STUK-YTO-TR 190 / AUGUST 2002

FAILURE MODE AND EFFECTS ANALYSIS OF SOFTWARE-BASED AUTOMATION SYSTEMS

Haapanen Pentti, Helminen Atte

(VTT Industrial Systems)

In STUK this study was supervised by Marja-Leena Järvinen

STUK • SÄTEILYTURVAKESKUS
STRÅLSÄKERHETSCENTRALEN
RADIATION AND NUCLEAR SAFETY AUTHORITY

Osoite/Address • Laippatie 4, 00880 Helsinki
Postiosoite / Postal address • PL / P.O.Box 14, FIN-00881 Helsinki, FINLAND
Puh./Tel. (09) 759 881, +358 9 759 881 • Fax (09) 759 88 500, +358 9 759 88 500 • www.stuk.fi

ISBN 951-712-584-4 (print)
ISBN 951-712-585-2 (pdf)
ISSN 0785-9325

Dark Oy, Vantaa/Finland 2002

HAAPANEN Pentti, HELMINEN Atte (VTT Industrial Systems). Failure mode and effects analysis of software-based automation systems. STUK-YTO-TR 190. Helsinki 2002. 35 pp + Appendices 2 pp.

Keywords: safety, safety analysis, reliability analysis, automation, programmable systems, software-based systems, reactor protection systems, nuclear reactor safety, failure mode and effects analysis

Abstract

Failure mode and effects analysis (FMEA) is one of the well-known analysis methods having an established position in the traditional reliability analysis. The purpose of FMEA is to identify possible failure modes of the system components, evaluate their influences on system behaviour and propose proper countermeasures to suppress these effects. The generic nature of FMEA has enabled its wide use in various branches of industry reaching from business management to the design of spaceships. The popularity and diverse use of the analysis method has led to multiple interpretations, practices and standards presenting the same analysis method.

FMEA is well understood at the systems and hardware levels, where the potential failure modes usually are known and the task is to analyse their effects on system behaviour. Nowadays, more and more system functions are realised on software level, which has aroused the urge to apply the FMEA methodology also on software based systems. Software failure modes generally are unknown—"software modules do not fail, they only display incorrect behaviour"—and depend on dynamic behaviour of the application. These facts set special requirements on the FMEA of software based systems and make it difficult to realise.

In this report the failure mode and effects analysis is studied for the use of reliability analysis of software-based systems. More precisely, the target system of FMEA is defined to be a safety-critical software-based automation application in a nuclear power plant, implemented on an industrial automation system platform. Through a literature study the report tries to clarify the intriguing questions related to the practical use of software failure mode and effects analysis.

The study is a part of the research project "Programmable Automation System Safety Integrity assessment (PASSI)", belonging to the Finnish Nuclear Safety Research Programme (FINNUS, 1999–2002). In the project various safety assessment methods and tools for software-based systems are developed and evaluated. The project is financed together by the Radiation and Nuclear Safety Authority (STUK), the Ministry of Trade and Industry (KTM) and the Technical Research Centre of Finland (VTT).

HAAPANEN Pentti, HELMINEN Atte (VTT Tuotteet ja tuotanto). Ohjelmoitavien automaatiojärjestelmien vikaantumis- ja vaikutusanalyysi. STUK-YTO-TR 190. Helsinki 2002. 35 s. + liitteet 2 s.

Avainsanat: turvallisuus, turvallisuusanalyysi, luotettavuusanalyysi, automaatio, ohjelmoitavat järjestelmät, reaktorin suojausjärjestelmät, reaktoriturvallisuus, vika- ja vaikutusanalyysi

Tiivistelmä

Vika- ja vaikutusanalyysi (VVA) on tunnettu analyysimenetelmä, jolla on vakiintunut asema perinteisissä luotettavuusanalyyseissä. VVA:n tavoitteena on tunnistaa järjestelmän komponennttien mahdolliset vikantumistavat, arvioida niiden vaikutuksia järjestelmän käyttäytymiseen ja ehdottaa sopivia vastatoimenpiteitä haitallisten vaikutusten estämiseksi. VVA:n yleispätevä luonne on mahdollistanut sen soveltamiseen mitä moninaisimpiin kohteisiin ulottuen liiketoiminnan hallinnasta avaruusalusten suunniteluun. Menetelmän laaja suosio ja erilaiset käyttötavat ovat johtaneet useisiin erilaisiin menettelyä koskeviin tulkintoihin, käytäntöihin ja standardien syntyyn.

VVA hallitaan hyvin järjestelmä- ja laitetasoilla, joilla mahdolliset vikaantumistavat yleensä tunnetaan ja tehtävänä on analysoida niiden vaikutuksia järjestelmän käyttäytymiseen. Nykyään yhä suurempi osa järjestelmien toiminnoista toteutetaan ohjelmistotasolla, mikä on herättänyt halun soveltaa VVA metodologiaa myös ohjelmoitaviin järjestelmiin. Ohjelmistojen vikaantumistapoja ei yleensä tunneta – ”ohjelmistot eivät vikaannu, ne vain voivat käyttäytyä ennakoimattomalla tavalla” – ja ne riippuvat sovelluksen dynaamisesta käyttäytymisestä.

Tässä raportissa on selvitelty vika- ja vaikutusanalyysin soveltamista ohjelmoitaviin järjestelmiin. Tarkemmin sanottuna analyysin kohteena on ajateltu olevan teolliseen automaatiojärjestelmään implementoitu ydinvoimalaitoksen turvallisuuskriittinen automaatiosovellus. Lähinnä kirjallisuuden perusteella on yritetty selvittää ohjelmoitavaan tekniikkaan liittyviä vika- ja vaiktusanalyysin erityisongelmia.

Tutkimus on ollut osa Suomen kansalliseen ydinturvallisuustutkimusohjelmaan (FINNUS 1999–2002) kuuluuvaa ”Ohjelmoitavan automaation turvallisuuden arviointi (PASSI)”-projektia. Projektissa on kehitetty ja arvioitu erilaisia ohjelmoitavien järjestelmien turvallisuuden arviointimenetelmiä. Projektia ovat rahoittaneet yhdessä Säteilyturvakeskus (STUK), Kauppa- ja teollisuusministeriö (KTM) ja Valtion teknillinen tutkimuskeskus (VTT).

Contents

Definitions

Terms

Analysis approach

Variations in design complexity and available data will generally dictate the analysis approach to be used. There are two primary approaches for the FMECA. One is the hardware approach that lists individual hardware items and analyzes their possible failure modes. The other is the functional approach that recognizes that every item is designed to perform a number of outputs. The outputs are listed and their failures analyzed. For more complex systems, a combination of the functional and hardware approaches may be considered.

Block diagrams

Block diagrams that illustrate the operation, interrelationships, and interdependencies of the functions of a system are required to show the sequence and the series dependence or independence of functions and operations. Block diagrams may be constructed in conjunction with, or after defining the system and shall present the system breakdown of its major functions. More than one block diagram is sometimes required to represent alternative modes of operation, depending upon the definition established for the system.

Compensating provision

Actions that are available or can be taken by an operator to negate or mitigate the effect of a failure on a system.

Contractor

A private sector enterprise engaged to provide services or products within agreed limits specified by a procuring activity.

Corrective action

A documented design, process, procedure, or materials change implemented and validated to correct the cause of failure or design deficiency.

Criticality

A relative measure of the consequences of a failure mode and its frequency of occurrences.

Criticality analysis (CA)

A procedure by which each potential failure mode is ranked according to the combined influence of severity and probability of occurrence.

Damage effects

The result(s) or consequence(s) a damage mode has upon the operation, function, or status of a system or any component thereof. Damage effects are classified as primary damage effects and secondary damage effects.

Damage mode

The manner by which damage is observed. Generally describes the way the damage occurs.

Damage mode and effects analysis (DMEA)

The analysis of a system or equipment conducted to determine—the extent of damage sustained from given levels of hostile damage mechanisms and the effects of such damage modes on the continued controlled operation—and mission completion capabilities of the system or equipment.

Design data and drawings

Design data and drawings identify each item and the item configuration that perform each of the system functions. System design data and drawings will usually describe the system's internal and interface functions beginning at system level and progressing to the lowest indenture level of the system. Design data will usually include either functional block diagrams or schematics that will facilitate construction of reliability block diagrams.

Detection

Detection is the probability of the failure being detected before the impact of the effect is realized.

Detection mechanism

The means or method by which a failure can be discovered by an operator under normal system operation or can be discovered by the maintenance crew by some diagnostic action.

Environments

The conditions, circumstances, influences, stresses and combinations thereof, surrounding and affecting systems or equipment during storage, handling, transportation, testing, installation, and use in standby status and mission operation.

Failure

Departure of a system from its required behaviour; failures are problems that users or customers see.

Failure cause

The physical or chemical processes, design defects, part misapplication, quality defects, part misapplication, or other processes which are the basic reason for failure or which initiate the physical process by which deterioration proceeds to failure.

Failure definition

This is a general statement of what constitutes a failure of the item in terms of performance parameters and allowable limits for each specified output.

Failure effect

The consequence(s) a failure mode has on the operation, function, or status of an item. Failure effects are classified as local effect, next higher level, and end effect.

Failure mode

The manner by which a failure is observed. Generally describes the way the failure occurs and its impact on equipment operation.

Failure mode and effects analysis (FMEA)

A procedure by which each potential failure mode in a system is analyzed to determine the results or effects thereof on the system and to classify each potential failure mode according to its severity.

FMECA—Maintainability information

A procedure by which each potential failure is analyzed to determine how the failure is detected and the actions to be taken to repair the failure.

FMECA planning

Planning the FMECA work involves the contractor's procedures for implementing their specified requirements. Planning should include updating to reflect design changes and analysis results. Worksheet formats, ground rules, assumptions, identification of the level of analysis, failure definitions, and identification of coincident use of the FMECA by the contractor and other organizational elements should also be considered.

Functional approach
The functional approach is normally used when hardware items cannot be uniquely identified or when system complexity requires analysis from the top down.

Functional block diagrams
Functional block diagrams illustrate the operation and interrelationships between functional entities of a system as defined in engineering data and schematics.

Ground rules and assumptions
The ground rules identify the FMECA approach (e.g., hardware, functional or combination), the lowest level to be analyzed, and include statements of what might constitute a failure in terms of performance criteria. Every effort should be made to identify and record all ground rules and analysis assumptions prior to initiation of the analysis; however, ground rules and analysis assumptions may be adjusted as requirements change.

Hardware approach
The hardware approach is normally used when hardware items can be uniquely identified from schematics, drawings, and other engineering and design data. This approach is recommended for use in a part level up approach often referred to as the bottom-up approach.

Indenture levels
The item levels which, identify or describe relative complexity of assembly or function. The levels progress from the more complex (system) to the simpler (part) divisions.

Initial indenture level
The level of the total, overall item which is the subject of the FMECA.

Other indenture levels
The succeeding indenture levels (second, third, fourth, etc) which represent an orderly progression to the simpler division of the item.

Interfaces
The systems, external to the system being analyzed, which provide a common boundary or service and are necessary for the system to perform its mission in an undegraded mode; for example, systems that supply power, cooling, heating, air services, or input signals.

Level of analysis
The level of analysis applies to the system hardware or functional level at which failures are postulated. In other words, how the system being analyzed is segregated (e.g., a section of the system, component, sub-component, etc.).

Occurrence
Rating scale that shows the probability or frequency of the failure.

Reliability block diagrams
Reliability block diagrams define the series dependence, or independence, of all functions of a system or functional group for each lifecycle event.

Risk Priority Number (RPN)
The risk priority number is usually calculated as the product of the severity, occurrence, and detection.

Severity
The consequences of a failure as a result of a particular failure mode. Severity considers the worst potential consequence of a failure, determined by the degree of injury, property damage, or system damage that could ultimately occur.

Single failure point
The failure of an item that would result in failure of the system and is not compensated for by redundancy or alternative operational procedure.

Threat mechanism
The means or methods which are embodied or employed as an element of a man-made hostile environment to produce damage effects on a system and its components.

Trade-off study reports

These reports should identify areas of marginal and state–of–the–art design and explain any design compromises and operating restraints agreed upon. This information will aid in determining the possible and most probable failure modes and causes in the system.

Undetectable failure

A postulated failure mode in the FMEA for which there is no failure detection method by which the operator is made aware of the failure.

Abbreviations

ACE	Abnormal Condition or Event
AIAG	Automotive Industry Action Group
BDA	Bi-directional Analysis
CCA	Cause-Consequence Analysis
CHA	Component Hazard Analysis
COTS	Commercial Off-the-Shelf
DFMEA	Design Failure Modes and Effects Analysis
DCD	Data Context Diagram
DFD	Data Flow Diagram
ETA	Event Tree Analysis
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects, and Criticality Analysis
FPA	Function Point Analysis
FPTN	Failure Propagation and Transformation Notation
FSE	Functions, Systems and Equipment
FTA	Fault Tree Analysis
HA	Hazard Analysis
HAZOP	Hazard and Operability Analysis
IEC	International Electrotechnical Commission
IEEE	Institute of Electronic and Electrical Engineers
ISO	International Standards Organisation
MOD	Ministry of Defence (UK)
O&SHA	Operating and Support Hazard Analysis
PHA	Preliminary Hazard Analysis
PHL	Preliminary Hazard List
RCA	Root Cause Analysis
RPN	Risk Priority Number
SAD	Software Architecture Description
SAE	Society of Automotive Engineers
SCA	Sneak-Circuit Analysis
SDD	Software Design Description
SFMEA	Software Failure Modes and Effects Analysis (SW FMEA)
SFMECA	Software Failure Modes, Effects, and Criticality Analysis (SW FMECA)
SFTA	Software Fault Tree Analysis
SHA	Software Hazard Analysis
SRS	Software Requirements Specification
SSCA	Software Sneak Circuit Analysis (SSCA)
SSP	Software Safety Plan
SSPP	System Safety Program Plan
SSSP	System Software Safety Process
SwHA	Software Hazard Analysis
ZNA	Zonal Hazard Analysis

1 Introduction

1.1 Overview

A failure mode and effects analysis (FMEA) can be described as a systematic way of identifying failure modes of a system, item or function, and evaluating the effects of the failure modes on the higher level. The objective is to determine the causes for the failure modes and what could be done to eliminate or reduce the chance of failure. A bottom-up technique such as FMEA is an effective way to identify component failures or system malfunctions, and to document the system under consideration.

The FMEA discipline was originally developed in the United States Military. Military procedure MIL-P-1629, titled procedures for performing a failure mode, effects and criticality analysis, is dated on November 9th, 1949. The method was used as a reliability evaluation technique to determine the effect of system and equipment failures. Failures were classified according to their impact on the military mission success and personnel/ equipment safety. The concept that personnel and equipment are interchangeable does not apply for example in the modern manufacturing context of producing consumer goods and therefore the manufacturers in different areas of industries have established new sets of priorities, guidelines and standards of their own. However, military procedure MIL-P-1629 has functioned as a model for latter military standards MIL-STD-1629 and MIL-STD-1629A, which illustrate the most widely used FMEA procedures.

Outside the military the formal application of FMEA was first adopted to the aerospace industry, where FMEA was already used during the Apollo missions in the 1960's. In the early 1980's, United States automotive companies began to formally incorporate FMEA into their product development process. A task force representing Chrysler Corporation, Ford Motor Company and General Motors Corporation developed QS 9000 standard in an effort to standardise supplier quality systems. QS 9000 is the automotive analogy to better known standard ISO 9000. QS 9000 compliant automotive suppliers must utilise FMEA in the advanced quality planning process and in the development of their quality control plans. The effort made by the task force led to an industry-wide FMEA standard SAE J-1739 issued by the Society of Automotive Engineers' in 1994.

Academic discussion on FMEA originates from the 1960's when studies of component failures were broadened to include the effects of component failures on the system of which they were a part. One of the earliest descriptions of a formal approach for performing a FMEA was given at the New York Academy of Sciences (see Coutinho, 1964). In the late 1960's and early 1970's several professional societies published formal procedures for performing the analysis. The generic nature of the method assisted the rapid broadening of FMEA to different application areas and various practices fundamentally using the same analysis method were created. Along with the digital revolution the FMEA was applied in the analysis of software-based systems and one of the first articles regarding to software failure mode and effects analysis (SWFMEA) was given in Reifer (1979). Even thought there is no explicit standard for SWFMEA, the standard IEC 60812 published in 1985 is often referred when carrying out FMEA for software-based systems. Closer review on standards related to FMEA and a literature survey on the topics are given in Chapters 2 and 3.

1.2 Failure Mode and Effects Analysis

A systematic thinking promoted by FMEA is relevant when a new product or system is developed. FMEA seeks for answer for questions like: what could go wrong with the system or process involved in creating the system; how badly might it go wrong; and what needs to be done to prevent failures? Kennedy (1998) lists the purposes of FMEA as follows:

- Identify potential design and process related failure modes. Ideally, the design or process is changed to remove potential problems in the early stages of development (Pries, 1998).
- Find the effects of the failure modes. Russomanno, Bonnell, and Bowles (1994) noted that FMEA allows a team to analyse the effect of each failure on system operation.
- Find the root causes of the failure modes. Barkai (1999) stated that an FMEA is designed to find the sources of the failures of a system.
- Prioritise recommended actions using the risk priority number. Kennedy noted that the risk priority number is computed using the probability of occurrence of the failure mode, the severity of the effect of the failure mode, and the probability of detection of the failure mode through manufacturing. Loch (1998) pointed out that FMEA could provide a prioritised list of potential failures.
- Identify, implement, and document the recommended actions. Pries noted that FMEA documents should be under formal document control. Kennedy stated that the recommended actions are to address failure modes with rankings that are considered unacceptable.

Other authors have presented some specifications and additional purposes for FMEA such as:

- Assess the safety of various systems and components (Russomanno, Bonnell, & Bowles, 1994). Pries (1998) pointed out that hazardous situations could be identified early in the development process. The developers can analyse the hazardous situation, the causes of the hazardous situation, and solutions to manage the hazardous situation.
- Highlight any significant problems with a design (Barkai, 1999). If feasible, the team modifies the design to avoid those problems. Johnson (1998) stated that FMEA could avoid expensive modifications to designs by identifying potential failures and preventing them.
- Serve as a prelude to test design (Pries, 1998). Pries noted that the software design FMEA is a listing of potential problems and is thus a listing of test cases. Pries pointed out that any effort put into the FMEA would be helpful in test case development. Pries stated that the test descriptions developed from the FMEA would challenge the software product at its weakest points by testing for anomalies.

In practice the answers are searched in FMEA through an iterative analysis process, for which the main phases are illustrated in Fig. 1. The analysis process starts from the identification of the scope of the system and the functions the FMEA is to be applied on. The development process flow charts and design plans of the system are used to support the identification. After the subject for the FMEA is confirmed the next step is to identify the potential failure modes in a gradual way. The technique of brainstorming has often proven to be a useful method for finding failure modes. There ex-

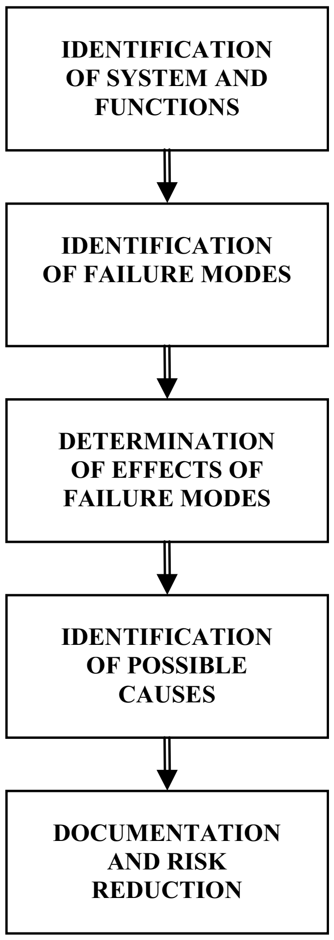
IDENTIFICATION OF SYSTEM AND FUNCTIONS
IDENTIFICATION OF FAILURE MODES
DETERMINATION OF EFFECTS OF FAILURE MODES
IDENTIFICATION OF POSSIBLE CAUSES
DOCUMENTATION AND RISK REDUCTION

Figure 1. *Main phases of FMEA.*

ists many different worksheets to support the brainstorm procedure and the documentation of FMEA overall. In the following phases the effects and causes of potential failures are determined. So called cause and effect diagrams can be used to help in these phases. The final step is to document the process and take actions to reduce the risks due to the identified failure modes.

The FMEA can generally be classified as either a product FMEA or a process FMEA depending upon the application. The product FMEA analyses the design of a product by examining the way that item's failure modes affect the operation of the product. The process FMEA analyses the processes involved in design, building, using, and maintaining a product by examining the way that failures in the manufacturing or service processes affect on the operation of the product. Some sources refer to a design and a process FMEA instead of a product and a process FMEA, however, both FMEA types focus on design – design of the product or design of the process – and therefore latter terminology is used in the text. The classification and different categories of failure mode and effects analysis of the product and process FMEA are presented in Fig. 2.

For software-based systems both the product and process FMEA are appropriate. In the process FMEA the production of the hardware of a software-based system may involve chemical processes, machining operations, and the assembly of subsystem components. The software production includes the production routines reaching from requirement specification to the final testing of the software. Use includes all of the ways a product may be used; it includes operator or other human interfaces, effects of over-stress conditions, and possible misuses of the system. Maintenance includes preventive and corrective maintenance as well as configuration control.

The product FMEA, which can be described as applying to the hardware or software of the product, or to the timing and sequencing of various system operations. Hardware includes, for example, a system's electrical, mechanical, and hydraulic subsystems and the interfaces between these subsystems. Software includes programs and their execution as tasks that implement various system functions. Software also includes the program interfaces with the hardware and those between different programs or tasks. Software can be embedded as a functional component in a self-contained system or executed on a general-purpose computer.

1.3 Software Failure Modes and Effects Analysis

Performing FMEA for a mechanical or electrical system is usually more straightforward operation than what it is for a software-based system. Failure modes of components such as relays and resistors are generally well understood. Reasons for the component failures are known and their consequences may be studied. Mechanical and electrical components are supposed to fail, due to some reason as wearing, ageing or unanticipated stress. The analysis may not always be easy, but at least, the safety engineers can rely on data provided by the component manufacturers, results of tests and feedback of available operational experience. For software-based systems the situation is different. The failure modes of software are generally un-

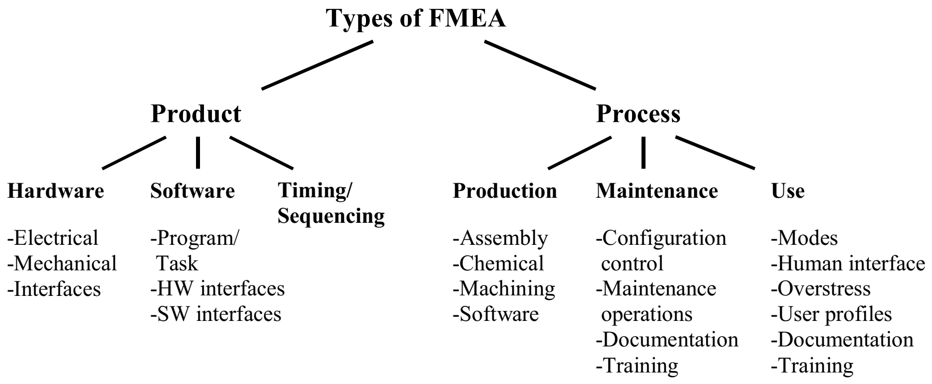

Figure 2. *Types of FMEA (SAG, 1996).*

known. The software modules do not fail they only display incorrect behaviour. To find out this incorrect behaviour the safety engineer has to apply his own knowledge about the software to set up an appropriate FMEA approach.

The main phases of SWFMEA are similar to the phases shown in Figure 1. The performer of SWFMEA has to find out the appropriate starting point for the analyses, set up a list of relevant failure modes and understand what makes those failure modes possible and what are their consequences. The failure modes in SWFMEA should be seen in a wide perspective reflecting the failure modes of incorrect behaviour of the software as mentioned above and not for example just as typos in the software code. The failure mode and effects analysis for hardware or software has certain distinguishing characteristics. Ristord et al. (2001) lists these characteristics as following:

Hardware FMEA:

- May be performed at functional level or part level.
- Applies to a system considered as free from failed components.
- Postulates failures of hardware components according to failure modes due to ageing, wearing or stress.
- Analyses the consequences of theses failures at system level.
- States the criticality and the measures taken to prevent or mitigate the consequences.

Software FMEA:

- Is only practicable at functional level.
- Applies to a system considered as containing software faults which may lead to failure under triggering conditions.
- Postulates failures of software components according to functional failure modes due to potential software faults.
- Analyses the consequences of these failures at system level.
- States the criticality and describes the measures taken to prevent or mitigate the consequences. Measures can, for example, show that a fault leading to the failure mode will be necessarily detected by the tests performed on the component, or demonstrate that there is no credible cause leading to this failure mode due to the software design and coding rules applied.

After the short historical review and general overview to the methodology of failure mode and effects analysis given in this chapter the structure of the report is as described below. In Chapter 2, a literature survey on the topic is carried out based on published information available. The key standards related to failure mode and effects analysis are covered in Chapter 3. In Chapter 4 the special features of the failure mode and effects analysis applied on software-based systems are discussed. Main conclusions on the topic are presented in Chapter 5.

2 Survey of literature

There is relatively little information published on the use of FMEA for information systems (Banerjee, 1995). Goddard (2000) noted that techniques for applying software FMEA to systems during their design have been largely missing from the literature. He pointed out that the number of papers dedicated to software FMEA has remained small. On the other hand, no papers have been written reporting on particular projects where FMEA were used unsuccessfully. The following survey has mainly been adapted from Signor (2000).

FMEA is a tool for identifying, analysing and prioritising failures. Pfleeger (1998) defined a failure as the departure of a system from its required behavior; failures are problems that users or customers see. Loch (1998) noted that potential failures may concern functionality, reliability, ease of repair, process efficiency, choice of manufacturing method, ease of detection, choice of tools, or the degree to which machines are used. Reifer (1979) provided several examples of software failures. An aircraft autoland system may fail because of a convergence in its vectoring algorithms during extreme operating conditions. A safety system used to monitor a nuclear power plant may have a logic error that allows a failure to dampen an overloaded reactor to go undetected. An error in a spacecraft program for controlling re-entry angle could cause skip-out and loss of mission.

FMEA provides a framework for a detailed cause and effect analysis (Russomanno, Bonnell, & Bowles, 1994). FMEA requires a team to thoroughly examine and quantify the relationships among failure modes, effects, causes, current controls, and recommended actions. Lutz and Woodhouse (1996) defined a failure mode as the physical or functional manifestation of a failure. Luke (1995) noted that software requirements not being met are failure modes. An effect in FMEA is the consequence of the failure mode in terms of operation, function, or status of the system (Bull, Burrows, Edge, Hawkins, and Woollons, 1996). The causes column in the FMEA worksheet contains the principal causes associated with each postulated failure mode (Onodera, 1997). Stamatis (1995) stated that current controls exist to prevent the cause of the failure from occurring. Bluvband and Zilberberg (1998) pointed out that current controls are used to detect failures. Controls include warning lights, gages, filters, and preventive actions such as design reviews and operator training. The recommended action in an FMEA may be a specific action or further study to reduce the impact of a failure mode (Stamatis, 1995).

By multiplying the values for severity, occurrence, and detection, the team obtains a risk priority number. Severity is the FMEA rating scale that shows the seriousness of the effect of the failure (McDermott, Mikulak, & Beauregard, 1996). Occurrence is the FMEA rating scale that shows the probability or frequency of the failure. Detection is the probability of the failure being detected before the impact of the effect is realized. Risk priority numbers help the team to select the recommended actions with the most impact.

Ease of Use

FMEA has many–to–many relationships among failure modes, effects, causes, current controls, and recommended actions (Bluvband & Zilberberg, 1998). A significant FMEA application soon bogs down in confusion due to large amounts of repetitive and redundant data. This problem is typical of FMEA applications. Montgomery, Pugh, Leedham, and Twitchett (1996) noted that entries in the FMEA worksheet are voluminous and, as a result, very brief. They pointed out that these numerous brief entries make FMEA reports difficult to produce, difficult to understand, and difficult to maintain. Passey (1999) pointed out that although some FMEA effects arise repeatedly, FMEA does not group together the items causing the effects. Passey noted that the placement of failure modes with the same effect on multiple pages tends to obscure the major failure modes that need to be addressed.

Cooper (1999) noted that a good system provides quick results. However, Montgomery et al. (1996) noted that the traditional brainstorming process for FMEA is tedious, time-consuming, and error-prone. Wirth, Berthold, Krämer, and Peter (1996) stated that FMEA is laborious and time-consuming to carry out. Parkinson, Thompson, and Iwnicki (1998) stated that FMEA is very time-consuming and must be focused to avoid masses of paperwork and workforce hostility. Tsung (1995) noted that FMEA often suffers from duplication of effort and large amounts of redundant documentation. The information overload from repetitive, redundant, and scattered data obscures the relationships among the rows and columns of the FMEA, adding to the confusion. Information overload is the inability to extract needed knowledge from an immense quantity of information (Nelson, 1994).

Extent of FMEA Usage

FMEA has been widely adopted and has become standard practice in Japanese, American, and European manufacturing companies (Huang et.al., 1999). Goddard (2000) stated that FMEA is a traditional reliability and safety analysis technique that has enjoyed extensive application to diverse products over several decades. Onodera (1997) investigated about 100 FMEA applications in various industries in Japan. He found that the FMEA is being used in the areas of electronics, automobiles, consumer products, electrical generating power plants, building and road construction, telecommunications, etc. Stamatis (1995) pointed out that FMEA is used in the electromechanical, semiconductor and medical device industries and for computer hardware and software. Hoffman (2000) noted that FMEA is a mainline reliability and maintainability tool. The three big United States automakers request that their suppliers use FMEA (Chrysler Corporation, Ford Motor Company, & General Motors Corporation, 1995).

Perkins (1996) stated FMEA applications in the aerospace and nuclear industries have seen an exponential increase in product software content and complexity. Stålhane and Wedde (1998) noted that several companies that develop embedded systems have been using FMEA for years. Goddard (2000) pointed out that application of FMEA to software has been somewhat problematic and is less common than system and hardware FMEA.

The FMEA in Information Systems

Goddard (2000) stated that there are two types of software FMEA for embedded control systems: system software FMEA and detailed software FMEA. System software FMEA can be used to evaluate the effectiveness of the software architecture without all the work required for detailed software FMEA. Goddard noted that system software FMEA analysis should be performed as early as possible in the software design process. This FMEA analysis is based on the top-level software design. Goddard stated that the system software FMEA should be documented in the tabular format used for hardware FMEA.

Goddard (2000) stated that detailed software FMEA validates that the software has been constructed to achieve the specified safety requirements. Detailed software FMEA is similar to component level hardware FMEA. Goddard noted that the analysis is lengthy and labor intensive. He pointed out that the results are not available until late in the development process. Goddard argued that detailed software FMEA is often cost effective only for systems with limited hardware integrity.

Banerjee (1995) provided an insightful look at how teams should use FMEA in software development. Banerjee presented lessons learned in using FMEA at Isardata, a small German software company. FMEA requires teamwork and the pooled knowledge of all team members. Many potential failure modes are common to a class of software projects. Banerjee also pointed out that the corresponding recommended actions are also common. Good learning mechanisms in a project team or in an organization greatly increase the effectiveness of FMEA. FMEA can improve software quality by identifying potential failure modes. Banerjee stated that FMEA can improve productivity through its prioritization of recommended actions.

Pries (1998) pointed out that a software FMEA can be used as a listing of potential test cases. The FMEA failure modes can be converted into test cases by developing procedures for stimulating the conditions that can lead to the failure. Pries argued that because each test case is bound to a particular failure mode, these test cases can become a particularly aggressive collection of test procedures.

Lutz and Woodhouse (1996) described their use of software FMEA in requirements analysis at the Jet Propulsion Laboratory. Software FMEA helped them with the early understanding of requirements, communication, and error removal. Lutz and Woodhouse noted that software FMEA is a time-consuming, tedious, manual task. Software FMEA depends on the domain knowledge of the analyst. Similarly, Fenelon and McDermid (1993) and Pfleeger (1998) pointed out that FMEA is highly labor intensive and relies on the experience of the analysts. Lutz and Woodhouse stated that a complete list of software failure modes cannot be developed.

Goddard (1993) described the use of software FMEA at Hughes Aircraft. Goddard noted that performing the software FMEA as early as possible allows early identification of potential failure modes. He pointed out that a static technique like FMEA cannot fully assess the dynamics of control loops. Goddard (1996) reported that a combination of Petri nets and FMEA improved the software requirements analysis process at Hughes Aircraft.

Moriguti (1997) provided a thorough examination of Total Quality Management for software development. Moriguti presented an overview of FMEA. The book pointed out that FMEA is a bottom-up analysis technique for discovering imperfections and hidden design defects. Moriguti suggested performing the FMEA on subsystem-level functional blocks. Moriguti noted that when FMEA is performed on an entire product, the effort often quite large. Moriguti pointed out that using FMEA before the fundamental design is completed can prevent extensive rework. Moriguti explained that when prioritization is emphasized in the FMEA, the model is sometimes referred to as Failure Modes, Effects and Criticality Analysis (FMECA).

Pries (1998) outlined a procedure for using software design FMEA. Pries stated that software design FMEA should start with system or subsystem outputs listed in the Item and Function (leftmost) columns of the FMEA. The next steps are to list potential failure modes, effects of failures, and potential causes. Pries noted that current design controls can include design reviews, walkthroughs, inspections, complexity analysis, and coding standards. Pries argued that because reliable empirical numbers for occurrence values are difficult or impossible to establish, FMEA teams can set all occurrences to a value of 5 or 10. Pries noted that detection numbers are highly subjective and heavily dependent on the experience of the FMEA team.

Luke (1995) discussed the use of FMEA for software. He pointed out that early identification of potential failure modes is an excellent practice in software development because it helps in the design of tests to check for the presence of the failure modes. In FMEA, a software failure may have effects in the current module, in higher level modules, and the system as a whole. Luke suggested that a proxy such as historical failure rate be substituted for occurrence.

Stamatis (1995) presented the use of FMEA with information systems. He noted that computer industry failures may result from software development process problems, coding, systems analysis, systems integration, software errors, and typing errors. Stamatis pointed out that failures may arise from the work of testers, developers, and managers. Stamatis noted that a detailed FMEA analysis may examine the source code for errors in logic and loops, parameters and linkage, declarations and initializations, and syntax.

Ammar, Nikzadeh, and Dugan (1997) used severity measures with FMEA for a risk assessment of a large-scale spacecraft software system. They noted that severity considers the worst potential consequence of a failure whether degree of injuries or system damages. Ammar, Nikzadeh, and Dugan used four severity classifications. Catastrophic failures are those that may cause death or system loss. Critical failures are failures that may cause severe injury or major system damage that results in mission loss. Marginal failures are failures that may cause minor injury or minor system damage that results in delay or loss of availability or mission degradation. Minor failures are not serious enough to cause injuries or system damage but result in unscheduled maintenance or repair.

Maier (1997) described the use of FMEA during the development of robot control system software for a fusion reactor. He used FMEA to examine each software requirement for all possible failure modes. Failure modes included an unsent message, a message sent too early, a message sent too late, a wrong message, and a faulty message. FMEA causes included software failures, design errors, and unforeseen external events. Maier noted that for software failures, additional protective functions to be integrated in the software may need to be defined. For design errors, the errors may need to be removed or the design may need to be modified. Maier stated that unforeseen external events may be eliminated by protective measures or changing the design. Maier recommended that the methodology he presented be applied at an early stage of the software development process to focus development and testing efforts.

Bouti, Kadi, and Lefrançois (1998) described the use of FMEA in an automated manufacturing cell. They noted that a good functional description of the system is necessary for FMEA. They recommended the use of an overall model that clearly specifies the system functions. They suggested the use of system modeling techniques that facilitate communication and teamwork. Bouti, Kadi, and Lefrançois argued that it is impossible to perform a failure analysis when functions are not well defined and understood. They pointed out that failure analysis is possible during the design phase because the functions are well established by then. Bouti, Kadi, and Lefrançois also noted that when several functions are performed by the same component, possible failures for all functions should be considered.

Becker and Flick (1996) applied FMEA in Lockheed Martin's development of a distributed system for air traffic control. They described the failure modes and detection methods used in their FMEA. The classes of failure modes for their application included hardware or software stop, hardware or software crash, hardware or software hang, slow response, startup failure, faulty message, checkpoint file failure, internal capacity exceeded, and loss of service. Becker and Flick listed several detection methods. A task heartbeat monitor is coordination software that detects a missed function task heartbeat. A message sequence manager checks message sequence numbers to flag messages that are not in order. A roll call method takes attendance to ensure that all members of a group are present. A duplicate message check looks for the receipt of duplicate messages.

Stålhane and Wedde (1998) used FMEA with a traffic control system in Norway. They used FMEA to analyse changes to the system. They noted that potentially any change involving an assignment or a procedure call can change system parameters in a way that could compromise the system's safety. The FMEA pointed out code segments or procedures requiring further investigation. Stålhane and Wedde also stated that for an FMEA of code modifications, implementation and programming language knowledge is very important.

FMEA Size

The sheer size of the FMEA presents a challenge. An FMEA worrksheet is at least 16 columns wide, is often several levels deep, and is sometimes dozens or hundreds of pages long. Montgomery et al. (1996) noted that entries in the FMEA worksheet are voluminous. Bull et al. (1996) pointed out that a manual FMEA is a bulky document. Parkinson, Thompson, and Iwnicki (1998) stated that FMEA must be focused to avoid masses of paperwork. Tsung (1995) noted that FMEA often suffers from large amounts of documentation. Large documents implemented in Excel require much scrolling and scrolling adversely affects ease of use (Plaisant et. al., 1997).

FMEA Basics

An FMEA worksheet has multiple rows with fixed column headings. The FMEA worksheet is often arranged in 16 columns, 11 for the FMEA Process and 5 for the Action Results (McDermott, Mikulak, & Beauregard, 1996). Passey (1999) pointed out that no universal FMEA format has emerged. The FMEA Process columns normally include Item and Function; Potential Failure Mode; Potential Effect(s) of Failure; Severity; Potential Cause(s) of Failure; Occurrence; Current Controls; Detection; Risk Priority Number; Recommended Action; and Responsibility and Target Completion Date. The Action Results columns include Action Taken, Severity, Occurrence, Detection, and Risk Priority Number. Kukkal, Bowles, and Bonnell (1993) stated that failure effects at one level of a system need to be propagated as failure modes to the next higher level.

Breyfogle (1999) described the road map for creating FMEA entries as follows. First, note an input or function of a process or design. Then, list two or three ways the input or function can go wrong. Next, list one or more effects of failure. For each failure mode, list one or more causes of input going wrong. Then, for each cause, list at least one method of preventing or detecting the cause. Last, enter the severity, occurrence, and detection values.

Stamatis (1995) wrote the most comprehensive reference available for FMEA (Schneider, 1996). Stamatis stated that there are four types of FMEA: the system FMEA, the design FMEA, the process FMEA, and the service FMEA. All types of FMEA produce a list of failure modes ranked by the risk priority numbers and a list of recommended actions to reduce failures or to improve failure detection.

The system FMEA is used to analyse systems and subsystems in the early concept and design phase (Stamatis, 1995). A system FMEA analyses potential failure modes between the functions of the system caused by system deficiencies. This FMEA includes the interactions between systems and the elements of the system. A significant output of the system FMEA is a list of potential system functions that could detect potential failure modes.

The design FMEA is used to analyse products before they are released to manufacturing (Stamatis, 1995). A design FMEA analyses failure modes caused by deficiencies in the design of a system. Significant outputs of the design FMEA are a list of critical and/or significant characteristics of the design; a list of potential parameters for testing, inspection, and/or detection methods; and a list of potential recommended actions for the critical and significant characteristics of the design.

The process FMEA is used to analyse manufacturing and assembly processes (Stamatis, 1995). A process FMEA analyses failure modes caused by process or assembly deficiencies. Significant outputs of the process FMEA are a list of critical and/ or significant characteristics of the process and a list of potential recommended actions to address the critical and significant characteristics.

The service FMEA is used to analyse services before they reach the customer (Stamatis, 1995). A service FMEA analyses failure modes such as tasks, errors, and mistakes caused by system or process deficiencies. Significant outputs of the service FMEA are a list of critical or significant tasks or processes, a list of potential bottleneck processes or tasks, and a list of potential functions for monitoring the system or process.

Hoffman (2000) noted that today's commercially available FMEA tools are nothing more than documentation aids. Russomanno, Bonnell, and Bowles (1994) and Montgomery et al. (1996) pointed out that these packages do not include any intelligence but merely assist with clerical functions, data collection, database manipulations, and automatic report generation.

Ristord et. al. (2001) describe the application of FMEA procedure on the new nuclear instrumentation system (NIS) at Tihange nuclear power plant in Belgium. The NIS-system is implemented on a generic software-based safety system platform SPINNLINE 3 by Schneider Electric Industries. The choice of a software-based technology has raised the issue of the risk of CCF due to the same software being used in redundant independent units. Implementing functional diversity or equipment diversity has been considered but found either not practicable or of little value within this context. Because of the possible consequences in case of the NIS not performing its protection function on demand, the licensing authority has required an FMEA oriented toward the SCCF risk as part of the safety case.

This FMEA has been performed on the NIS architecture, the SPINLINE 3 Operational System Software and the three application software packages (i.e. source, intermediate and power range). The software FMEA experience included the adaptation of the principles of FMEA to analyse a software program, the choice of a "block of instruction" (BI) approach to identify the components to analyse, the definition of the software failure modes associated with the BIs and the feedback of experience.

3 FMEA standards

3.1 Overview

As mentioned already in the previous chapters various industries have established their own FMEA standards to provide the best support for the product and process FMEA related to a specific branch of industry. Aerospace and defence companies have usually referred to the MIL-STD-1629A standard. Automotive suppliers have mostly used SAE J-1739 or the Chrysler, Ford, and General Motors provided FMEA methodologies. Other industries have generally adopted one of these or the IEC 60812 standard customising the standards to meet the specific requirements and needs of the industry. A short review to the standards referred in the text is given in this chapter.

3.2 IEC 60812

IEC 60812 published by the International Electrotechnical Commission describes a failure mode and effects analysis, and a failure mode, effects and criticality analysis. The standard gives guidance how the objectives of the analysis can be achieved when using FMEA or FMECA as risk analysis tools. The following information is included in the standard:

- procedural steps necessary to perform an analysis
- identification of appropriate terms, assumptions, criticality measures, failure modes
- determining basic principles
- form for documenting FMEA/FMECA
- criticality grid to evaluate failure effects.

3.3 MIL-STD-1629A

MIL-STD-1629A is dated on November 24th, 1980, and has been published by the United States Department of Defence. The standard establishes requirements and procedures for performing a failure mode, effects, and criticality analysis. In the standard FMECA is presented to systematically evaluate and document the potential impacts of each functional or hardware failure on mission success, personnel and system safety, system performance, maintainability and maintenance requirements. Each potential failure is ranked by the severity of its effect in order that appropriate corrective actions may be taken to eliminate or control the risks of potential failures. The document details the functional block diagram modelling method, defines severity classification and criticality numbers. The following sample formats are provided by the standard:

- failure mode and effects analysis
- criticality analysis
- FMECA- maintainability information
- damage mode and effects analysis
- failure mode, effects, and criticality analysis plan

MIL-STD-1629A was cancelled by the action of the standard authority on August 4th, 1998. Users were referred to use various national and international documents for information regarding failure mode, effects, and criticality analysis.

3.4 SAE J-1739

The document provides guidance on the application of the failure mode and effects analysis technique. The focus is on performing the product, process and plant machinery FMEA. The standard outlines the product and process concepts for performing FMEA on plant machinery and equipment, and provides the format for documenting the study. The following information is included in the document:

- FMEA implementation
- what is an FMEA?
- format for documenting product/process FMEA on machinery
- development of a product/process FMEA
- suggested evaluation criteria for severity, detection and occurrence of failure.

4 SWFMEA procedure

The Failure Mode and Effects Analysis procedures were originally developed in the post World War II era for mechanical and electrical systems and their production processes, before the emergency of software based systems in the market. Common standards and guidelines even today only briefly consider the handling of the malfunctions caused by software faults and their effects in FMEA and often state that this is possible only to a limited extent (IEC 60812). No specific standard or guideline concentrating on the special issues of software-based system FMEA has yet been published.

A general FMEA/FMECA procedure is described in the IEC 60812 standard. Present official version of this standard stems from the year 1985. IEC TC56/WG 2 is, however, working on a revision of the standard; a Committee Draft (CD) has been distributed to the committee members in spring 2001.

The general FMEA/FMECA procedure presented in IEC 60812 agrees well with other major FMEA standards and guidelines presented in Chapter 3. These procedures constitute a good starting point also for the FMEA for software-based systems. Depending on the objectives, level etc. of the specific FMEA this procedure can easily be adapted to the actual needs case by case.

In this chapter the whole structure of the FMEA procedure is not presented (refer to the IEC 60812). Instead, only the special aspects connected to the incorporation of the software failure modes and effects in the failure mode and effects analysis of a software-based automation system application are considered. These considerations are based on the findings from the (rather sparse) literature on the specific subject.

A complete FMEA for a software-based automation system shall include both the hardware and software failure modes and their effects on the final system function. In this report we, however, confine only on the software part of the analysis, the hardware part being more straightforward (if also difficult itself).

As a starting point for the following considerations the main procedural steps of a FMEA are defined to be (adapted from IEC 60812):

- Define system boundaries for analysis,
- understand system requirements and function,
- define failure/success criteria,
- breakdown the system into elements,
- determine each element's failure modes and their failure effects and record these,
- summarise each failure effect, and
- report findings.

FMEA is documented on tabular worksheet; an example of typical FMEA worksheet (derived from IEC 60812) is presented in App. 1; this can readily be adapted to the specific needs of each actual FMEA application. App. 2. gives an example of a SWFMEA form.

Criticality analysis is a quantitative extension of the (qualitative) FMEA. Using the failure effects identified by the FMEA, each effect is classified according to the severity of damages it causes to people, property or environment. The frequency of the effect to come about together with its severity defines the criticality. A set of severity and frequency classes are defined and the results of the analysis is presented in the criticality matrix (Fig. 3, IEC 60812). The SAE J-1739 standards adds a third aspect to the criticality assessment by introducing the concept of Risk Priority

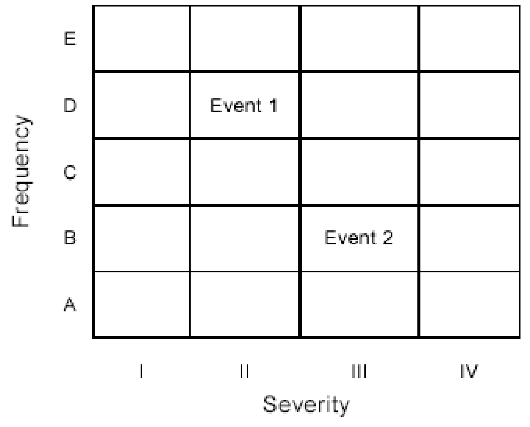

Figure 3. *The criticality matrix.*

Number (RPN) defined as the product of three entities, Severity, Occurrence (i.e. frequency) and Detection.

4.1 Level of analysis

The FMEA is a bottom-up method, where the system under analysis is first hierarchically divided into components (Fig. 4). The division shall be done in such a way that the failure modes of the components at the bottom level can be identified. The failure effects of the lower level components constitute the failure modes of the upper level components.

The basic factors influencing the selection of the proper lowest level of system decomposition are the purpose of the analysis and the availability of system design information.

When considering the FMEA of a software-based automation system application, the utmost purpose of the analysis usually is to find out if there are some software faults that in some situation could jeopardise the proper functioning of the system. The lowest level components from which the analysis is started are then units of software executed sequentially in some single processor or concurrently in parallel processor of the system.

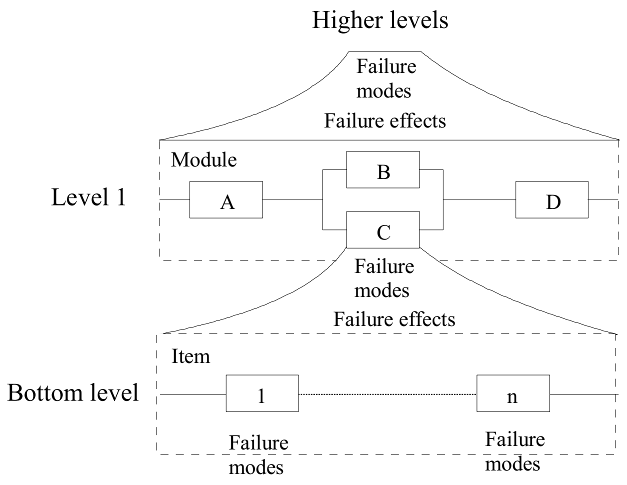

Figure 4. *The FMEA level hierarchy (adapted from IEC 60812).*

A well-established way to realise software-based safety-critical automation applications nowadays is to implement the desired functions on an automation system platform, e.g. on a programmable logic system or on a more general automation system. The software in this kind of realisation is on the first hand divided into *system software* and *application software.* The system software (a simple operating system) can further be divided into *the system kernel* and *system services.* The kernel functions include e.g. the system boot, initialisation, self-tests etc. whereas the system services e.g. take care of different data handling operations. The platform includes also a library of standardised software components, *the function blocks ("modules")*, of which the application is constructed by connecting ("configuring") adequate function blocks to form the desired *application functions*, which rest on the system service support (see Fig. 5). The automation system platform also contains a graphical design tool. Using this tool the designer converts the functional specifications of the application to a functional block diagram which is then automatically translated to the executable object code. The total design documentation of the application thus consists crudely of the requirement specification, the functional specification, the functional block diagrams and the (listing) of the object code.

A natural way of thinking would then suggest that the FMEA of a software-based application could be started from the function block diagrams by taking the individual function blocks as the lowest level components in the analysis. In practice this procedure, however, seems unfeasible. Firstly, this approach in most cases leads to rather extensive and complicated analyses, and secondly,

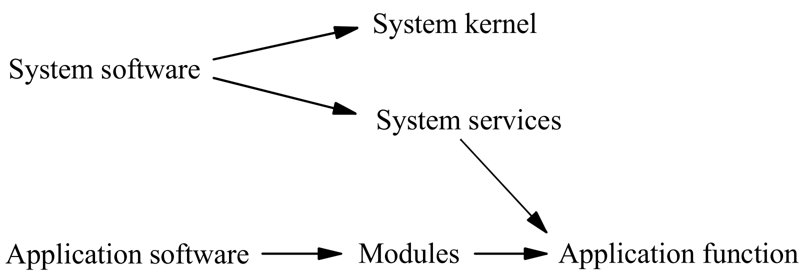

Figure 5. *Construction of the application function.*

the failure modes of the function blocks are not known.

E.g. Ristord et. al. (2001) state that the software FMEA is practicable only at the (application) function level. They consider the SPINLINE 3 application software to be composed of units called Blocks of Instructions (BIs) executed sequentially. The BIs are defined by having the following properties:

- BIs are either "intermediate"—they are a sequence of smaller BIs—or "terminal"—they cannot be decomposed in smaller BIs.
- They have only one "exit" point. They produce output results from inputs and possibly memorized values. Some BIs have direct access to hardware registers.
- They have a bounded execution time (i.e. the execution time is always smaller than a fixed value).
- They exchange data through memory variables. A memory variable is most often written by only one BI and may be read by one or several BIs.

Their system decomposition of the SPINLINE 3 software is presented in Fig. 6.

4.2 Failure modes

When a proper way of decomposing the system under analysis is found the next step is to define the failure modes of the components. For the hardware components this in general is straightforward and can be based on operational experience of the same and similar components. Component manufacturers often give failure modes and frequencies for their products.

For the software components such information does not exists and failure modes are unknown (if a failure mode would be known, it would be corrected). Therefore, the definition of failure modes is one of the hardest parts of the FMEA of a software-based system. The analysts have to apply their own knowledge about the software and postulate the relevant failure modes.

Reifer (1979) gives the following general list of failure modes based on the analysis of three large software projects.

- Computational
- Logic
- Data I/0
- Data Handling
- Interface
- Data Definition
- Data Base
- Other.

Ristord et. al. (2001) give a the following list of five general purpose failure modes at processing unit level:

- the operating system stops
- the program stops with a clear message
- the program stops without clear message
- the program runs, producing obviously wrong results
- the program runs, producing apparently correct but in fact wrong results.

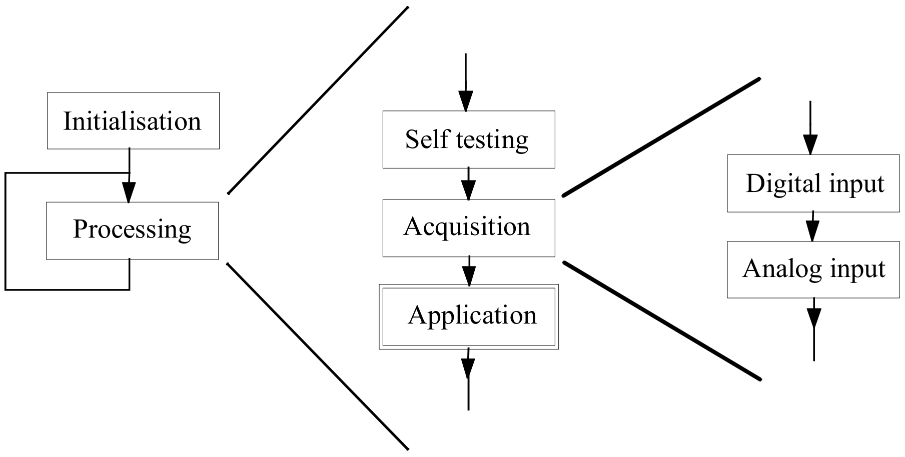

Figure 6. *First levels of the decomposition of the SPINLINE 3 software (Ristord et. al., 2001).*

More specifically, they define for the SPINLINE 3 BIs the following list of failure modes:

- the BI execution does not end through the "exit" point
- the BI execution time does not meet time limits
- the BI does not perform the intended actions or performs unintended actions:
    - it does not provide the expected outputs
    - it modifies variables that it shall not modify
    - it does not interact as expected with I/O boards
    - it does not interact as expected with CPU resources
    - it modifies code memory or constants.

Lutz et.al. (1999b) divide the failure modes concerning either the data or the processing of data. For each input and each output of the software component, they consider of the following four failure modes:

- Missing Data (e.g., lost message, data loss due to hardware failure)
- Incorrect Data (e.g., inaccurate data, spurious data)
- Timing of Data (e.g., obsolete data, data arrives too soon for processing)
- Extra Data (e.g., data redundancy, data overflow).

For each event (step in processing), on the other hand, they consider of the following four failure modes:

- Halt/Abnormal Termination (e.g., hung or deadlocked at this point)
- Omitted Event (e.g., event does not take place, but execution continues)
- Incorrect Logic (e.g., preconditions are inaccurate; event does not implement intent)
- Timing/Order (e.g., event occurs in wrong order; event occurs too early/late).

Becker et. al. (1996) give the following classes of failure modes:

- hardware or software stop
- hardware or software crash
- hardware or software hang
- slow response
- startup failure
- faulty message
- checkpoint file failure
- internal capacity exceeded
- loss of service.

They also listed the following detection methods:

- a task heartbeat monitor is coordination software that detects a missed function task heartbeat
- a message sequence manager checks the sequence numbers for messages to flag messages that are not in order
- a roll call method takes attendance to ensure that all members of a group are present
- a duplicate message check looks for the receipt of duplicate messages.

Still further, a very comprehensive classification of software failure modes is given by Beizer (1990).

IEC 60812 also gives guidance on the definition of failure modes and contains two tables of examples of typical failure modes. They are, however, largely rather general and/or concern mainly mechanical system thus not giving much support for software FMEA. The same goes also for other common standards and guidelines.

The FMEA also includes the identification and description of possible causes for each possible failure mode. Software failure modes are caused by inherent design faults in the software; therefore when searching the causes of postulated failure modes the design process should be looked at. IEC 60812 gives a table of possible failure causes, which largely are also applicable for software.

4.3 Information requirements

The IEC 60812 standard defines rather comprehensively the information needs for the general FMEA procedure. It emphasises the free availability of all relevant information and the active co-operation of the designer. The main areas of information in this standard are:

- system structure
- system initiation, operation, control and maintenance
- system environment
- modelling
- system boundary

- definition of the system's functional structure
- representation of system structure
- block diagrams
- failure significance and compensating provisions.

A well-documented software-based system design mostly covers these items, so it is more the question of the maturity of the design process than the specialities of software-based system.

4.4 Criticality analysis

As stated earlier, the Criticality Analysis is a quantitative extension of the FMEA, based on the severity of failure effects and the frequency of failure occurrence, possibly augmented with the probability of the failure detection. For an automation system application, the severity is determined by the effects of automation function failures on the safety of the controlled process. Even if difficult, the severity of a single low level component failure mode can in principle be concluded backwards from the top level straightforwardly.

The frequency of occurrence is much harder to define for a software-based system. The manifestation of an inherent software fault as a failure depends not only on the software itself, but also on the operational profile of the system, i.e. on the frequency of the triggering even that causes the fault to lead to failure. This frequency is usually not known. Luke (1995) proposed that a proxy such as McCabe's complexity value or Halstead's complexity measure be substituted for occurrence. Luke argued that there is really no way to know a software failure rate at any given point in time because the defects have not yet been discovered. He stated that design complexity is positively linearly correlated to defect rate. Therefore, Luke suggested using McCabe's complexity value or Halstead's complexity measure to estimate the occurrence of software defects.

Also the probability of detection is hard to define, since only a part of software failures can be detected with self-diagnostic methods.

5 Conclusions

Failure mode and effects analysis (FMEA) is a well-established reliability engineering tool widely used in various fields of industry. The purpose of FMEA is to identify possible failure modes of the system components, evaluate their influences on system behaviour and propose proper countermeasures to suppress these effects. FMEA is primarily adapted to the study of material and equipment failures and can be applied to systems based on different technologies (electrical, mechanical, hydraulic etc.) and combinations of these technologies (IEC 60812).

FMEA is well understood at the systems and hardware levels, where the potential failure modes usually are known and the task is to analyse their effects on system behaviour. Nowadays, more and more system functions are realised on software level, which has aroused the urge to apply the FMEA methodology also on software-based systems. Software failure modes generally are unknown ("software modules do not fail, they only display incorrect behaviour") and depend on dynamic behaviour of the application. These facts set special requirements on the FMEA of software based systems and make it difficult to realise. A common opinion is that FMEA is applicable to software-based systems only to a limited extent.

Anyhow, the total verification and validation (V&V) process of a software-based safety critical application shall include also the software failure mode and effects analysis of the system at a proper level. FMEA can be used in all phases of the system life cycle from requirement specification to design, implementation, operation and maintenance. Most benefit from the use of FMEA can be achieved at the early phases of the design, where it can reveal weak points in the system structure and thus avoid expensive design changes.

Perhaps the greatest benefit from the FMEA applied on a software-based system is the guidance it can give for other V&V efforts. By revealing the possible weak points it can e.g. help generating test cases for system testing.

The FMEA can not alone provide the necessary evidence for the qualification of software-based safety critical applications in nuclear power plants, but the method should be combined with other safety and reliability engineering methods. Maskuniitty et. al. (1994) propose a stepwise refined iterative application of fault tree analysis (FTA) and failure mode and effects analysis procedure as described in Fig. 7. This is similar with the Bi-directional Analysis (BDA) method proposed by Lutz et. al. (1997, 1999b). The preliminary fault tree analysis and determination of minimal cut sets directs the identification of failure modes to those that are most significant for the system reliability, the effects analysis of these failures, then steer the refinement of the fault trees and the final detailed FMEA.

The fault trees and reliability block diagrams are basically static approaches that can not reach all the dynamical aspects of the software. Other approaches, e.g. Petri net analysis, Dynamic flowgraph analysis etc. have been proposed to catch these effects in the reliability and safety analysis of software-based system (Pries, 1998).

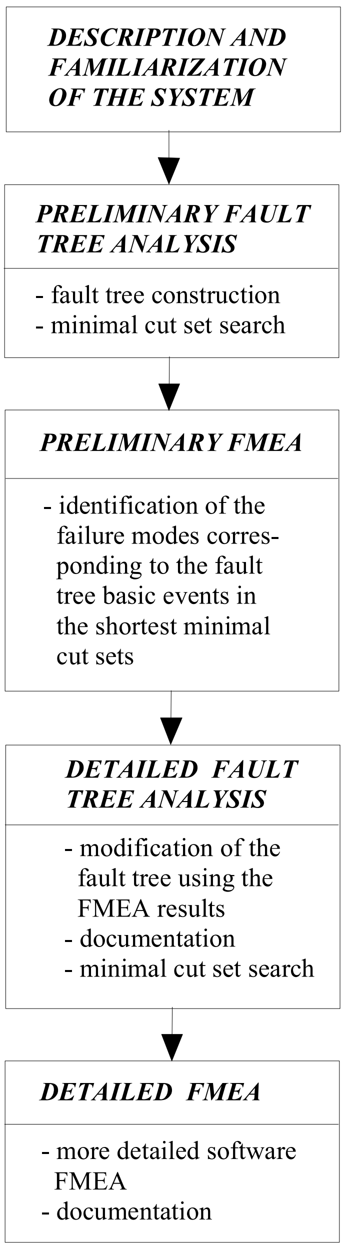
DESCRIPTION AND FAMILIARIZATION OF THE SYSTEM
PRELIMINARY FAULT TREE ANALYSIS
- fault tree construction
- minimal cut set search
PRELIMINARY FMEA
- identification of the failure modes corresponding to the fault tree basic events in the shortest minimal cut sets
DETAILED FAULT TREE ANALYSIS
- modification of the fault tree using the FMEA results
- documentation
- minimal cut set search
DETAILED FMEA
- more detailed software FMEA
- documentation

Figure 7. *The analysis steps of combined FTA/FMEA procedure.*

Bibliography

Addy, E.A., 1991, *Case Study on Isolation of Safety-Critical Software.* In Proceedings of the 6th Annual Conference on Computer Assurance, NIST/IEEE, Gaithersburg, MD, pp. 75–83.

Alur, R., Henzinger, T.A. & Ho, P.H., 1996, *Automatic Symbolic Verification of Embedded Systems.* In IEEE Transactions on Software Engineering 22, 3, pp. 181–201.

Ammar, H.H., Nikzadeh, T. & Dugan, J.B., 1997, *A methodology for risk assessment of functional specification of software systems using colored Petri nets*. Proceedings, Fourth International Software Metrics Symposium, pp. 108–117.

Atlee, J.M. & Gannon, J., 1993, *State-Based Model Checking of Event-Driven System Requirements.* IEEE Transactions on Software Engineering 19, 1, pp. 24–40.

Banerjee, N., 1995, *Utilization of FMEA concept in software lifecycle management*. Proceedings of Conference on Software Quality Management, pp. 219–230.

Barkai, J., 1999, *Automatic generation of a diagnostic expert system from failure mode and effects analysis (FMEA) information*. Society of Automotive Engineers, 1449, pp. 73–78.

Becker, J.C. & Flick, G., 1996. *A practical approach to failure mode, effects and criticality analysis (FMECA) for computing systems*. High–Assurance Systems Engineering Workshop, pp. 228–236.

Beizer, B., 1990, *Software Testing Techniques.* John Wiley & Sons

Bell, D., Cox, L., Jackson, S. & Schaefer, P., 1992, *Using Causal Reasoning for Automated Failure Modes & Effects Analysis (FMEA).* IEEE Proc Annual Reliability and Maintainability Symposium, pp. 343–353.

Ben–Daya, M. & Raouf, A., 1996, *A revised failure mode and effects analysis model.* International Journal of Quality & Reliability Management, 13(1), pp. 43–47.

Bestavros, A.A., Clark, J.J. & Ferrier, N.J., 1990, *Management of sensori-motor activity in mobile robots.* In Proceedings of the 1990 IEEE International Conference on Robotics and Automation, IEEE Computer Society Press, Cincinnati, OH, pp. 592–597.

Binder, R.V., 1997, *Can a manufacturing quality model work for software?* IEEE Software, 14(5), pp. 101–105.

Bluvband, Z. & Zilberberg, E., 1998, *Knowledge base approach to integrated FMEA.* ASQ's 52nd Annual Quality Congress Proceedings, pp. 535–545.

Bondavalli, A. & Simoncini, L., 1990, *Failure classification with respect to detection.* in First Year Report, Task B: Specification and Design for Dependability, Volume 2. ESPRIT BRA Project 3092: Predictably Dependable Computing Systems, May 1990.

Bouti, A., Kadi, D.A. & Lefrançois, P., 1998, *An integrative functional approach for automated manufacturing systems modeling*. Integrated Computer-Aided Engineering, 5(4), pp. 333–348.

Bowles, J.B., 1998, *The new SAE FMECA standard*. 1998 Proceedings Annual Reliability and Maintainability Symposium, pp. 48–53.

Burns D.J. & Pitblado R.M., 1993, *A modified HAZOP methodology for Safety Critical System Assessment.* 7th meeting of the UK Safety Critical Club, Bristol, February 1993.

Burns, D. & McDermid, J.A., 1994, *Real-time safety-critical systems: analysis and synthesis.* Software Engineering Journal, vol. 9, no. 6, pp. 267–281, Nov. 1994.

Cha, S.S., Leveson, N.G. & Shimeall, T.J., 1998, *Safety Verification in Murphy Using Fault Tree Analysis.* Proc. 10th Int. Conf. on S/W Eng., Apr 1988, Singapore, pp. 377–386.

Cha, S.S., Leveson, N.G. & Shimeall, T.J., 1991, *Safety Verification of Ada Programs Using Fault Tree Analysis.* In IEEE Software, 8, 4, pp. 48–59.

Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B. & Wong, M.Y., 1992, *Orthogonal Defect Classification—A Concept for In-Process Measurements.* IEEE Trans-actions on Software Engineering 18, 11, pp. 943–956.

Cichocki, T. & Gorski, J., 2001, *Formal support for fault modelling and analysis.* SAFECOMP 2001 Budapest, Hungary, 26–28 September 2001.

Coutinho, J.S., 1964, *Failure-Effect Analysis*. Trans. New York Academy of Sciences, pp.564–585.

Crow, J. & Di Vito, B.L., 1996, *Formalizing Space Shuttle Software Requirements.* In Proceedings of the ACM SIGSOFT Workshop on Formal Methods in Software Practice, San Diego, CA.

DeLemos, R., Saeed, A. & Anderson, T., 1995, *Analyzing Safety Requirements for Process Control Systems.* IEEE Software, Vol. 12, No. 3, May, 1995, pp. 42–52.

Dhillon, B. S., 1992, *Failure modes and effects analysis-bibliography*. Microelectronics Reliabi-ity, 32(5), pp. 719–731.

Dugan, J.B. & Boyd, M.A., 1993, *Fault Trees & Markov Models for Reliability Analysis of Fault-Tolerant Digital Systems.* Reliability Engineering & System Safety 39 (1993), pp. 291–307.

Ezhilchelvan, P.D. & Shrivastava, S.K., 1989, *A classification of faults in systems.* University of Newcastle upon Tyne.

Fencott, C. & Hebbron, B., 1995, *The Application of HAZOP Studies to Integrated Requirements Models for Control Systems.* ISA Transactions 34, pp. 297–308.

Fenelon P., McDermid J.A., Nicholson M. & Pumfrey D.J,, 1994, *Towards Integrated Safety Analysis and Design.* ACM Applied Computing Review, 2(1), pp. 21–32, 1994.

Fenelon, P. & Hebbron, B.D., 1994, *Applying HAZOP to software engineering models.* In Risk Management And Critical Protective Systems: Proceedings of SARSS 1994, Altrincham, England, Oct. 1994, pp. 1/1–1/16, The Safety And Reliability Society.

Fenelon, P. & McDermid, J.A., 1993, *An integrated tool set for software safety analysis.* The Journal of Systems and Software, 21, pp. 279–290.

Fenelon, P. & McDermid, J.A., 1992, *Integrated techniques for software safety analysis.* In Proceedings of the IEE Colloquium on Hazard Analysis. Nov. 1992, Institution of Electrical Engineers.

Fragola, J.R. & Spahn, J.F., 1973, *The Software Error Effects Analyis; A Qualitative Design Tool.* In Proceedings of the 1973 IEEE Symposium on Computer Software Reliability, IEEE, New York, pp. 90–93.

Gilchrist, W., 1993, *Modelling failure modes and effects analysis*. International Journal of Quality and Reliability Management, 10(5), pp. 16–23.

Goble, W.M. & Brombacher, A.C., 1999, *Using a failure modes, effects and diagnostic analysis (FMEDA) to measure diagnostic coverage in programmable electronic systems*. Reliability Engineering & System Safety. 66(2), pp. 145–148, 1999 Nov.

Goddard, P.L., 2000, *Software FMEA techni-ques*. 2000 Proceedings Annual Reliability and Maintainability Symposium, pp. 118–123.

Goddard, P.L., 1996, *A combined analysis approach to assessing requirements for safety critical real-time control systems*. 1996 Proceedings Annual Reliability and Maintain-ability Symposium, pp. 110–115.

Goddard, P.L., 1993, *Validating the safety of embedded real-time control systems using FMEA*. 1993 Proceedings Annual Reliability and Maintainability Symposium, pp. 227–230.

Hall, F.M., Paul, R.A. & Snow, W.E., 1983, *Hardware/Software FMECA*. Proc Annual Reliability and Maintainability Symposium, pp. 320–327.

Hansen, K.M., Ravn, A.P. & Stavridou, V., 1993, *From Safety Analysis to Software Requirements*. IEEE Transactions on Software Engineering, Vol. 21, July, 1993, pp. 279–290.

Hebbron, B.D. & Fencott, P.C, 1994, *The application of HAZOP studies to integrated requirements models for control systems*. In Proceedings of SAFECOMP '94, Oct. 1994.

Heimdahl, M.P.E. & Leveson, N.G., 1996, *Completeness and Consistency in Hierarchical State-Based Requirements*. IEEE Transactions on Software Engineering 22, 6, pp. 363–377.

Heitmeyer, C., Bull, B., Gasarch, C. & Labaw, B., 1995, *SCR: A Toolset for Specifying and Analyzing Requirements*. In Proceedings of the 10th Annual Conference on Computer Assurance, IEEE, Gaithersburg, MD, pp. 109–122.

Herrman, D.S., 1995, *A methodology for evaluating, comparing, and selecting software safety and reliability standards*. Proc. Tenth Annual Conference on Computer Assurance, pp. 223–232.

Hu, A.J., Dill, D.L., Drexler, A.J. & C. Han Yang, 1993, *Higher-Level Specification and Verification with BDDs*. In Proceedings of Computer Aided Verification: Fourth Internatio-nal Workshop, Lecture Notes in Computer Science, Eds. G. v. Bochmann and D.K. Probst, vol. 663, Springer-Verlag, Berlin.

Ippolito, L.M. & Wallace, D.R., 1995, *A Study on Hazard Analysis in High Integrity Software Standards and Guidelines*. Gaithersburg, MD: U.S. Dept. of Commerce, National Institute of Standards and Technology, NISTIR 5589.

Johnson, S.K., 1998, *Combining QFD and FMEA to optimize performance*. ASQ's 52nd Annual Quality Congress Proceedings, pp. 564–575.

Jordan, W.E., 1972, *Failure modes, effects, and criticality analysis*. Proceedings Annual Relia-ility and Maintainability Symposium, pp. 30–37.

Kennedy, M., 1998, *Failure modes & effects analysis (FMEA) of flip chip devices attached to printed wiring boards (PWB)*. 1998 IEEE/PMT International Electronics Manufacturing Technology Symposium, pp. 232–239.

Knight, J. & Nakano, G., 1997, *Software Test Techniques for System Fault-Tree Analysis*.

Kukkal, P., Bowles, J.B. & Bonnell, R.D., 1993, *Database design for failure modes and effects analysis*. 1993 Proceedings Annual Reliability and Maintainability Symposium, pp. 231–239.

Lamport, L. & Lynch, N., 1990, *Distributed Computing Models and Methods*. In Handbook of Theoretical Computer Science, vol. B, Formal Models and Semantics, J. van Leeuwen, Ed. Cambridge/Amsterdam: MIT Press/Elsevier, 1990, pp. 1157–1199.

Leveson, N.G., 1995, *Safeware, System Safety and Computers*. Addison-Wesley, Leveson, N.G., Cha, S.S. & Shimeall, T.J, 1991, *Safety Verification of Ada Programs using software fault trees.* IEEE Software, 8(7), pp. 48–59, July 1991.

Leveson, N.G. & Stolzy, J.L., 1987, *Safety Analysis Using Petri Nets.* IEEE Transactions on Software Engineering, vol. 13, no. 3, March 1987.

Leveson, N.G. & Harvey, P.R., 1983, *Software fault tree analysis.* Journal of Systems and Software, vol. 3, pp. 173–181.

Loftus, C., Pugh, D. & Pyle, I., 1996, *Software Failure Modes, Effects and Criticality Analysis-Position Paper*.

Luke, S.R., 1995, *Failure mode, effects and criticality analysis (FMECA) for software.* 5th Fleet Maintenance Symposium, Virginia Beach, VA (USA), 24–25 Oct 1995*,* pp. 731–735.

Lutz, R.R. & Shaw, H–Y., 1999a, *Applying Adaptive Safety Analysis Techniques*. Proceedings of the Tenth International Symposium on Software Reliability Engineering, Boca Raton, FL, Nov 1–4, 1999.

Lutz, R.R. & Woodhouse, R.M., 1999b, *Bi-directional Analysis for Certification of Safety-Critical Software*. Proceedings, ISACC'99, International Software Assurance Certification Conference, Chantilly, VA, Feb. 28–Mar 2, 1999.

Lutz, R.R. & Woodhouse, R.M., 1998a, *Failure Modes and Effects Analysis.* Encyclopedia of Electrical and Electronics Engineering, ed. J. Webster, John Wiley and Sons Publishers.

Lutz, R.R. & Shaw, H-Y., 1998b, *Applying Integrated Safety Analysis Techniques (Software FMEA and FTA).* JPL D–16168, Nov. 30, 1998.

Lutz, R.R., Helmer, G., Moseman, M., Statezni, D. & Tockey, S., 1998c, *Safety Analysis of Requirements for a Product Family.* Proc. Third IEEE International Conference on Requirements Engineering.

Lutz, R.R. & Woodhouse, R.M., 1997, *Require-ments Analysis Using Forward and Backward Search.* Annals of Software Engineering , 3, pp. 459–475.

Lutz, R.R & Woodhouse, R.M., 1996, *Experience Report: Contributions of SFMEA to Requirements Analysis.* Proceedings of ICRE '96, April 15–18, 1996, Colorado Springs, CO, pp. 44–51.

Lutz, R.R., 1996a, *Targeting safety-related errors during software requirements analysis.* Journal of Systems and Software 34, pp. 223–230.

Lutz, R.R., 1996b, *Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems.* The Journal of Systems and Software.

Lutz, R. & Ampo, Y., 1994, *Experience Report: Using Formal Methods for Requirements Analysis of Critical Spacecraft Software.* In Proceedings for the 19th Annual Software Engineering Workshop, NASA Goddard Space Flight Center, Greenbelt, MD, pp. 231–236.

Maier, T., 1997, *FMEA and FTA to support safe design of embedded software in safety–critical systems.* Safety and Reliability of Software Based Systems. Twelfth Annual CSR Workshop, pp. 351–367.

Maskuniitty, M. & Pulkkinen, U., 1994, *Fault tree and failure mode and effects analysis of a digital safety function.* AVV(94)TR2, Technical Research Centre of Finland, 35 p.+app.

Mazur, M.F., 1994, *Software FMECA.* Proc. 5th International Symposium on Software Reliability Engineering, Monterey, CA, Nov. 6–9.

McDermid, J.A., Nicholson, M., Pumfrey, D.J. & Fenelon, P., 1995, *Experience with the application of HAZOP to computer-based systems.* In Proceedings of the 10th Annual Conference on Computer Assurance, IEEE, Gaithersburg, MD, pp. 37–48.

McDermid, J.A. & Pumfrey, D.J., 1994, *A development of hazard analysis to aid software design.* In COMPASS '94: Proceedings of the Ninth Annual Conference on Computer Assurance. IEEE / NIST, Gaithersburg, MD, June 1994, pp. 17–25.

McDermott, R.E., Mikulak, R.J. & Beauregard, M.R., 1996, *The basics of FMEA.* New York: Quality Resources.

Modugno, F., Leveson, N.G., Reese, J.D., Partridge, K. & Sandys, S.D., 1997, *Integrated Safety Analysis of Requirements Specifications.* Proc Third IEEE International Symposium on Requirements Engineering, pp. 148–159.

Montgomery, T.A. & Marko, K.A., 1997, *Quantitative FMEA automation*. 1997 Proceedings Annual Reliability and Maintainability Symposium, pp. 226–228.

Montgomery, T.A., Pugh, D.R., Leedham, S.T. & Twitchett, S.R., 1996, *FMEA Automation for the Complete Design Process.* IEEE Proc Annual Reliability and Maintainability Symposium, Annapolis, MD, Jan 6–10, pp. 30–36.

Moriguti, S., 1997, *Software excellence: A total quality management guide*. Portland, OR: Productivity Press.

Mullery, G.P., 1987, *CORE—a method for COntrolled Requirements Expression.* In System and Software Requirements Engineering, R.H. Thayer and M. Dorfman, Eds., pp. 304–313. IEEE Press.

Nakajo, T. & Kume, K., 1991, *A Case History Analysis of Software Error Cause-Effect Relationship.* IEEE Transactions on Software Engineering, 17, 8, 830–838.

Onodera, K., 1997, *Effective techniques of FMEA at each life-cycle stage*. 1997 Proceedings Annual Reliability and Maintaina-bility Symposium, pp. 50–56.

Ostrand, T.J. & Weyuker, E.J., 1984, *Collecting and Categorizing Software Error Data in an Industrial Environment.* The Journal of Systems and Software, 4, pp. 289–300.

Parkinson, H.J., Thompson, G. & Iwnicki, S. 1998, *The development of an FMEA methodology for rolling stock remanufacture and refurbishment*. ImechE Seminar Publication 1998-20, pp. 55–66.

Parzinger, M.J. & Nath, R., 1997, *The effects of TQM implementation factors on software quality*. Proceedings of the Annual Meeting Decision Sciences Institute, 2, pp. 834–836.

Passey, R.D.C., 1999, *Foresight begins with FMEA*. Medical Device Technology, 10(2), pp. 88–92.

Perkins, D., 1996, *FMEA for a REaL (resource limited) design community*. American Quality Congress Proceedings, American Society for Quality Control, pp. 435–442.

Pfleeger, S.L., 1998, *Software engineering: Theory and practice.* Upper Saddle River, NJ: Prentice Hall.

Price, C.J. & Taylor, N.S., 1998, *FMEA for multiple failures*. 1998 Proceedings Annual Reliability and Maintainability Symposium, pp. 43–47.

Price, C.J., 1996, *Effortless incremental design FMEA*. 1996 Proceedings Annual Reliability and Maintainability Symposium, pp. 43–47.

Price, C.J., Pugh, D.R., Wilson, M.S. & Snooke, N., 1995. *The Flame system: Automating electrical failure mode & effects analysis (FMEA).* 1995 Proceedings Annual Reliability and Maintainability Symposium, pp. 90–95.

Pries, K.H., 1998, *Failure mode & effects analysis in software development.* (SAE Technical Paper Series No. 982816). Warren-dale, PA: Society of Automotive Engineers.

Pugh, D.R. & Snooke, N., 1996, *Dynamic Analysis of Qualitative Circuits for Failure Mode and Effects Analysis.* Proc Annual Reliability and Maintainability Symposium, pp. 37–42.

Raheja, J., 1991, *Assurance Technologies: Principles and Practices.* McGraw-Hill.

Raheja, D., 1990, *Software system failure mode and effects analysis (SSFMEA)–a tool for reliability growth*. Proceedings of the Int'l Symp. On Reliability and Maintainability (ISRM'90), Tokyo, Japan (1990), pp. 271–277.

Ratan, V., Partridge, K., Reese, J. & Leveson, N.G., 1996, *Safety analysis tools for requirements specifications.* Proc Eleventh Annual Conference on Computer Assurance, pp. 149–160.

Reese, J.D., 1996, *Software Deviation Analysis*, Ph.D. Thesis, University of California, Irvine, 1996.

Reifer, D.J., 1979, *Software Failure Modes and Effects Analysis.* IEEE Transactions on Reliability, R-28, 3, pp. 247–249.

Ristord, L. & Esmenjaud, C., 2001, *FMEA Per-ored on the SPINLINE3 Operational System Software as part of the TIHANGE 1 NIS Refurbishment Safety Case*. CNRA/CNSI Workshop 2001–Licensing and Operating Experience of Computer Based I&C Systems. Ceské Budejovice–September 25–27, 2001.

Roberts, N.H., Vesely, W.E., Haasl, D.F. & Goldberg, F.F., 1981, *Fault Tree Handbook.* Systems and Reliability Research Office of U.S. Nuclear Regulatory Commission, Jan. 1981.

Roland, H.E. & Moriarty, B., 1990, *System Safety Engineering and Management*. New York, New York: John Wiley and Sons.

Rushby, J., 1993. *Formal Methods and Digital Systems Validation for Airborne Systems*, SRI-CSL-93-07.

Russomanno, D.J., Bonnell, R.D. & Bowles, J.B., 1994, *Viewing computer-aided failure modes and effects analysis from an artificial intelligence perspective*. Integrated Computer-Aided Engineering, 1(3), pp. 209–228.

Saeed, A., de Lemos, R. & Anderson, T., 1995, *On the safety analysis of requirements specifications for safety-critical software.* ISA Transactions, 34(3), pp. 283–295.

Selby, R.W. & Basili, V.R., 1991, *Analyzing Error-Prone System Structure.* IEEE Trans-actions on Software Engineering, 17, 2, 141–152.

Signor, M.C., 2000, *The Failure Analysis Matrix: A Usable Model for Ranking Solutions to Failures in Information Systems.* A Formal Dissertation Proposal, School of Computer and Information Sciences, Nova Southeastern University.

Sommerville, I., 1996, *Software Engineering.* Fifth Edition, Addison-Wesley, Reading, MA.

Stamatis, D. H., 1995, *Failure mode and effect analysis: FMEA from theory to execution.* Milwaukee, WI: ASQC Quality Press.

Stephenson, J., 1991. *System Safety 2000: A practical guide for planning, managing and conducting system safety programs*. New York, New York: Van Nostrand Reinhold.

Storey, N., 1996, *Safety-Critical Computer Systems*. Addison-Wesley.

Stålhane, T. & Wedde, K.J., 1998, *Modification of safety critical systems: An assessment of three approached.* Microprocessors and Microsystems, 21(10), pp. 611–619.

Talbert, N., 1998, *The Cost of COTS*: An Interview with John McDermid, Computer, 31, 6, June, pp. 46–52.

Tanenbaum, A.S., 1992, *Modern Operating Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Taylor, J.R., 1982, *Fault Tree and cause Consequence Analysis for Control Software Validation*. Technical report, Risø National Laboratory, Denmark, January 1982.

Tsung, P. W., 1995, *The three "F"s in quality and reliability: Fault tree analysis, FMECA, FRACAS*. International Symposium on Quality in Non–Ferrous Pyrometallurgy, pp. 115–124.

Vesley, W.E., Goldberg, F.F., Roberts, N.H. & Haasl, D.L., 1981, *Fault Tree Handbook*. NUREG-0942, U.S. Nuclear Regulatory Commission.

Voas, J., 1998, *Recipe for Certifying High Assurance Software*, RST Corp.

Wallace, D.R., Ippolito, L.M. & Kuhn, D.R., 1992, *High Integrity Software Standards and Guidelines*, Gaithersburg, MD: U.S. Department of Commerce, National Institute of Standards and Technology, NIST Special Publication 500–204, September.

Wilson, S.P., Kelly, T.P. & McDermid, J.A., 1995, *Safety Case Development: Current Practice, Future Prospects*. In R. Shaw, editor, Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, Bruges, Belgium, September 1995. ENCRESS, Springer.

Wunram, J., 1990, *A Strategy for Identification and Development of Safety Critical Software Embedded in Complex Space Systems.* IAA 90–557, pp. 35–51.

Yang, K. & Kapur, K.C., 1997, *Customer driven reliability: Integration of QFD and robust design*. 1997 Proceedings Annual Reliability and Maintainability Symposium, pp. 339–345.

Standards & guidelines

Chrysler Corporation, Ford Motor Company & General Motors Corporation, 1995, *Potential failure mode and effects analysis (FMEA) reference manual* (2nd ed.). AIAG, (810) 358–3003.

DOD, 1980, *Military Standard, Procedures for Performing a Failure Mode, Effects and Criticality Analysis.* MIL-STD-1629A, U.S. Department of Defense, Washington, D. C. (Cancelled 4.8.1998).

DOD, 1984, *Military Standard, System safety program requirements.* MIL-STD-882B: U.S. Department of Defense, Washington, D C.

Electronic Industries Association, 1983, *System Safety Analysis Techniques.* Safety Engineering Bulletin No. 3-A, Engineering Department, Washington, D. C.

Electronic Industries Association, 1990, *System Safety Engineering in Software Development.* Safety Engineering Bulletin No. 6-A, Washington, D. C.

IEC, 2001, *Analysis techniques for system reliability—Procedure for failure mode and effects analysus (FMEA).* IEC 60812, Ed. 2. 56/735/CD, 27.4.2001.

IEEE, 1990, *Standard Glossary of Software Engineering Terminology.* IEEE Std 610.12-1990, IEEE, New York.

JPL, 1990. *Project Reliability Group Reliability Analyses Handbook.* D-5703, Jet Propulsion Laboratory/California Institute of Technology, Pasadena, CA.

JPL, 1998, *Earth Observing System Microwave Limb Sounder, System–Level Failure Modes, Effects, and Criticality Analysis.* JPL D-16088, Version 1.0, September 1, 1998.

JPL, 1995, *Software Failure Modes & Effects Analysis.* Software Product Assurance Handbook. Jet Propulsion Laboratory D-8642.

MOD, 1993, *Draft Defence Standard 00-56: Safety Management Requirements for Defence Systems Containing Programmable Electronics.* UK Ministry of Defence, Feb. 1993.

MOD, 1991, *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment.* Interim Defence Standard 00-56, Issue 1, Ministry of Defence, Directorate of Standardization, Kentigern House, 65 Brown Street, Glasgow, G2 8EX, UK, April 1991.

NASA, *FEAT (Failure Environment Analysis Tool).* NASA Software Technology Transfer Center, Cosmic #MSC–21873 and #MSC-22446.

NASA, *FIRM (Failure Identification and Risk Management Tool).* NASA Software Technology Transfer Center, Cosmic #MSC-21860.

NRC, 1981, *Fault Tree Handbook.* W Veseley. Nuclear Regulatory Commission Washington D.C. 1981. NUREG-0942.

RTCA, Inc., 1992, *Software Considerations in Airborne Systems and Equipment Certification.* RTCA/DO-178B.

SAE, 1996, *Aerospace Recommended Practice: Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* ARP 4754.

SAE, 1996, *Aerospace Recommended Practice: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* ARP 4761.

SAE, 2000, *Potential Failure Mode and Effects Analysis in Design (Design FMEA) and Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA) Reference Manual*. SAE J-1739.

SAG, 1996, *Failure Modes and Effects Analysis and Failure Modes, Effects, and Criticality Analysis Procedures.*

System Safety Society, 1993, *System Safety Analysis Handbook*.

APPENDIX 1 EXAMPLE OF THE FMEA-WORKSHEET (IEC 60812)

Indenture level: Sheet no: Operating mode:			Design by: Item: Revision:					Prepared by: Approved by:				FMEA
Item ref.	Item description-function	Failure entry code	Failure mode	Possible failure causes	Symptom detected by	Local effect	Effect on unit output	Compensating provision against failure	Severity class	Failure rate F/Mhr	Data source	Recommendations and actions taken

APPENDIX 2 An example of a SWFMEA form (TÜV Nord)

Subsystem: ____________

Ref-No	Component	Fault	Cause	Failure effect	Safety