Jacques Hérard, Jan Tegehall, Andreas Söderberg, SP
Timo Malm, Maarit Kivipuro, VTT

Machine Control Systems – Safety Life Cycle and Probability

Guidelines for System Validation

Abstract

Machine control systems – Safety Life Cycle and Probability
Guidelines for system validation

The current IEC 62061 standard, under development, will close the gap between the IEC 61508 standard and the EN 954 standard by giving specific requirements for the development and design of safety-related electrical, electronic and programmable control systems (SRECS) for machines.
This report will provide some support to designers in the development of machinery control systems to fulfil the safety requirements according to the IEC 62061 standard.

Key words: Electrical, electronic and programmable electronic control systems, failure rate, IEC 62061, safety, programmable systems, safety-related control systems, safety integrity level, safety in machinery, validation.

SP Sveriges Provnings- och Forskningsinstitut
SP Rapport 2002:35
ISBN 91-7848-926-1
ISSN 0284-5172
Borås 2003

SP Swedish National Testing and Research Institute
SP Report 2002:35

Postal address:
Box 857,
SE-501 15 BORÅS, Sweden
Telephone: +46 33 16 50 00
Telex: 36252 Testing S
Telefax: +46 33 13 55 02
E-mail: info@sp.se

Contents

Abstract		2
Contents		3
Preface		4
Summary		5
1	**Introduction**	7
2	**Scope**	7
3	**Definitions**	7
4	**System analysis**	12
4.1	Background, tools and methods	12
4.2	When to use the methods	33
5	**Failure rate calculation**	39
5.1	Background	39
5.2	Reliability block modelling	45
6	**Validation of the safety lifecycle**	51
6.1	Validation of development process	51
6.2	Validation of the implemented system	53
6.3	Validation results	55
6.4	Analysis of results	56
7	**Conclusion**	57
Appendix A		58
Appendix B		61

Preface

This publication is the result of the project " Validation of programmable control systems in machine automation" (VAPOCO). The work was made in co-operation between SP Swedish National Testing and Research Institute (SP) and VTT Industrial Systems (VTT).

The SP group consisted of Andreas Söderberg, Jan Tegehall and Jacques Hérard. The SP group has contributed in the presentation of analysis and design methods that are applicable in the realisation of safety features. Focus on our work was set on the validation process and on the evaluation of safety functions integrated in the system under validation. The Swedish company Tetra Pak has supported the SP group in its work.

During the project the SP group had the opportunity to discuss different issues treated in the IEC 62061 standard with Johnny Persson of Tetra Pak. It has also been possible to present our comments and to have a presentation of the concepts of the standard during a meeting organised at Tetra Pak. The Swedish part of the project was funded jointly by the Swedish Agency for Innovation Systems (VINNOVA) and by SP.

The Finnish part of the project was realised by Timo Malm and Maarit Kivipuro from VTT Industrial Systems. Other participants and supporters were Timo Sorsa from KCI Konecranes International Oy, Hannu Heman from Metso Minerals Oy, Jouni Törnqvist from Bronto Skylift Oy, Juha Herrala from Sandvik Tamrock Oy, Matti Sundquist from Ministry of Social Affairs and Health, Risto Tuominen, Helge Palmen and Marita Hietikko from VTT Industrial Systems. Risto Tuominen helped in assessing the fault trees and in computing fault trees with a commercial program. Helge Palmen helped in calculating failure rates of components according to MIL-HBK 217F and SN 29500. Marita Hietikko helped in the validation methods. The case study was provided by KCI Konecranes International Oy. The Finnish part of the project was funded by the Finnish Work Environment Fund, VTT and the participating companies.

Summary

The present work presents a list of methods and techniques applicable during the design of safety-related electronic control systems (SRECS). This work also addresses techniques for failure rate calculation and provides some guidance for the evaluation of the safety life cycle of SRECS, according to the IEC 62061 standard.

1 Introduction

The Machinery Directive states the essential health and safety requirements that all equipments in the machine sector shall comply to. The purpose of harmonised standards is to draw up the specific requirements for the current technological field. However, the constant development of modern technology implies that the standards are developed as well, so that new systems can be compatible and comparable with each other and with older systems. This is also the case with safety-related programmable control systems for which the only applicable standard is the IEC 61508 standard. This standard is relatively new and is not fully used by the manufacturers.

Many manufacturers use the EN 954 standard when developing safety-related electrical control systems (SRECS), but the EN 954 standard applies only to non-programmable safety-related systems. The succeeding standard ISO 13849 is only partly applicable to programmable systems.

The current IEC 62061 standard, under development, will close the gap between the IEC 61508 standard and the EN 954 standard by giving specific requirements for the development and design of SRECS for machinery. It is therefore important that the IEC 62061 standard is easy to understand by manufacturers and designers. The goal of this report is to give complementary assistance and support for the implementation of the IEC 62061 standard.

The validation of safety in programmable systems is usually considered to cover all phases of the safety life cycle of a programmable system. The reason is that in programmable control systems, design failures are much more important than stochastic failures and therefore methods, that are related to the design process are important. Another reason is that programmable systems can be so complex that it is not possible to consider and validate the system completely at the end of the design process.

Programmable control systems are therefore validated by using several methods during the whole design process in order to minimise design failures. The methods related to phases posterior to the actual validation of the system are not considered here.

2 Scope

There is a need of a specific standard for design and development of electrical, and programmable electronic control systems in machinery. For SRECS the generic standard IEC 61508 is applicable but is not sector specific. All necessary requirements needed by machinery system developers are not available. This is the reason why the IEC 62061 standard is being developed.

This document gives a list of tools that can be applied when designing and validating programmable control systems in machinery. The list includes analysis methods, which are part of the design process. These analysis methods support the validation process during the different phases of the programmable system life cycle.

3 Definitions

Channel: An element or a group of elements that independently performs a function

Common cause failure: Failure, which is the result of one or more events, causing coincident failures of two or more separate channels in a multiple channel system, leading to system failure

Dangerous failure: Failure which has the potential to put the safety-related system in a hazardous or fail-to-function state

Demand mode of operation : a way in which a safety-related electrical control system is intended to be used, with respect to the frequency of demands made upon it, which may be either:
— **low demand mode:** where the frequency of demands for operation made on a safety-related electrical control system is no greater than one per year and no greater than twice the proof-test frequency
— **high demand or continuous mode:** where the frequency of demands for operation made on a safety-related electrical control system is greater than one per year or greater than twice the proof-test frequency (derived from IEC 61508-4, 3.5.12)

Dependent failure : A failure for which the probability cannot be expressed as the simple product of the unconditional probabilities of the individual events which caused it.
Two events A and B are dependent, where P(z) is the probability of event z, only if:
P(A and B) > P(A) x P(B)

Diagnostic coverage: Fractional decrease in the probability of dangerous hardware failure resulting from the operation of the automatic diagnostic tests.
The definition may also be represented in terms of the following equation, where DC is the diagnostic coverage, λ_{DD} the probability of detected dangerous failures and λ_{Dtotal} is the probability of total dangerous failures:

$$DC = \frac{\lambda_{DD}}{\lambda_{Dtotal}} * 100\%$$

Diagnostic coverage may exist for the whole or parts of a safety-related system. For example diagnostic coverage may exist for sensors and/or logic system and/or final elements.

The term safe diagnostic coverage, or diagnostic coverage including safe failures, is used to describe respectively the fractional decrease in the probability of safe hardware failure, or of both safe and dangerous hardware failures, resulting from the operation of the automatic diagnostic tests.

Equipment under control (EUC): Equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities

Error: Part of a system state that is liable to lead to a subsequent failure. Manifestation of a fault in the system [LAP]

EUC control system: System that responds to input signals from the process and/or from an operator and generates output signals causing the EUC to operate in the desired manner

Event-triggered system: A real-time computer system is event-triggered (ET) if all communication and processing activities are triggered by an event other than a clock tick. [KOP]

Fail active: The ability of a system to recover and continue execution after the occurrence of a failure.

Fail halt: The ability of a system to stop after the occurrence of a failure.

Fail-operational (FO): The ability of a system to continue to deliver service in degraded mode and with known safety risks after the occurrence of a failure.

Fail-safe (FS): The ability of a system to reach a safe state after the occurrence of a failure.

Fail-silent: A subsystem is fail-silent if it either produces a correct result or no result at all. [KOP]

Fail soft: The ability of a system to enter a safe state and continue to deliver service in degraded mode after the occurrence of a failure.

Fail passive: The ability of a system to close down after the occurrence of a failure.

Fail stop: The ability of a system to signal a failure and then stop after the occurrence of a failure.

Failure: termination of the ability of a functional unit to perform a required function. [IEC 61508-4], 3.6.4

Failure Modes: Failure modes must be considered in systems designed for safety protection applications. Two failure modes are important: safe and dangerous.

Failure rate (λ): Instantaneous failure rate, often called „hazard rate" is a commonly used measure of reliability that gives the number of failures per unit time from a quantity of components exposed to failure.
$\lambda(t)$ = Failures per Unit Time/Quantity Exposed.

Fault: Abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function. [IEC 61508-4], 3.6.1

Fault-tolerance: Ability of a functional unit to continue to perform a required function in the presence of faults or errors. [IEC 61508-4], 3.6.3

FMEA: Failure mode and effects analysis

FO/FS: Fail-operational / fail-safe. A system is FO/FS if it continues to give service according to its specifications in the presence of one failure, but returns to a safe state in the presence of two failures.

FO/FO: Fail-operational / fail-operational. A system is FO/FO if it continues to give service according to its specifications in the presence of two failures.

FTA: Fault tree analysis

Functional safety: Part of the overall safety relating to the EUC and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities. [IEC 61508]

Hazard: A hazard is an undesirable condition that has the potential to cause or contribute to an accident. [KOP]

HazOp: Hazard in Operation Analysis

Machine Control System (MCS): Electrical/electronic/programmable electronic system that controls an equipment, a machinery, an apparatus or a plant used for manufacturing, process, transportation, medical or other activities

Mean Time between Failures (MTBF): Like MTTF (mean time to failure) and MTTR (mean time to restoration), it is an average value, but it is the time between failures. This implies that a component has failed and then has been repaired.
MTBF = MTTF + MTTR

Mean Time to Failure (MTTF): "Expected" failure time. It is defined from the statistic definition of expected value.
MTTF = $1/\lambda$ is valid for single components with a constant failure rate or a series of components, all with constant failure rates.

Mean Time to Restoration (MTTR): "Expected value" of the random variable restoration time. The definition includes the time required to detect that a failure has occurred, as well as the time required to repair once the failure has been detected and identified.

Module: Routine, discrete component or a functional set of encapsulated routines or discrete components belonging together

Random hardware failure: Failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware

Redundancy: Means, in addition to the means which would be sufficient, for a functional unit to perform a required function or for data to represent information

Reliability: Reliability is a measure of success. Reliability is defined as "the probability that a device will perform its intended function when required to do so if operated within its specified design limits. " This definition includes four important aspects:

1. The device's "intended function" must be known.
2. "When the device is required to function" must be judged.
3. "Satisfactory performance" must be defined.
4. The "specified design limits" must be known.

All four aspects must be addressed when defining a situation to be a success or a failure.

Safe failure: Failure that does not have the potential to put the safety-related system in a hazardous or fail-to-function state.

Safety function: Function to be implemented by an MCS safety-related system, other technology safety-related system or external risk reduction facilities, which is intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event.

Safety integrity level (SIL): Discrete level (one out of a possible three) for specifying the safety integrity requirements of the safety functions to be allocated to the SRECS, where safety integrity level 4 (in machinery level 3 is the highest level) has the highest level of safety integrity and safety integrity level 1 has the lowest.

Safety-related electrical control system (SRECS): Electrical part of a machine control system for which failure can increase the probability of the machine causing physical injury or damage to health

[IEC 62061]

System: Set of elements which interact according to a design, where an element of a system can be another system, called a subsystem, which may be a controlling or a controlled system and may include hardware, software and human interaction.

Systematic failure: Failure related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors.

Abbreviation	Term (units)	
T_1	Proof test period (hour)	
DC	Diagnostic coverage (percentage)	$DC = \frac{\Sigma \lambda_{DD}}{\Sigma \lambda_{Dtotal}}$
β	Fraction of failures having a common cause	(It is assumed $\beta = 2 \times \beta_D$)
λ	Average probability of failure (per demand or per hour)	
λ_D	Probability of dangerous failure (per demand or per hour)	$0.5\,\lambda$ Assumption made: 50% dangerous failures and 50% safe failures)
λ_{DD}	Probability of detected dangerous failure (per demand or per hour)	Sum of all the probabilities of detected dangerous failure within the channel
λ_{DU}	Probability of undetected dangerous failure (per demand or per hour	Sum of all the probabilities of undetected dangerous failure within the channel
λ_{SD}	Probability of detected safe failure (per demand or per hour)	Sum of all the probabilities of detected safe failure within the channel
t_{DE}	Device equivalent mean down time (hour)	Combined down time for all the components in a channel
t_{SE}	System equivalent mean down time (hour)	Combined down time for all the channels in a system
PFD_{AVG}	Average probability of failure on demand	$PFD_{AVG} = \frac{1}{T_i}\int_0^{T_i} PFD(t)dt$
λ_{AVG}	Average probability of failure per hour	

[IEC 61508-6], Appendix B

4 System analysis

Machinery, process plants and other equipment may, in the case of malfunction present risks resulting from hazardous events such as fires, explosions, and exposure to radiation and machinery traps.

Assurance that the safety integrity target has been reached for dangerous random hardware failures is partly based on safety features such as:
- the hardware fault tolerance requirements
- the diagnostic coverage
- the frequency of proof tests of subsystems and components
- the hardware architecture of the SRECS.

The designer has at his disposal a large score of applicable tools in the developing and implementing processes. These tools of which a list is provided in this section are very useful in the validation process.

4.1 Background, tools and methods

4.1.1 The Machinery Directive

The Machinery Directive gives in Annex 1, the essential health and safety requirements that must be fulfilled by the manufacturer. Since the requirements are quite general, more information is needed to implement the safety functions. It must be stated in the technical files of the machine how each requirement has been fulfilled.

Harmonised standards interpret the essential health and safety requirements and give the requirements for the current technological field. However, the constant development of modern technology implies that the standards are developed as well, so that new systems can be compatible and comparable with each other and with older systems. This is also the case with safety-related programmable control systems.

For safety-related electrical control system (SRECS), the standard IEC 61508 is applicable. This standard is relatively new and is not fully used by the manufacturers. Many manufacturers use the EN 954 standard when developing SRECS. But the EN 954 standard is intended to be used only in non-programmable safety-related systems. The manufacturers rely on this standard since it is more established in the industrial sector and is easier to comprehend. This is a problem especially because the EN 954 standard focuses on fault detection and system behaviour with no specific requirements regarding safety principles.

The current IEC 62061 standard, under development, shall close the gap between the IEC 61508 standard and the EN 954 standard by giving specific requirements for the development and design of SRECS for machinery. It is therefore important that the IEC 62061 standard is easy to understand by manufacturers and designers. The goal of this report is to give complementary assistance and support for the implementation of the IEC 62061 standard.

The harmonised standard EN 1050 Risk Assessment gives a general list of hazards that reflects the essential health and safety requirements of the Machinery Directive. The analyser should find all the hazards of the machine by studying systematically hazard by hazard. [EN 1050 1996]

The method can be used to risk assessment of the machine just before defining safety requirements. It can also be used after the design phase to check that all the hazards are sufficiently taken care of.

4.1.2 Risk assessment

Risk assessment is a series of logical steps to enable, in a systematic way, the examination of the hazards associated with machinery. Risk assessment is followed, whenever necessary, by risk reduction as described in clause 5 of EN 292-1:1991. When this process is repeated it gives the iterative process for eliminating hazards as far as possible and for implementing safety measures [EN 1050]

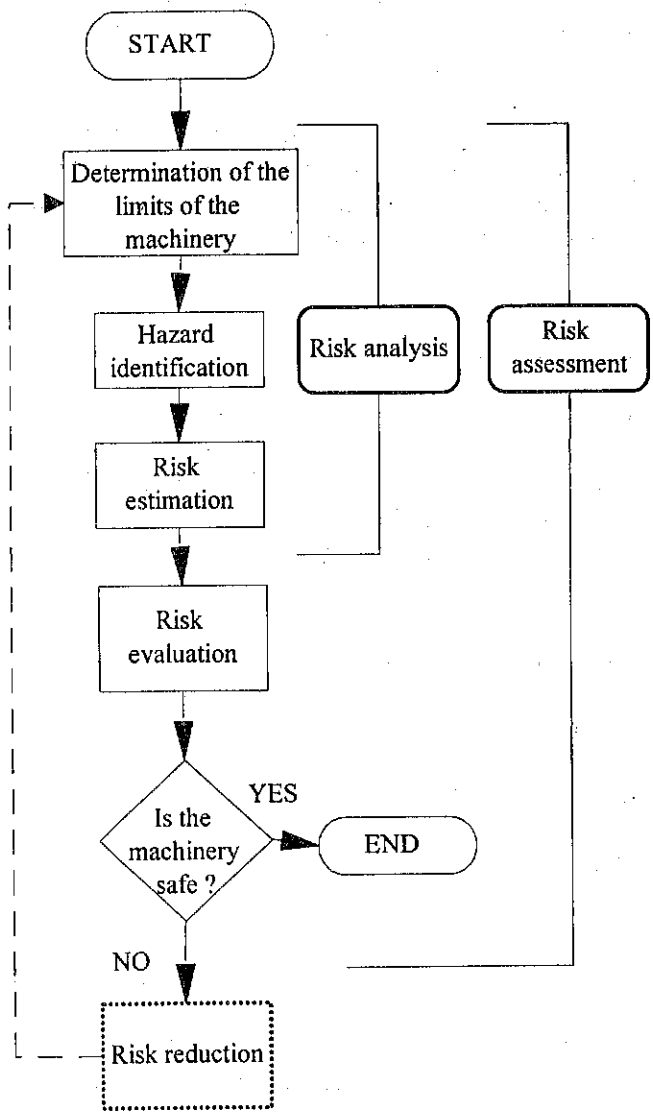

Figure 4.1.2.1 The iterative process to achieve safety [EN 1050]

Risk assessment includes (see fig. 4.1.1.1)

- Risk analysis,
  - a) determination of the limits of the machinery
  - b) hazard identification
  - c) risk estimation

- Risk evaluation

Risk analysis provides the information required for the risk evaluation that in turn allows judgements to be made on the safety of machinery (see 3.4 of EN 292-1:1991).
Risk assessment relies on decisions that shall be supported by quantitative methods.
Quantitative methods are particularly appropriate when the foreseeable severity and extent of harm are high.
Quantitative methods are useful to assess safety measures and to determine the applicable protection alternative.

The application of quantitative methods is restricted by the amount of useful data available and in many applications only qualitative risk assessment will be possible.
The risk assessment shall be conducted so that it is possible to document the procedure that has been followed and the results that have been achieved.

When the probability for risk and hazard is not negligible, the following strategy is applicable for their elimination. In the first place, try to redesign the machine in order to eliminate the risk. In the second place, the operator is protected against hazard by adding a guard or protective equipment. If it is impossible to eliminate the risk a warning should be given by using, for example, signs and intermittent light.

4.1.3 Preliminary hazard analysis (PHA)

PHA is an inductive analysis method, the objective of which is to identify hazards, dangerous situations and events that can cause harm for activity, equipment or system under study. As the name suggests, this procedure is carried out early in projects. It can also be useful when analysing existing systems or prioritising hazards where circumstances prevent a more extensive technique from being used.

The analysis includes:
- determining hazards that might exist and possible effects
- determining a clear set of guidelines and objectives to be used during design
- creating plans to deal with critical hazards
- assigning responsibility for hazard control
- allocating time and resources to deal with hazards

Table 4.1.3.1 shows an example of a PHA.

Table 4.1.3.1. Example of PHA.

VTT		PHA System:		Date: Compiled by: Status:	
Hazardous Condition	Hazard Cause	Hazard Effect	Hazard Level	Safety requirements	Hazard Control
Motor over temperature	Motor over temperature is not detected	The motor burns	1	Motor over temperature supervision	Temperature sensor. Movement prevented.
Unexpected start up	Short-circuit	Collision danger	3	Unexpected start up must be prevented	Distances between terminals. Initial checking in start up.

Too high a speed	HW/SW failure	Collision danger	1	-----	The user must supervise the operation
Condition monitoring does not work	HW/SW failure, parameter error	For instance brake failure	1	Regular maintenance required.	Regular maintenance.

4.1.4 HazOp

Hazard and operability study (HazOp) is a guided and structured brainstorming process. The guidance is provided by the guiding words while the functions of the system, also called study nodes, provide the structure.

Guide words drive the HazOp process by focusing the brainstorming on a set of deviations from normal operation, which can occur during operation or maintenance.

In some sense the guide words can be viewed as generalised failure modes. The study nodes can be system components or system functions. HazOp performed on functions is called Functional HazOp.

The method is used to get an overview of all potential hazards. It can be used in two ways:
- for safety assessment, where we check that all potential hazards can be handled by the system. The output will be a list of potential hazards and an assessment of the implemented defence mechanisms.
- for system design, where we start with a preliminary HazOp and then enter into a loop consisting of design, HazOp and fault tree analyses in order to check whether the goals can be reached. The output will be a set of safety requirements for the implemented system. Table 4.1.4.1 shows an example of a HazOp sheet.

Table 4.1.4.1. HazOp.

HazOp study made to a system				Compiler Date/version	
Guide word	Element	Conditions of occurrence	Consequences	Safety requirements	Comments
No input	Output module	Broken wire, failure inside the output module	Output oscillation stops. Message is sent back to central unit. Relay re-energises. Fails to safety.	Failsafe principles, oscillation input required, output calculations.	
No output	Output module	Failure inside the output module. Old contact surfaces and high current weld both contact poles.	Failure is detected by duplication and failsafe principles. No information to central module. Failure is not detected by central unit.	Duplicated output and failsafe principle needed.	

4.1.5 Cause – Consequence Analysis

The causes of a critical event are determined using a top-down search strategy. The consequences that could result from the critical event are then worked out using a forward search technique in a Cause - Consequence diagram.
The Cause – Consequence diagram uses gates to describe the relations between causal events. Such diagrams illustrate the consequences of critical events in a more tractable format than many pages of natural language.

In Cause - Consequence analysis, separate diagrams are required for each critical event. Since there may be a large number of contributory factors to an accident, several diagrams will be required. Such characteristics frustrate the application of these diagrams to represent and account for the complex interaction between human and system failure during major accidents.

4.1.6 Event tree analysis

Event tree analysis is a graphic method that makes it possible to describe the sequence of events initiated by a top event. The event tree results in a diagram that indicates how serious consequences can occur.

On the top of the diagram is written the sequence conditions that are relevant in the progression of events that follow the initiating event.

Starting under the initiating event, which is the target of the analysis, a line is drawn to the first condition in the sequence. There, the diagram branches off into "yes" and "no" branches, describing how future events depend on the condition. For each of these branches, one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence.

The event tree can be used to compute the probability of the various consequences, based on the probability and number of conditions in the sequence.

4.1.7 Computer-aided specification tools

General
The technique produces a specification in the form of a database that can be automatically inspected to assess consistency and completeness. The specification tool can animate various aspects of the specified system to the user. In general, the technique supports not only the creation of the specification but also of the design and other phases of the project lifecycle. Specification tools can be classified according to the following sub-clauses.

Tools oriented towards no specific method
The specification tool takes over some routine work from the user and supports the project management. It does not enforce any particular specification methodology. The relative independence with regard to method allows users a great deal of freedom but also gives them little of the specialised support necessary when creating specifications. This makes familiarisation with the system more difficult.

Model orientated procedure with hierarchical analysis
This method gives a functional representation of the desired system (structured analysis) at various levels of abstraction (degree of precision). The analysis at various levels acts

on both actions and data. Assessment of ambiguity and completeness is possible between hierarchical levels as well as between two functional units (modules) at the same level.

Entity models
The desired system is described as a collection of objects and their relationships. The tool enables to determine which relationships can be interpreted by the system. In general, the relationships permit a description of the hierarchical structure of the objects, the data flow, the relationships between the data, and which data are subject to certain manufacturing processes. The classical procedure has been extended for process control applications. Inspection capabilities and support for the user depend on the variety of relationships illustrated. On the other hand, a large number of representation possibilities make the application of this technique complex.

Incentive and answer
The relationships between the objects of the system are specified in a notation of "incentives" and "answers". A simple and easily expanded language is used which contains language elements that represent objects, relationships, characteristics and structures.

4.1.8 Static analysis

In static analysis, the program to be studied is not executed, in contrast to dynamic analysis. The evaluation of a system is based on the analysis of the structure, and the documentation of the software. The primary analysis tools for the analysis of a real-time system are the data flow diagrams (DFD). The purpose of a DFD is to describe the data flow between the major functional components of the system. These components are determined by the system requirement specifications and relate the system to its operating environment. The clause 4.1.15 on structured design / analysis technique (SADT) describes one technique for static analysis

4.1.9 Dynamic analysis

In dynamic analysis the program to be studied is executed. The dynamic analysis of a safety-related system is carried out by subjecting a near-operational prototype of a safety-related system to input data, which is typical of the intended operating environment. The analysis is satisfactory if the observed behaviour of the safety-related system conforms to the required behaviour. Any failure of the safety-related system must be corrected and the new operational version must then be reanalysed.

4.1.10 FMEA

The Failure Mode and Effect Analysis (FMEA) is the most common analysing tool used for assessing the safety and performance of a control system. The method is defined in an international standard [IEC 812]. FMEA is a bottom-up (inductive) method, which begins with single failures, and then the causes and consequences of the failures are considered. In the FMEA, all components, elements or subsystems of the system under control are listed. FMEA can be done at different levels and in different phases of the design, which affects the depth of the analysis. In an early phase of the design, a detailed analysis cannot be done, and also some parts of the system can be considered so clear and harmless that deep analysis is not necessary. However, in the critical parts, the analysis needs to be deep and it should be made at a component level. If the safety of the system really depends on a certain complex component, the analysis may even include some inner parts of the component, for example software analysis or consideration of typical failures related to a certain logical function.

In the ISO 13849-2 standard, there are useful lists for FMEA on failures of common components in different types of control systems. The standard gives probable component failures and the analyst decides if the failures are valid in the system considered or if there are other possible failures. If functional blocks, hybrid circuits or integrated circuits are analysed, the list stated in the ISO 13849-2 standard is not enough. Additionally, systematic failures and typical failures in microprocessors, memories, communication circuits etc, have to be considered since those failures are more common than basic random hardware failures.

FMEA is intended mainly for single random failures and so it has the following weak points:

- It does not support detection of common cause failures and design failures (systematic failures).
- Human errors are usually left out; the method concentrates on components and not the process events. A sequence of actions causing a certain hazard is difficult to detect.
- Sequential failures causing a hazard can also be difficult to detect, since the basic idea of the method is to consider one failure at a time. If the analysis is made with strict discipline it is also possible to detect sequential failures. If a failure is not detected by the control system, other failures (or events) are studied assuming the undetected failure has happened.
- Systems with a lot of redundancy can be difficult to consider since sequential failures can be important.
- The method treats failures equally, and so even failures with very low probability are considered carefully. This may increase the workload and cause a lot of paper-work.
- In a large analysis documentation, it can be difficult to identify the critical failures. It can be difficult to see which failures have to be considered first, and how to handle the critical failures.

However, FMEA is probably the best method to detect random hardware failures, since it considers all components (individually or as blocks) systematically. Some critical parts can be analysed at a detailed level and some at a system level. If the method seems to become too laborious, it is recommended to carry out the analysis at a higher level. There is, though, a risk that some failure effects remain undetected.

The FMEA table always includes the failure modes of each component and the effects of each failure mode. Since the analysis is carried out to improve the system or to show that the system is safe or reliable enough, some remarks and future actions are always needed in the table. Severity ranking is needed to ease the comparison between failure modes, and therefore it helps to rank the improvement actions. When the analysis includes criticality ranking it is called Failure Mode, Effects, and Criticality Analysis (FMECA). The criticality and probability factor can be a general category, like; impossible, improbable, occasional, probable, frequent. Exact failure probability values can also be used if available. In many cases exact values are not available.

Table 4.1.10.1 shows an example of an FMECA. [STSARCES2]

Table 4.1.10.1. Example of an FMECA.

VTT Safety Engineering		FMECA System: Coffee mill Subsystem:		Page: Date: Compiled by: Approved by:		
Item and function	Failure mode	Failure cause	Failure effects	Detection method	Probability & Severity	Remarks
Switch	Short circuit	- Foreign object, animal or liquid - isolation failure (moisture, dirt, ageing - bad, loose connection, vibration, temperature changes - overheating (lack of cooling)	a) Coffee mill cannot be stopped by actuating the switch. Someone may cut his finger. b) When the plug is connected, the coffee mill starts up although the switch is in the off state. Someone may cut his finger.	Coffee mill does not stop. Switch may become dark.	a) 3C b) 4C	The coffee mill can be stopped by un-plugging it.
Switch	Open circuit	- The switch mechanism fails to operate, mechanism jams, breaks	The coffee mill cannot start up.		1D	

4.1.11 FTA

Fault Tree Analysis (FTA) is a deductive technique that focuses on one particular accident or top event at a time and provides a method for determining causes of that accident. The purpose of the method is to identify combinations of component failures and human errors that can result in the top event. The fault tree is expressed as a graphic model that displays combinations of component failures and other events that can result in the top event. FTA can begin once the top events of interest have been determined. Such information is the result of a preliminary hazard analysis (PHA) or some other analysing method.
The advantages of FTA are:

- It can reveal single point failures, common cause failures and multiple failure sequences leading to a common consequence.
- It can reveal the point when a safe state becomes unsafe.
- The method is well known and standardised.
- The method is suitable for analysing the entire system including hardware, software, and any other technologies, events and human actions.
- The method provides a clear linkage between qualitative analysis and the probabilistic assessment.
- It shows clearly the reasons for a hazardous event.

The disadvantages of FTA are:

- It may be difficult to identify all hazards, failures and events of a large system.
- The method is not applicable to systems with a large number of safety critical failures.

- It is difficult to introduce timing information into fault trees.
- The method can become large and complex [IPPOLITO].
- The method gives a static view to system behaviour.
- The method assumes independence of events, although dependencies can be present; this affects the probability calculations. The dependencies also increase the work.
- Different analysts end up with different representations that can be difficult to compare.

Quite often probability calculations are included in the FTA. FTA can be performed with special computer programs that easily provide the proper documentation. There are also programs, which can switch the method. The analysis only needs to be fed once, and the program provides the information as an FTA or an FMEA. Figure 4.1.11.1 shows an example of one hazardous event in an FTA format. The figure on the right illustrates the system. [HAMMER] and [STSARCES2]

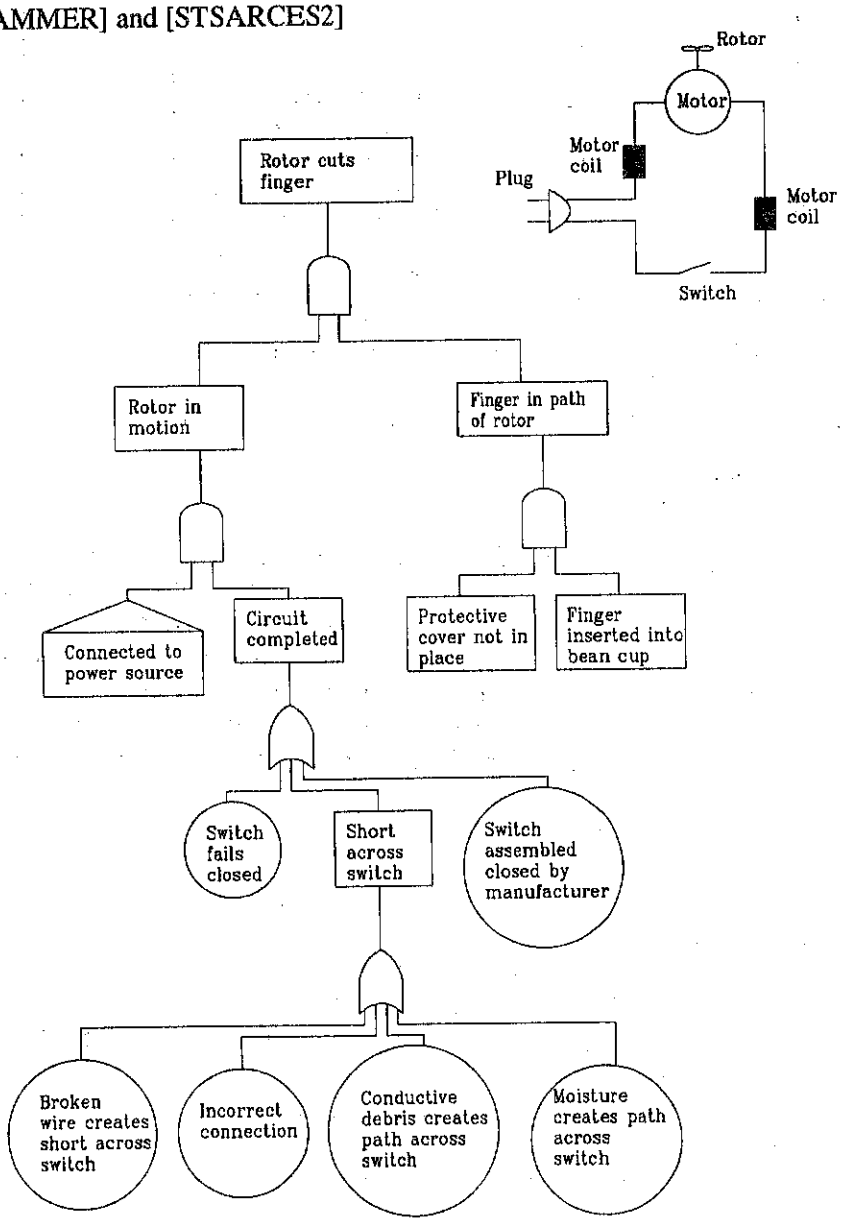

Figure 4.1.11.1 Example of a fault tree [HAMMER]

4.1.12 Common mode failure analysis

Common mode failure analysis (CMF) and common mode analysis (CMA) are both types of common cause analysis. This text is concentrating on CMF. The aim of this method is to identify potential failures in redundant systems or redundant sub-systems, which would undermine the benefits of redundancy because of the appearance of the same failures in the redundant parts at the same time.

Computer systems intended to take care of the safety of a plant often use redundancy in hardware and majority voting. This technique is used to avoid random component failures, which would tend to prevent the correct processing of data in a computer system.

However, some failures can be common to more than one component. For example, if a computer system is installed in one single room, shortcomings in the air-conditioning might reduce the benefits of redundancy. The same is true for other external effects on the computer system such as fire, flooding, electromagnetic interference, plane crashes, and earthquakes. The computer system may also be affected by incidents related to operation and maintenance. It is, therefore, essential that adequate and well-documented procedures are provided for operation and maintenance. Extensive training of operating and maintenance personnel is also essential.

Internal effects are also major contributors to Common-Mode Failures (CMF). They can stem from design errors in common or identical components and their interfaces, as well as from ageing components. CMF-analysis is used to search such potential common failures. Methods of CMF-analysis are general quality control, design reviews, verification and testing by an independent team, and analysis of real incidents with feedback of experience of similar systems. The scope of the analysis, however, goes beyond hardware. Even if "diverse software" is used in different chains of a redundant computer system, there might be some commonality in the software approaches, which could give rise to CMF, such as errors in the common specification.

When CMFs do not occur at exactly the same time, precautions can be taken by comparing the redundant chains. This should lead to detection of a failure, before it is common to all chains. CMF analysis takes this technique into account.

The advantages of CMF are:

- Covers areas not addressed by other techniques

The problems or disadvantages of CMF are:

- Problems arise from the fact that CMF analysis cannot be formalised, but always needs human creativity to identify CMF situations. A resulting disadvantage is the man-hours effort involved in striving for completeness.

Results of the analysis are identification of potential CMFs together with quantification where possible and qualification of their effect. From the quantification, the calculated frequencies of common mode failures would be assessable against criteria, if existing, and cost of CMF effects and counter-measure costs can be balanced. Assessment of defences employed against CMF.

CMF should be used with other methods. It profits from a previous Event Tree, or Cause Consequence Analysis, as well as Failure Modes and Effect Analysis in that a thorough understanding of the system is a prerequisite to CMF analysis. Identified potential CMF

can be further evaluated in these techniques. Probabilistic methods are applicable whenever CMF rates can be determined. In cases where CMFs do not occur simultaneously, probabilistic methods for individual failures can be a substitute to CMF analysis. [BISHOP]

Table 4.1.12.1 shows an example of CMF.

Table 4.1.12.1 Example of a CMF table

Common mode type	Common mode source	Common mode failure/error	Requirement/ Comment
Software development error	Common structure and design in duplicated modules	One error in used compiler, language, system code or application code reflects to duplicated sister processor since it is identical	Diversity in duplicated modules is lacking.
Hardware development error	Common structure and design in duplicated modules	One design failure reflects to duplicated sister module since it is identical	Diversity in duplicated modules is lacking

4.1.13 Walkthroughs and reviews

In walkthrough methods the designer(s) and analyser(s) manually go through the system design together and check the correctness. Usually some test cases, checklists and guidelines are used in the process.

In reviews the system is presented to someone whose approval or comments are expected.

4.1.14 Fagan inspections

Fagan inspection is a highly structured, clearly defined process by which software documents are reviewed in detail. The inspection includes planning, overview, preparation, inspection meeting, inspection analysis meeting, re-work and follow-up. In the process the programmer reads the source code to a group who asks questions and analyses the program by using a checklist. The inspection is a cost-effective way of finding errors.

4.1.15 Structured design / analysis technique (SADT)

The aim of this technique is to model and identify, in a diagrammatical form using information flows, the decision making processes and the management tasks associated with a complex system.

In SADT the concept of an activity-factor diagram plays a central role. An A/F diagram consists of activities grouped in so called "action boxes". Each action box has a unique name, and is linked to other action boxes by factor relations (drawn as arrows), which are also given unique names. Each action box can be hierarchically decomposed into

subsidiary action boxes and relations. There are four types of factors: inputs, controls, mechanisms and outputs.

Input: indicated by an arrow that enters an action box at the left hand side. Inputs can represent material or immaterial things and they are suitable for manipulation by one or more activities in an action box.

Control: are instructions, procedures, choice criteria and so on. Controls guide the execution of an activity and they are shown by arrows entering the top side of an action box.

Mechanism: is a resource such as personnel, organisational units or equipment that is needed for an activity to perform its task.

Output: can denote anything that can activity produces, and it is pictured by an arrow leaving an action box at the right hand side.

Figure 4.1.15.1 shows an example of an SA diagram.

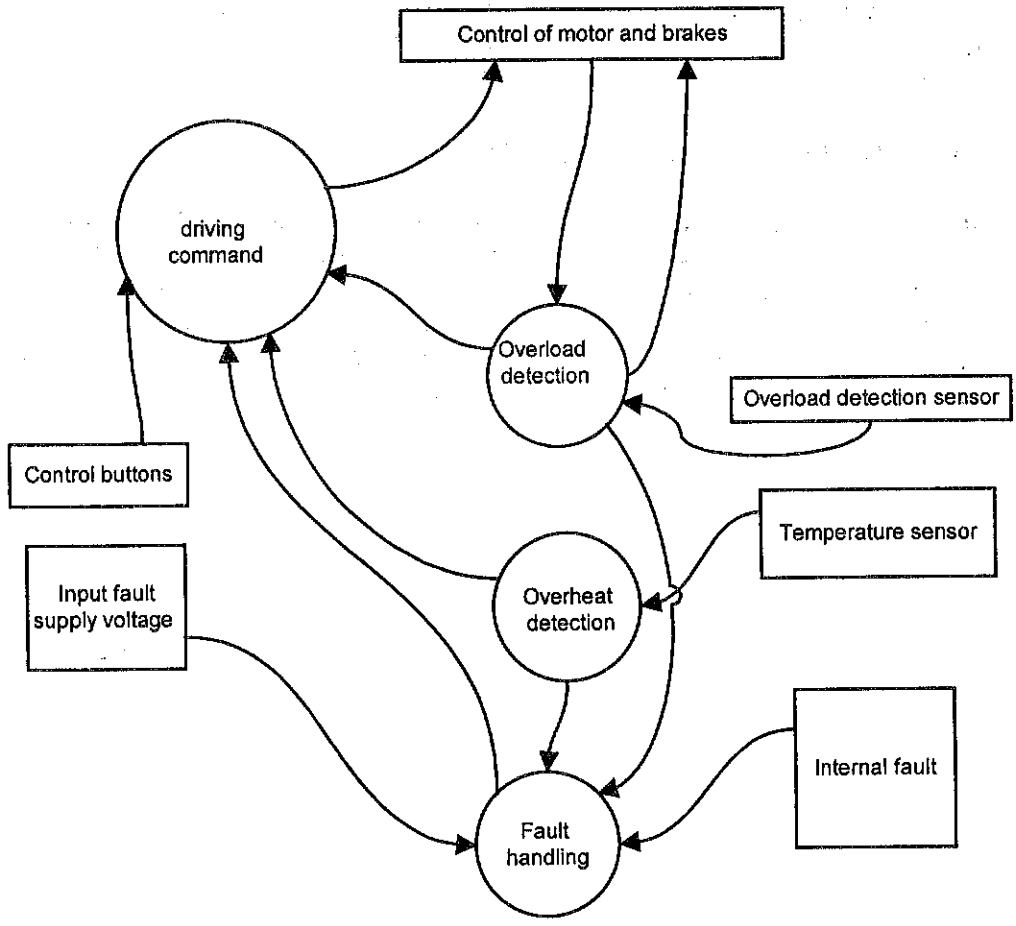

Figure 4.1.15.1 SA diagram

When activities are strongly related to each other by many factor relations it is better to consider these activities as an indivisible group that is contained in one action box which does not lend itself to further detailing of its content. The guiding principle for grouping of activities into action boxes is that the resulting boxes are coupled pair wise by only a few factors.

The model hierarchy of A/F diagrams is pursued until further detailing of the action boxes is meaningless. This stage is reached when the activities within the boxes are inseparable or when further detailing of the action boxes falls outside the scope of the system analysis.

The advantages of SADT are:

- SADT permits the analysis of any activity in a complex system. The resulting analysis may be viewed at different levels of detail.

Problems or disadvantages of SADT are:

- When activities are grouped together in action boxes, care must be taken that no boxes overlap, i.e. one activity is assigned to more than one box.
- The unique naming convention makes it difficult for different activities to share potentially common mechanisms (e.g. action boxes containing pre-programmed functions).

ADT is a good analysis tool for existing systems, and can also be used in the design specification of systems. SADT is recommended in assessment of safety critical functions and relationships in complex systems. [BISHOP]

4.1.16 State transition diagrams

The state transition diagrams (state charts) aim to describe the different states of the program and how to move from one state to another. State transition diagrams make it easier to understand the operation of the program. It is also easier to start programming and the reliability of the code can be improved when well-inspected state transition diagrams are available. It may be difficult to describe all program entities on diagrams. Figure 4.1.16.1 shows an example of a state transition diagram.

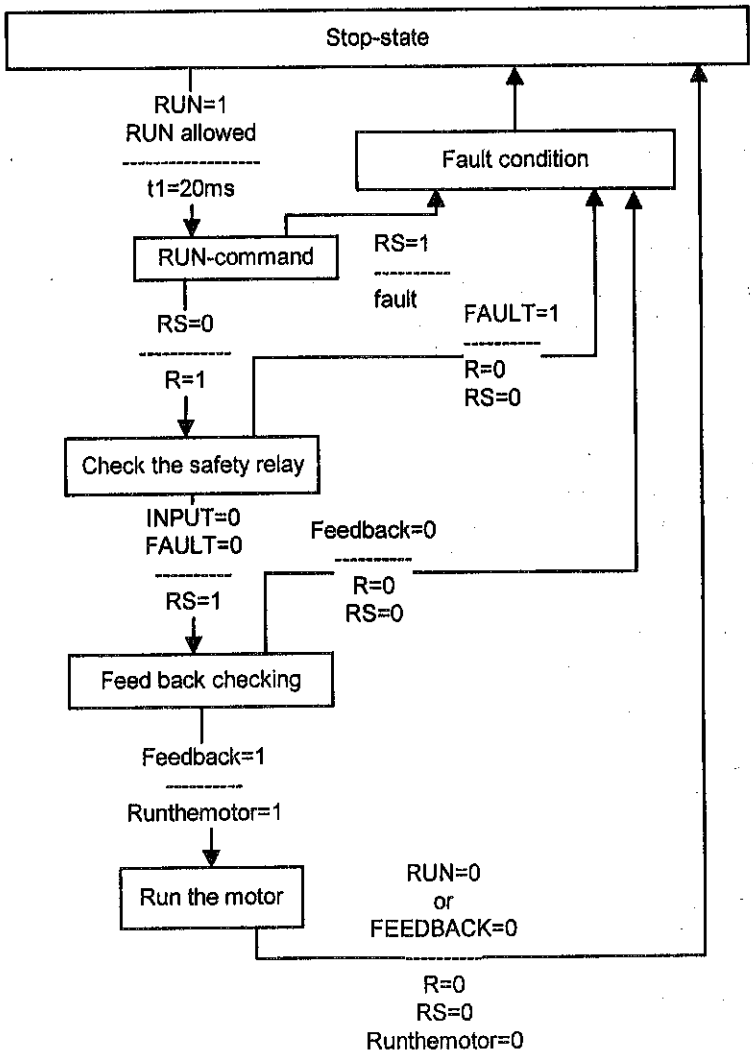

Figure 4.1.16.1 Example of a state transition diagram

4.1.17 Unified modelling language (UML)

UML is designed for modelling and describing object-oriented software systems. UML diagrams are visual tools for describing system design views, components, processes and human-computer interactions. UML diagrams consist of visual elements that support modelling and visualization of a software system.

A summary of types of UML diagrams can be described below:
- Class
- Object
- Sequence
- Collaboration
- State chart
- Activity
- Use-case
- Component
- Deployment

State chart is used to describe both static and dynamic system activities and behaviours. System interaction can be described by using sequence charts and collaboration charts. System implementation can be described by using components and deployment diagrams.

4.1.18 Formal methods

Formal methods consist of a set of techniques and tools based on mathematical modelling and formal logic that are used to specify and verify requirements and designs for computer systems and software.

The use of formal methods can assume various forms, ranging from occasional mathematical notation embedded in specifications stated in text form, to fully formal specifications using specification languages with precise semantics. The most rigorous, formal methods involve computer-assisted proofs of key properties regarding the behaviour of the system.

Formal specification is the use of notations derived from formal logic to:

1. describe the assumptions about the world in which a system will operate
2. draw up the requirements that the system is to achieve
3. choose a design that fulfils those requirements.

Formal verification is the use of proof methods from formal logic to:

1. analyse specifications for certain forms of consistency and completeness
2. prove that the design will satisfy the requirements, given the assumptions

The mathematics of formal methods include:

1. predicate calculus (order logic)
2. recursive function theory
3. lambda calculus
4. programming language semantics
5. discrete mathematics-number theory, abstract algebra, etc.

Formal methods can be classified according to whether their primary purpose is descriptive or analytic.

Descriptive methods focus largely on specification as a tool for review and discussion.

Analytic methods focus on the utility of specification as a mathematical model for analysing and predicting the behaviour of (hardware and software) systems, in addition to their utility for communication.

4.1.18.1.1 Formal language favoured by each of the two methods

Descriptive formal methods
Descriptive formal methods generally use the notations of conventional mathematics, most commonly, notations based on set theory, with quantification restricted to first order functions that are essentially partial, and types imposed on an inherently untyped foundation. The language choice does not readily support automation; descriptive methods typically offer attractive user interfaces and little in the way of deductive machinery.

Analytic formal methods
Analytic formal methods put considerable emphasis on mechanization and general design specification languages capable of supporting efficient automated deduction. These methods can be further classified according to the degree of automation provided by the proof checker.
There are:

- formal method systems with automatic theorem proving and virtually no user interaction
- formal method systems with proof checking and virtually no automatic proof steps
- formal methods that combine elements of both.

This spectrum of methods reflects tradeoffs with respect to specification language, proof development and user-accessibility.

Formal method systems that support fully automatic theorem proving have restricted specification languages and powerful theorem provers that can be difficult to control and offer little feedback on failed proofs, but perform impressively in the hands of experts. Most state exploration tools also fall into this category.

Formal method systems based on proof checking generally offer more expressive languages, but require significant manual input for theorem proving.
Formal method systems that combine a significant level of automation and user input fall somewhere in between, depending on language characteristics and proof methodology.

A typical analytic formal method tool consists of the following components:

- **User Interface**: integrates tool components and manages input and output.
- **Parser**: checks specifications for syntactic consistency and builds an internal representation used by other components of the system.
- **Prettyprinter: or unparser**: translates the internal representation of the specification into a standard format for user display and output.
- **Typechecker**: checks specifications for semantic consistency, possibly adding semantic information to the internal representation built by the parser. Systems in which the typechecker and prover are closely integrated, attempt to prove type correctness theorems automatically.
- **Proof checker**: performs proofs over a syntactically and semantically correct specification.

Most formal method systems also offer some or all of the following:

- **Browser**: produces cross-reference and displays cross-reference information; particularly useful for large specifications possibly spread across several files
- **Status Recorder, Reporter**: maintains and reports status of specification (e.g., parsed, type checked) and proofs (e.g., proof complete or incomplete, axiomatic foundation, status of lemmas used in proof).
- **Output Generator**: provides customized and possibly user-customizable formatting for specifications and proofs.

Formal methods offer significant potential for improving error detection early in the life cycle.
Formal methods are used when developing e.g. safety-critical programmable control systems with high demands on dependability performance.

Following are some of the benefits realisable with the effective use of formal methods:

- Formal methods help finding errors
  When applied to complex programmable systems, formal methods are efficient in finding errors that remain undetected during extensive testing.
- Formal methods contribute efficiently to detect errors in requirements and errors in the design process are identified earlier than would be the case otherwise.
- Formalized statements can be analysed and their consequences evaluated systematically in a repeatable manner. With mathematical proof methods, the risk of drawing the wrong conclusions about the behaviour of a system as a result of extrapolation from a finite number of tests is eliminated.

[NASA]

4.1.19 Semi-formal methods

As well as formal methods, semi-formal methods present the specifications of a system unambiguously and consistently in order to detect omissions and mistakes. The distinction between formal and semi-formal methods is not always clearly drawn but is nevertheless an important one.

While formal methods are based on a precise notation with strong logical properties of provability, the stringency of semi-formal methods is not sufficient to support proof of consistency. Semi-formal methods may include graphical notations such as state charts and may support limited static analysis and simulation in system modelling.

4.1.20 Data flow analysis

The dataflow analysis is a method for analysing the data path through the software. The method focuses on the parameters and processes used to realise specific functions and does not regards conditions for these parameters. This is a graphical and descriptive method, which produces flow-charts.

When performing the method certain primitive symbols are used such as data stores, processes, initiating event(s) and final event(s). The primitives are connected by arrows that show the data-streams.

The outcome of the method gives a survey of interesting parts (functions) of the software and more or less detailed information on which parameters initiate the function(s) and on which parameters are finally affected by the function(s). The primary purpose of the analysis is to verify unwanted or especially critical data-streams or processes affecting the functions.

4.1.21 Control flow analysis

The control flow analysis is a method for analysing the behaviours and states of a control system. The method focuses on the separate modules included in the system, their behaviour (states) and the conditions for these states. Commonly used means for this analysis are representation of the system as a finite state machine and development of state transition diagrams.

The outcome of the method(s) gives information about e.g. safety-related states and how they are accessed and a survey of the functionality of the system.

4.1.22 Worst case analysis

Worst case analysis aims to find problems, which may arise from unfavourable combination of conditions and component tolerances. In the context of software analysis, worst case can refer to memory needs and execution time.

The purpose of Worst Case Execution Time (WCET) analysis is to provide a priori information about the worst possible execution time of a program before using the program in a system. WCTE estimates are used in real-time systems development to perform scheduling and schedulability analysis, to determine whether performance goals are met for periodic tasks, and to check that interrupts have sufficiently short reaction times.
The goal of WCET analysis is to estimate the worst-case execution time of the program. To generate a WCET estimate, the program must be processed through the following phases:
- program flow analysis
- low level analysis
- calculation

4.1.22.1 Program flow analysis

The program flow analysis phase determines the dynamical behaviour of the program. The result is information about which functions get called, how many times loops iterate, dependencies between if-statements, etc.

4.1.22.2 Low level analysis

The purpose of the low level analysis is to determine the execution time for each basic unit of flow (e.g. an instruction or a basic block), given the architecture and features of the target system. Examples of features to consider are pipelines and caches.

4.1.22.3 Calculation

The calculation phase generates the final WCET estimate for the program, given the program flow and the low-level analysis results. Following are the three main categories of WCET calculation methods:

Path-based calculation
In a path-based calculation, the WCET estimate is generated by calculating times for the different paths in a program, searching for the path with the longest execution time. The defining feature is that possible execution paths are explicitly represented.

Tree-based methods
In Tree-based methods, the final WCET is generated by a bottom-up path through the tree representing the program. The analyses for smaller parts of the program are used to make timing estimates for larger parts of the program.

Implicit Path Enumeration Technique (IPET)–based methods
IPET-based methods express program flow and atomic execution times using algebraic and/or logical constraints. The WCET estimate is calculated by maximising an objective function, while satisfying all constraints.

4.1.23 Consistency analysis

The aim of this method is to avoid systematic faults that can lead to breakdowns in the system under test, either early or after many years of operation. This systematic and possible computer-aided approach inspects specific static characteristics of the prototype system to ensure completeness, consistency, and clearness of the requirements specification (for example construction guidelines, system specification and an appliance data sheet).

4.1.24 Markov modelling

The aim of this method is to evaluate the reliability, the safety or the availability of the system. A graph of the system is constructed. The graph represents the status of the system with regard to its failure states (the failure states are represented by the nodes of the graph). The edges between nodes, which represent the failure events or repair events, are weighted with the corresponding failure rates or repair rates. It is assumed that a transition from the state N to the next state N + 1, is independent of the previous state N-1. The failure events, states and rates can be detailed in such a way that a precise description of the system is obtained, e.g. detected or undetected failures, manifestation of a larger failure, etc.

The Markov technique is suitable for modelling redundant systems in which the level of redundancy varies with the time due to component failure and repair.

For simple systems, the expressions, that describe the transition probabilities, are available in literature or can be calculated manually. For more complex systems, methods of simplification exist but in the case of very complex systems the graph is simulated by computer calculation.

Figure 4.1.24.1 shows the Markov representation of a duplicated system. The Common cause factor (β) gives the proportion of common cause failures (design failures and failures which can cause the failure of several components). In the example the failure rate is assumed to be constant. When the Markov representation is ready, the matrix for calculations can be made. [Dorra2]

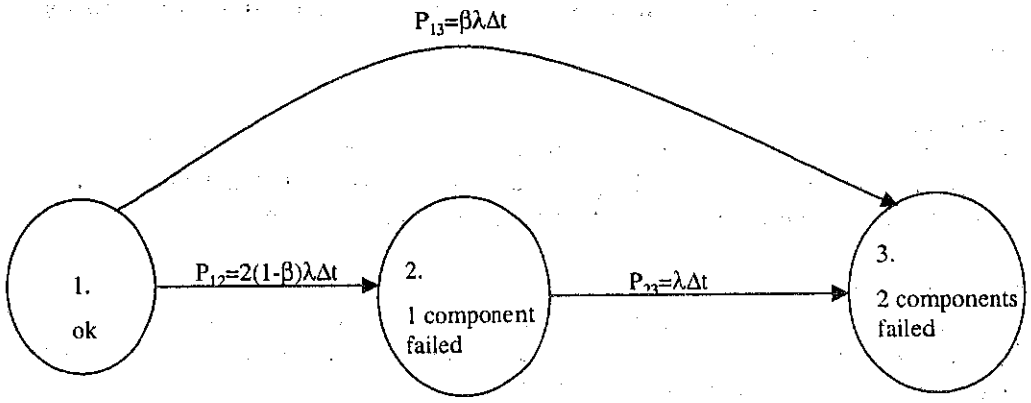

Figure 4.1.24.1. Markov representation of dual redundancy with common cause effect.

λ= probability of failure
β= common cause effect
Δt=observation time

The advantages of Markov Modelling are:
Markov Modelling provides an easy way to estimate the dependability of systems in which the level of redundancy varies during the life-cycle of the system.

Parametric studies can be performed to assess the impact of:
- different system configurations
- different rates
- different repair strategies

Problems and disadvantages
- A Markov simulation is necessary for most practical applications
- Some results may not be truly representative when the rates differ by orders of magnitude (for example, failure rate $=10^{-6}h^{-1}$ and repair rate = $1h^{-1}$) depending on the simulation method applied
- The number of states may become very large.

The method is probabilistic. It can be used as a complement to other analysis methods such as; Failure Mode and Effect Analysis, Fault Tree Analysis and Cause-Consequence Diagrams.

The Markov Modelling is recommended for dependability evaluation of redundant hardware. Tools exist to simulate or compute Markov graphs.

4.1.25 Sneak circuit analysis

Sneak circuit analysis is used to find unexpected paths or logic flows that may initiate undesired functions or inhibit desired functions. Sneak circuits are not the result of hardware failures but are latent conditions inadvertently introduced in the system or coded into the software programs. Such conditions can lead to malfunction. In the analysis process, the system is described with topological network trees that clearly show the relationship between components. Checklists are then used and corresponding questions stated.

In the figure 4.1.25.1 is an example of sneak circuit in a car. Only the current going through the ignition switch should open the car radio but also the current marked by dash line which passes the ignition switch can open the radio, when the brake pedal is pressed and the emergency switch is closed. Figure 4.1.25.2 shows the same circuit as a topological network tree.

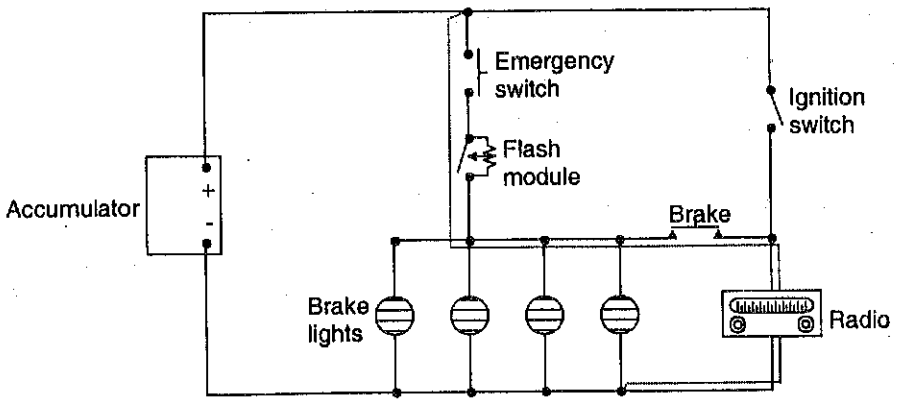

Figure 4.1.25.1 Example of a sneak circuit in a car [RANKIN]

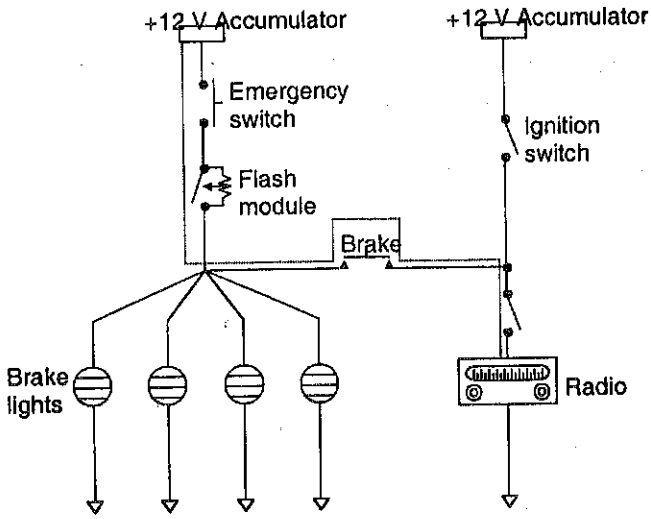

Figure 4.1.25.2 Network tree of the example.

Advantages of sneak circuit analysis
- applicable to hardware, software and the combined system
- can be applied to programs written in any language
- directed towards system design flow rather than component failures

Problems and disadvantages
- it is labour intensive
- it is likely to be performed late in the development cycle, so changes will be expensive

4.1.26 Simulation

Simulation techniques can be used to analyse hardware, software and systems before any actual system exists.

The aim of hardware simulation is to carry out a systematic and complete inspection of an electrical/electronic circuit, of both the functional performance and the correct dimensioning of the components.

The function of a safety-related system is simulated on a computer via a software behavioural model. Individual components have their own simulated behaviour, and the response of the circuit to which they are connected is examined by looking at the marginal data of each component.

4.1.27 Fault injection techniques

Fault injection techniques aim to detect any difference between the specified and observed behaviour in the presence of faults. Fault injection techniques can be used to a simulation based model or to a prototype. In a prototype it is possible to use physical fault injection to the hardware by using pin level fault injection or selected radiation. However, software based fault injection techniques are more common. [GIL]

The fault injection techniques reveal errors or inadequate error handling mechanisms in a system, which is almost complete. However, the amount of tests is limited and the type of tests that can be used depends on the current system.

Advantages:
- Possibility of injecting faults to a real system
- Possibility of revealing real errors/failures/deficiencies
- Performance of a complex system can be verified.

4.1.28 Functional testing

Functional testing can be done in parts of the system or in a complete system when the system is almost ready. In functional testing all common use is tested and then extreme use. Some kind of functional testing is made to all systems whenever it is possible.

If the system under tests should be "proven in use" type the use of the system should last at least a certain amount of hours (see IEC 61508 – 7).

4.2 When to use the methods

The methods that can be used in the validation process can be divided into groups in several ways. One way is to classify the methods according to the phase of the system life cycle in which it is useful. First, some methods are useful in finding hazards and making the specification to the system, second, a group of methods are useful in making the system easier to understand, design and validate, and, third, some methods are good in finding errors and failures. The methods can also be divided into groups according to the applicable technology hardware, software and/or system.

Table 4.2.1 shows whether the validation technique is useful for hardware, software or for the complete system and the design phase at which it is applicable. It also shows for which purpose each validation technique is suitable. R means (recommended or) suitable for certain purposes in this phase and HR means (highly recommended or) suitable to use in this phase.

R or HR refers only to the usefulness of the method in a certain phase of the design. Marking "-" means that the method is not normally used in this phase of the design.

Table 4.2.1 Validation techniques

Technique	**Useful in SW/HW/ system**	**Risk analysis and requirements**	**Design, error/failure finding**	**Implementation**
Risk assessment (EN 1050)	system	HR	-	-
Computer-aided specification tools	HW/SW/ system	R	-	-
Cause – consequence analysis	HW/SW/ system	R	-	-
Event tree analysis	HW/SW/ system	R	-	-
PHA	HW/SW/ system	HR	-	-
HAZOP	HW/SW/ system	HR	R	-
Structured design/analysis	HW/SW/ system	R	R	-
State transition diagrams	SW	R	HR	-
Unified modelling language (UML)	HW/SW/ system	R	R	-
Data flow analysis	SW	R	HR	-
Control flow analysis	SW	R	HR	-
Formal methods	SW	R	HR	-
Worst case analysis	HW/SW	R	R	-
FMEA	HW/SW/ system	HR	HW:HR, system:HR SW: R	-
FTA	HW/SW/ system	HR	HW:HR, system:HR, SW: R	-
Common mode analysis	HW/SW/ system	(R)	R	-
Markov modelling	HW/ system	R	HR	-
Sneak circuit analysis Software sneak analysis	HW/SW/ system	-	HR	-
Fagan inspections	SW (/HW)	R	HR	HR
Walkthroughs	HW/SW/ system	HR	HR	HR
Simulation	HW/SW/ system	R	R	R
Fault injection	HW/SW	-	HR	HR

Table 4.2.2 gives short descriptions of methods, which can be used in the validation process. More detailed information can be found in chapter 4 and in the references.

Table 4.2.2 Description of each method

Method	Purposes	Applicability	Efficiency	References
Risk assessment (EN 1050)	Risk assessment according to EN 1050 is made to machines by proceeding hazard by hazard. Checklist presented in the standard helps to find hazards.	The method is useful in finding hazards. The method covers hazards presented in Machine Directive.	Method is effective in finding hazards generated by machines.	EN 1050, Machine Directive
Computer-aided specification tools	Set of tools based on mathematical modelling used to specify requirements for computer systems and software.	In the safety life cycle, following the hazard and risk analysis, such tools support the correspondence between hazards and requirements specification.	Facilitates the traceability of safety design steps taken during safety life cycle development.	IEC 61508-7
Cause – consequence analysis	Cause – consequence analysis is used to find and describe hazardous situations.	Method is good in describing hazards.	The method describes the hazards well.	IEC 61508-7
Event tree analysis	Graphic representation of events sequence initiated by a top event.	Together with the fault tree analysis and the cause consequence diagram, constitutes a powerful technique to determine design weaknesses.	The method describes the hazards well.	IEC 60300-3-9.2000. Risk analysis of technological systems.
PHA	PHA is used to find hazards and hazardous situations in early phase of the development process.	The method is good in finding hazards.	The method is effective in finding hazards. The knowledge of the group performing the analysis affect strongly to the results.	IEC 60300-3-9 Dependability management. Part3: Application guide. Section 9: Risk analysis of technological systems.
HAZOP	HAZOP is used to find hazards by Using guide words.	By using practical guide words the method can be used in many kind	The method is effective in finding hazards when proper guide	Draft IEC 61882 Ed. 1: Hazard and operability (HAZOP) studies

Method	Purposes	Applicability	Efficiency	References
		of applications for finding hazards.	words and experts are available.	– Guide word approach. [IEC 60300]
Structured design/analysis	Tool for analysis and design based on graphical representation.	There are several approaches to structured software development, many of them not suitable for real-time systems. At the same time such systems might gain quite a lot from using structured methods and tools.	When using methods such as e.g. Yourdon, JSD, MASCOT, SADT, the system is refined in several levels of abstraction, from context diagram down to mini specifications and data dictionaries.	[IEC 61508-7], C.2.1.6
State transition diagrams	One type of diagram used in structured analysis and design of real-time systems	The state transition diagrams describe the states of the different processes which constitute the other diagrams in a structured analysis/design representation model.		[IEC 61508-7], B.2.3.2
Unified modelling language (UML)	Tool for modeling and describing object-oriented software systems.	UML diagrams are visual tools for describing system design views, components, processes and human-computer interactions.	Method is easy to use and has visual expression capability.	[RUMBAUGH]
Data flow analysis	Determine the characteristics of program structures	Static technique that combines the information obtained from the control flow analysis.	Specific faults are not necessarily pointed out, but anomalies that could lead to program errors are detected.	[IEC 61508-7], C.5.10
Control flow analysis	Control flow analysis is a widely used approach for analysing functional and object oriented programs.	Static technique for predicting safe and computable set of functions that the objects of a program may assume during execution.	Identifies and provides an adequate understanding of the manner functions should be construct for a reliable and efficient system.	[IEC 61508-7], C.5.9

Method	Purposes	Applicability	Efficiency	References
Formal methods	As a notation for describing systems. As an analytical tool for calculating and predicting the behaviour of systems.	If applied from requirements specification to implementation, a complete verification is required. This is rarely done in practice	In principle, formal methods can accomplish the equivalent of exhaustive testing, if applied all the way from requirements to implementation.	[IEC 61508-3], A.1, A.2, A.4 & B.5
Worst case analysis	Exposes the performance of a system under extreme conditions of operation.	HW- Used for design analysis of MOS and BJT IC circuits. SW- identifies the time, memory or communication critical features.	Determines the worst case performance, but does not determine the statistical weighting of performance.	[IEC 61508-7], B.6.9
FMEA	FMEA is used for identifying failure modes of a system, or piece-part, and determining the effects at the next higher level.	FMEA can be used for SW, HW to support other analysis techniques such as FTA, DD, or MA.	FMEA is effective in finding single HW failures. It can reveal critical parts of the software.	[IEC 812]
FTA	FTA is used to show critical paths to hazardous situations. Focuses on one particular event (top event or failure condition) and provides a method the causes of this event.	FTA can be used for HW, SW, systems and it may include e.g. human actions and reliability calculations.	FTA gives a clear view of critical parts of the system.	[IEC 1025] [IEC 60300
Common mode analysis	Common mode analysis is used to complete other failure analysis e.g. FTA, Dependence diagram or Markov analysis.	Qualitative assessment to verify independence of Logically ANDed events in the FTA, dependence diagram, or Markov analysis.	Items containing identical hardware and/or software may be susceptible to common or generic faults which may cause simultaneous malfunctions in multiple items.	[IEC 61508-7], C 6.4.3
Markov modelling	Markov modelling is used to reliability calculations in a system. Markov Analysis technique represents various states and the rela-	Applicable method to model the failure performance of a repairable system over a wide range of failure rates and repair times	The method can be used to calculate realistic repair times, various system configurations, and realistic system features,	[IEC 61508-7], C.6.4 [IEC 61165] Application of Markov techniques

Method	Purposes	Applicability	Efficiency	References
	tionships among them.		including diagnostic testing for fault tolerant systems.	
Sneak circuit analysis Software sneak analysis	Sneak circuit analysis is used to detect an unexpected path within a system, which can initiate an undesired function or inhibit a desired function.	Applicable method to hardware, software and the combined system. Can be applied to programs written in any language.	It is labour intensive method. It is also likely to be performed late in the development cycle, so changes will be expensive.	[IEC 61508-7], C.5.11
Fagan inspections	Fagan inspection is a formal walkthrough method. The method has certain fixed phases.	Fagan inspection covers the whole design process	Fagan inspection is a cost effective method to find errors or failures in design. Errors in details may be hard to find.	[IEC 61508-7], C.5.15
Walkthroughs	Walkthroughs are used to ensure that the safety-related system complies with the requirements given in the specification.	The manufacturer of the product examine regarding doubts and weak points concerning the realisation and use of the product. Doubts and weak points are documented.	Walkthroughs helps the actor to find doubts and weak points so that they may be resolved.	[IEC 61508-7], B.3.8
Simulation	Simulation is used for a systematic and complete inspection of an electrical or electronic circuit. Including both the functional performance and the correct dimensioning of the components.	The function of the safety-related circuit's are simulated on a computer using behavioural model software.	Simulation informs about functional performance and the correct dimensioning of the components.	[IEC 61508-7], B.3.6
Fault injection	Fault injection techniques is used to introduce or simulate faults into system software or in some cases into hardware. The response is then documented.	Detailed functional block and source code is used to describe the location and type of fault and how fault injection is introduced.	Fault injection is a qualitative method of assessing dependability. By applying fault injection methods to seriously stress the software design during execution, system	[GIL

Method	Purposes	Applicability	Efficiency	References
			designers gain valuable information about new hazards unaccounted in earlier analysis.	

5 Failure rate calculation

The present chapter gives the background and the support for the calculation of failure rate. The computed probability of a dangerous failure ultimately will provide a value for the comparison with the target failure value of the specified SIL.

5.1 Background

The probability of a failure for components is a time dependant function and it resembles a so-called bathtub curve (see Figure 5.1.1). At first there may be some quality problems in the product and therefore the failure rate is high. This period lasts usually less than 1000 hours.
Then comes the useful life period and the failure rate stays constant. In many cases the failure rate tends to increase slightly as time increases, but this phenomenon is usually ignored. It is a constant failure rate value that is practice used for the reliability calculations. This period lasts often several years. The constant failure rate value does not show how long the component lasts. It only shows how often the object fails per time unit in average.
At the end comes the wear-out period or the end of the life period characterised by an increase in the failure rate. In safety related systems the components are usually removed and changed before reaching this period. [HØYLAND], [DHILLON]

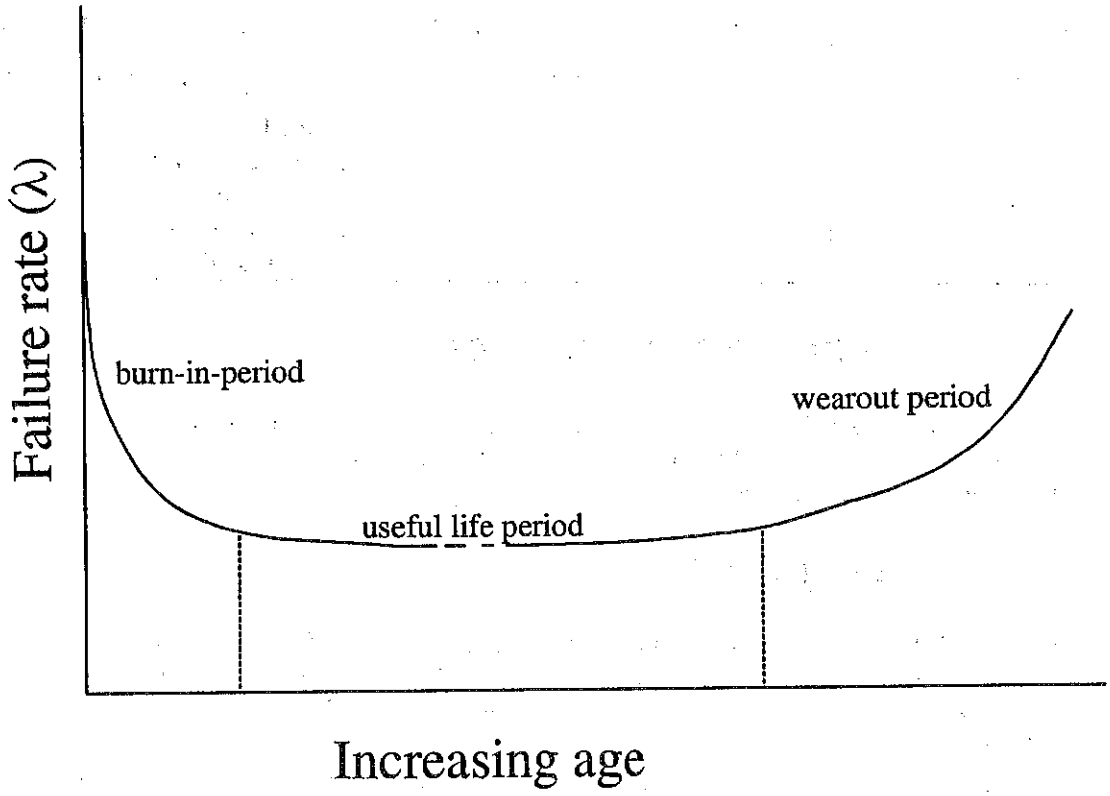

Figure 5.1.1 Failure rate

The failure rate is described as a time dependent function. It is a so-called "bathtub" curve. The "useful life" period is much longer than the "burn-in" period but longer than the "wear-out" period.

Initial values

The MTTF (mean time to failure) value, the $MTTF_d$ value (where $_d$ stands for dangerous) and the current constant failure rate value are used in safety calculations. When the failure rate is constant the relation between failure rate (λ) and MTTF is:

$$\lambda=1/MTTF \quad (1)$$

The MTTF values are provided by different sources. These values may vary substantially from one source to another. In some sources, factors such as component temperature, load and quality class are needed to calculate the MTTF. If, for example, the component load and the temperature are not estimated correctly the difference between two extreme values of MTTF may be two orders of magnitude.

Fault trees and reliability calculations

The reliability function describes the probability that the object has succeeded in performing its task within a specified time. At start (t=0) the probability value is 1 (the object operates perfectly) and at eternity the probability value is zero (the object has failed). The reliability function for a constant failure rate in an exponential distribution is:

$$R(t)= e^{-\lambda t} \quad (2)$$

In the equation λ is the failure rate (failures/hour) and t is the time value in hours. [DHILLON]

In these FTA calculations only OR- and AND-gates are considered. Here are some remarks for solving the equations.

For OR gate the total reliability is:

$$R(t)_{tot}=R_1(t)*R_2(t)*...R_n(t) = e^{-\lambda 1 t} * e^{-\lambda 2 t} *... * e^{-\lambda n t} = e^{-(\lambda 1+\lambda 2+...+\lambda n)*t} \quad (3)$$

In the equation $R(t)_{tot}$ means the reliability in the output of the gate and $R_1(t)$, $R_2(t)$... are the inputs to the gate.

The failure rate (λ) value of an OR gate is derived from equation (3) by using natural logarithm. The failure rate is received by summing the input λ values. [DHILLON]

$$\lambda_{total}=\lambda_1+\lambda_2+...+\lambda_n \quad (4)$$

The reliability function for an AND gate is the following:

$$R(t)_{tot}=1-(1-R_1(t))*(1-R_2(t))*...(1-R_n(t)) \quad (5)$$

If one combines equations (2) and (5) it can be seen that the result is complex and time-dependent function. The MTTF value can be estimated by integrating the reliability function R(t) from zero to eternity. This gives an average value over all time. It is not possible to calculate the reliability values during a specific time period by using the single MTTF value, because the failure rate is not a constant but a time dependent function. [HØYLAND]

$$MTTF= {}_0\int^{\infty} R(t)dt \quad (6)$$

By using equations (2) and (5) for calculating $R(t)_{tot}$, and integrating equation (6), and then finding the equivalent numerical series the formula can also be expressed in the following way: [Dhillon et. al. 1981]

$$MTTF_{total}=1/\lambda_1+1/\lambda_2+...+1/\lambda_n -(1/(\lambda_1+\lambda_2)+1/(\lambda_1+\lambda_n)+...+ 1/(\lambda_{n-1}+\lambda_n))+...+(-1)^{n-1}/(\Sigma\lambda_n) \quad (7)$$

Equations (5), (6) and (7) do not consider the cases of a failure occurring within a certain time period. In some cases the failure is revealed if the failures do not happen in a certain critical order. In some cases the failures are revealed during a periodic test function or normal operation. It is possible that a failure is revealed immediately after it has occurred. All these cases result in a specific reliability function.

It is not well defined how to calculate the MTTF for redundant devices. The reason is that failure rates may have different definitions and assumptions and the failure rate for redundant systems is a time dependent function. An agreement is needed to have common

definitions. It would be possible to compare the MTTF calculations of two devices if the calculation method and the input source were the same.

Discussion

Several factors cause uncertainty in reliability calculations. Quite often the absolute values are not exact, but the values can well be compared to other calculations, which are calculated in the same way. If the results are used only for comparison, many of the uncertainty factors are minimised. Figure 2 shows the main factors, which cause uncertainty to reliability estimations.

Using single values when comparing the reliability of systems is much criticised because it may give misleading results if the shape of the reliability function is not the same for all systems. If the system has no redundancy and all components are expected to comply with exponential distribution there is no problem. In this case the calculation of the freedom of critical events is made by using fault trees. There is redundancy and the curves do not have exactly the same shape. Figure 5.1.2 (Relex Software) shows the problem although the devices comply with the Weibull distribution. The figure shows the reliability function for three different devices, which have the same MTTF value. It can be seen from the figure that when the reliability is e.g. 0.5 the expected time is totally different. The example shows an extreme but possible case. The shape of the reliability functions of complex fault trees also can be complicated.

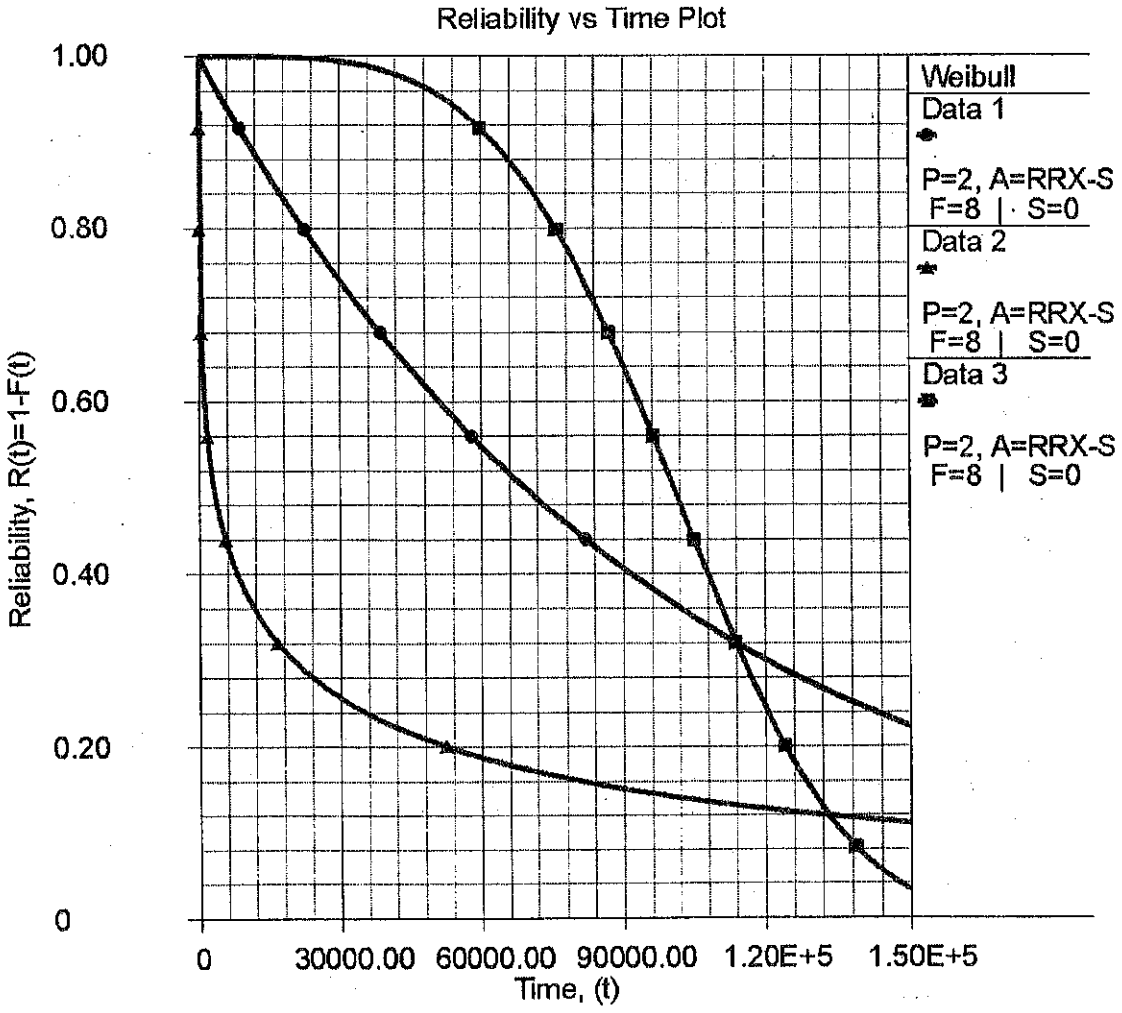

Figure 5.1.2 Reliability function (Weibull distribution)

The accuracy in the calculation of failure models for single components has been discussed. Parts count method and part stress analysis give average values, but they do not take into account e.g. design. The physics-of-failure method gives accurate results, but it is more difficult to get initial values for the calculations. In the method for each failure mode a failure model is created. [Alanen et. al. 2001]

Table 5.1.1 Factors which cause uncertainty to failure rate calculations. The factors are related to component failure rates, component failure distributions and system models

Cause for uncertainty	Means to better results
Component quality is not always the same. This causes deviation to the reality.	Quality problems are usual in the beginning. A trial period should last so long that all quality problems are detected.
Most of the failure models are approximations of the real components.	Consider which data is most suitable or should an experimental data be used. Physics of failure method considers different failure mechanisms first separately. The method gives good results if the input data is accurate.
How to treat electromechanical components, which age mainly according to switching number and load, not time.	Consider which data is most suitable. If the usage differs from general case it should be considered when choosing the failure rates for the components.
Components develop continuously and all models do not consider this.	It is better to use an updated date in calculations. It is also possible to test the components and calculate the failure rate values from the test results.
If one uses data from different sources then usually the most pessimistic values dominate the results.	It would be best to use same source for all components. If this is not possible, the difference between the sources should be considered.
Most of the failure distribution data are not accurate. Failure mechanisms in different circumstances affect the failure modes and the models do not always consider this.	If the distribution is not known and one wants to make a pessimistic approximation, the value 100% can be used. For some unusual failures the value 50% is commonly used. The effect on the results is usually not noticeable.
It is difficult to estimate a failure distribution for integrated circuits, which may have complex critical failures (e.g. short circuit between certain pins).	A pessimistic assumption of the value for a general short circuit should be used for any short circuit in the integrated circuit. The same principles can also be used for other failure modes.
The environment of the system may be unpredictable	A worst-case environment should be used in calculations to match the most pessimistic assumption.
The circuit architecture is usually not considered in the models	Protective components, circuit layout and proper enclosure increase the life length of the system although they may increase the amount of components.
The fault tree or system model is simplified and this may cause errors	Check that the model covers the worst-case situations.
The equations used for calculating failure rate for the complete system may have been different. MTTF may refer e.g. to simulation result to first critical failure, integral of reliability function from 0 to ∞ or certain reliability figure.	When comparing different systems the same calculation method should be used. E.g. in simulation process several parameters need to be defined. These definitions affect the results.
Usually single figures are used to estimate the reliability of a system. However, the shape of the reliability functions may differ and may give a misleading result.	When comparing two systems consider also the shape of the reliability function. Complete reliability functions may show better results than single figures.
How to combine failure rates of sub-functions? The functions may depend on each other.	The failure rates need to be independent to make calculations. If both functions independently and individually may cause the event for which the failure rate is calculated, the result is the sum of the failure rates. This gives also the worst-case result because the dependencies mitigate the result. On the other hand the total function may include some factors, for which the found sub-functions are lacking.

Reliability estimations can be related to the IEC 61508 standard. To reach a certain SIL it is necessary to conform to all the requirements of that level, not only the probability requirements of a hardware failure.

The result would be too optimistic if one top event was divided into several sub-events while the MTTF values were calculated separately. In some cases the consequences of the failures can be different and therefore it is difficult to summarise the complete risk. The load limiting function could be divided into two sub-functions:

- device fails totally to limit any overload or
- device fails to limit small overloads.

The consequences are different and also the failure rates for such events are different. More severe consequences are less probable than minor consequences. If the function is divided furthermore, the failure rate for the load limiting function is even more difficult to express.

5.2 Reliability block modelling

The reliability block modelling is a method for the quantification of the fault rate in terms of probability of dangerous hardware failure for a system.

The safety integrity level of a system is the synthesis of the integrity levels of its subsystems and components. This assumption is the guiding concept for the block modelling calculation process.

The calculation process is carried out in three steps according to the following:

A) The first step consists in the division of the system in subsystems and blocks. This arrangement characterises the system; it is the architecture of the system. According to the IEC 62061 standard each safety-related control function is defined in the SRECS safety requirement specification.

The architecture of the SRECS is extracted from the allocation of the safety requirements of the function blocks into subsystems.

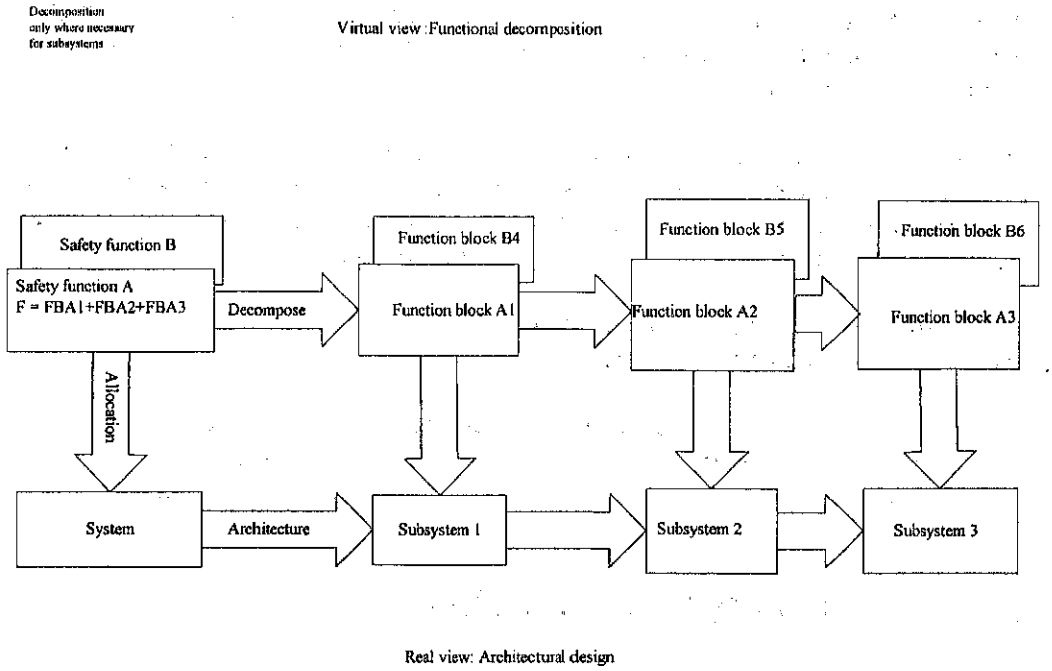

Figure 5.2.1 System decomposition
[IEC 62061]

B) The calculation of the fault rate for each specific block and subsystem is carried out.

C) The configuration of the system depending on how the blocks and subsystems interface with each other determines the final value of the fault rate for the whole system.

The block modelling enables the decomposition of a system into safety function blocks. Each function block shall correspond to the implementation of the corresponding safety function. The quantitative approach resides in the fact that the functionality of each function is associated to the condition of the electronic components used in that function block. The impact of component failures within a function block is analysed in the fault tree according to appendix B of this report.

5.2.1 Assumptions for the reliability block modelling

The calculations are based on the following assumptions (listed in the IEC 61508 standard):

- component failure rates are constant over the life of the system
- redundant components have similar failure rates
- the base β-factor for common cause failures, β, is two times the β-factor resulting from the diagnostic tests, β_D
- the mean time to restoration is an order of magnitude less than the expected demand rate
- for components or subsystems operating in high demand or continuous mode of operation, the SRECS always shuts down safely after detecting a dangerous fault. Consequently, the diagnostic test interval shall be of a magnitude less than the expected demand rate
- when a power supply failure removes power from a de-energize-to-trip SRECS and initiates a system trip to a safe state, the power supply does not affect the average probability of failure on demand of the SRECS; if the system is energized to trip, or the

power supply has failure modes that can cause unsafe operation of the SRECS, the power supply must be included in the evaluation
- for components or subsystems operating in high demand mode or continuous mode of operation , the SRECS always shuts down safely after detecting a dangerous fault; to achieve this, the diagnostic test interval is a magnitude less than the expected rate.

5.2.2 System configurations

The purpose of a SRECS is to control its output circuit. Arrangements are in some cases designed to minimize the probability of failure with output energized, others are designed to minimize the probability of failure with output de-energized. Some arrangements are designed to protect against specific failure modes. These various configurations are referred to as system architectures.

The single controller with a single processor unit and a single I/O is the simplest model for a SRECS. As a single repairable unit, any component failure will cause the controller to fail.

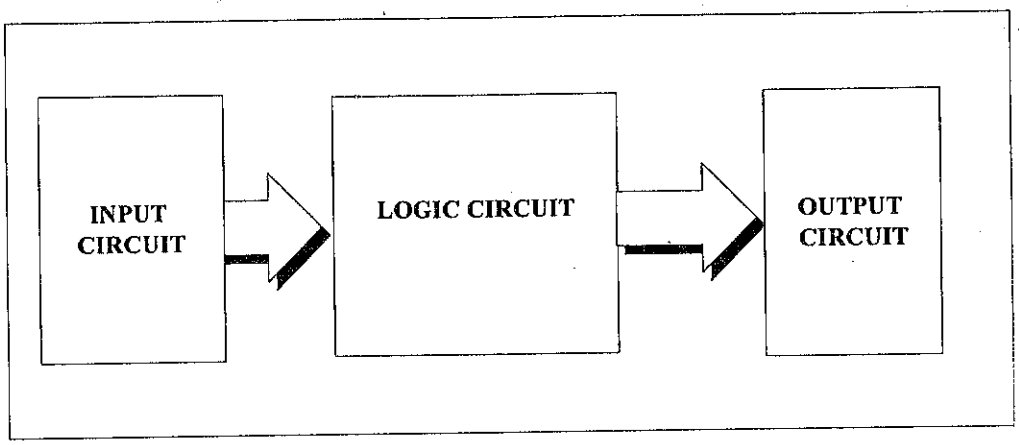

Figure 5.2.2.1 Basic programmable electronic controller

The failure rate for the basic controller is found by adding the failure rates of its constituting blocks.

$$\lambda_{controller} = \sum_{i=1}^{3} \lambda_{block\,i}$$

The types of failure rates to be considered are related to the different states that can occur under operation. Referring to IEC 61508 and IEC 62061, two types of failures are possible:

Safe failure ($\lambda_S = \lambda_{SU} + \lambda_{SD}$); a safe failure may cause a degraded level of operation but the safety integrity level is maintained.

Dangerous failure ($\lambda_D = \lambda_{DU} + \lambda_{DD}$); such a failure reduces the safety integrity level during operation and may result into a hazardous situation.

The probability of failure for each block is given by: $\lambda_{block} = \lambda_S + \lambda_D = \lambda_{SU} + \lambda_{SD} + \lambda_{DU} + \lambda_{DD}$

Assuming that dangerous failures and safe failures are equally distributed;

$\lambda_D = \lambda_S = 0.5\ \lambda$;

A fault-three analysis permits the identification of potentially dangerous faults for all the safety-related functions within a block.

$\lambda_{DU} + \lambda_{DD} = 0.5\ \lambda$ where λ is the failure rate for the current block.

Safety considerations are the only aspects that are taken into consideration in the calculation of failure probability of safety-related functions provided by a SRECS. Consequently, the probability for dangerous failures sums up to:

$\lambda_D = \lambda_{DU}$

This mathematical simplification is easily understood when considering the following diagram:

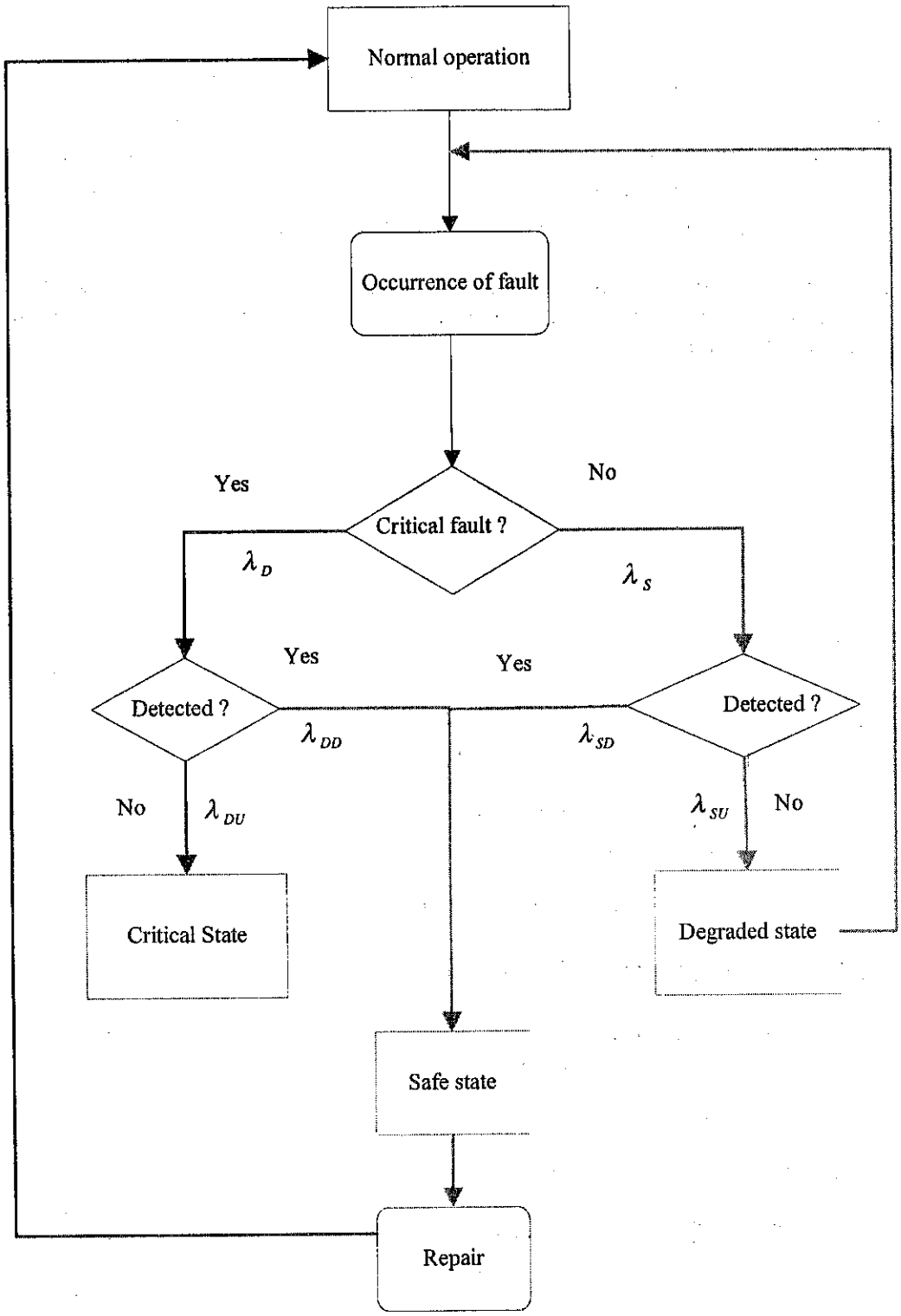

Figure 5.2.2.3 Fault occurrence diagram
Reference: [INRS]

A critical state occurs only when a dangerous fault is not detected. The remaining states can be considered as safe states.
In other words, λ_S sums up to:

$$\lambda_S = \lambda_{SU} + \lambda_{SD} + \lambda_{DD}$$

The average probability of failure for a SRECS in the simplest case is equivalent to the sum of individual λ_{DU} of each block. This is also the worst case since no consideration is given to the implementation of diagnostic tests for fault detection. To increase the accuracy in the estimation of λ_{DU}, the diagnostic coverage of the diagnostic tests as well as the associated test interval shall be taken into account.

Moreover, in redundant system configurations, common mode faults are likely to occur. Such faults affect the estimation of the probability of failure and should be accounted for in the block modelling.

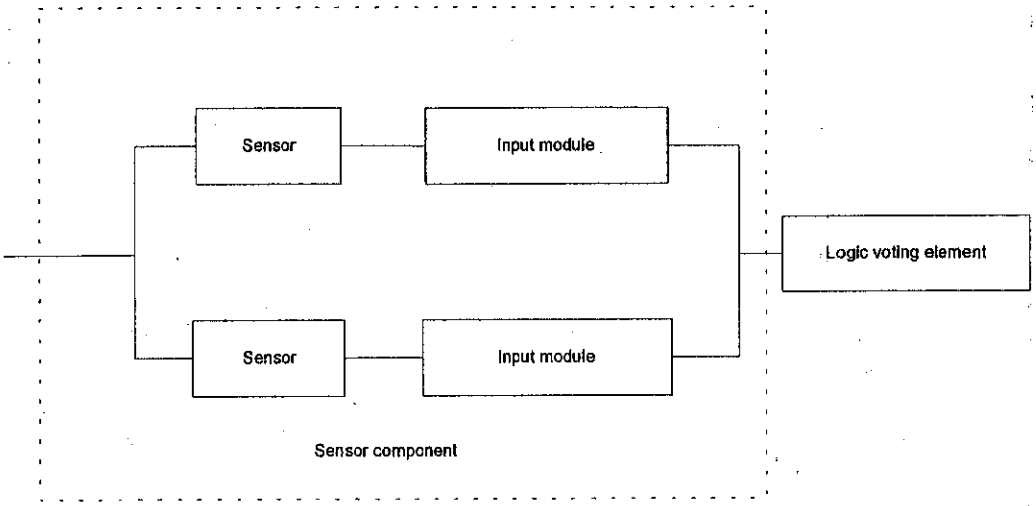

Figure 5.2.2.4 Example: configuration for two sensor channels

5.2.3 Average probability of failure per demand

A complete SRECS combines components in an architecture that has been analysed. Generically such a system can be represented by the following block configuration.

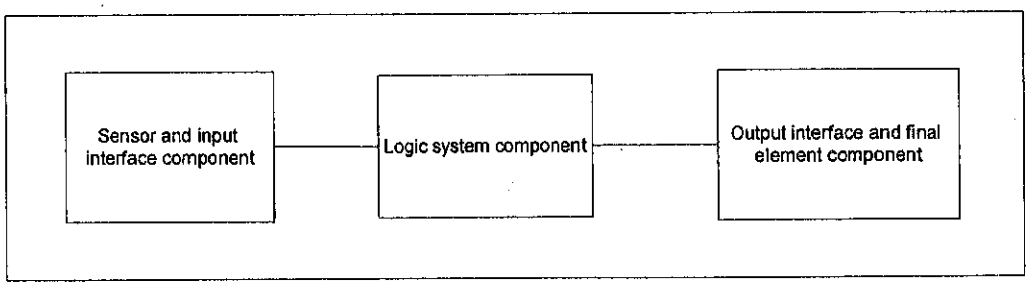

Figure 5.2.3.1 Block configuration

6 Validation of the safety lifecycle

The purpose of the validation of the SRECS is to determine if the SRECS has been designed and developed according to the functional requirements and the safety integrity level requirements that are in line with the IEC 62061 draft standard.

The aim of this process is to validate all the phases of the safety lifecycle and not to present hints on how to perform each design or development phase. Two aspects are considered:
- the implementation process
- the implemented system.

The following clauses address the validation process. They are intended to provide some support for the application of the IEC 62061 standard that is currently under development. The recommendations provided will remain valid even if substantial changes are expected in the final version of the IEC 62061 standard.

In the following part of this report, references are made to the current version of the standard. Since this standard is under development, the references may be subject to changes.

6.1 Validation of development process

This part of the validation shall primarily consider the fulfilment of the requirements for the management of functional safety (stated in IEC 62061 clause 4) regarding the establishment of a safety plan.

The second purpose of the validation is to follow the execution of the functional safety plan and to verify the execution of activities (prescribed in clause 5 of IEC 62061) regarding the requirements for the specification of safety-related control functions.

The third purpose of the validation is to ensure that the requirements (stated in clause 6 of IEC 62061) concerning the workflow for the design and the development process of a SRECS are fulfilled.

The validation shall focus on the following phases that concern the safety life cycle.

6.1.1 System description

A prerequisite to the validation of the system is the availability of a clear system description. [IEC 62061]
The sections 5.2.2 and 5.2.3 of the IEC 62061 standard specify the information required.

The description of the SRECS is such that all subsystems and their relationship are clearly documented. The following methods presented earlier in this work are applicable.
4.1.17 Unified modelling language (UML)
4.1.16 State transition diagram

6.1.2 Risk assessment

The goal of this activity is to ensure that the hazard and risk analysis has been carried out in a systematic way and that all possible risks and hazards are identified and documented.

References in this document:

4.1.3 Preliminary hazard analysis (PHA)
4.1.4 HazOp
4.1.2 Risk assessment
4.1.8 Static analysis
4.1.9 Dynamic analysis
4.1.5 Cause consequence analysis
4.1.6 Event tree analysis
4.1.20 Data flow analysis
4.1.21 Control flow analysis
4.1.22 Worst-case analysis
4.1.11 FTA
4.1.12 Common mode failure analysis
4.1.10 FMEA

[EN 1050]
[IEC 62061,] 6.10.2.7

6.1.3 Safety requirements

The goal of this activity is to make sure that there is a one to one correspondence between the identified risk and the safety requirement. The safety requirements shall specify the following safety-related control functions:

- functions that enable the machine to achieve or maintain a safe state
- functions related to the detection, monitoring and fault handling in all the subsystems of the SRECS
- functions related to the periodic testing of safety-related control functions on-line and off-line
- functions that allow the SRECS to be safely modified
- interfaces to non-safety control functions
- capacity and response time performance

The deduction of safety requirements is ultimately the result of the risk assessment. Hence the final set-up of requirements is a process that evolves from the impact of changes introduced in the course of system development. The following methods contribute to establish the safety requirement specification:

4.1.5 Cause consequence analysis
4.1.6 Event tree analysis
4.1.20 Data flow analysis
4.1.21 Control flow analysis
4.1.22 Worst-case analysis
4.1.11 FTA
4.1.12 Common mode failure analysis
4.1.10 FMEA

6.1.4 Safety-related control functions

The goal of this activity is to verify that the safety-related control functions necessary to fulfil the functional safety requirements for each identified hazard have been specified and implemented.

The methods to achieve that goal have been described earlier in this document.
The following method is applicable.

4.1.13 Walkthroughs and reviews

An extensive list of requirements is stated in IEC 62061, clause 5.2.3.

6.1.5 Safety integrity requirements for safety-related control functions

The goal of this activity is to verify that the claimed safety integrity level for each safety-related function complies with the requirements set up resulting from the risk and hazard analysis.

Each safety-related control function shall be decomposed into function blocks and each function block shall be allocated to a subsystem. The requirements (functionality, integrity) for each block are verified and an analysis of the definition of inputs and outputs of each function block is carried out.

The methods to achieve that goal have been described earlier in this document.
Any one of the following methods can be applied.

4.1.24 Markov modelling
4.1.11 Fault tree analysis
5 Failure rate calculation

The examination of intermediary results shall be considered as well.

The structure of each safety-related control function shall be defined according to clause 6.6.2.1 of IEC 62061.

6.2 Validation of the implemented system

The validation process concerns the design activities related to the risk and hazard analysis, the establishment of the safety requirements, the identification of the safety-related functions and the safety integrity set up for each safety function.

The second part of the evaluation shall focus on the definition and boundaries of the safety-related control system and the implementation of the necessary safety-related control functions that have been selected and developed.

The goal of this validation is to verify the compatibility of the subsystem elements that are interconnected for the realisation of subsystems. This activity includes the analysis of subsystem design tools and methods as well as the inspection of the design of subsystems and provision for diagnostic functions.

[IEC 62061], clause 6.4

6.2.1 Systematic safety integrity

The systematic safety integrity embraces system architecture, the choice of safety principles as well as the features developed to control systematic faults. Focus will be set on the identification of safety-related software modules and their interaction with hardware components. Safety-related faults/errors shall be defined in order to prevent their possible propagation in the system and to ensure a transition to any of the predefined safe states. The robustness of the SRECS shall be provided by an appropriate system architecture strengthened when necessary by techniques such as redundancy or diversity.

[IEC 62061], clause 6.7.8, 6.10

6.2.1.1 Safety principles

The goal of this activity is to verify if the measures and techniques used to enhance functional safety comply with the safety requirements and are sufficient to achieve the specified safety integrity level for each safety-related control function. The normal operation of a SRECS depends on the continuous or periodic monitoring of safety critical units or components, such as processor, memory, power supply, bus sensors, final elements and input/output.

[IEC 62061], clause 6.8
[IEC 61508-2], Table A.1- Faults or failures to be detected

6.2.1.2 Architecture

The aim of this activity is to analyse how the subsystems are combined to each other to implement the safety functions of the SRECS. The goal of the analysis is to assess the robustness of the current architecture in relation to the level of subsystem complexity.

[IEC 62061], clauses 6.7.6, 6.11.3.3

6.2.1.3 Control of systematic faults

The goal of this activity is to ensure that the design is such that systematic faults are detected and taken care of.

[IEC 62061], clause 6.4.2

6.2.2 Hardware safety integrity

The first purpose of this activity is to assess the methods used to calculate the failure rate of critical components used in the implementation of each safety-related control function. The second purpose of this activity is to check if the calculated failure rate is equal or less than the target failure value specified in the safety requirements specification.

[IEC 62061], clause 6.6.3.2
[IEC 61508-2], clause 7.4.3.2

6.2.2.1 Avoidance of hardware failures

The goal of this activity is to verify that basic features are applied to avoid potential faults in hardware.

[ISO 13849-2], B1, B2, B3, [IEC 62061], clause 6.4.1

6.2.2.2 Behaviour on detection of a fault in the SRECS

The goal of this activity is to carry out an examination of the requirements for system behaviour in occurrence of failures. The safety functions are tested by means of the following methods:
4.1.28 Functional testing
4.1.26 Simulation
4.1.27 Fault injection techniques

[IEC 62061], clause 6.3

6.2.2.3 Assessment of safety integrity level

The goal of this activity is to determine if the probability of failure claimed for each safety-related control function is achieved. The probability of dangerous hardware failure associated to a subsystem that implements the safety function shall be compared to the target failure value specified for that subsystem. The failure rate calculation results are inspected and put in relation to the diagnostic coverage of the components of that subsystem and the interval between all the tests that contribute to the diagnostic coverage.

[IEC 62061], clauses 5.2.4, 6.6.3

6.3 Validation results

6.3.1 The implementation process

The results of the validation shall give a judgement of the implementation process concerning the following activities:

System description
The subsystems and their relationship shall be clearly and fully documented. Without such a description the possibility to carry out the validation is limited.

Risk assessment
The risk analysis shall be carried out in a systematic way.
The context (operator, maintenance personnel, modes of operation, etc.) in which the system operates shall be defined.
The completeness of relevant risks listed is analysed.

Safety requirements
To each risk a safety requirement shall be assigned and each safety requirement shall be considered during the entire development process. The verification of the requirement specification shall ensure its consistency and completeness for the whole lifecycle of the control system.

Safety-related control functions
All safety-related control functions shall be implemented according to the safety requirements specification.

Safety integrity requirements for safety-related control functions
The claimed safety integrity level for each safety related function shall comply with the requirements set up resulting from the risk and hazard analysis.

6.3.2 The implemented system

The results of the validation shall give a judgement of the implemented system. The following points are considered:

- systematic safety integrity
- safety principles
- architecture
- control of systematic faults
- hardware safety integrity
- avoidance of hardware failures
- behaviour on detection of a fault in the SRECS
- assessment of hardware safety integrity level

The calculated failure rate values for the safety-related control functions are compared with the target failure values listed in Table 1 of the IEC 62061 standard. This comparison shall show if the SIL target has been met or not.

The safety-related control functions necessary to fulfil the functional safety requirements for each identified hazard shall be specified and implemented.

The analysis of the techniques and measures applied for the control of random hardware failures, systematic, environmental and operational failures shall determine if the consistency criteria are met or not

The validation results shall express if the detection of any critical faults results in the performance of the specified fault reaction function within a maximum time threshold.

The validation results shall express if the SIL claim for each safety-related control function of the system is in agreement with the architectural constraints and the estimated values of the probability of dangerous random hardware failures.

6.4 Analysis of results

The validation results shall express if:

- the specification of safety requirements according to the IEC 62061 standard clause 5.2 including the specification of software safety requirements clause 6.10 are met
- the architectural constraints on hardware safety integrity and the requirements for the probability of dangerous random hardware failures are met
- the requirements for the avoidance and control of failures are met
- the requirements for system behaviour on detection of a fault are met

7 Conclusion

Programmable electronic systems are increasingly used to perform safety functions in machinery equipment. If such systems are to be effectively and safely exploited, it is essential that designers and manufacturers have the adequate guidance when taking decisions concerning safety aspects. According to the IEC 62061 standard, the validation of safety in programmable systems shall cover all the phases of the safety life cycle.

The IEC 62061 standard specifies the requirements for the development and the validation of programmable systems. This report presents a list of qualitative methods and techniques applicable at different phases of development. The results obtained from such analysis are valuable when carrying out the validation of safety in programmable systems. A quantitative approach to safety evaluation is also covered with the introduction of the failure rate estimation, and illustrated by an example of calculation of the probability of dangerous failure for the stop function in a monitoring crane system.

An attempt to draw up a sequence of activities to be carried out in the validation process is presented in chapter six. Design failures and systematic failures may affect the execution of safety-related functions. By combining qualitative and quantitative techniques it is possible to estimate the reliability of a safety-related system. The present report highlights the numerous factors that affect the correctness of the failure rate calculations.

Appendix A References

[ALANEN] Alanen, J., Nevalainen, O., Palmen, H. Dependability of Electronics. Smart - Käke. Report VTT-AUT3 C-9907 January 2001.

[BISHOP] Bishop, P. G. Dependability of Critical Computer Systems 3. Elsevier Applied Science. 1990. ISBN 1-85166-544-7.

[CARA] Cara -FaultTree demo version v4.1. Sydvest Software.

[DENSON] Denson, W., Crowell, W., Jaworsky, P. &Mahar, D. 1997. Failure Mode/Mechanism Distributions. Reliability Analysis Center (RAC). FMD 97. New York. 440 p.

[DHILLON] Dhillon, B. & Singh, C. 1981. Engineering Reliability - New Techniques and Applications. A Wiley - Interscience publication. New York. 339 p. ISBN 0-471-05014-8.

[prEN1200] prEN12100-1:2001(E) Safety of machinery – Basic concepts, general principles for design – Part 1: Basic terminology, methodology.

[EN1050] EN 1050 Safety of machinery – Principles for risk assessment, November 1996.

[EN292-1] EN 292-1 Safety of machinery – Basic concepts, general principles for design – Part 1: Basic terminology, methodology.

[FMECA] ESA; Guidelines for Considering a Software Intensive System within FMECA Studies.

[HAMMER] Hammer, W. 1980. Product Safety Management and Engineering. Prentice-Hall, Inc., Englewood Cliffs, N.J. 324 p.

[HARJU] Harju, H. The quallitative assessment of software dependability. (In Finnish) VTT Research notes 2066. http://www.inf.vtt.fi/pdf/ 111 p.

[HAZOP] Felix Redmill, Morris Chudleigh and James Catmur; Systems Safety: HazOp and Software HazOp.

[HØYLAND] Høyland, A. & Rausand, M. 1994. System Reliability Theory. John Wiley & Sons, Inc. New York. 518 p. ISBN 0-471-59397-4

[IEC 1025] IEC 1025. 1990. Fault tree analysis (FTA). International Electrotechnical Commission 40 p.

[IEC 60300] IEC 60300-3-9.2000. Dependability management. Part 3: Application guide. Section 9: Risk analysis of technological systems.

[IEC 61508];
[IEC 61508–1] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 1: General requirements (1998).

[IEC 61508-2] Functional safety of electrical/electronic/programmable electronic safety-related system Part 2: Requirements for electrical/electronic/ programmable electronic safety-related systems (2000).

[IEC 61508–3] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 3: Software requirements (1998).

[IEC 61508–4] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 4: Definitions and abbreviations (1998).

[IEC 61508–5] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 5 Guideline on the application of Part 1.

[IEC 61508-6] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 6: Guidelines on the application of parts 2 and 3.

[IEC 61508-7] Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 7: Overview of techniques and measures.

[IEC 62061] Safety of Machinery - Functional safety of electrical, electronic and programmable control systems for machinery. 44/380/CD - 90 p, 2002.

[IEC 812] IEC 812. 1985. Analysis techniques for system reliability - Procedure for failure mode and effect analysis (FMEA). International Electrotechnical Commission. 41 p.

[INRS] Probabilité de défaillance dangereuse d'un système: explications et exemple de calcul. Pascal Lamy ISNN 0397 – 4529

[IPPOLITO] Ippolito, L. & Wallace, D. 1995. A study on Hazard Analysis in High Integrity Software Standards and Guidelines. NIST, Gaithersburg, U.S. Department of commerce. http://hissa.ncsl.nist.gov/HHRFdata/Artifacts/ITLdoc/5589/hazard.html#36_ SEC, 45 p.

[ISO 13849];
[ISO 13849-1]
[EN 954-1] prEN ISO 13849-1:2002
Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design (ISO/DIS 13849-1:2002)

[ISO 13849-2]
[EN 954-2] ISO/FDIS 13849-2:2002
Safety of machinery - Safety-related parts of control systems - Part 2: Validation (ISO/FDIS 13849-2:2002(E))

[KOP] Kopetz, H.; "Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, 1997.

[KOTEL] KOTEL 234. 1998. Failure rate of electronic components (In Finnish). KOTEL Ry. 17 p.

[LAP] Laprie, J. C., (ed.) "Dependability: Basic Concepts and Terminology", Springer-Verlag, Wien, 1992.

[MIL] MIL-HBK 217F. 1991. Military Handbook - Reliability Prediction of Electronic Equipment. Department of Defence. Washington DC.

[NASA] Formal Methods Specification and Verification Guidebook for Software and Computer Systems
Volume I: Planning and Technology Insertion

[NT] Hérard, J., Jacobson, J., Kivipuro, M., Malm, T., Ståhlhane, T. & Tiusanen, R. Guideline for the validation of Functional Safety according to IEC 61508. Nordtest report NT TECHN REPORT 459. http://www.vtt.fi/nordtest/technrap/tec459.pdf 48 p.

[PENG] Peng, W. & Wallace, D. Software Error Analysis. 1993. NIST. http://hissa.nist.gov/HHRFdata/Artifacts/ITLdoc/209/error.htm 83 p.

[RANKIN] Rankin, J.P. 1984. Sneak cicuits – A class of random, unrepeatable glitch. Plant/Operations Progress, Vol. 3, No. 3, pp. 175-178.

[RUMBAUGH] Rumbaugh, J., Jacobson, I & Booch, G. 1998. The Unified Modeling Language Reference manual, ACM Press, New York.

[SAN] Tom DeMarco; Structured Analysis and System Specification.

[SN 29500] Siemens (Ed.): Failure Rates of Components

[STSARCES] Final Report Safety-Related Complex Electronic Systems – European Project STSARCES Contract SMT 4CT97-2191.

[STSARCES1] Dorra, M. & Reinert, D. Quantitative Analysis of Complex Electronic Systems using Fault Tree Analysis and Markov Modelling. STSARCES final report annex 6. 52 p.

[STSARCES2] Malm, T. & Kivipuro, M. 1999. Safety Validation of Complex Components - Validation by Analysis. STSARCES final report annex 8. 53 p.

[STSARCES3] Gil, P. & Badiola, J. 1999. Safety Validation of Complex Components. Intercomparison black box/white box tests. STSARCES final report annex 9. 106 p.

Appendix B FTA and failure rate of CID

Load limiting devise (CID) description

CID is made for monitoring crane functions and to initiate some safety functions. The main function is to limit hoisting in a case of overload, but it also measures e.g. number of hoists, cumulative operation time, loads, supply voltage, motor temperature and possible wear out of brakes. Some of the inputs can cause alarm for the driver and some can prevent hoisting. CID is constructed of computer-based technology and it consists of a lot of parameters, which must be set by the manufacturer.

Several fault trees were made to estimate the failure rate of critical events and also to compare different methods. This report addresses one fault tree: Unexpected start-up in two-step control. Actually it covers the failure of a stop function.

Tasks

FTA is used in order to estimate the expected time to some critical events. The chosen events are the most critical events, but they are not the only possible events. The calculations are also made to compare different sources of failure rate values and to compare different methods in estimating the MTTF. Figure B1 shows the tasks that are used in making the FTA calculations.

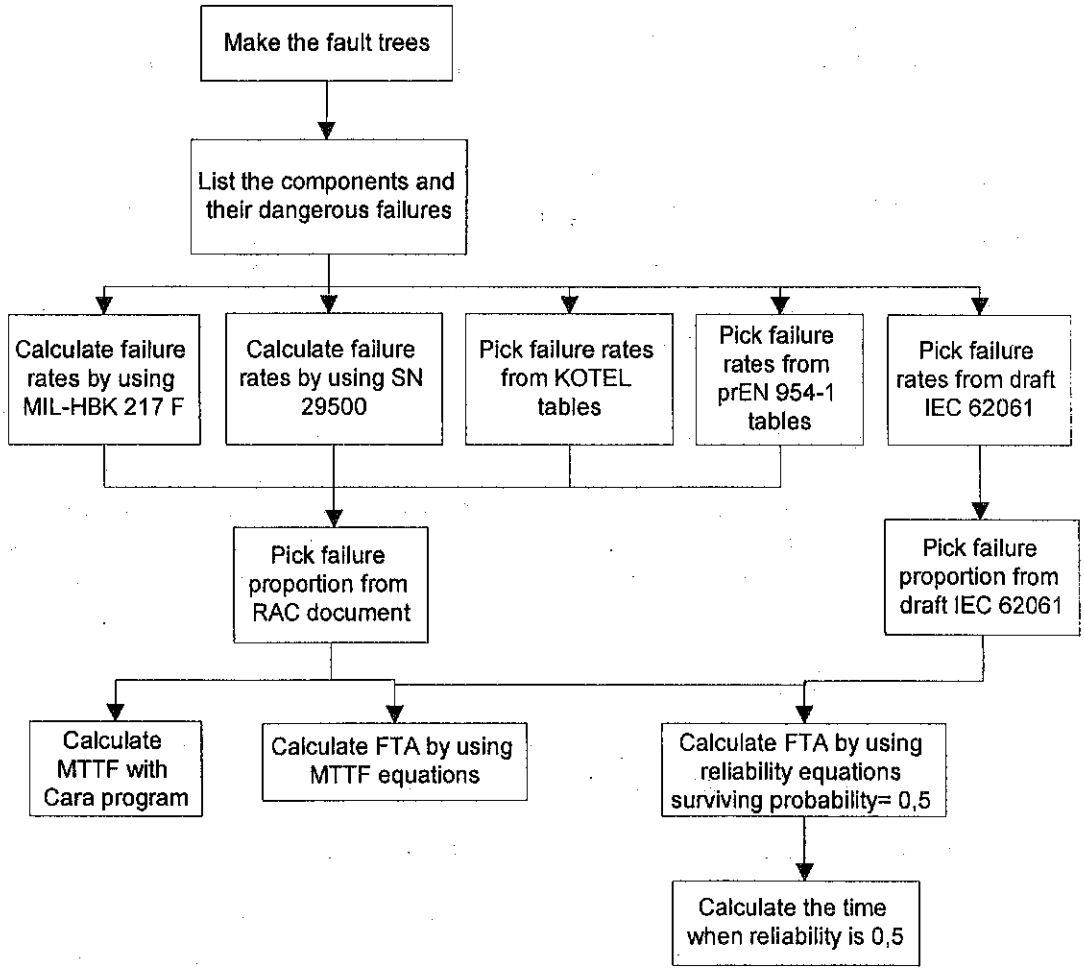

Figure B1. The tasks in calculating the FTA

Initial values

For all components used in the FTA an MTTF (mean time to dangerous failure) value is calculated. MTTF (or $MTTF_d$ where $_d$ stands for dangerous) values or actually constant failure rate values are used later in fault trees.

MTTF values are gathered from different sources and calculated separately in order to compare the differences in FTA results. The more specific values are calculated according to MIL-HBK 217F and SN 29500. For these calculations some component and environmental values are needed. Such values are component type, component class, component temperature and component load. All these values are estimated according to general use in measurement circuits (not heavy use). However, in real applications the conditions may vary and wrong environmental values in equations may cause over 100 fold results.

Experimental values are gathered from the KOTEL report. The component data has been collected between 1993 and 1995 from maintenance reports of two companies. The amount of service hours for each component type is something between 10^8 to 10^{11} hours. The failure rate for each component type is calculated by using statistical methods and 60 % certainty factor. Therefore the calculated failure rate was never zero even if no failures were detected. [KOTEL]

Initial values are gathered also from standard drafts IEC 62061 and ISO 13849-1,which, however, were not complete at the time of calculation. In ISO 13849-1 there was no data for any integrated circuits and in IEC 62061 there was no data for processors. When no data was available for a certain component, the worst values related to other components were used. Therefore, the calculations related to the standards give only an approximation.

When the failure rate for each component is known, the dangerous failures are set forth by the FTA and the failure distribution related to any specific failure mode is gathered from the RAC document Failure Mode/Mechanism Distributions. In IEC 62061 there is data on failure distribution. The initial values in the document are calculated by multiplying the constant hazard rate and the proportion value. Finally we obtain the failure rates for dangerous failures.

For integrated circuits it is difficult to estimate failure rates of dangerous failures since there are so many failure modes. In many cases e.g. only one short circuit in a certain pin may initiate a hazardous situation. The component may have 112 pins and an enormous variety of failure modes. If data about failure rate is available, dividing the current failure rate with all failure modes would give a negligible result. Such a procedure would not be appropriate since there are in the complex integrated circuit i.e. transistors controlling the output with a defined failure rate ($\neq 0$).

Fault trees and MTTF

The aim of the calculation is to get a failure rate or MTTF for some critical events caused by a failure in the CID system. Fault trees are used in these calculations.

Failure rates for a fault tree model are calculated in two steps. First the MTTF values for all possible OR-functions are computed, the AND-functions are put together and finally equations (1), (4) and (6) are used for calculating the final MTTF value for the whole tree (see chapter 5). The final value is an estimate over the time period from 0 to ∞. If there are AND-functions in the fault tree the value is not accurate in a short time period since the failure rate is in this case a time dependent function. [Dhillon]

Another way to calculate the failure rate is to use reliability functions. The fault tree is first simplified by calculating the failure rate for all possible OR-functions from the bottom. Then the failure rates are inserted in equation (2). The reliability for AND-

functions is calculated by using equation (7) and for OR-functions equation (3). To make the calculations, the fault tree reliability functions are inserted in a single cell of the Excel spreadsheet program. By inserting a time value, the cell gives the reliability factor as an output. The time value for which the reliability factor is 0,5 is calculated by using Excel Solver. A reliability of 0,5 means that when the time period has elapsed, half of the devices have failed and caused the top event.

There are also some assumptions that affect the results. A worst-case environment should be used in calculations to match the most pessimistic assumption.

- The fault tree for unexpected start-up is a little simplified since the top event would occur only if the initial events happen in a certain order. The right order is not considered, and so, the calculations are on the safe side.
- In some cases the failure remains invisible until the second failure causes the top event. In these calculations the failures are expected to remain invisible.
- Manual testing can reveal some failures. This procedure has been ignored.
- Other failures do not affect the top event calculations, i.e. other failures do not weaken the system so that the top event would be more probable.
- It is assumed that the failure rate for all components is constant and if the device fails in a particular way (not specified in the fault tree) it is replaced with a similar device with similar failure rates.
- Maintenance time is neglected in calculations.

Calculation results

Unexpected start-up in two-step control

Figure B2 shows the fault tree for an unexpected start-up and the failure possibilities related to the stop function. The component list under each basic event presents the components and their failure modes. Each failure mode can give rise to the basic event (components are combined with OR-function). The fault tree is simplified in order to handle only AND- and OR-functions. If the top event is initiated the commands up- or down remain activated without any possible intervention of CID. Table B1 shows the input values (MTTF) for the fault tree.

Simple relay failures were also calculated so that both alternatives S1 and S2 or S1 and S3 fail. The relays are important factors in the fault tree and their MTTF values have a greatest impact compared with the MTTF values of nearly all other components.

Table B1 shows the components with their MTTF value and failure distribution related to the following fault tree. Input values in the columns associated with the EN 954-1 and IEC 62061 standards are under development and the values for CPU were estimated in line with other components.

Table B1. Components with their MTTF value and failure distribution

Component	Type	Proportion of dangerous failures	Proportion ... IEC 62061	Kotel	SN 29500	MIL-HBK 217F	ISO 13849-1	Draft IEC 62061
C11	CAPACITOR 47N 275VAC 20% R15	0,059	0,4	1,64E+08	4,44E+09	4,42E+08	2,28E+04	3,00E+05
R16	RESISTOR 3K32 1 % 200 V	0,375	0,8	1,00E+11	1,05E+11	3,67E+09	5,00E+09	2,00E+05
R17	-"-	0,375	0,8	1,00E+11	1,05E+11	3,67E+09	5,00E+09	2,00E+05
C23	CER CAPACITOR 20 % 500 V	0,3	0,4	1,64E+08	3,50E+10	4,31E+08	2,28E+04	3,00E+05
N2	SMD OPTOISOLATOR	1	1	4,35E+08	4,51E+09	5,16E+07	*1,48E+04*	6,00E+05
R23	RESISTOR 1K00 1 % 50 V	0,375	0,8	1,00E+11	1,05E+11	6,87E+09	5,00E+09	2,00E+05
V18	SMD NPN 160 V 300 mA	0,48	0,25	2,00E+09	1,23E+10	3,64E+08	6,00E+08	6,00E+04
CPU input	Motorola MC68HC912DG128A	0,279	0,2	6,71E+07	7,33E+08	7,60E+06	*1,48E+04*	*3,00E+05*
S2	15 A RELAY 12 VDC	0,1	0,25	6,25E+08	2,51E+08	7,66E+07	1,26E+04	4,00E+05
S3	-"-	0,1	0,25	6,25E+08	2,51E+08	7,66E+07	1,26E+04	4,00E+05
S1	15 A RELAY 48 VDC	0,1	0,25	6,25E+08	2,51E+08	8,31E+07	1,26E+04	4,00E+05
CPU input	Motorola MC68HC912DG128A	0,279	0,2	6,71E+07	7,33E+08	7,60E+06	*1,48E+04*	*3,00E+05*

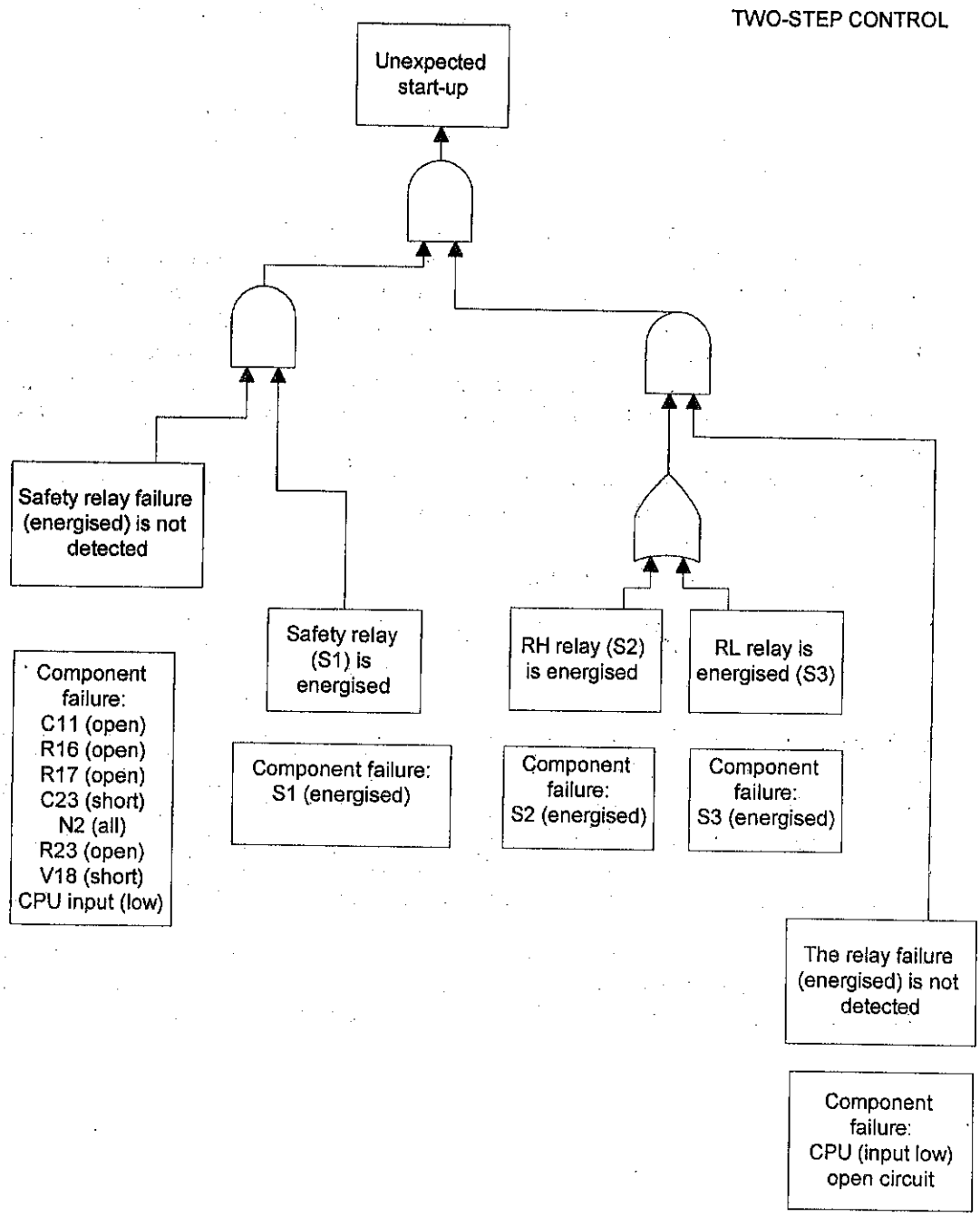

Figure B2. FTA of unexpected start-up in two-step control.

Table B2 shows the calculation results. Here are some explanations to the table:
MTTF calc.: Equations (1), (4) and (6) have been used to estimate MTTF.

- Surviving probability 0,5: MTTF is estimated by calculating the time when the reliability function value is 0,5.
- Cara MTTF: Cara is a commercial FTA program, which estimates the time for the first failure by simulating the fault tree. The used time period for calculation was 10^{14}. The time was chosen so that it is clearly longer than MTTF for any component. Then several failures exist during the simulation period. The selected time period always affects the results.

- Simplified FT: In this case there are only relays in the fault tree. The result gives the probability to critical relay failure. In the top event the CPU can still operate and it detects the failure. The calculations are made according to "MTTF calc.".

- MTTF all failure: In this case failure distribution information is removed and the failure proportion is 100% for all failure modes. The calculation is made according to "MTTF calc."

Table B2 shows that since the input values of draft IEC 62061 and ISO 13849-1 differ from other input values, the results are also very different. For the other input values the highest and lowest results differ from each other by an order of magnitude clearly less than one. The "Simplified FT" row shows the results for a simple fault tree, with only three relays. Since the relays are so important in the tree, the results resemble the first row. The "MTTF all failure" row shows the results when the failure modes are ignored and the results differ from the "MTTF calc." results by an order of magnitude close to one.
In Table 2, the row "MTTF all failure" shows the results with no information on failure distribution by simply using the same formulas as in "MTTF calc.".

Table B2. MTTF calculation results

	Kotel	SN 29500	MIL-HBK 217F	Draft IEC 62061	ISO 13849-1
MTTF calc.	7,29E+09	4,30E+09	9,52E+08	2,46E+06	1,57E+05
Surviving prob. 0,5	5,68E+09	3,67E+09	7,34E+08	2,06E+06	1,27E+05
Cara MTTF	5,73E+09	3,21E+09	7,32E+08	1,91E+06	1,22E+05
Simplified FT	7,29E+09	2,93E+09	9,52E+08	1,87E+06	1,46E+05
MTTF all failure	*7,31E+08*	*9,81E+08*	*9,55E+07*	*5,68E+05*	*2,19E+04*

Figure B3 shows the reliability function shape. When the time value is zero the reliability is 1 and when time increases the reliability slowly approaches zero. When the MTTF is calculated according to "MTTF calc" the surviving probability figure is about 0,37.

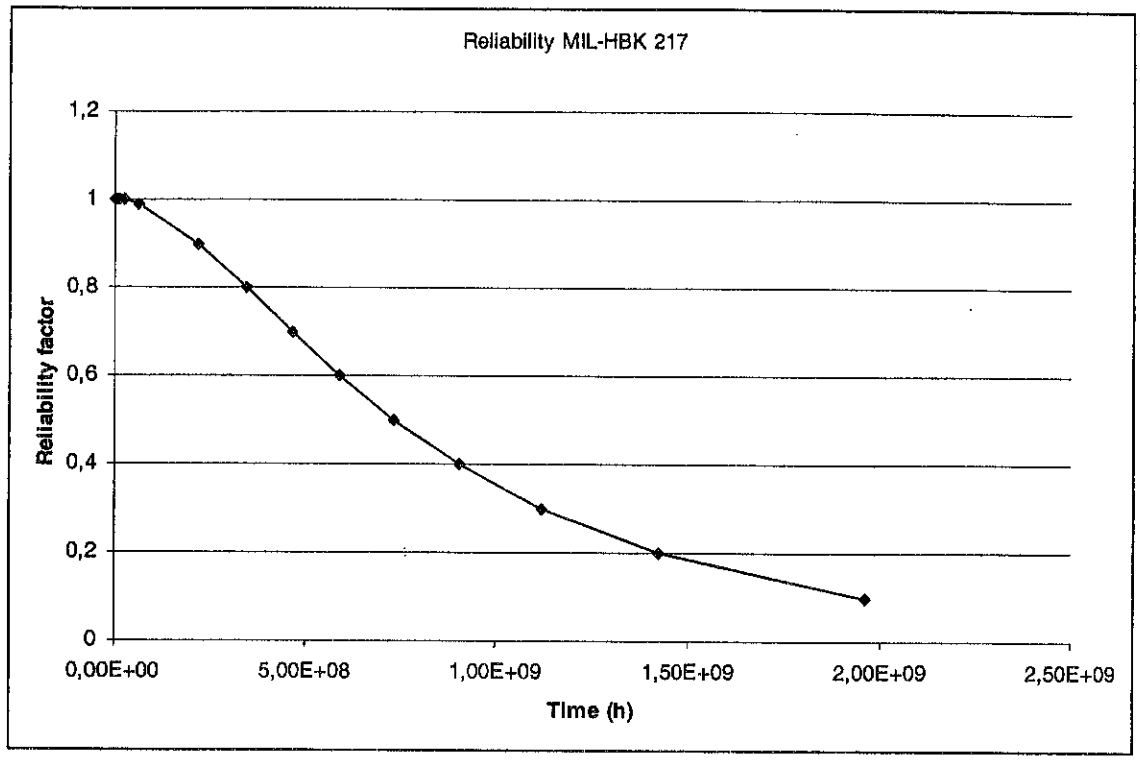

Figure B3. The reliability function is calculated by using the MIL-HBK 217 F figures as input.

Summary of calculation results

When comparing input values it can be seen that the most pessimistic data is provided by standard drafts IEC 62061 and ISO 13849-1. However, neither data base source of these two standards is complete and since the standards give initial values for safety-related calculations the values need to be on the safe side, i.e. very pessimistic. The next pessimistic values are in the MIL-HBK 217 and then in SN 29500. The most optimistic values are in the KOTEL data base. According to literature the MIL-HBK 217F gives too pessimistic results. [KOTEL],[Høyland]

The calculations show that input values have a determining effect on the final results. Therefore, it is not possible to compare the reliability or the safety of two devices if the failure rate sources are different. SN 29500 and MIL HBK 217F need e.g. information about component type and operating conditions for the calculations.