design decisions can be proven applicable. See: "Formal Methods Aided Design and Verification of Joint Behaviour". 3. Verification. This is the traditional use of formal methods, and there are two separate base technologies, model checking and theorem proving. A more general term instead of model checking is state space methods. The idea is to prove in a mathematically solid way that the (formal model of) implementation satisfies the requirements. <p>In these areas, formalisms, tools and methods varies, but all requires some sort of formal modelling and after learning one, others come more easily. Selection of the first area is thus arbitrary and can be based on the potential values which can be gained.</p>
Resulting Context	A considered application of formal methods.
Related Patterns	<p>Formal Methods Aided Design and Verification of Joint Behaviour</p> <p>Sometimes it might be feasible to use formal methods in the context of Generic Glue pattern, because formal models can be simulated and tested using test automation.</p>
Standard References	<p>IEC 61508-3 (2nd ed.), Tables A.1, A.2, A.4, and A.9 present recommend techniques for requirements specification, architecture design, detailed design, and software verification at different SIL levels.</p> <p>IEC 61508-7 (1st ed.), subsections B 2.2 and B 2.3 define related terms. IEC 61508-7 (1st ed.), subsection C 2.4 gives examples of formal methods.</p>
Authors	Heikki Virtanen
Status	Version 2011-04-29

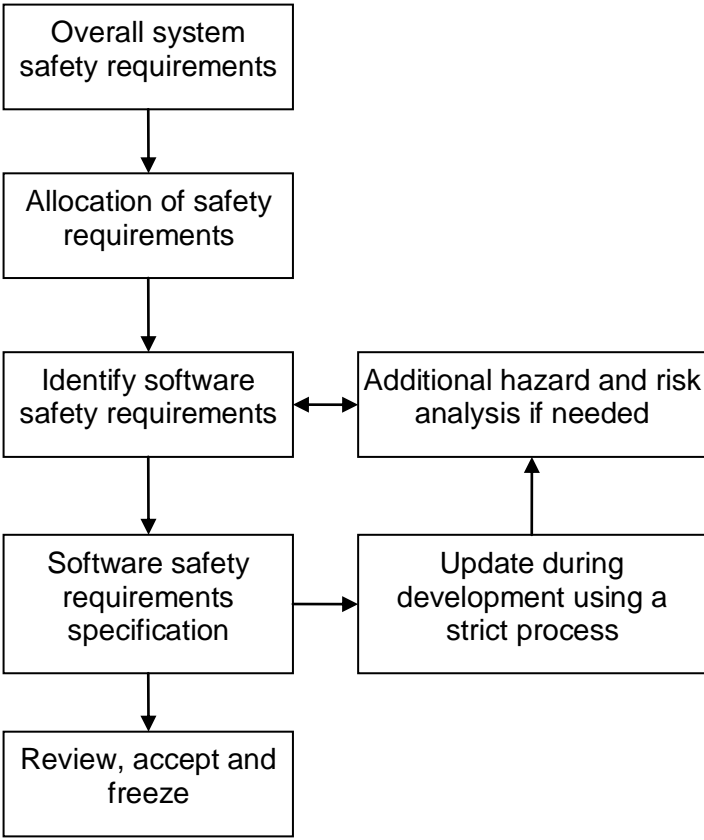
Name	3.1.3 Use of Formal Methods
Notes	<p>The term "formal methods" refers to mathematically precise notations and techniques. From the standard' perspective, they have to be used in an exact manner also. If not, or if the notation or proving methods let any ambiguities or omissions, the standard uses the term semi-formal method.</p> <p>IEC 61508-3 (2nd ed.), Table A.10, Functional safety assessment, does not directly mention formal methods, but the use of them can make assessment more robust.</p> <p>See also Wikipedia article Formal methods: http://en.wikipedia.org/wiki/Formal_methods</p>
Tags	development, techniques, tools, formal methods

Name	3.1.4 Selection of Support Tools and Development Languages
Context	The basic design and requirements for the design have been drafted. Before the design continues, the design and implementation tools need to be defined, based on the project's defined SIL level. The basic decisions are usually decided during project planning, but can be specified in detail closer to the design phase.
Problem	How to select a set of support tools and languages that fulfil safety requirements and can be proven to produce reliable results.
Forces	Tools are important as they reduce the probability of human error in design, affect the auditability of designs and implementations and robustness of software.
Solution	<p>Selection of tools based on SIL level, the development task, developers' competence and other factors.</p> <pre> graph TD A[Tools and languages selection mind map] --- B[Hardware compatibility] A --- C[Support for techniques] A --- D[Developer and subcontractor competence] A --- E[Familiarity] A --- F[Cost] A --- G[Availability] A --- H[Efficiency] A --- I[SIL level] A --- J[Experience and trust in tools] </pre>
Resulting Context	The tools have been chosen. The tools have the required documentation for use and their validation for use. The tools have been defined in the project's Safety Plan or similar.
Related Patterns	Selection of Methods / Techniques Coding Detailed Module Design
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.4.4 defines the selection process</p> <p>IEC 61508-3 (2nd ed.), table A.3 presents recommend techniques for tool selection at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.3 describes the strictness of various ways of application of techniques for tool selection</p>
Authors	Matti Vuori
Status	Version 2011-04-29

Name	3.1.4 Selection of Support Tools and Development Languages
Notes	The development organization does not usually specify tools on a project basis, but defines a toolset consisting of accepted tools to be used in all “similar” projects – based on the target system, technologies and the SIL level.
Tags	development, techniques, tools, languages

4 Software Safety Requirements Specification

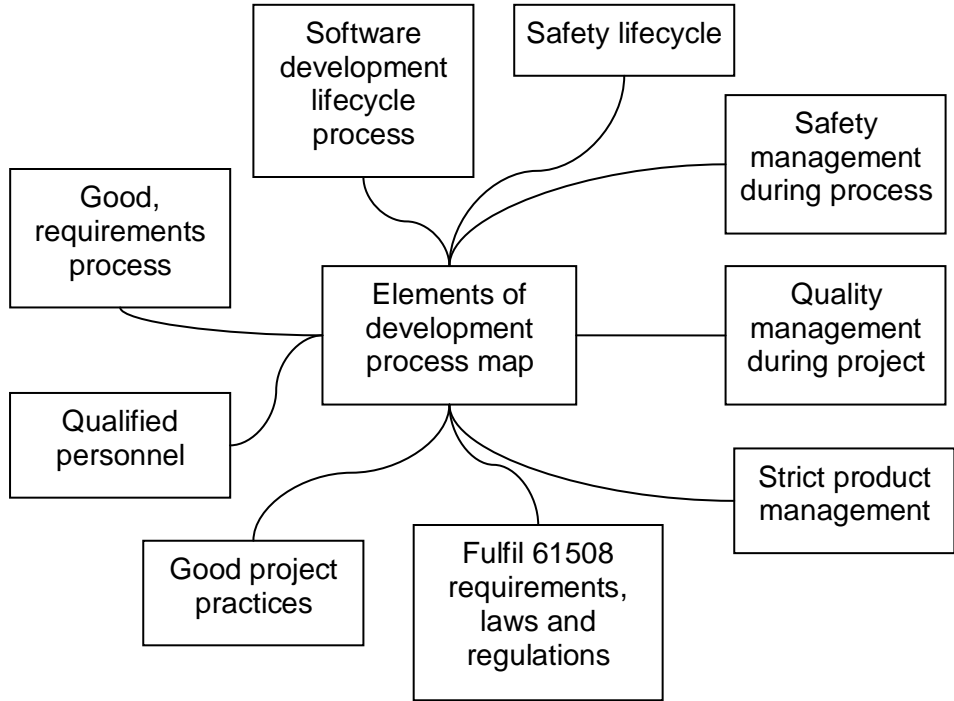
Name	4.1.1 Software Safety Requirements Specification
Context	The overall system requirements have been specified. The safety requirements have been allocated to hardware, PE and software systems. The SIL level of overall system has been defined.
Problem	How to specify safety requirements for the software system.
Forces	Understanding the safety requirements is critical for software development, as the requirements form the basis for systematic verification and validation.
Solution	<p>The safety requirements are created from the overall system safety requirements, considering the allocation of those requirements and the SIL level.</p> <p>Therefore, the safety requirements specification phase is really a design phase where we specify what the software system should do and how it should perform regarding safety.</p> <p>The basic process outline is:</p> <ul style="list-style-type: none"> • Understanding the overall system's safety requirements. • Understanding the allocation of safety related functions and requirements for hardware, E/E/PE system and software. • Extracting the thus known requirements. • Further identification and analysis of requirements. • Making additional hazard and risk analyses if needed. • Writing a Safety Requirements Specification document. • Inspection and review of the document. • Freezing the safety requirements specification. <p>The most important output of the phase is Software Safety Requirement Specification.</p>

Name	4.1.1 Software Safety Requirements Specification
	 <pre> graph TD A[Overall system safety requirements] --> B[Allocation of safety requirements] B --> C[Identify software safety requirements] C <--> D[Additional hazard and risk analysis if needed] C --> E[Software safety requirements specification] E <--> F[Update during development using a strict process] E --> G[Review, accept and freeze] </pre> <p>When using modern information systems, the resulting document may not be a traditional monolithic “document”, but a view to the safety requirements, collected from requirements assigned to various elements of the system. Thus, it can be reviewed, accepted and frozen in parts.</p> <p>Of course, some requirement specifications may change during a project for various reasons. The safety requirement specification is a critical document both for safety and for validation and thus for using the system and its changes need a very strict process.</p>
Resulting Context	Safety requirements for software have been defined, reviewed and accepted and software development can start.
Related Patterns	<p>In Software Validation Planning it is planned how the requirements will be validated, in the context of the overall system.</p> <p>In Software Validation the validation is carried out.</p> <p>Software Development continues the development branch of the V-model based on the safety requirements.</p>
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.2.2 defines the process.</p> <p>IEC 61508-3 (2nd ed.), table C.1 describes the strictness of various ways application of techniques for the requirement specification</p>
Authors	Matti Vuori
Status	Version 2011-04-29

Name	4.1.1 Software Safety Requirements Specification
Notes	For the approach of IEC 61508 and SIL levels, see Wikipedia article IEC 61508: http://en.wikipedia.org/wiki/IEC_61508
Tags	safety requirements, requirements specification, safety, risk

5 Software Design & Development

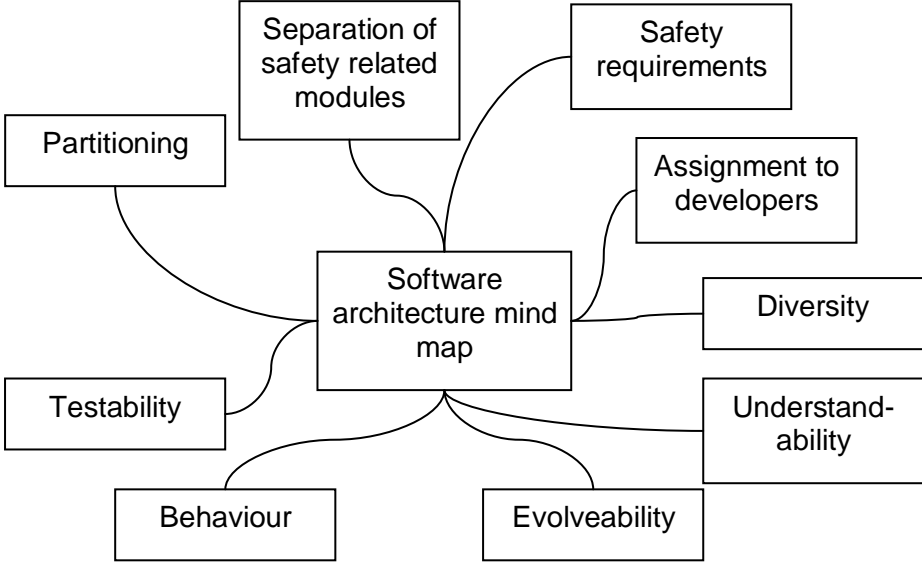
5.1 General

Name	5.1.1 Software Development
Context	Software safety requirements specification has been produced.
Problem	How to create a software system that fulfils the specified requirements with respect to the required safety integrity level?
Forces	Software design is based on software requirements and should transfer those into system specifications that can be implemented.
Solution	<p>Development needs to follow a systematic process that combines of software development lifecycle and the safety lifecycle presented in IEC 61508.</p> <p>The IEC 61508 series has an underlying assumption that a V-model based development model is used, but any model can be used as long as the organisation can fulfil the requirements of IEC 61508 with it – and produce safe software.</p> <p>Obviously, while this document concentrates on IEC 61508, the development work and process also need to fulfil other standards, regulations and laws too.</p> <p>Otherwise – just good, professional development work is carried out, guided by (among others) the elements in the next mind map.</p> 

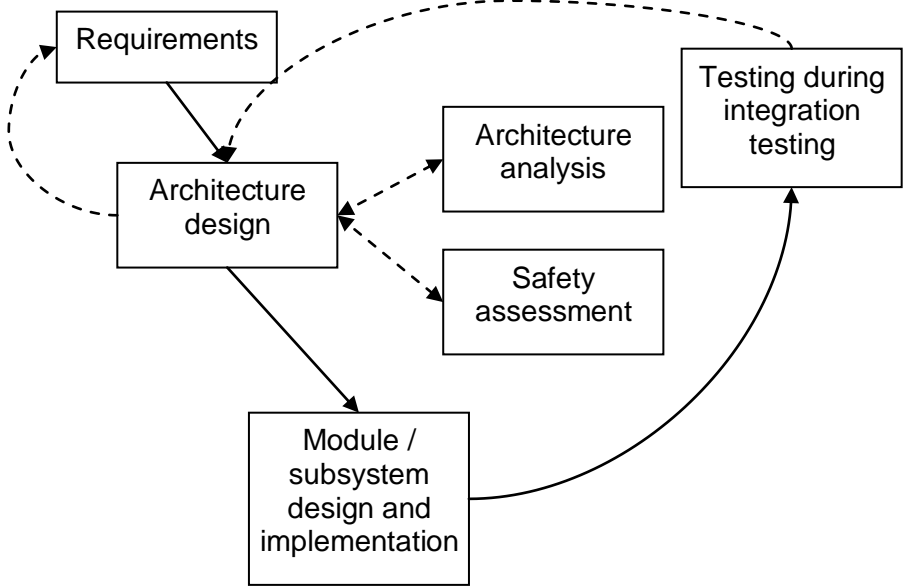
Name	5.1.1 Software Development
Resulting Context	A software system integrated into the PE system, verified to meet safety requirements.
Related Patterns	(Several)
Standard References	IEC 61508-1 (2 nd ed.) presents the general context and requirements IEC 61508-3 (2 nd ed.) presents the software development requirements
Authors	Matti Vuori, Johannes Koskinen
Status	Version 2011-04-29
Notes	See Wikipedia article Software development process: http://en.wikipedia.org/wiki/Software_development_process Safety engineering is a critical aspect in the development of safety-critical software. See Wikipedia article Safety engineering: http://en.wikipedia.org/wiki/Safety_engineering
Tags	principles, software development, process, lifecycle, safety lifecycle

5.2 Software Architecture Design & Verification

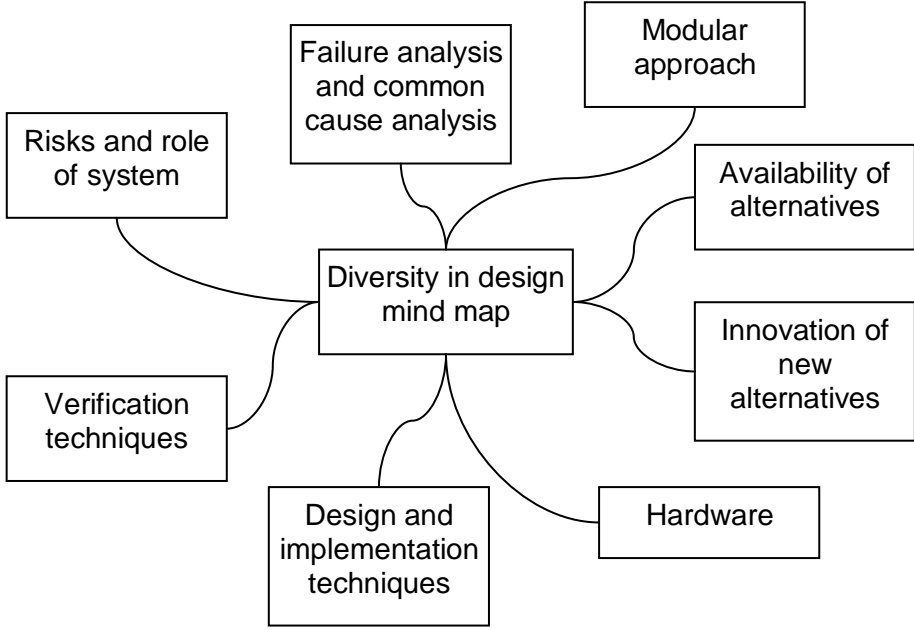
Name	5.2.1 Software Architecture Design
Context	Software safety requirements specification has been finalised and the hardware architecture of the system has been designed.
Problem	How to create a software architecture that fulfils the specified requirements with respect to the required safety integrity level?
Forces	Good architecture is a very important factor in the safety critical system.
Solution	<p>The software architecture design is based on a partitioning into elements and subsystems.</p> <p>Key issues in architecture design:</p> <ul style="list-style-type: none"> • Safety architecture is very strictly based on safety requirements and will be verified against them. • Obviously, the software architecture must be compatible with hardware architecture and those two should often be developed in close collaboration. • Safety architecture and “operational architecture” need to be kept separate, yet understanding that the operational architecture requires just as much attention as its functioning is a potential cause of hazards. • Separation of safety-related and non-safety related elements greatly helps in the verification, validation and certification of the system. • Separation should also lead to independence of the development teams that do the design and implementation of the modules. • There may be a need for diversity of design and implementation. Diversity is important for preventing common cause failures (like a failure mode or a disturbance that is not addressed in any redundant components due to similar design principles). • Besides redundancy, other means of reaching fault tolerance need to be implemented. The architecture needs to be robust against disturbances and if it fails, it should fail in a safe manner. • Diversity may be needed even in implementation tools, so the architectural ideas should not be dependent on just one approach. • The architecture should be very much independent of implementation techniques and technologies. (Even though in practice there will always be some ideas of those.) • Modularity is a key factor as it aids both in utilising diversity and redundancy in design and in verification and validation, as well as configuration management. • Simplicity and understandability is an essential factor. If a system is hard to understand, it will not be safe. • Analysability and verifiability needs to be considered as analysis is needed in design and verification already during the design phase is very important for the project to succeed. • All systems need to evolve. The above principles make modification of the system easier.

Name	5.2.1 Software Architecture Design
	<ul style="list-style-type: none"> • Based on good design, the behaviour of the architecture is predictable. • The design requires certain design and verification techniques, depending on the SIL level. • Simplicity and modularisation are essential <p>See IEC 61508-3 (2nd ed.), table A.2 about design techniques to use to reach these goals.</p>  <pre> graph TD A[Software architecture mind map] --- B[Partitioning] A --- C[Separation of safety related modules] A --- D[Safety requirements] A --- E[Assignment to developers] A --- F[Diversity] A --- G[Understandability] A --- H[Evolveability] A --- I[Behaviour] A --- J[Testability] </pre>
Resulting Context	The software architecture has been designed and software design can begin.
Related Patterns	Software Architecture Verification Formal Methods Aided Design and Verification of Joint Behaviour
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.4.3 describes the process</p> <p>IEC 61508-3 (2nd ed.), table A.2 presents recommended techniques for the architecture design at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table B.9 presents recommended principles of modular approach at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.2 describes the strictness of various ways of applying techniques in the architecture design</p>
Authors	Matti Vuori, Johannes Koskinen
Status	Version 2011-04-29
Notes	<p>See also Wikipedia article of Software architectures: http://en.wikipedia.org/wiki/Software_architecture</p>
Tags	development, architecture, design, principles

Name	5.2.2 Software Architecture Verification
Context	Software safety requirements specification has been finalised and the software architecture design has been completed by applying Software Architecture Design pattern.
Problem	How to ensure that the software architecture design adequately fulfils the software safety requirements specification?
Forces	The software architecture defines the major elements and subsystems of the software, how they are interconnected, and how the required (safety integrity) attributes will be achieved. It also defines the overall behaviour of the software, and how the software elements interface and interact. To carry on to the next phase, the information from the current software safety lifecycle phase shall be verified. All essential information from the current phase of the software safety lifecycle needed for the correct execution of the next phase should be available and must be verified. The information should include the adequacy of the specifications, design and validation plans in the current phase. The verification configuration should be precisely defined and the verification activities shall be repeatable.
Solution	<p>Software architecture verification should consider whether the software architecture design adequately fulfils the software safety requirements specification. Verification is done in many parts:</p> <ol style="list-style-type: none"> 1. During design the architecture plans are iterated. The iterations are flexibly analysed and simulated using appropriate techniques. <ul style="list-style-type: none"> • Execution of use cases through the design. • Use of failure analysis. (Common causes is an important aspect to analyse.) 2. Review of architecture. Main issues: <ul style="list-style-type: none"> • The basic software architecture design. The software architecture design should fulfil the software safety requirements specification and other solid design principles. • Attributes of elements / subsystems. The attributes of major elements and subsystems should be <ul style="list-style-type: none"> • Adequate for the safety performance required. • Testable for further verification. • Readable by the development and verification team. • Safe to modify. • Incompatibilities between design and specification. • Architecture design versus safety requirements specification • Architecture design versus integration tests. • Integration tests versus verification and validation plans. 3. Testing. <ul style="list-style-type: none"> • Integration testing is the final verification of architecture.

Name	5.2.2 Software Architecture Verification
	 <pre> graph TD Requirements[Requirements] --> ArchitectureDesign[Architecture design] ArchitectureDesign --> ArchitectureAnalysis[Architecture analysis] ArchitectureDesign --> SafetyAssessment[Safety assessment] ArchitectureDesign --> ModuleDesign[Module / subsystem design and implementation] ArchitectureAnalysis -.-> ArchitectureDesign SafetyAssessment -.-> ArchitectureDesign ModuleDesign --> Testing[Testing during integration testing] Testing -.-> Requirements Testing -.-> ArchitectureDesign </pre> <p>The diagram illustrates the Software Architecture Verification process. It starts with 'Requirements', which leads to 'Architecture design'. From 'Architecture design', the process branches into 'Architecture analysis', 'Safety assessment', and 'Module / subsystem design and implementation'. 'Architecture analysis' and 'Safety assessment' have dashed feedback arrows pointing back to 'Architecture design'. 'Module / subsystem design and implementation' leads to 'Testing during integration testing'. Finally, 'Testing during integration testing' has dashed feedback arrows pointing back to 'Requirements' and 'Architecture design'.</p>
Resulting Context	Software architecture design that adequately fulfils the software safety requirements specification.
Related Patterns	Software Architecture Design Software System Design Verification Module Integration Testing Software Validation Formal Methods Aided Design and Verification of Joint Behaviour
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.9.2.9 defines requirements for the software architecture verification.</p> <p>IEC 61508-3 (2nd ed.), sub-clause 7.4.3 defines requirements for software architecture design.</p> <p>IEC 61508-3 (2nd ed.), sub-clause 7.9 gives general requirements for software verification.</p>
Authors	Matti Vuori, Johannes Koskinen
Status	<p>A rewrite 2011-04-12 of the previously published version</p> <p>Last update 2011-04-18</p>
Notes	
Tags	architecture, verification, process

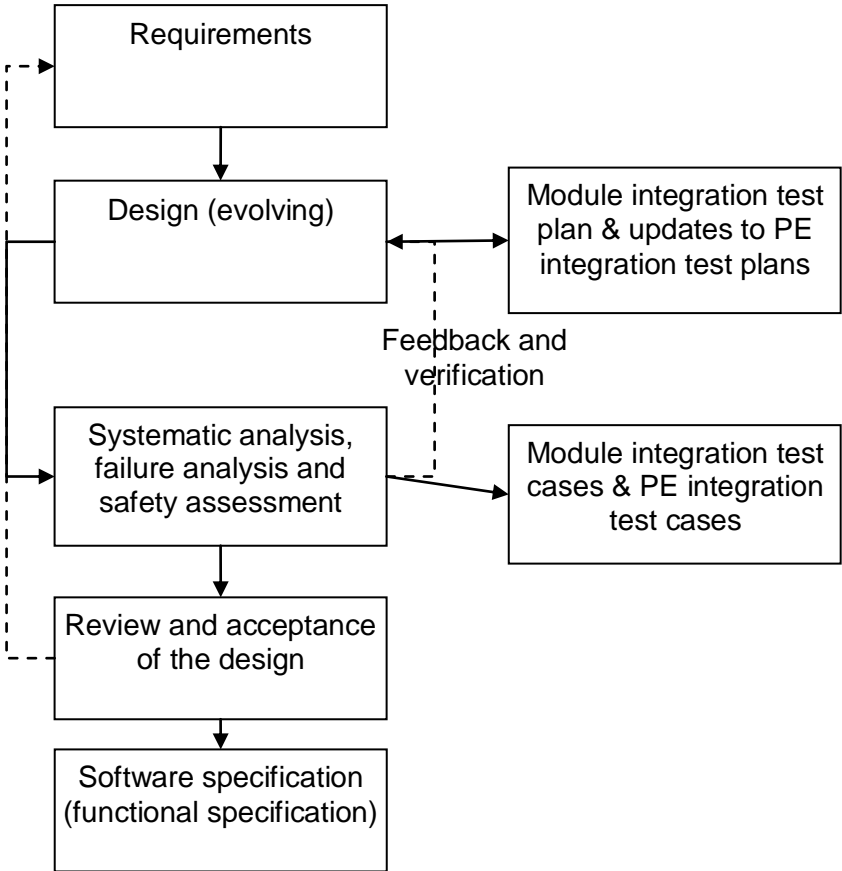
Name	5.2.3 Technical Diversity
Context	All design phases.
Problem	How to create such a system that not more than one element of it fails due to a disturbance? Or: how to avoid common cause failures?
Forces	Especially for a safety system it is essential that it does not fail even if the production system fails. And in the case of redundant systems, they must not all fail due to a similar disturbance (like a non-working communication channel), that is, there should be no possibility of common cause failures. Diversity in design and implementation is a key to this, in both hardware and in software systems.
Solution	<p>Thinking patterns:</p> <ul style="list-style-type: none"> • Keeping an open mind to alternatives. Anything can be done in various ways. • Innovation. Finding new ways require creative thinking. • Modular approach to architecture and design allows for diversity. <p>Organisational patterns:</p> <ul style="list-style-type: none"> • Independence of teams and individuals so that they can reach independent designs. • Coordination of approaches of different individuals and teams. <p>Analysis of needs for diversity:</p> <ul style="list-style-type: none"> • Requirements for diversity are based on risk – the SIL level, safety requirements and the role of a subsystem or module. • Identification of external and internal failures that would require redundancy and diversity in the redundant subsystems. <p>Technology management:</p> <ul style="list-style-type: none"> • Availability of diverse technologies and basic designs. • Creation of new alternatives in each project. <p>Examples of diversity:</p> <ul style="list-style-type: none"> • Communication technologies and channels. • Communication protocols. • Algorithms and data structures. • Languages and compilers and code generators. • Libraries and software components. • Sensors and monitoring. • Data storage. • User interfaces and access. • Verification methods and test techniques. Test automation and manual testing. Different test approaches supplement each other.

Name	5.2.3 Technical Diversity
	<p>Diversity is an issue to check in design reviews.</p>  <pre> graph TD Center[Diversity in design mind map] --- Risks[Risks and role of system] Center --- Verification[Verification techniques] Center --- Design[Design and implementation techniques] Center --- Hardware[Hardware] Center --- Innovation[Innovation of new alternatives] Center --- Availability[Availability of alternatives] Center --- Modular[Modular approach] Center --- Failure[Failure analysis and common cause analysis] </pre>
Resulting Context	A diverse system that is not prone to common cause failures.
Related Patterns	Software Safety Requirements Specification Software Architecture Design Diversity in Team Practices Failure Analysis
Standard References	<p>IEC 61508-1 (2nd ed.), sub-clauses 7.6.2.7 and 7.6.2.8 describe requirements for independence of design solutions considering common cause failures</p> <p>IEC 61508-3 (2nd ed.), table A.2 presents recommendations for diversity in architecture at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.2 presents achievable strictness for diversity techniques in architecture at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table A.10 presents common cause analysis as a technique of functional safety assessment if diverse software is used</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	principles, diversity

Name	5.2.4 Formal Methods Aided Design and Verification of Joint Behaviour
Context	There is good understanding of requirements, informal description of the expected behaviour of the system, and a sketch of the architecture.
Problem	How should the parts (components) defined in the architecture behave in order to obtain the expected behaviour of the total (sub)system?
Forces	For reactive and concurrent systems, it is almost impossible to verify architecture reliably by means of inspection or manual reasoning.
Solution	For design: <ul style="list-style-type: none"> • Modelling abstract behaviour of parts using cooperative state machines. For verification: <ul style="list-style-type: none"> • Verify design using visual verification or model checking.
Resulting Context	Formal functional models of the parts of the system, which in combination produce expected behaviour.
Related Patterns	Software Architecture Design Software Architecture Verification
Standard References	
Authors	Heikki Virtanen
Status	Version 2011-04-29
Notes	The technique is outlined in "Visualisation of Reduced Abstracted Behaviour as a Design Tool", available at http://doi.ieeecomputersociety.org/10.1109/EMPDP.1996.500586 and http://www.cs.tut.fi/ohj/VARG/publications/96-2.ps
Tags	verification, techniques, formal methods, modelling, architecture

5.3 Software System Design

Name	5.3.1 Software System Design – general
Context	The software architecture has been designed and verified. Now the process can continue in the design phase.
Problem	How to design the software so that it can be implemented, verified and validated.
Forces	This is the phase where requirements and architecture are turned into a software system that can be implemented, verified and finally validated.
Solution	<p>Based on requirements, a design for the software is drafted. Note that this same process can be used at many abstraction levels, as defined by the architecture.</p> <ul style="list-style-type: none"> • It is verified by review that the design meets all safety requirements. • The behaviour of the design is analyzed using applicable methods, like simulation, FMEA or cause and consequence analysis. This analysis is in part regular design analysis and in part formal safety assessment (see patterns Functional Safety Assessment and Failure Analysis) • Safety assessment is carried out against the design and the design is changed accordingly. This supplements other analysis activities and confirms that the design should meet the SIL level requirements. • The design is verified to meet safety requirements and architecture design. <p>The analysis is used to design test cases, to be executed during integration testing.</p> <p>This process is highly iterative. The design evolves from a preliminary draft (usually based on experience) to detailed design.</p> <p>The test plans for that design are evolving, not just by test cases, but also test approach (depending on the design situation). The testing applicable at this phase is module integration testing and PE integration testing.</p> <p>After the design has been assessed to be safe, the design and its associated plans and documentation are reviewed and its implementation can begin.</p> <p>The basic flow is shown in the next figure:</p>

Name	5.3.1 Software System Design – general
	 <p>Obviously, the design phase can lead to the proposing of changes of requirements, returning the flow back to the previous phase.</p> <p>All design elements are linked to requirements and thus tracked that all requirements assigned to that design are handled and fulfilled appropriately (as known at the design phase).</p> <p>The end result of this phase is a specification, often called functional specification.</p>
Resulting Context	The software modules have been designed, and assessed (from the design point of view) to be sufficiently safe. Implementation can begin.
Related Patterns	Software System Design Verification Detailed Module Design Functional Safety Assessment Failure Analysis
Standard References	IEC 61508-3 (2 nd ed.), sub-clause 7.4.5 describes the design process IEC 61508-3 (2 nd ed.), table A.4 presents recommended techniques for detailed software design at different SIL levels
Authors	Matti Vuori
Status	Version 2011-04-29

Name	5.3.1 Software System Design – general
Notes	
Tags	development, design, software system

Name	5.3.2 Software System Design Verification
Context	The software system design specification has been produced. Before designing the software module specification, the system design specification needs to be verified.
Problem	How to ensure that there are no incompatibilities between the software system design specification and software architecture design?
Forces	Following the architectural design decisions is critical for safety-critical development. Therefore, the software designs need to be verified against the architecture.
Solution	<p>The designs are verified to ensure that the software system design matches the software architecture design and fulfils its requirements. After integration the system is checked to ensure that it fulfils the software system design. The attributes of all the major elements of the software system are concerned.</p> <p>Verification is done in several parts:</p> <ul style="list-style-type: none"> • Analytical verification. Analysis, inspection and review of design documents and artefacts (including prototypes). • Verification by testing. This is mostly done in integration testing (all levels). • Auditing of the integrated system to see that its implementation fully matches architecture and design.
Resulting Context	Software designs that are verified and can be passed to detailed design.

Name	5.3.2 Software System Design Verification
Related Patterns	<p>Software System Design – general presents the context of this.</p> <p>Detailed Module Design describes the next phase.</p> <p>Module Integration Testing</p> <p>PE Integration Testing</p>
Standard References	<p>IEC 61508-3 (2nd ed.), clause 7.9.2 describes generic software verification</p> <p>IEC 61508-3 (2nd ed.), table A.9 presents recommended techniques used in verification at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table A.8 presents recommended static analysis techniques at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.9 describes the strictness of various ways of application of techniques used in verification</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>For generic information about verification see Wikipedia article Verification and Validation http://en.wikipedia.org/wiki/Verification_and_validation</p> <p>For generic information about software testing see Wikipedia article Software Testing: http://en.wikipedia.org/wiki/Software_testing</p>
Tags	software system, verification

Name	5.3.3 Generic Glue
Context	Before accepting the deliverable of the phase.
Problem	How can one be sure that the given requirements for the next phase are well defined i.e. complete and do not contain any contradiction?
Forces	<p>In normal circumstances, answering this kind of problem does not pay back. When issues arise later, things are simply changed.</p> <p>In safety critical software development, it is not that straightforward. It is required that the chain of prerequisites is solid and traceable. On the other hand, approved deliverables may not be changed without the formal and rather laborious modification procedure.</p>
Solution	<p>Partially the challenge can be responded to by doing extra work within the phase before approving a deliverable. The extra work can produce a prototype of the next task, where approximately the same work is done. However, the prototype is not as complete and is done more quickly using more informal and/or domain specific notations. For example, before coding the prototype task can be programming with pseudo code. Anything which can be inspected, simulated or tested and which ties deliverables of the two successive phases more tightly together, goes.</p> <p>This prototype task is a natural context for testing, too. Along the whole development process, test cases and proof obligations are found and all of them have to be satisfied before final validation and certification. Therefore there are always some tests, against which the deliverables of the prototype task can be reviewed in order to lower risk.</p> <div style="text-align: center;"> <p>The diagram illustrates the 'Generic Glue' concept. It shows a vertical flow from 'Previous phase' to 'Next phase' with a 'Glue' box in between. To the left, a solid arrow points down from 'Previous phase' to 'Next phase', with text: 'Shows what the next phase would look like', 'Allows simulation', 'Helps connect phase product elements and shows rationale'. To the right, a dashed arrow points up from 'Next phase' to 'Previous phase', with text: 'Verifies work product', 'Allows analysis'.</p> </div>
Resulting Context	Understanding that the phase product can be used with a minimum of problems

Name	5.3.3 Generic Glue
Related Patterns	Verification of a Work Product Software Modification Single Development Task Control Workflow Glue Design and Implementation is a specialised instance of this
Standard References	(No specific reference in the IEC 61508 standard series)
Authors	Heikki Virtanen, Matti Vuori
Status	Version 2011-04-29
Notes	Semi-formal models and methods are natural in this context, because they are more accurate than the natural language and not as tedious as formal models and methods.
Tags	design, glue, prototyping, semi-formal methods, domain notation, analysis, validation

5.4 Module Design and Implementation

Name	5.4.1 Detailed Module Design
Context	The overall software system has been designed. Now, the individual modules are designed.
Problem	How to develop an individual software module so that it can be implemented safely and reliably.
Forces	The detailed design phase transforms functionality into instructions for implementation. In this phase the development is prone to many traditional problems, like coding errors or similar, which may, if left unnoticed and uncontrolled, lead to system misbehaviour, failures and accidents.
Solution	<p>The detailed module design is based on higher level specification and also on safety assessments. During this phase the details of the module are designed (like protocols, algorithms, internal data structures) and the development tools are defined for the module. In practice, coding and design are performed concurrently by the same developer.</p> <pre> graph TD Req[Requirements (functional specification)] --> Design[Detailed design (evolving) and coding.] Design --> Decisions[Decisions on implementation technologies and tools.] Design --> TestPlan[Module test plan] Design --> Simulation[Systematic analysis and simulation] Simulation --> Review[Review and acceptance of the design] Review --> Spec[Detailed Module Specification] Simulation -.-> Feedback and verification Design TestPlan --> TestCases[Module test cases] TestCases --> Testing[Module Testing] Testing -.-> Feedback and verification Simulation </pre> <p>While the requirements should be frozen, the detail design will evolve, with the help of analysis and simulation, and in modern practice, also module tests are executed constantly. Simulation and testing may be combined when using</p>

Name	5.4.1 Detailed Module Design
	<p>model-based test techniques – even as simply as making a desk-top simulation with the help of a state transition diagram.</p> <p>At the beginning of the phase, as soon as the design and implementation problem has been understood, the technologies and tools (like what PLC language should be used) are selected.</p> <p>The details of coding and module testing are described separately.</p> <p>The outputs of this phase are the Module Specification, a Module Test Plan and a collection of module test cases, which are required to be executed.</p>
Resulting Context	A designed module.
Related Patterns	<p>Coding will describe the task of coding using a programming language.</p> <p>Glue Design and Implementation presents use of description that link design to implementation.</p> <p>Module Testing and Simulation will describe the testing tasks at this phase.</p> <p>Model-Based Testing</p>
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.4.6 describes the phase</p> <p>IEC 61508-3 (2nd ed.), table C.19 describes the strictness of various ways application of techniques for modular approach to design</p> <p>IEC 61508-3 (2nd ed.), table C.4 describes the strictness of various ways of application of techniques for module design and coding</p>
Authors	Matti Vuori
Status	Version 2011-05-26
Notes	
Tags	module design, detailed design

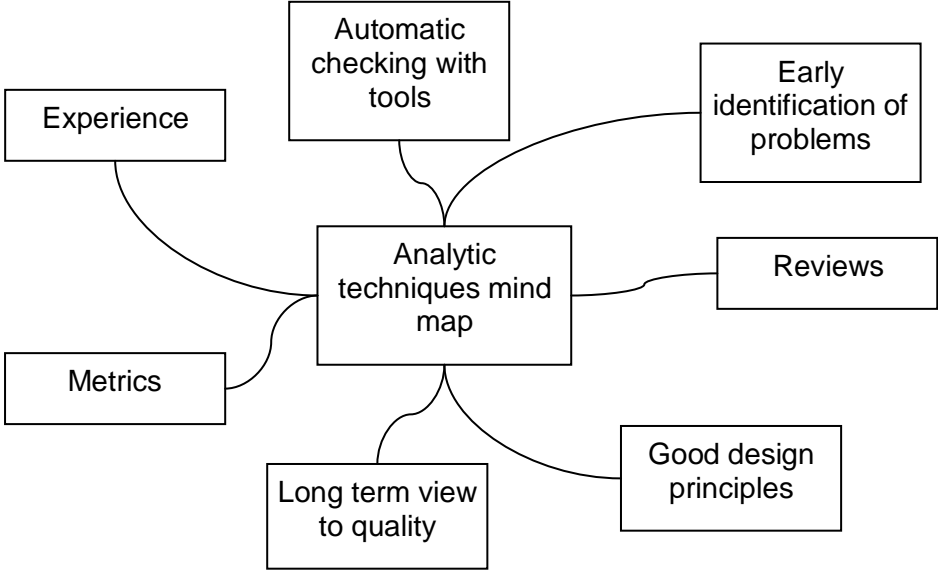
Name	5.4.2 Glue Design and Implementation
Context	Moving from the detailed design phase to implementation.
Problem	How can one be sure that design is correct, detailed enough, can be implemented with reasonable effort and without extra design decisions?
Forces	In safety critical software development the preceding phase must be completed and approved before one can move to the next phase, and tedious modification procedure have to be followed if the approved document has to be changed.
Solution	<p>The solution is an extra document which is more closely related to the end product of the next phase than the end product of the previous phase, but not so laborious to write than the end product of the next phase and does not contain all the details.</p> <p>In between detailed design and coding glue can be pseudo code or code fragments inside the design document (iterate programming).</p>
Resulting Context	More trust that the details can be successfully implemented. Good guidance for the programmers.
Related Patterns	This is an instance of Generic Glue pattern Detailed Module Design Coding
Standard References	(No direct reference in the IEC 61508 standard series)
Authors	Heikki Virtanen
Status	Version 2011-04-29

Name	5.4.2 Glue Design and Implementation
Notes	
Tags	detailed design, implementation, glue, pseudo code, coding

Name	5.4.3 Coding
Context	An individual module has been designed and now it is implemented by coding in a programming language.
Problem	How to create reliable and safe program code, which is easy to modify when the need arises and which is also easy to audit – for purpose and for safety and security.
Forces	Code is the basic block of any software system. Its quality is very critical for the quality of the system and also for the efficiency of the development process. Especially in agile development, quality of code is very important.
Solution	<p>Preliminary tasks:</p> <ul style="list-style-type: none"> • Analysis of implementability of module design (simulation, pseudo code). • Selection of a suitable language for each task. • Creation and training of coding standards. <p>Coding:</p> <ul style="list-style-type: none"> • Following of coding standards. • Aiming for understandability, maintainability and testability. • Utilising good practices, recommended by IEC 61508-3, such as defensive programming. <p>Considering testing:</p> <ul style="list-style-type: none"> • Constant consideration for module tests; continuous module test design; frequent running of tests. • Using module test automation so tests can be executed during module integration tests. • In agile development, coding is often “test driven”, meaning that tests are written first and after that the module’s code. <p>Assuring code quality:</p> <ul style="list-style-type: none"> • Code reviews especially during early phases of development. • If the language supports it, running static analysis tools against the code.

Name	5.4.3 Coding
	<pre> graph TD MD[Module design] --> C[Coding] S[Selection of languages and tools Coding standards] --> C C --> MI[Module integration] C --> MR[Code review] C --> MT[Module tests] MT -.-> C </pre>
Resulting Context	A programmed and documented module, module tested, available for low level integration.
Related Patterns	Selection of Support Tools and Development Languages Glue Design and Implementation Module Testing and Simulation Analytic Design and Code Quality Assessment Module Integration Tests
Standard References	IEC 61508-3 (2 nd ed.), sub-clause 7.4.6 describes the phase IEC 61508-3 (2 nd ed.), table C.4 describes the strictness of various ways of application of techniques for module design and coding IEC 61508-3 (2 nd ed.), table C.15 describes the strictness of various ways of application of techniques for static analysis
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	For test-driven development see Wikipedia article Test-driven development: http://en.wikipedia.org/wiki/Test-driven_development
Tags	implementation, coding, code, glue, module design

Name	5.4.4 Analytic Design and Code Quality Assessment
Context	An individual module has been designed and now it is implemented by coding in a programming language.
Problem	How to create reliable and safe program code, which is easy to modify when the need arises and which is also easy to audit – for purpose and for safety and security?
Forces	While testing is the ultimate verification method, quality of design and code needs to be assessed by human to ensure all qualities of good design and implementation.
Solution	<p>Goals:</p> <ul style="list-style-type: none"> • Using appropriate analysis techniques and tools, assess that the designs and implementations are of good quality. • Consider: good design principles, understandability, adherence to standards and rules, testability, documentation, potential problems. • Utilize: knowledge of experienced designers and testers, automatic tools. <p>Some traditional ways to do analysis:</p> <ul style="list-style-type: none"> • Design and code reviews. • Using metrics (like module size, complexity). • Static analysis with tools to identify potential problems. • Failure analysis and safety assessments. • Tools to detect adherence to architecture design, use of forbidden functions, etc. <p>Analysis target:</p> <ul style="list-style-type: none"> • Analysis requires some shared presentation. Simple modelling techniques can aid in analysis. <p>Remember:</p> <ul style="list-style-type: none"> • The earlier we find problems, the easier they are to correct. • Analysing things in suitable sized parts will make analysis more effective. • Any analysis that can be done automatically using tools should be done so, leaving human thinking for challenging issues.

Name	5.4.4 Analytic Design and Code Quality Assessment
	 <p>The diagram is a mind map with a central box labeled "Analytic techniques mind map". Six other boxes are connected to this central box by curved lines: "Experience" (top-left), "Automatic checking with tools" (top), "Early identification of problems" (top-right), "Reviews" (right), "Good design principles" (bottom-right), and "Long term view to quality" (bottom-left). "Metrics" is also connected to the central box from the left side.</p>
Resulting Context	A programmed and documented module, module tested, available for low-level integration.
Related Patterns	Detailed Module Design Glue Design and Implementation Coding
Standard References	IEC 61508-3 (2 nd ed.), table B.1 presents some recommended design and coding standards at different SIL levels IEC 61508-3 (2 nd ed.), table C.18 describes the strictness of various ways of application of techniques for static analysis
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	techniques, analytic techniques, quality, design, code

5.5 Verification Testing

Name	5.5.1 Verification Testing
Context	After a software module has been implemented or integrated, the resulting executable program is verified by testing.
Problem	How to verify programs and their components at all abstraction levels?
Forces	During the course of software system development, all work products and implementations are verified. For implemented software, testing is the most important general verification method. Due to the nature of safety-critical development, testing needs to be both high quality and highly efficient.
Solution	<p>Verification testing of software in the context of IEC 61508 (2nd ed.) mostly follows the traditional V-model.</p> <p>Verification relations are drawn with dashed lines -.--</p> <p>Key features of this scheme:</p> <ul style="list-style-type: none"> • There are several testing levels. • Testing at each level is based on a design item or the same level. • Software must pass the lower test level before testing of it at the next level can begin.

Name	5.5.1 Verification Testing
	<ul style="list-style-type: none"> • There are strict rules for passing each test level. If the software doesn't pass the testing, it needs to be corrected and the testing repeated. • If the correction requires design changes, the designs need to be updated with Software Modification process. • If the requirements or specifications that a test or test case is based on changes, the verification obtained with the tests is invalidated and the tests need to be repeated. This includes any regression testing based on impact analysis. <p>Patterns for the test levels will describe the process more.</p>
Resulting Context	<p>A software item or system that has been verified by testing.</p> <p>A software system configuration, behaviour or which is known and understood due to the testing.</p>
Related Patterns	<p>The test level patterns: Module Testing and Simulation, Module Integration Testing, PE Integration Testing</p> <p>Regression Testing</p> <p>Software Validation</p>
Standard References	<p>(See the patterns for test levels)</p> <p>IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques</p> <p>IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques</p> <p>IEC 61508-3 (2nd ed.), table B.5 presents recommended modelling techniques at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.15 describes the strictness of various ways of application of techniques for modelling</p> <p>IEC 61508-3 (2nd ed.), table C.13 describes the strictness of various ways of application of functional and black-box testing techniques</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>See Wikipedia article Software Testing:</p> <p>http://en.wikipedia.org/wiki/Software_testing</p>
Tags	verification, testing, V-model, process

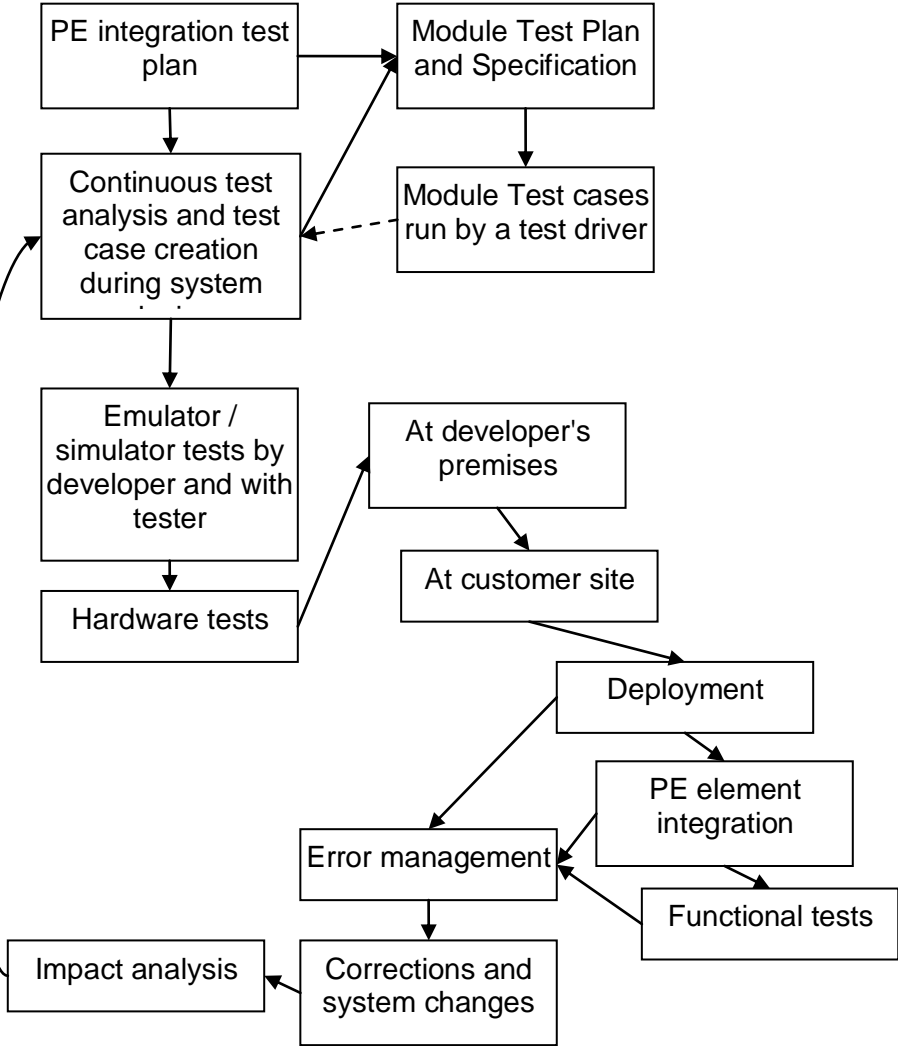
Name	5.5.2 Module Testing and Simulation
Context	A module has been tested and the quality of program code has been assessed. Now, the module will be tested individually using appropriate module testing practices.
Problem	How to test the module to verify that it meets the requirements?
Forces	A module is the basic unit of software architecture and its proper working is critical to how the system behaves though it may also be monitored and controlled by separate safety systems.
Solution	<p>Module testing is based on module design and thus the Module Specification.</p> <pre> graph TD A[Module Design Specification] --> B[Module Test Specification] A --> C[Implementation] B --> D[Module Test cases run by a test driver] D -.-> C </pre> <p>In the V-model tests are designed based on a specification and run by a test driver (a ready-made unit test tool or an ad-hoc tool) against the implementation, usually on the developer's workstation, perhaps using a hardware emulator.</p> <p>In practice, the implementation affects test case design. Tests are often implemented simultaneously with implementation. Module tests are often integrated in the developer's IDE so that they are executed automatically when the local version of the software is built.</p> <p>Module testing is an incremental process. Tests are executed and the software corrected based on tests in a flexible manner. Execution of the tests shows a need for more tests.</p> <p>After the tests (at least for implemented functionality – see Module Integration Testing for more on this) pass, the module can be passed to integration testing.</p>

Name	5.5.2 Module Testing and Simulation
	<pre> graph TD A[Module Design Specification] --> B[Implementation] A --> C[Module Test Plan and Specification] C --> D[Module Test cases run by a test driver] D -.-> B B --> E{Tests} E --> B E --> F[Module Integration] </pre>
Resulting Context	A tested module, which can be passed to integration testing.
Related Patterns	Detailed Module Design Coding Model-Based Testing
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.4.7 describes the module testing phase</p> <p>IEC 61508-3 (2nd ed.), table B.2 presents recommended test techniques suitable for module testing at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table B.3 presents recommended test techniques suitable for module testing at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.5 describes the strictness of various ways of application of techniques for module testing and integration</p> <p>IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques</p> <p>IEC 61508-3 (2nd ed.), table C.15 describes the strictness of various ways of application of techniques for modelling</p> <p>IEC 61508-3 (2nd ed.), table C.13 describes the strictness of various ways of application of functional and black-box testing techniques</p>
Authors	Matti Vuori
Status	Version 2011-05-26
Notes	<p>Module testing is often called unit testing.</p> <p>See also Wikipedia article Unit testing: http://en.wikipedia.org/wiki/Unit_testing</p>
Tags	verification, testing, module testing, simulation

Name	5.5.3 Module Integration Testing
Context	A module has been developed and is now integrated to the software system. During integration, integration tests are executed.
Problem	How to test the collection of modules in the architecture to verify that the system meets functional requirements?
Forces	Module integration testing is a task where the modules produced by various developers and subcontractors meet. Thus it is a very important first step in assuring that the system has been implemented correctly and works as planned.
Solution	<p>Modules are integrated using a pre-decided strategy, usually one at a time. Tests are executed to verify that the modules together fulfil the functional requirements and match the architecture design.</p> <pre> graph TD Architecture[Architecture] --> Implemented[Implemented modules] Functional[Functional Specification] --> Implemented Architecture --> TestPlan[Module Integration Test Plan and Test Specification] Functional --> TestPlan TestPlan --> TestCases[Integration Test cases run by a test driver] TestCases -.-> Implemented </pre> <p>Key points:</p> <ul style="list-style-type: none"> • Integration is usually carried out by a designated person responsible for it, or, in larger projects, an integration team. • As this is the first step of executing many modules together and to performing the system's functionalities (at some level) this is also a learning experience and may produce changes to specifications. • In modern software development, module integration is a continuous activity, executed every day or even when new source code is checked in the version control system. • A module needs to pass module testing before integration testing. This does not mean that the module needs to be implemented fully. "Continuous integration" is a principle where a module is integrated every time the version control system receives an updated version of it from the developer ("check in"). Thus, the module still evolves and not all of its planned tests are yet executed. • Often, module tests are executed again in the integration phase. • Integration testing is a good place to run static analyses of the source code of all integrated modules to check their quality and to assess that they implement coding standards and follow the architecture rules.

Name	5.5.3 Module Integration Testing
	<ul style="list-style-type: none"> Modelling can be used at this phase.
Resulting Context	An integrated software system, which can now be integrated into a higher level system and further verified.
Related Patterns	Software System Design – general Software System Design Verification Module Testing And Simulation Model-Based Testing
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.4.7 describes the module testing phase</p> <p>IEC 61508-3 (2nd ed.), table A.5 presents recommended techniques for module testing and integration at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table B.2 presents recommended test techniques suitable for module integration testing at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table B.3 presents recommended test techniques suitable for module integration testing at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.5 describes the strictness of various ways of application of techniques for module testing and integration</p> <p>IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques</p> <p>IEC 61508-3 (2nd ed.), table C.15 describes the strictness of various ways of application of techniques for modelling</p> <p>IEC 61508-3 (2nd ed.), table C.13 describes the strictness of various ways of application of functional and black-box testing techniques</p>
Authors	Matti Vuori
Status	Version 2011-05-26
Notes	<p>See Wikipedia article Integration Testing: http://en.wikipedia.org/wiki/Integration_testing</p> <p>About continuous integration see Wikipedia article Continuous integration: http://en.wikipedia.org/wiki/Continuous_integration</p>
Tags	verification, testing, module integration testing, integration testing

Name	5.5.4 PE Integration Testing
Context	The software modules have been integrated and now the software can be integrated with programmable electronics and tested.
Problem	How to test the integrated system so that its functioning and functional safety can be verified?
Forces	PE integration is a critical phase as it verifies that the software can actually be executed in the target hardware in a correct manner and thus it will bring to light many problems that need to be solved.
Solution	<p>Key points of the process:</p> <ul style="list-style-type: none"> • Planning and design of PE integration tests during designing (at all design levels). • The configuration of software and hardware needs to be clearly defined. <p>Goals of tests:</p> <ul style="list-style-type: none"> • Verify compatibility of hardware and software. • Verify fulfilment of software safety requirements. <p>Specific requirements for tests:</p> <ul style="list-style-type: none"> • The configuration of software and hardware needs to be clearly defined. • Tests need to be repeatable. • Test specifications need to distinguish activities that the developers can carry on at their premises and those that need to be carried out at customer's premises. • Test specifications need to distinguish integration steps a) merging of software into hardware (installation, deployment), b) PE integration such as adding sensors, c) applying the E/E/PE safety-related system to the equipment. • If there are changes to the system, impact analysis is required. • Strict error management needs to be practiced during PE integration testing. • Testing using an emulator and simulators even on the developer's desktop can be flexible and with fewer process requirements.

Name	5.5.4 PE Integration Testing
	 <pre> graph TD A[PE integration test plan] --> B[Continuous test analysis and test case creation during system] A --> C[Module Test Plan and Specification] C --> D[Module Test cases run by a test driver] D -.-> B B --> E[Emulator / simulator tests by developer and with tester] E --> F[Hardware tests] F --> G[At developer's premises] G --> H[At customer site] H --> I[Deployment] I --> J[Error management] I --> K[PE element integration] K --> L[Functional tests] L --> J J --> M[Corrections and system changes] M --> N[Impact analysis] N --> B </pre> <p>The flowchart illustrates the PE Integration Testing process. It begins with the 'PE integration test plan', which leads to 'Continuous test analysis and test case creation during system' and 'Module Test Plan and Specification'. 'Module Test Plan and Specification' leads to 'Module Test cases run by a test driver', which in turn feeds back into 'Continuous test analysis and test case creation during system'. From 'Continuous test analysis and test case creation during system', the process moves to 'Emulator / simulator tests by developer and with tester', followed by 'Hardware tests'. 'Hardware tests' lead to testing 'At developer's premises', which then leads to testing 'At customer site'. From 'At customer site', the process proceeds to 'Deployment'. 'Deployment' leads to 'Error management' and 'PE element integration'. 'PE element integration' leads to 'Functional tests', which also feeds into 'Error management'. 'Error management' leads to 'Corrections and system changes', which then leads to 'Impact analysis'. Finally, 'Impact analysis' feeds back into 'Continuous test analysis and test case creation during system'.</p>
Resulting Context	A tested integrated software + hardware system
Related Patterns	Flow Between Design Levels and Tests Impact Analysis

Name	5.5.4 PE Integration Testing
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.5.2 describes the PE integration testing phase</p> <p>IEC 61508-3 (2nd ed.), table A.6 presents recommended techniques for programmable electronics integration at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.6 describes the strictness of various ways of application of techniques for programmable electronics integration</p> <p>IEC 61508-3 (2nd ed.), table C.12 describes the strictness of various ways of application of dynamic analysis and testing techniques</p> <p>IEC 61508-3 (2nd ed.), table C.13 describes the strictness of various ways of application of functional and black-box testing techniques</p> <p>IEC 61508-3 (2nd ed.), table C.15 describes the strictness of various ways of application of techniques for modelling</p> <p>IEC 61508-3 (2nd ed.), table B.6 presents recommended techniques for performance testing at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.16 describes the strictness of various ways of application of techniques for performance testing</p> <p>IEC 61508-3 (2nd ed.), table B.4 presents recommended failure analysis techniques at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.14 describes the strictness of various ways of application of techniques for software failure analysis</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	
Tags	verification, testing, PE integration testing, integration testing, hardware

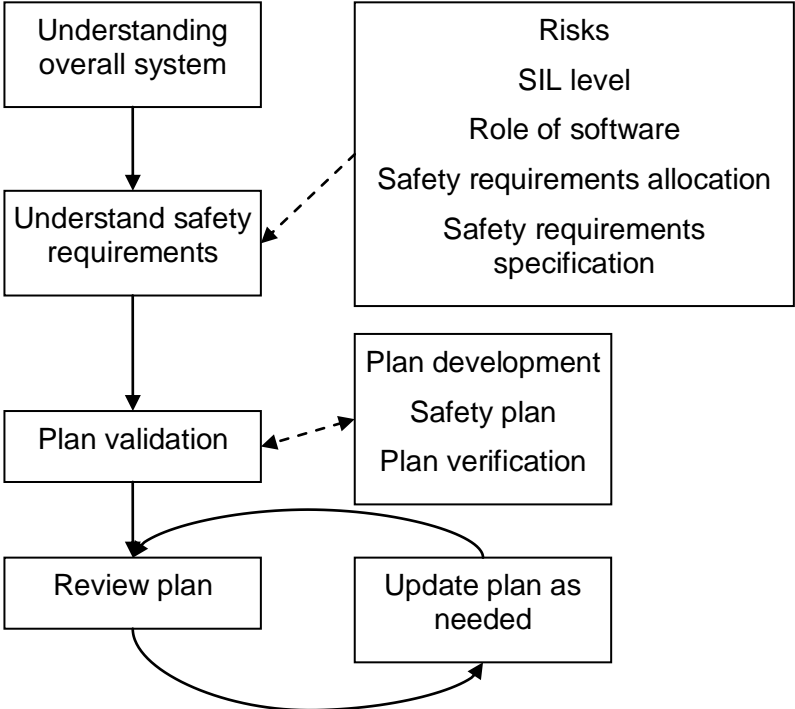
Name	5.5.5 Regression Testing
Context	Some part of software implementation has changes.
Problem	A change in software can lead to problems in other parts of the system. To identify those effects, regression testing is used.
Forces	Modern software systems are complex and prone to problems caused by even a small, seemingly trivial change.
Solution	<p>Regression testing is done at all test levels, following the V-model from module testing to high test levels, like system testing.</p> <pre> graph TD A[Changed implementation of module] --> B[Integration – resulting in changed system] B --> C[System level] D[Re-run all module tests] -.-> A E[Integration tests] -.-> B F[Specific regression tests] -.-> B G[General regression tests] -.-> C H[Specific regression tests] -.-> C </pre> <p>Regression tests can consist of “standard” regression test suites for each level and specific regression tests, designed to be used in that particular situation. These can be selected with the help of impact analysis.</p> <p>Executing regression tests is an everyday, informal activity, tightly linked to implementation. But as an official verification task, its activities need to be logged and results saved.</p> <p>A key issue in regression testing is test automation:</p> <ul style="list-style-type: none"> • Pre-programmed module tests are the first “regression safety net” and are run by the developer, usually for one module only. • During integration, integration tests are run. With continuous integration, catching regression can be immediate. • Due to automation, logging and storing of test reports is automatic. • If model-based testing is used, regression testing can be continuous as new test cases and system conditions can be generated practically indefinitely. • On the system level, manual testing has an important role and exploratory testing is often used.

Name	5.5.5 Regression Testing
Resulting Context	Software system verified for regression effects.
Related Patterns	Software Modification Impact Analysis Module Testing and Simulation Module Integration Testing
Standard References	<p>IEC 61508-3 (2nd ed.), clause 7.8.2 presents regression testing as a tool in handling changes in software change process</p> <p>IEC 61508-3 (2nd ed.), table A.8 presents recommendations for regression testing at different SIL levels in the software change process</p> <p>IEC 61508-3 (2nd ed.), table C.8 describes the strictness of various ways of regression testing in the software change process</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>See also Wikipedia article Regression testing: http://en.wikipedia.org/wiki/Regression_testing</p> <p>For Test automation see Wikipedia article Test Automation: http://en.wikipedia.org/wiki/Test_automation</p>
Tags	verification, testing, regression testing, modification

Name	5.5.6 Model-Based Testing
Context	Conformance between a specification and its implementation needs to be verified.
Problem	How to create tests that cover the specification and can be maintained with reasonable effort when the specification changes.
Forces	The number of tests required increases with the complexity of the specification. Larger the test suite, more problems there are in the test asset maintenance.
Solution	Instead of manually designing the tests, tests are automatically produced from formal models that have been created based on the specifications and the requirements. When something changes, the models can be updated accordingly and the tests regenerated to reflect the new design. Typically, 2/3 of the defects found with the approach are found already in the manual modelling phase, which entails early defect detection.
Resulting Context	Implementation verified for conformance against the specifications. Easy to modify test assets.
Related Patterns	Module Testing and Simulation Module Integration Testing Software Modification
Standard References	IEC 61508-3 (2 nd ed.), table A.5 presents techniques and measures for software module testing and integration IEC 61508-3 (2 nd ed.), table B.2 presents techniques and measures for dynamic analysis and testing IEC 61508-3 (2 nd ed.), table B.3 presents techniques and measures for functional and black-box testing IEC 61508-3 (2 nd ed.), table C.12 and C.13 describe the strictness of model-based testing as a technique
Authors	Mika Katara
Status	Version 2011-05-25
Notes	See also Wikipedia article on Model-Based Testing: http://en.wikipedia.org/wiki/Model-based_testing Another description of the technique: https://goldpractice.thedacs.com/practices/mbt/
Tags	verification, testing, modelling, modification

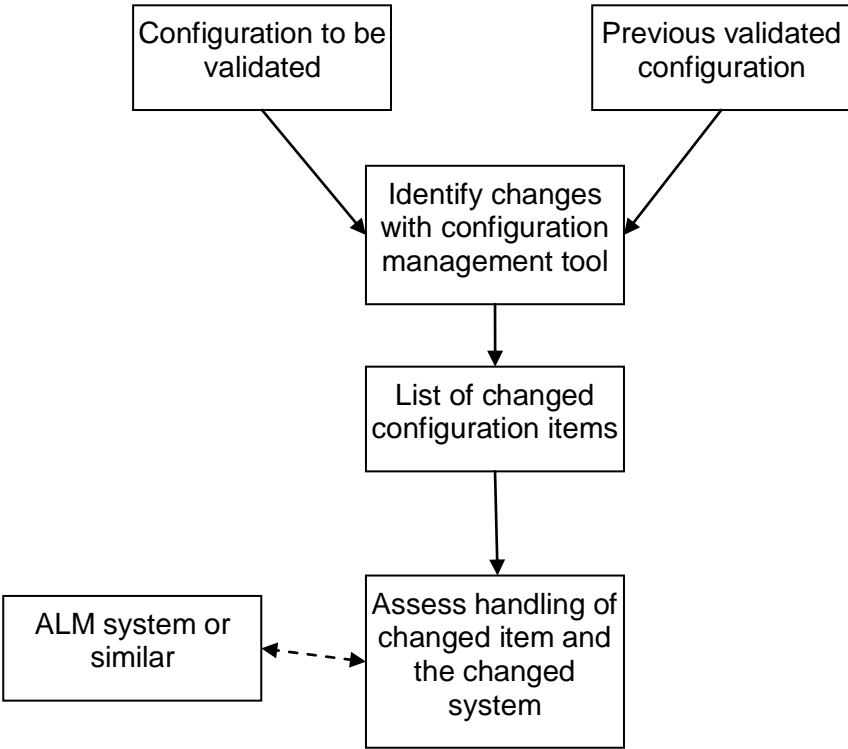
6 Software Aspects of System Safety Validation

Name	6.1.1 Software Validation Planning
Context	Software safety requirements specification has been finalised.
Problem	How to develop a plan for validating the safety-related software aspects of system safety?
Forces	Validation is a task that needs to be a planned activity so that the plans can be assessed to be sufficient for the requirements of IEC 61508 and so that the validation can afterwards be compared with the plan to see that it has been carried out properly. Validation of software is also done in the context of the overall system.
Solution	<p>The main process:</p> <ul style="list-style-type: none"> • Understand the overall context and the system and the role of software in it. • Understand the validation requirements, based on the project's SIL level. • Make a clear distinction in all plans between the validation of safety requirements and the validation of other product requirements. • Decide on the parties who make the validation, considering the required independence (for example, independent company unit, external validator) and a need for certification. • Create an overall safety plan that ensures that the development and safety assurance process is sufficient. • Plan all verification steps so that they ensure that the validation will proceed smoothly. • Plan the validation, leaving sufficient calendar time for its activities. This is usually in a form similar to a project plan. Have close collaboration in this with the party who will be doing the validation. • Consider in the plans that the validation process may not pass the first time and thus changes may need to be made and validation repeated. • Plan some coordinated collaboration with the party doing the validation so that the development process can be guided into a positive direction (yet maintaining the independence of the validator). • Review the plan with all stakeholders and ensure that everyone understands the criticality of validation – without it the product cannot be taken into use.

Name	6.1.1 Software Validation Planning
	 <pre> graph TD A[Understanding overall system] --> B[Understand safety requirements] B --> C[Plan validation] C --> D[Review plan] D --> E[Update plan as needed] E --> C E --> D F[Risks SIL level Role of software Safety requirements allocation Safety requirements specification] -.-> B G[Plan development Safety plan Plan verification] -.-> C </pre> <p>The flowchart illustrates the Software Validation Planning process. It begins with 'Understanding overall system', which leads to 'Understand safety requirements'. This step is influenced by a box containing 'Risks', 'SIL level', 'Role of software', 'Safety requirements allocation', and 'Safety requirements specification'. From 'Understand safety requirements', the process moves to 'Plan validation', which is influenced by a box containing 'Plan development', 'Safety plan', and 'Plan verification'. The process then proceeds to 'Review plan', which leads to 'Update plan as needed'. There is a feedback loop from 'Update plan as needed' back to 'Plan validation', and another from 'Update plan as needed' back to 'Review plan'.</p>
Resulting Context	A planned validation process, which can be executed when the product is ready for validation.
Related Patterns	Software Validation
Standard References	<p>IEC 61508-1 (2nd ed.), clause 7.14 describes safety validation requirements</p> <p>IEC 61508-3 (2nd ed.), clause 7.7 defines the process for system validation.</p> <p>IEC 61508-3 (2nd ed.), table A.7 presents recommended techniques for software aspects and properties of system safety validation at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.7 describes the strictness of various ways of application of the software aspects and properties of system safety validation</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>While validation plan should in an “ideal world” be based on stable requirements, things change and evolve and thus the validation plan needs to be updated during the development process.</p> <p>See Wikipedia article Verification and Validation http://en.wikipedia.org/wiki/Verification_and_validation</p>
Tags	validation, software system, software aspects, overall system

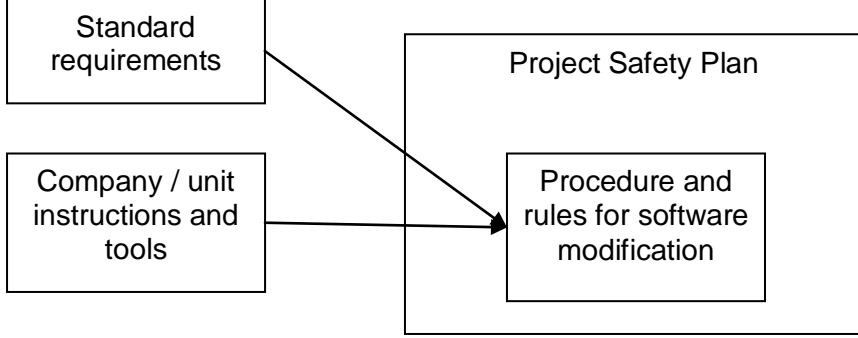
Name	6.1.2 Software Validation
Context	A plan for validating the software aspects of system safety has been developed by applying the Software Validation Planning pattern. The software for the system has been developed by applying the Software Development pattern.
Problem	How to ensure that the integrated system complies with the software safety requirements specification at the required safety integrity level?
Forces	It should be precisely defined when the software fulfils the safety requirements. The process should also be repeatable and well documented. The safety of the software part of E/E/PE safety related systems usually cannot be ensured without the underlying hardware and system environment.
Solution	<p>Validation is the final confirmation that the total system meets all the required objectives and that all the design procedures have been followed. If compliance with the requirements for safety-related software has already been established in the safety validation planning (IEC 61508-2 (2nd ed.), clause 7.7), then the validation need not be repeated. The division of responsibility should be documented during safety planning (IEC 61508-1 (2nd ed.), chapter 6), as responsibility for conformance with safety validation may rest with multiple parties.</p> <p>The validation configuration should be precisely defined and software validation should be repeatable. It should also be clear when the validation is complete and has been successfully completed. Unlike normal software validation, the software part of E/E/PE safety related systems usually cannot be validated separately from the underlying hardware and system environment.</p> <p>The validation is carried out mainly by testing. Analysis, animation and modelling can be used to supplement the validation activities. The used software tools are defined in IEC 61508-3 (2nd ed.), sub-clause 7.4.4. The requirements for safety-related software are specified in software safety requirements specification (IEC 61508-3 (2nd ed.), clause 7.2).</p> <ol style="list-style-type: none"> 1. Carry out the validation plan. The validation activities are carried out as specified in the validation plan. The validation should meet the requirements given in IEC 61508-3 (2nd ed.), sub-clause 7.7.2.7: The tools used should meet the requirements of IEC 61508-3 (2nd ed.), sub-clause 7.4.4. 2. Document the responsibilities. If the responsibility for conformance rests with multiple parties, the division of the responsibility is documented during safety planning. 3. Document the results. The results of the validation are documented. For each safety function, the following results are documented: <ul style="list-style-type: none"> • Record of validation activities as it is important to be able to retrace the sequence of activities. • The version of the validation plan used. • The safety function being validated. • The tools used. • The results. • Discrepancies between expected and actual results.

Name	6.1.2 Software Validation
	<p>4. Analyse the results. The result of validating software aspects of system safety shall be documented. If there are discrepancies between expected and actual results, the validation can be continued or aborted with a change request issued. The decision shall also be documented. To pass the validation, the tests shall show that all of the specified requirements for safety related software are correctly met and the software does not perform unintended functions. The documented results should state whether the software has passed the validation or not. In the latter case, the reasons for not passing the validation shall be documented.</p> <p>If compliance with the requirements for safety-related software has already been established in the safety validation planning, then the validation does not need to be repeated. The results of the validation of the software aspects of system safety should meet the following requirements (IEC 61508-3 (2nd ed.), sub-clause 7.7.2.9):</p> <ul style="list-style-type: none"> • The tests show that all of the specified requirements for safety-related software are correctly met and the software does not perform unintended functions. • Test cases and their results are documented for subsequent analysis and independent assessment as required by the safety integrity level. • The documented results of validation states either that the software has passed the validation or the reasons for not passing the validation.
Resulting Context	An integrated system that complies with the software safety specification at the required safety integrity level.
Related Patterns	Verification of a Work Product Software Verification Testing Software Validation Planning
Standard References	<p>IEC 61508-1 (2nd ed.), clause 7.14 describes safety validation requirements</p> <p>IEC 61508-3 (2nd ed.), clause 7.7 defines the process for system validation.</p> <p>IEC 61508-3 (2nd ed.), table A.7 presents recommended techniques for the software aspects and properties of system safety validation at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.7 describes the strictness of various ways of application of the software aspects and properties of system safety validation</p>
Authors	Johannes Koskinen; small edits Matti Vuori
Status	Published
Notes	See Wikipedia article Verification and Validation http://en.wikipedia.org/wiki/Verification_and_validation
Tags	validation, software system, software aspects, overall system

Name	6.1.3 Configuration Auditing
Context	For validation, the configuration needs to be audited to see what changes to the system have been made and how the changes have been handled.
Problem	How can we assess how the system differs from a last validated baseline?
Forces	Each change needs to be verified and validated properly and analysed for effects to the software system and also the overall system.
Solution	<p>A Configuration Audit is a starting point in determining the system's state. It is carried out by comparing the configuration to be validated against the configuration previously validated. After the configuration changes have been identified, the changes to the configuration items and the actual tasks can be traced and assessed.</p>  <pre> graph TD A[Configuration to be validated] --> B[Identify changes with configuration management tool] C[Previous validated configuration] --> B B --> D[List of changed configuration items] D --> E[Assess handling of changed item and the changed system] F[ALM system or similar] -.-> E </pre> <p>The outputs of this task are:</p> <ul style="list-style-type: none"> • A list of changed configuration items. • Access to all associated tasks – design, risk analysis, implementation, verification, safety assessment
Resulting Context	The configuration of the system, its state and changes to the (previous) baseline are known and validation can proceed.
Related Patterns	Configuration Management Software Modification Software Validation

Name	6.1.3 Configuration Auditing
Standard References	
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	For Application Lifecycle Management systems, see Wikipedia article Application lifecycle management: http://en.wikipedia.org/wiki/Application_lifecycle_management
Tags	configuration, auditing, validation

7 Software Modification

Name	7.1.1 Software Modification Planning
Context	Software modification should be planned before carrying out the actual modification activities.
Problem	How to plan modification of the software so that the modification activities can be performed safely and so that the resulting product is fully understood and can be validated?
Forces	Software modification changes the validated configuration that designs and implementations are based on. Thus, it invalidates aspects of the system and may cause a lot of work and unexpected problems if not carried out in a systematic, risk-mitigating manner.
Solution	<p>Software modifications are carried out using a well planned systematic process.</p>  <pre> graph LR SR[Standard requirements] --> PRSM[Procedure and rules for software modification] CI[Company / unit instructions and tools] --> PRSM subgraph PSP [Project Safety Plan] PRSM end </pre> <p>Often, a company or unit has instructions for the modification process which consider the information systems and other tools used. The modification process is integrated or linked in the project's Safety Plan.</p>
Resulting Context	A plan for making modification in the software. The process can be project-based, created during safety planning or a general instruction for the software development unit or company and chosen for the project with a clear, documented decision.
Related Patterns	The plan is executed in the Software Modification pattern.
Standard References	<p>IEC 61508-3 (2nd ed.), sub-clause 7.8.2 present software modification principles</p> <p>IEC 61508-3 (2nd ed.), table A.8 presents recommended techniques and properties used in modifying at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.8 describes the strictness of various ways of application of techniques and properties used in modifying software</p>
Authors	Matti Vuori
Status	Version 2011-04-29

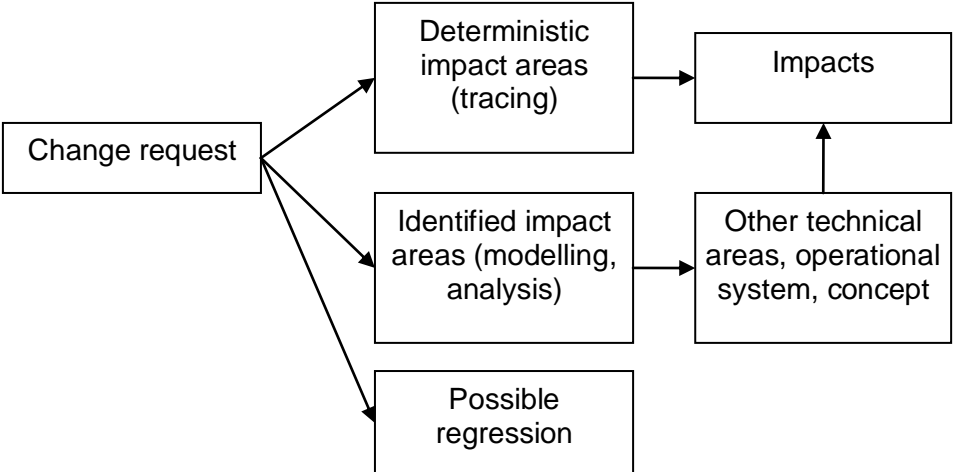
Name	7.1.1 Software Modification Planning
Notes	
Tags	modification, software modification, impact analysis, planning

Name	7.1.2 Software Modification
Context	There is an E/E/PE safety-related system where Software Validation has been applied. The software may need modifications, for example because of modifications to the overall safety requirements.
Problem	How to ensure that the required software systematic capability is sustained when the validated software is modified?
Forces	Based on IEC 61508-3 (2 nd ed.) standard, software cannot be maintained – it is always modified. When the validated software is modified (e.g. corrected or enhanced), it should be ensured that the functional safety is appropriate after the modification. Modification should be complete and correct with respect to its requirements. Unwanted behaviour caused by the modification shall be avoided. In addition, the design of the modified software should be verifiable and testable.
Solution	<p>A systematic modification procedure goes in the following way:</p> <ul style="list-style-type: none"> • A change request is made. • Impact analysis is made for the proposed modification. • It is determined whether a hazard and risk analysis is required and which software safety lifecycle phases will be repeated. It may even be necessary to implement a full hazard and risk analysis. • The modifications and verification are planned in detail. The planning should cover the identification of required staff, the detailed specification for the modifications, verification, and testing of the modifications (see IEC 61508-1 (2nd ed.), chapter 6). • After the planning, the modifications are carried out as planned. • During the modification, analysis and safety assessment are applied in the same manner as when developing a new functionality. • The modification will result in changed configuration and will require verification, regression testing and re-validation (unless it is a non-safety related feature). • Details of all modifications, including log files and re-verification and re-validation of data and results, shall be documented.

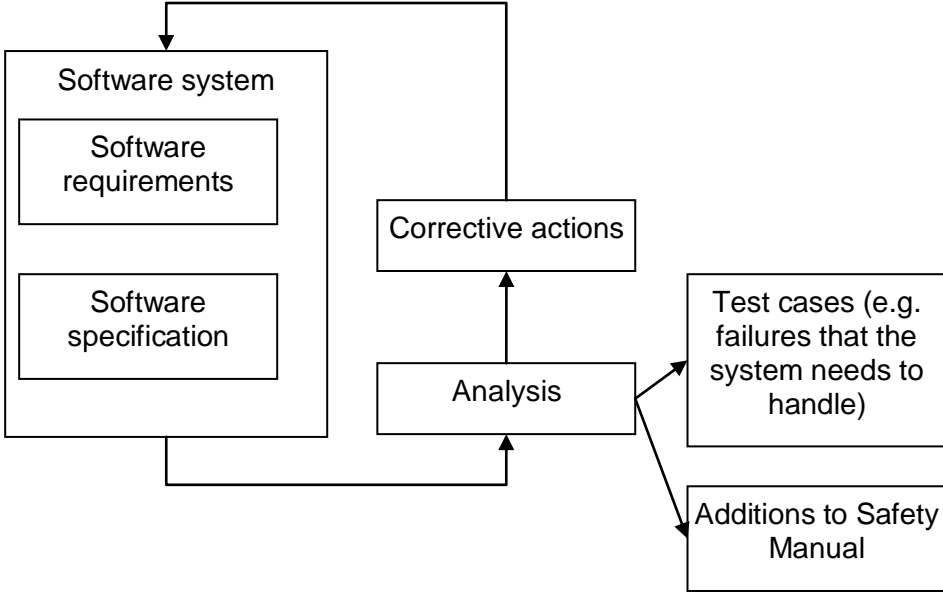
Name	7.1.2 Software Modification
	<pre> graph TD CR[Change request] --> IA[Impact analysis] IA --> AHR[Assess need of hazard and risk analysis update] AHR --> RAR[Risk analysis and return to previous phases] AHR --> DDI[Detailed design and implementation] DDI --> ASA[Analysis and safety assessment] ASA -.-> AHR DDI --> V[Verification] V --> RT[Regression testing] V --> CC[Changed configuration] RT --> RV[Re-validation] CC --> RV </pre>
Resulting Context	A modified software system, verified appropriately. A changed configuration. Information for re-validation of the system.
Related Patterns	Software Modification Planning Impact Analysis Regression Testing Model-Based Testing
Standard References	<p>IEC 61508-3 (2nd ed.), clause 7.8 describes guidelines to corrections, enhancements or adaptations to the validated software.</p> <p>IEC 61508-1 (2nd ed.), clause 7.16 defines software modification procedures.</p> <p>IEC 61508-3, table A.8 presents recommended techniques and properties used in modifying at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.8 describes the strictness of various ways of application of techniques and properties used in modifying software</p>
Authors	Johannes Koskinen, Matti Vuori

Name	7.1.2 Software Modification
Status	Version 2011-05-26
Notes	Wikipedia article Change management (engineering) describes in generic level the change management process: http://en.wikipedia.org/wiki/Change_management_(engineering)
Tags	modification, software modification, impact analysis

Name	7.1.3 Impact Analysis
Context	When a change is considered to any element of software requirements, architecture, design or implementation, its impact on the system and safety needs to be assessed.
Problem	How can we best assess how a proposed change impacts the system?
Forces	In any system, of any complexity, changes impact many other things, even outside the scope of the change. We need to identify where and how the change impacts so we can verify and validate the system after the change. Based on the analysis we may see that the change may not be feasible due to the amount of work or increased risk, compared with the benefits.
Solution	<p>The analysis consists of many tasks:</p> <ul style="list-style-type: none"> • Usually, the proposal for change (change request) includes a “pre-understanding” of what impacts the change might have. After that, an expert or a small team can consider the impact. • Tracing of design links. When, for example, a requirement changes, we need to look into what designs and implementations it may invalidate or affect. Forward tracing is used for that. • Effects of code changes. Where a code module or function is used or linked to, can be analysed with special analysis tools, or manually. All other modules will need to be looked into for known affects and also for unknown regression effects. • Decision of whether the change would cause a need for hazard and risk analysis (especially if it changes a safety-related functionality of the system, the risks of which would need to be reanalysed). <p>Tools:</p> <ul style="list-style-type: none"> • Modern tools will greatly help in analysing the impact as they can automatically report all known relations between the part to be changed and other parts of the system. • When using model-based development, simulations can be used in analysing the impact. • A fishbone diagram is a traditional tool to assess how changes would affect all parts of the production system: technology, people, operation, processes, overall system, etc. Other analytical tools can be suitable, depending on the situation. This analysis helps in understanding the safety percussions and need for a formal safety analysis. • Many companies have a checklist to aid in identifying impacts and also in reviewing the change proposal in question.

Name	7.1.3 Impact Analysis
	 <pre> graph LR CR[Change request] --> DIA[Deterministic impact areas (tracing)] CR --> IIA[Identified impact areas (modelling, analysis)] CR --> PR[Possible regression] DIA --> IMP[Impacts] IIA --> OTS[Other technical areas, operational system, concept] OTS --> IMP </pre> <p>The outputs of Impact Analysis include:</p> <ul style="list-style-type: none"> • A report, to be used in accepting the modification presenting the impacts, the specification, design and implementation required (or to be repeated) and how the change would affect safety; especially whether a safety analysis would be required. • Possibly a list of software components that should be regression tested due to the change. Understanding of this will be updated when the change gets into detailed design.
Resulting Context	A change request fully understood for its impact.
Related Patterns	Software Modification
Standard References	IEC 61508-3 (2 nd ed.), sub-clause 7.8.2.3
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	<p>Impact analysis also a tool to help project stakeholders to understand that there are no trivial changes, especially in safety-critical development. A simple looking change in code or user interface requires a lot of analysis, testing (including regression tests), documentation and other work.</p> <p>See also Wikipedia article Change Impact Analysis: http://en.wikipedia.org/wiki/Change_impact_analysis</p>
Tags	impact analysis, modification, software modification

8 Functional Safety Assessment

Name	8.1.1 Functional Safety Assessment
Context	A software design has been drafted and its safety needs to be assessed
Problem	How to analytically assess the functional safety of software?
Forces	Safety assessment is obviously a very critical phase in safety-critical development.
Solution	<p>Safety is assessed in numerous ways that complement each other. It is based on the requirements and specification of the system. Through analysis, a team will assess the behaviour of the software system and identify possible safety issues. Corrective actions are planned for these. The assessment also produces test cases that verify and validate the correct handling of system failures and disturbances identified in the analysis.</p>  <pre> graph TD subgraph Software_System [Software system] SR[Software requirements] SS[Software specification] end Analysis[Analysis] CA[Corrective actions] TC[Test cases (e.g. failures that the system needs to handle)] ASM[Additions to Safety Manual] SS --> Analysis Analysis --> CA CA --> Software_System Analysis --> TC Analysis --> ASM </pre> <p>Key points:</p> <ul style="list-style-type: none"> • The object of analysis can be the whole software system or a subsystem. Thus, the analysis should be carried out at various abstraction levels and repeated many times during the course of development. • While the analysis is based on specification, it requires realistic understanding of the environment where the system is used, possible deviations and disturbances, etc. • Systematic methods / techniques are used. Those include checklists and failure analysis techniques like FMEA, Cause and Consequence Analysis. IEC 61508-3 (2nd ed.), tables A.10 and C.10 present possible techniques. • The analysis team benefits from the participation of people who have not participated in the design of the system analysed, as they have less

Name	8.1.1 Functional Safety Assessment
	<p>presuppositions and can more easily identify possible issues.</p> <ul style="list-style-type: none"> • The analysis needs to be documented. • Depending on the time of analysis, the corrective actions may require processing through the Software Modification process pattern.
Resulting Context	An analysed software system or subsystem. A number of corrections to specifications or requirements. Information to be included in the Safety Manual of the product. Test cases that verify handling of disturbances.
Related Patterns	Software System Design Failure Analysis
Standard References	<p>IEC 61508-3 (2nd ed.), chapter 8 describes the principles of safety assessment</p> <p>IEC 61508-3 (2nd ed.), table A.10 presents recommended techniques for functional safety assessment at different SIL levels</p> <p>IEC 61508-3 (2nd ed.), table C.10 describes the strictness of various ways of application of techniques for functional safety assessment</p>
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	For safety analysis see Wikipedia article Safety Engineering: http://en.wikipedia.org/wiki/Safety_engineering
Tags	safety assessment, functional safety assessment, safety, risk, assessment

Name	8.1.2 Failure Analysis
Context	Failure analysis is a generic paradigm used in analysing identified failures (in testing or operation) or analysis of designs of how they handle failures.
Problem	<p>How can we analyse software errors and system failures in order to prevent them occurring again?</p> <p>How can we understand how the system handles failures and whether it does it properly?</p> <p>How can we understand how failures propagate through the system?</p>
Forces	Failures are a very critical issue to handle correctly in all safety-critical systems and thus analysing them requires a systematic approach.
Solution	<p>Systematic analysis in the design phase, using techniques like:</p> <ul style="list-style-type: none"> • Cause consequence analysis. • Event tree analysis. • Fault tree analysis. • Software functional failure analysis. <p>The picture shows some generic ways of how failure analysis helps understand the system and how to make it more robust.</p> <pre> graph TD Experience[Experience] --> SubsystemFailures[Subsystem failures] ModellingSystems[Modelling of systems and interactions] --> SubsystemFailures ModellingSystems --> Handling[Handling] SubsystemFailures --> Handling SubsystemFailures --> Disturbances[Disturbances] ModellingEvents[Modelling of events] --> CommonCauses[Common causes] CommonCauses --> Handling CommonCauses --> CorrectiveActions[Corrective actions] Disturbances --> Handling Handling <--> Safety[Safety?] Safety --> CorrectiveActions </pre>
Resulting Context	With the help of analysis, the system and its development are understood better. Failures identified in testing can be avoided.
Related Patterns	Functional Safety assessment Suspect and Prohibit

Name	8.1.2 Failure Analysis
Standard References	IEC 61508-3 (2 nd ed.), table B.4 presents recommended failure analysis techniques at different SIL levels IEC 61508-3 (2 nd ed.), table C.14 describes the strictness of various ways of application of techniques for software failure analysis
Authors	Matti Vuori
Status	Version 2011-04-29
Notes	As an example of analysis approach see Wikipedia article Failure mode and effects analysis: http://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis
Tags	failure analysis, analysis, design analysis, failures, suspect, safety assessment, FMEA, Cause consequence analysis, fault tree analysis, event tree analysis

9 Software Operation & Maintenance Procedures

Name	9.1.1 Writing of the Safety Manual
Context	During the course of software system development, its use is understood and possible safety issues are identified, as well as rules to follow in operation to ensure safety.
Problem	How can we communicate our knowledge of safe use to the users?
Forces	Every system needs instructions for use. In safety-critical development, safety issues have an important role in the instructions.
Solution	<p>The safety related issues are presented in a Safety Manual and other operation manuals.</p> <pre> graph TD A[Hazard and risk analysis] --> B[Software related hazards] B --> C[Software safety requirements] B --> D[Functional specification] B --> E[Safety Manual for operation] C --> D D --> F[Functional safety assessment] F --> G[Safety issues] G --> E H[Other documentation (Operating instructions, installation, configuration...)] --> E </pre> <p>Key points:</p> <ul style="list-style-type: none"> • The information included in the Safety Manual is collected in all phases of the development. • Writing of the manual should be clear and concise, as the manual should be understood by its users. • When the software is embedded, much of its safety information is expressed through the Safety Manuals of the PE system and the overall system.

Name	9.1.1 Writing of the Safety Manual
	<ul style="list-style-type: none"> All safety issues that can be resolved by technical means should be resolved so, and not by warnings in the Safety Manual, which is the last resort of handling hazards. User documentation is an important part of the overall system and is addressed in validation.
Resulting Context	A software system which is documented for operation so that it can be safely applied in the context of the overall system.
Related Patterns	Software Safety Requirements Specification Functional Safety Assessment
Standard References	<p>IEC 61508-1 (2nd ed.), table A.3 presents an example of a documentation structure for information related to the software safety lifecycle</p> <p>IEC 61508-1 (2nd ed.), Annex D describes the requirements for the Safety Manual</p>
Authors	Matti Vuori
Status	Updated 2011-04-26
Notes	
Tags	safety manual, operation, maintenance, instructions, safety, risk



Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland