SQUALE

Dependability Assessment Criteria

(4th draft)

January 1999

ACTS95/AC097

This document is issued by the SQUALE Consortium. Members of the SQUALE Consortium are CR2A-DI (co-ordinator), Admiral, IABG, LAAS-CNRS and Matra Transport International.

SQUALE is project AC097 of the
Advanced Communications Technologies and Services (ACTS) programme
of the European Commission.

Main contributors to this document are:

For CR2A-DI :	Pierre Corneillie, Sylvain Moreau, Claudine Valentin
For Admiral :	John Goodson, Alan Hawes, Tim Manning
For IABG :	Helmut Kurth, Goetz Liebisch, Angelika Steinacker
For LAAS-CNRS :	Yves Deswarte, Mohamed Kaâniche
For Matra Transport International :	Paul Benoit

This work was partly funded by the European Commission.

Printed at LAAS-CNRS

LAAS Report n°98456

January 1999

Table of Contents

1 INTRODUCTION 7

1.1 GENERAL REQUIREMENTS FOR THE DEPENDABILITY ASSESSMENT CRITERIA 7
1.2 OBJECT STATUS 8

2 SCOPE AND OBJECTIVES 9

2.1 PURPOSE OF THE DOCUMENT 9
2.2 STRUCTURE OF THE DOCUMENT 9
2.3 INTENDED AUDIENCE 10
2.4 BACKGROUND 11
2.5 APPLICABILITY 11

3 DEPENDABILITY ASSESSMENT AND CONSTRUCTION FRAMEWORK 13

3.1 GENERAL PRINCIPLES 13
3.2 DEPENDABILITY ASSESSMENT FRAMEWORK 13
3.2.1 Introduction 13
3.2.2 Roles 13
3.2.3 The Dependability Assessment Process 15
3.3 DEPENDABLE SYSTEM DEVELOPMENT 19
3.3.1 Overview 19
3.3.2 Dependability Construction 21
3.3.3 Confidence Providing Processes 25
3.4 DEPENDABILITY PROFILES 28
3.4.1 Introduction 28
3.4.2 Selection of Confidence Levels 28
3.4.3 Confidence levels in existing Security and Safety Standards 29
3.4.4 Dependability Attributes and Confidence Levels 29
3.4.5 Confidence Levels and the Criteria 30
3.5 THE DEPENDABILITY TARGET 31
3.5.1 Content of a Dependability Target 31
3.5.2 Use of the Dependability Target 33
3.6 COMPLEX SYSTEMS 34
3.6.1 Dependability Construction Framework 35
3.6.2 Subcontractors 35
3.6.3 COTS Products 37

4 CRITERIA FRAMEWORK 39

4.1 STRUCTURE OF THE PRESENTATION OF PROCESSES AND ACTIVITIES 39
4.2 DEPENDABILITY REQUIREMENTS VALIDATION 40
4.2.1 Motivation and Scope 40
4.2.2 Dependability requirements validation confidence providing activities 41
4.3 CORRECTNESS VERIFICATION 66
4.3.1 Motivation and scope 66
4.3.2 Correctness Verification Objectives and Lifecycle stages 68
4.3.3 Correctness Verification Planning 69
4.3.4 Assessors Activities 70
4.3.5 Correctness Verification Confidence Providing Activities 70
4.4 DEPENDABILITY VALIDATION 94
4.4.1 Motivation and scope 94
4.4.2 Dependability Validation Confidence Providing Activities 95
4.5 PROCESS QUALITY 109
4.5.1 Objective 109
4.5.2 Quality Assurance 111

4.6 MAPPING BETWEEN ATTRIBUTES, LEVELS AND ACTIONS 114
4.6.1 The Mapping Table 114
4.6.2 Selecting the Appropriate Dependability Profile 115

5 MAPPING TO EXISTING STANDARDS 119

5.1 ITSEC 119
5.1.1 Introduction 119
5.1.2 Overview and Scope of ITSEC 119
5.1.3 Similarities 120
5.1.4 General differences 120
5.1.5 Confidence Levels 121
5.1.6 Dependability Target 122
5.1.7 Lifecycle 123
5.2 IEC 1508 123
5.2.1 Introduction 123
5.2.2 Overview 123
5.2.3 General Similarities 124
5.2.4 General Differences 124
5.2.5 Dependability Target 124
5.2.6 Confidence Levels 125
5.2.7 Lifecycle 128
5.2.8 Assessment activities 128
5.3 EN 50126 129
5.3.1 Introduction 129
5.3.2 Overview 129
5.3.3 General similarities 130
5.3.4 General differences 131
5.3.5 System lifecycle 131
5.3.6 Confidence levels used in EN 50126 132
5.3.7 Dependability target 133
5.3.8 Confidence providing activities 133
5.3.9 Criteria applicability 134
5.4 EN 50128 134
5.4.1 Introduction 134
5.4.2 Overview 134
5.4.3 General similarities 135
5.4.4 General differences 135
5.4.5 Software lifecycle 135
5.4.6 Confidence levels used in EN 50128 137
5.4.7 Dependability target 138
5.4.8 Confidence providing activities 138
5.4.9 Criteria applicability 139
5.5 IEC 880 139
5.5.1 Introduction 139
5.5.2 Overview 139
5.5.3 General Similarities 139
5.5.4 General differences 139
5.5.5 Confidence levels used in IEC 880 140
5.5.6 Dependability Target 142
5.5.7 Software lifecycle 143
5.5.8 Miscellaneous 146
5.6 ETR 367 148
5.7 AVIONICS STANDARDS: RTCA DO178B 148
5.7.1 Introduction 148
5.7.2 Overview of DO-178B 149
5.7.3 General Similarities 150

5.7.4 General Differences ... 150
5.7.5 Confidence levels ... 151
5.7.6 Dependability Target ... 151
5.7.7 Confidence providing processes and activities ... 152
5.7.8 Levels of rigour, detail and independence ... 152
5.7.9 Lifecycle ... 153
5.7.10 Mapping between DO-178B requirements and the Criteria ... 154

ANNEXE A - DEPENDABILITY CONCEPTS AND TERMINOLOGY ... 159

A.1 BASIC DEFINITIONS ... 159
A.2 DEPENDABILITY CONCEPT RATIONALE ... 160
A.3 THE ATTRIBUTES OF DEPENDABILITY ... 161
A.4 DEPENDABILITY IMPAIRMENTS ... 162
A.4.1 Failures and Failure Modes ... 163
A.4.2 Fault Classes and Fault Assumptions ... 164
A.5 THE MEANS FOR DEPENDABILITY ... 166
A.5.1 Fault prevention ... 166
A.5.2 Fault tolerance ... 166
A.5.3 Fault removal ... 168
A.5.4 Fault forecasting ... 169
A.6 DEPENDABILITY PRINCIPLES ... 170
A.6.1 Identification and Authentication ... 170
A.6.2 Access Control ... 171
A.6.3 Accountability and Audit ... 171
A.6.4 Object Re-use ... 172
A.6.5 Levels of Redundancy ... 172
A.6.6 Design Diversity ... 173
A.6.7 Fail-Halt System ... 173
A.6.8 Graceful Degradation ... 174
A.6.9 Defence in Depth ... 174
A.6.10 Defensive Programming ... 174
A.6.11 Partitioning and Separation ... 175

ANNEXE B - GLOSSARY ... 177

ANNEXE C - BIBLIOGRAPHY ... 183

Tables

TABLE 1: CONFIDENCE LEVEL SCALE 29
TABLE 2: DEFINITION OF CONFIDENCE LEVELS FOR HAZARD ANALYSIS 51
TABLE 3: DEFINITION OF CONFIDENCE LEVELS FOR COMMON CAUSE ANALYSIS 59
TABLE 4: CONFIDENCE LEVELS FOR PROBABILISTIC QUANTITATIVE EVALUATION 64
TABLE 5: RELATION BETWEEN DEPENDABILITY REQUIREMENTS VALIDATION ACTIVITIES AND CONFIDENCE LEVELS .66
TABLE 6: DEFINITION OF CONFIDENCE LEVELS FOR STATIC ANALYSIS 74
TABLE 7: DEFINITION OF CONFIDENCE LEVELS FOR TEST METHODS 78
TABLE 8: DEFINITION OF CONFIDENCE LEVELS FOR FORMAL METHODS 84
TABLE 9: DEFINITION OF CONFIDENCE LEVELS FOR BEHAVIOURAL ANALYSIS 88
TABLE 10: DEFINITION OF CONFIDENCE LEVELS FOR TRACEABILITY ANALYSIS 92
TABLE 11: MAPPING BETWEEN CORRECTNESS VERIFICATION ACTIVITIES AND CONFIDENCE LEVELS 94
TABLE 12: DEFINITION OF CONFIDENCE LEVELS FOR PENETRATION ANALYSIS 99
TABLE 13: DEFINITION OF CONFIDENCE LEVELS FOR COVERT CHANNEL ANALYSIS 104
TABLE 14: CONFIDENCE LEVELS FOR EXPERIMENTAL EVALUATION 107
TABLE 15: RELATION BETWEEN DEPENDABILITY VALIDATION ACTIVITIES AND CONFIDENCE LEVELS 109
TABLE 16: SELECTION OF CPAS BY DEPENDABILITY CONFIDENCE PROFILE 117
TABLE 17: MAPPING BETWEEN IEC 1508 ACTIVITIES AND CRITERIA CPPS AND CPAS 128
TABLE 18: COMPARISON OF IEC DOCUMENTS AND SQUALE DOCUMENTS 145
TABLE 19: MAPPING OF IEC 880 DETAILED RECOMMENDATIONS AND SQUALE CRITERIA 148
TABLE 20: MAPPING OF DO-178B REQUIREMENTS TO THE CRITERIA 157

Figures

FIGURE 1: ROLES OF THE PARTIES PARTICIPATING IN THE ASSESSMENT 13
FIGURE 2: ASSESSMENT ACTIVITIES ORGANISATION 17
FIGURE 3: DEPENDABILITY CONSTRUCTION FRAMEWORK 22
FIGURE 4: MAJOR DEVELOPMENT STEPS 26
FIGURE 5: DEPENDABILITY TARGET STRUCTURE 32
FIGURE 6: DEPENDABILITY TARGET REVIEW 33
FIGURE 7: EXAMPLE OF SYSTEM DECOMPOSITION 34
FIGURE 8: SUBCONTRACTED COMPONENT 36
FIGURE 9: EXAMPLE OF DIFFERENT DEPENDABILITY PROFILES 115
FIGURE 10: SOFTWARE REQUIREMENTS BRANCH OF THE SYSTEM DEVELOPMENT LIFECYCLE 144
FIGURE 11: CERTIFICATION GUIDANCE DOCUMENTS COVERING SYSTEM, SAFETY, SOFTWARE AND HARDWARE PROCESSES 149
FIGURE 12: DEPENDABILITY FRAMEWORK 160
FIGURE 13: THE FAILURE MODES 163
FIGURE 14: THE ELEMENTARY FAULT CLASSES 164
FIGURE 15: THE COMBINED FAULT CLASSES 165

1 Introduction

The increasing dependency of our society on computer systems becomes obvious when critical systems fail to provide some important functions thereby endangering human lives or leading to severe economic losses. It is therefore necessary to develop methods which help to ensure that failures of computer systems which have disastrous consequences are avoided to the highest extent possible. This is true regardless if the failure was caused accidentally or deliberately.

The development and operation of dependable computing systems calls for the combined use of a set of processes, methods and techniques during the different life cycle phases of the system in order to fulfil the objectives of the project. Besides the traditional development processes which cover the classical development steps: Requirements Elicitation and Specification, Design, Realisation and Integration, additional processes are also defined in order to:

- plan and co-ordinate the activities carried out within each life cycle process, and
- ensure the correctness, the control and the confidence of the life cycle processes and their outputs.

Assessment of all these processes is necessary to reach a sufficient confidence on the system ability to fulfil its objectives. Several international industry and government accepted practices and standards relating to dependability have been proposed over the past years and some others are under development. But most of them are specific to an industrial sector (e.g. railway transportation) and specific to one aspect of dependability, namely safety or security. Yet, in most cases, Dependability Objectives encompass several other attributes such as availability or maintainability. The assessment criteria developed in the SQUALE project aim to cover all dependability attributes and all industrial sectors.

These criteria do not define a new life cycle or system development model. The purpose of the criteria is the definition of investigation, proofing and assessment activities, within any life cycle and system environment, that are necessary to gain confidence that the system meets its Dependability Objectives.

It is not the intention of these criteria to define a totally new approach incompatible with existing criteria from the safety or security sector. Within the SQUALE project an analysis of existing safety and security standards was performed to identify the existing approaches and their relevance for this harmonisation work. An analysis of the correspondences between the most relevant standards and the SQUALE criteria is given in this document.

1.1 General Requirements for the Dependability Assessment Criteria

The criteria defined in this document require a set of activities that need to be performed for each dependable system. These activities are:

- A Preliminary Hazard Analysis has to be performed for all parts of the system. All identified hazards shall be described together with their potential impact on the system or the environment.

- The hazards identified in the analysis shall be rated for their criticality. The assumptions made on the system as well as on the environment during the rating process shall be clearly described.
- Dependability Objectives shall be derived describing the overall dependability goals that the system has to achieve. These objectives shall completely address all hazards rated as unacceptable.
- Those Dependability Objectives as well as the results from the hazard analysis and rating process shall be used to define the specific dependability requirements for all parts of the system. These requirements shall cover all dependability attributes and are summarised in a dependability profile. This dependability profile has impacts on the development, operation, maintenance and decommissioning of the system as well as on the assessment activities.
- *Confidence providing activities* have to be performed to achieve confidence that the system meets its Dependability Objectives. The individual confidence providing activities depend on the level of confidence needed for a specific part of the system as well as on the Dependability Objectives that this part of the system contributes to.
- Assessment activities have to be defined that allow to continuously monitor whether the objectives are achieved. Continuous monitoring is necessary, since:
    - the assumptions made in the hazard analysis and hazard rating process may be incomplete or wrong
    - changes in the environment may result in additional hazards or may affect the rating of an already identified hazard
    - changes to the system itself may result in additional hazards or may affect the rating of an already identified hazard. Those changes may also affect the dependability requirements for specific parts of the system and can therefore lead to the need to adjust the dependability profile required for those parts of the system.

The criteria try to define a general framework as well as the individual assessment activities needed to address all the activities and aspects mentioned in the list above.

1.2 Object Status

This document is the fourth version of a working draft that presents the SQUALE dependability assessment Criteria and Framework. Trial evaluations have been performed according to the second version, and lessons have been learned from these experiments to improve the flexibility, the clarity and the structure of the criteria, resulting in the third version. The latter has been discussed during the open final SQUALE Workshop organised in Toulouse on November 21st-22nd and distributed to several standardisation bodies and working groups as well as to key players in the dependable systems sector. This fourth draft incorporates some improvements resulting from those discussions.

2 Scope and Objectives

This chapter describes:

- the purpose of these criteria,
- the structure of the document,
- the intended users of the document,
- the main existing standards from Safety and Security and how they influenced the work,
- why an assessment framework for dependability is needed.

2.1 Purpose of the Document

This document covers the assessment and certification aspects of dependable systems containing Information Technology (IT) components. Such assessment activities have been defined in the past in several standards and guidelines but they either focus on security or safety aspects solely. The intention of this document is to provide a framework for the assessment of dependable IT systems that cover both safety and security aspects besides other dependability attributes. In addition individual activities for this assessment process are identified and described. Some of those activities depend on the overall scope and functionality of the system to be assessed and are marked appropriately.

2.2 Structure of the Document

This document contains the following chapters:

Chapter 3 defines the general framework for the activities that are necessary to develop dependable systems by identifying the roles which contribute to dependability, and by relating those activities to the steps of a general development process and to a generic life cycle model. Furthermore this chapter defines the structure and content of a Dependability Target document, that serves as the basic plan for the assessment of a critical system. Such a document has to be prepared for each system that is to be assessed using the criteria defined in this document. Although this seems to be a new document not defined in any other standard, its content is composed of results from activities that are well defined and accepted for safety or security related systems.

Chapter 4 is the main part of this document and defines the individual assessment criteria and confidence providing activities. These activities are structured as:

- Dependability Requirement Validation activities
- Correctness Verification activities
- Dependability Validation activities and
- Process Quality activities

Dependability Requirement Validation activities are performed to check if the dependability requirements and the Dependability Policy have been derived correctly and cover all aspects of the overall Dependability Objectives.

Correctness Verification covers all aspects related to the assessment of correct refinement between different levels of abstraction in the system design as well as all aspects related to the assessment that all directly verifiable requirements have been implemented.

Dependability Validation covers all additional aspects to Correctness Verification that assess, if a system, as implemented, is compliant with the overall objectives. This looks especially for side-effects of the system implementation, which do not violate the system specification but could lead to failure situations.

Process Quality activities cover all aspects related to the construction / operation / maintenance / decommissioning process(es). They insure that well-defined and appropriate processes are in place and applied over the whole system life cycle.

Chapter 5 shows how the framework and the criteria defined in this document map to some well know standards within the safety and security sector. It is intended to show the mapping to the following standards:

- ITSEC
- IEC 1508
- IEC 880
- EN 50128
- ETR 367
- RTCA DO 178B

Mappings to additional standards, already existing or future, should be possible and may be derived in a similar way as the mappings defined in this chapter.

Annexe A defines the dependability concepts used in the overall assessment framework as well as in the specific criteria. **Annexe B** gives a glossary, and **Annexe C** presents a bibliography.

2.3 Intended Audience

This document is primarily aimed at persons responsible for the assessment and certification of dependable IT-systems or such systems with IT-components. It provides those persons with a framework for their tasks and describes assessment activities. The document is also aimed at developers of dependable systems and those persons responsible for operation / maintenance of dependable systems. They can use this document as a guideline to pass a dependability assessment successfully.

This document shall also serve as a basis to develop more detailed sector specific standards. For the development of those standards, the framework described in this document can be taken as a starting point. A sector specific standard will normally only need a subset of the dependability attributes defined in the general framework and therefore also need only a subset of the assessment activities (usually with some prescribed methods).

2.4 Background

The framework and criteria in this document were derived after the analysis of a wide range of standards and other material from the safety as well as the security sector. It was then decided to take one standard from each sector as a basis for the development:

- the draft IEC 1508 standard, which is intended to set out a generic approach for all Safety Lifecycle activities for electrical/electronic/programmable electronic systems;
- the ITSEC standard, which provides the basis for the Security Evaluation of IT-systems and products throughout Europe and several other countries.

The authors had the opinion that both standards in general cover their specific sector with a sufficient level of completeness but they agreed that both standards have specific weaknesses that should be addressed by these criteria.

Inputs from several additional standards have been taken for the definition of the individual activities within the criteria as well as to address those aspects which, in the opinions of the authors, were not covered or not adequately covered by the two reference standards.

Since the terminology in the safety and security sector are different due to the different development of both sectors, it was necessary to define a harmonised terminology first which is then used throughout this document. This terminology is defined in Annexe A of this document.

2.5 Applicability

This document is intended to be applied for the assessment and certification of systems that are critical for safety or security, or that have high requirements on availability, reliability or maintainability. All phases of the lifecycle of a dependable system are taken into account. The assessment criteria are intended for the assessment of hard- and software components within systems, but the interfaces to other parts of the system as well as to organisational, personnel and infrastructure measures are also taken into account.

3 Dependability Assessment and Construction Framework

3.1 General Principles

The Dependability Assessment Framework and the Dependability Construction Framework show which investigations are necessary to produce a dependable system and which proofs are required so that the system can be assessed.

3.2 Dependability Assessment Framework

3.2.1 Introduction

This chapter presents a general model applied during the assessment of a dependable system. The general model defines the roles and the responsibilities of the parties involved in the dependability assessment process, and describes the main assessment tasks.

The general model does not impose a fixed scheme, however it includes requirements that any scheme should comply with.

3.2.2 Roles

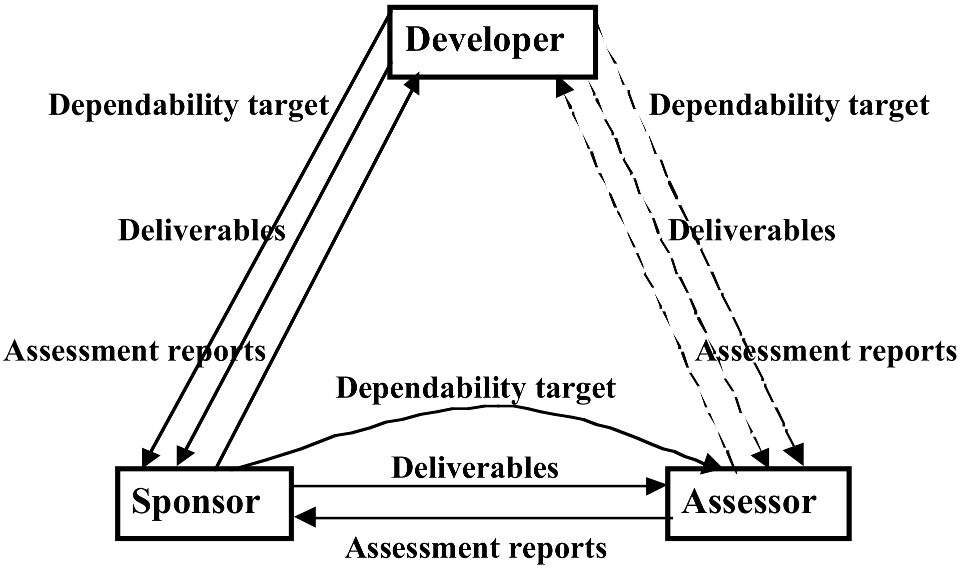

Figure 1: Roles of the parties participating in the assessment

The following parties are directly involved in the dependability assessment:

- the sponsor,
- the developer,
- the assessor.

Sponsor

The sponsor of a dependability assessment is the organisation requesting and funding the assessment. It will usually be the vendor of a product, or the user or the supplier of a system, or a certification authority.

The sponsor is responsible for:

- establishing the necessary agreements for the assessment,
- preparing and supplying the Dependability Target,
- assuring that the assessor is provided with the deliverables and support requested for the assessment.

Developer

The developer represents the organisation that produces the system to be assessed. It includes all the company departments involved in the assessment process. The main department playing a role in the assessment process is the development team but other departments, in particular the quality team, work in co-ordination with the development team. The development team develops his system in conformance with the quality team recommendations and the quality team gives some guidance to the development team about the development process. Verification and validation activities are also included in the developer's activities.

The responsibilities of the developer are:

- developing the system,
- supporting the Dependability Target preparation,
- supporting the assessment,
- providing deliverables requested for the assessment,
- developing, providing and maintaining the assessment deliverables

Assessor

The assessor is the one responsible of the assessment of the system. The Criteria require that the assessor shall be independent from the sponsor and the developer. Independence means the absence of any kind of pressure (be it financial, organisational, etc.) that might prevent him from doing an objective assessment.

The responsibilities of the assessor include:

- receiving the assessment evidence,
- requesting and receiving assessment support,
- performing if necessary an assessment feasibility study,
- supplying assessment verdicts with justification.

The assessment reports and the deliverables will be described later in Section 3.2.3.2.

3.2.3 The Dependability Assessment Process

3.2.3.1 Phases of the dependability assessment process

The assessment process can be decomposed into three main phases:

- preparation phase which initiates the assessment,
- conduct phase in which the assessment is performed,
- conclusion phase in which the assessment results are delivered.

Preparation

The preparation phase includes the initial contact between the sponsor and the assessor. The sponsor supplies the assessor with the Dependability Target. The assessor performs a feasibility study to assess likelihood of a successful assessment, requesting relevant information (deliverables) from the other parties involved in the assessment. The sponsor or the developer contributes to this study by supplying the assessor with deliverables. This set of deliverables defines the inputs and outputs of the confidence providing processes which are described later in Section 3.3.3. The feasibility study will confirm that the sponsor and the developer are well prepared for the conduct of the assessment, and it will involve, as a minimum, a review of the Dependability Target.

When a successful assessment seems to be feasible, the assessor shall prepare an Assessment Workplan (AW) agreed by all the parties and containing the list of required assessment deliverables, their delivery schedule and the assessor's actions.

Depending on the scope of the system to be assessed the assessment must be considered as every other project, so other assessment tasks shall be defined during the preparation phase to address if adequate:

- liaison with development team,
- liaison with accreditation and or certification authority,
- tailoring of criteria lifecycle and deliverables to system lifecycle,
- quality assurance,
- assessment project management,
- assessment planning,
- management of critical information.

An agreement is usually concluded between the sponsor and the assessor during this stage to define the dependability assessment framework.

Conduct

The conduct phase is the main activity of the assessment process. During this stage, the assessor reviews the assessment deliverables given by the sponsor (or directly by the developer) and performs adequate checks to get a sufficient level of confidence that all the requirements of the criteria for the corresponding confidence level have been met. The assessor's activities are to examine the results of the providing activities performed by the developer and/or to perform complementary confidence providing activities in order to verify the evidence provided by the sponsor. All problems found during this phase are discussed and resolved by the relevant parties. These problems and the sponsor or the developer proposals for their resolution are described in problem reports by the assessor. If no resolution can be agreed, the sponsor may wish to give up the assessment or accept potential limitations in the assessment report.

The assessor prepares an Assessment Technical Report (ATR) recording the overall verdict and the justification for the verdict.

Conclusion

In the conclusion phase, the assessor delivers the ATR as the final output of the assessment to the sponsor. Since the ATR includes sensitive or proprietary information, it is not a public domain document and it may be expurgated before it is given to the sponsor who may not have access to developer proprietary data.

3.2.3.2 Main assessment tasks and links

Figure 2 illustrates the division of assessment tasks between the developer, the sponsor and the assessor of a system.

Deliverables

Deliverables as used in the Criteria are all kind of information that must be delivered or made available to assessors for the purpose of their assessment[1]. These deliverables may be configuration items of hardware, software, firmware, procedures or technical documentation generated during system lifecycle or describing products used to build and maintain the system. . Deliverables may include evidence of former certificates for some system components, organisation processes (quality process, development security), production processes (mounting procedure or assembly equipment), testing processes (specific laboratory accreditation, assembly verification), justification of personal skills required for specific work (certified welder), etc.. Special attention should be paid by assessor in requiring deliverables documenting processes used to construct, operate, maintain, and dismantling the system, but also in requiring some sets of construction documentation (including specific studies) of the overall system in order to verify assumptions, to validate separation of dependability related functions, to validate hazards generated by the system through dependability related functions interfaces, and those that may be introduced by the way of the construction processes themselves (design, realisation, mounting, assembly, etc..).

[1] A different set of deliverables is usually agreed upon between developer and the client ordering the development.

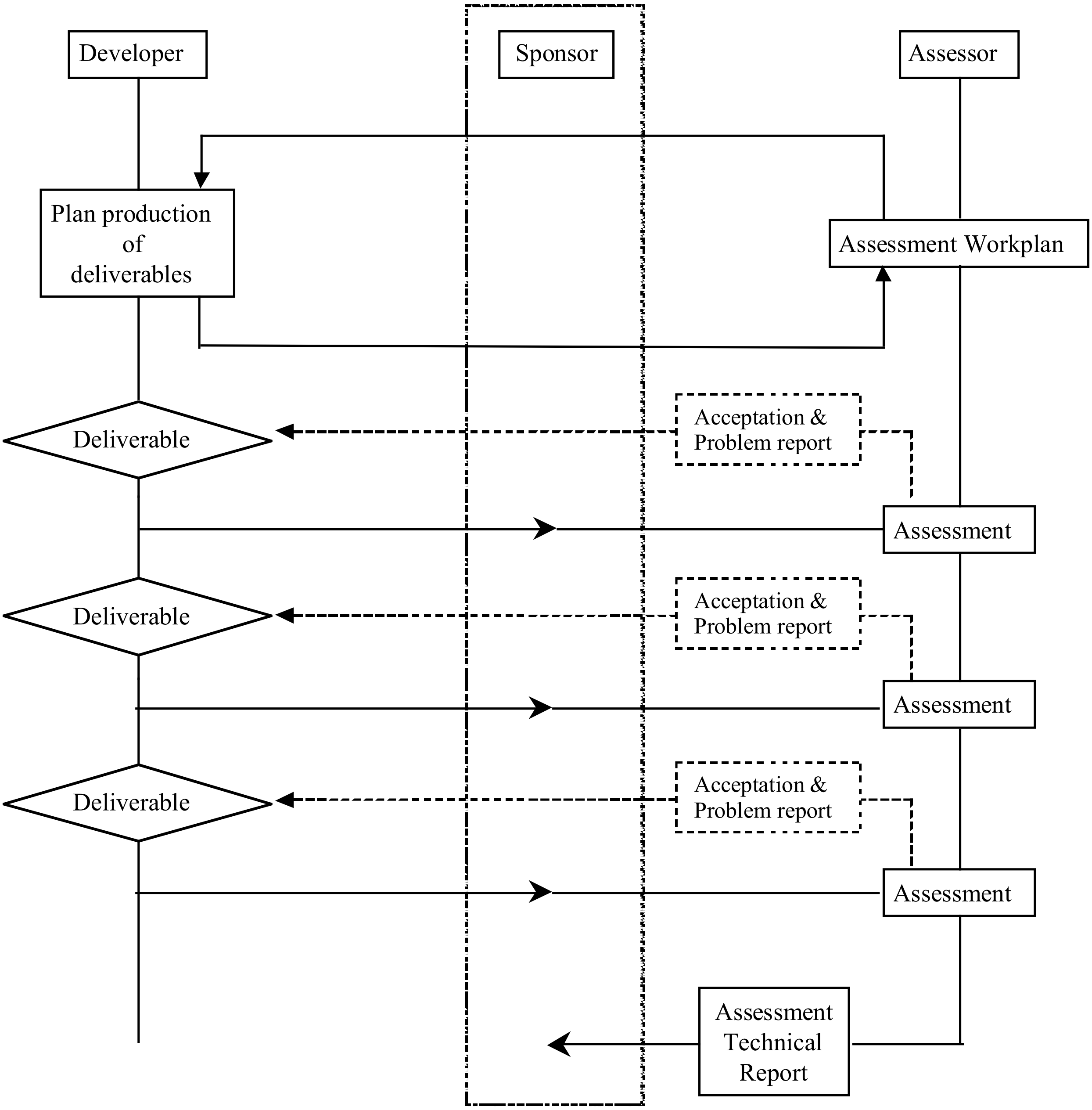
Developer
Sponsor
Assessor
Plan production of deliverables
Assessment Workplan
Deliverable
Acceptation & Problem report
Assessment
Deliverable
Acceptation & Problem report
Assessment
Deliverable
Acceptation & Problem report
Assessment
Assessment Technical Report

Figure 2: Assessment Activities Organisation

Deliverables provide the interface between a developer or a sponsor and an assessor and is the key to define the developer/assessor or sponsor/assessor work split. The developer produces the target system, the developer and the sponsor produce deliverables, and the assessor uses these deliverables to produce verdicts, which are contributions to the establishment of confidence.

Criteria define a dependable construction framework including a generic system construction lifecycle and confidence providing processes. This framework suggests a list of deliverables to be produced at each level of assessment in conformance to the assessment activities. However, probably the developer will have an existing documentation or will use a development cycle that will produce an associated documentation. In that case, the assessor is responsible to appreciate the difference between what it is requested by the assessment and the contents of the internal documentation.

The full set of deliverables needed for the assessment shall be defined by the assessor. The initial set of deliverables defined on the Dependability Target basis produced during the preparation phase shall be improved corresponding to the system decomposition and according to the components confidence level. This process shall be reviewed during the assessment conduct phase.

Planning

Assessment depends on the timely supply of deliverables to the assessors. Failure to provide timely, complete, consistent and correct deliverables is likely to result in an assessment being temporarily suspended or incurring extra costs.

The following issues should be considered when planning an assessment:

- preparation of deliverables,
- time for problem resolution,
- technical and logistical support to the assessment team,
- access to previous assessment results for the assessment team,
- access to the development site by the assessment team,
- access to the system in its operational environment,
- test requirements,
- time allowance for the assessment team to prepare the Assessment Technical Report,
- concurrent or consecutive assessment.

An assessment might be performed after development of the system has been completed, which is called a consecutive assessment, or in parallel with the development of the system, which is called a concurrent assessment. The main difference between concurrent and consecutive assessments is the availability of the deliverables. In a consecutive assessment all deliverables required by the Criteria are normally available right from the start of the assessment. In a concurrent assessment the deliverables will be as the development progresses. Concurrent assessments provide the opportunity for the developer to react rapidly to problems discovered. The difference between the two types of assessment affects the organisation of an assessment, i.e. the Assessment Workplan. In the concurrent assessment both the order and the timescale of the assessment activities are oriented towards the delivery of the deliverables. Testing cannot be performed before appropriate parts of the system are available, so the potential consequences of delays and iterations need to be considered.

Assessment Workplan

During the assessment preparation phase, the assessor shall establish an Assessment Workplan. The assessor records in this document a list of assessment deliverables and their delivery schedule which are constructed from the plan production of deliverables produced by the sponsor or the developer.

Further to documentation given by the sponsor or the developer, the assessor describes in the Assessment Workplan which actions have to be performed for each stage of the system lifecycle. The plan will identify the assessment activities for all four confidence providing processes detailed in Section 3.3.3 and define the most appropriate assessment methods and tools.

Problem Report

While performing assessment work, errors may be discovered, in which case a problem report shall be raised. These problems shall be treated specifically to identify consequences to the assessment. The assessors shall notify the developer or the sponsor of any problems found during the assessment, and the developer or the sponsor shall provide proposals for resolution and the timescales for such. If it is not possible to resolve a problem, the assessors shall determine the impact to the assessment and how it will be affected, so then the sponsor may:

- abandon the assessment,
- continue the assessment while accepting the problem and its implications,
- reschedule the assessment and, if adequate modify the Dependability Target.

Assessment Technical Report

In order to repeat, reproduce and re-use the assessment results, the assessor must write an Assessment Technical Report. This report shall contain the conclusions of the feasibility study, the Assessment Workplan, all verdicts, their justifications and any findings derived from the assessment work. ATR is given to the sponsor and it may be read by others assessors charged with performing re-assessment.

Meetings

During the assessment a number of meetings shall be arranged with the sponsor and/or the developer in order to discuss the progress, problems and plans. These meetings are:

- Start-up meeting,
- Progress meetings,
- Closedown meeting.

3.3 Dependable System Development

3.3.1 Overview

This section describes the dependability construction framework that will be the subject of the dependability assessment and presents the processes to be performed during the dependability assessment. This description does not prescribe a process according to which a dependable system must be constructed but only details those aspects that such a process must have according to the Criteria and that will be assessed.

The dependability construction framework can be applied at different levels of design. For each level, it takes as an input a corresponding system description to refine dependability needs at a lower level. These needs will be detailed recursively more and more in parallel with the system.

The dependability construction starts from a description of the system and of its environment. On this basis, threats and hazards to be considered are listed as the result of a preliminary hazard analysis. They are what the system has to protect against and what the system has to protect, they constitute the needs in terms of dependability. Then, the Dependability Objectives for the system have to be established which specify, in a general way, how to handle these hazards and threats. The Dependability Policy specifies high level measures that meet Dependability Objectives. The measures of the Dependability Policy are then put in concrete terms by specifying functions which then are projected on the system breakdown structure to allocate them on system components. Then, possible Targets of Dependability Assessment (TDAs) are identified (see Section 3.6.1).

The whole process is conducted again for each component, which is a TDA.

In order that confidence can be gained in the ability of the system to satisfy its Dependability Objectives and Requirements, it is necessary to put in place a number of activities, called Confidence Providing Activities (CPAs), which can be grouped into four processes, called Confidence Providing Processes (CPPs):

- The **Dependability Requirement Validation** aims to proof that the system level requirements related to intolerable threats and hazards are correctly mapped to the components of system decomposition.
- The goal of the **Correctness Verification** is to verify that the dependability related functions implemented conform with their specifications.
- The purpose of the **Dependability Validation** is to check the effectiveness of the TDA in meeting its Dependability Objectives.
- The **Process Quality** consists in ensuring that methods, tools and procedures applied during all stages of the system lifecycle are suitable to reach the Dependability Objectives.

The dependability construction framework expounded here is only described for the construction phase of the system lifecycle. However this framework is compatible with all the lifecycle phases including operation & maintenance and decommissioning. This means that following the Criteria, a system is designed so that it can be assessed during all the life cycle phases. In the same way, the Criteria do not require a specific life cycle model or development model. In the following, several phases are considered: construction, operation, decommissioning. The construction phase itself is supposed to include requirements, design, implementation, integration and commissioning, but these steps can be organised diversely.

Operation & maintenance phase is the day to day exploitation of the dependable system. It includes start of operation and operational control. Those steps are taken into account by the measures described in the Dependability Policy. The assessment of these measures is done with the dependability assessment of the whole system during the construction phase. Different forms of maintenance may occur during operation: perfective, adaptive, corrective (see Appendix A.3). The level of system reassessment following maintenance activities will depend on the level of change to the system. For instance, corrective maintenance which replaces like components with like, may need no reassessment, whilst a significant change of functionality may require a significant level of reassessment.

3.3.2 Dependability Construction

This section presents the dependability construction framework following a black box approach. Each box included in Figure 3 will be shortly described below. The description includes the justification for the box, the task to be performed, the inputs needed for this task and the outputs produced.

This dependability construction framework is applied at various levels of system decomposition. Therefore it has to be distinguished between requirements, functions and components at the input side and at the output side of the framework.

3.3.2.1 Initial needs and Environment Analysis

The initial needs and Environment Analysis includes the identification of the overall system, if the Target of Dependability Assessment is a subsystem or a component, and a description of the physical, organisational and personnel environment of the Target of Dependability Assessment. It contains a description of the interfaces of the Target of Dependability Assessment to the environment (including interfaces to other systems, interfaces to procedures, organisations and the physical environment, human interfaces).

Assumptions for the environment that concern the dependability aspects of the Target of Dependability Assessment (i. e. if it is assumed that specific hazards can not occur because they should be eliminated by the environment), are included in this analysis.

3.3.2.2 Preliminary Hazard Analysis

The preliminary hazard analysis aims to identify against what the system has to be protected and what the system has to protect. This step is a prerequisite since hazards and threats identified constitute the dependability needs that the system has to achieve.

The preliminary hazard analysis is based on the system and environment description relevant to the level of detail which the preliminary hazard analysis will be performed. It produces a list of threats and hazards for the system and its environment that shall be taken into account. It includes the assumptions made for each hazard in the analysis and the rating of the severity of each hazard with a description of the rating method and the assumptions made.

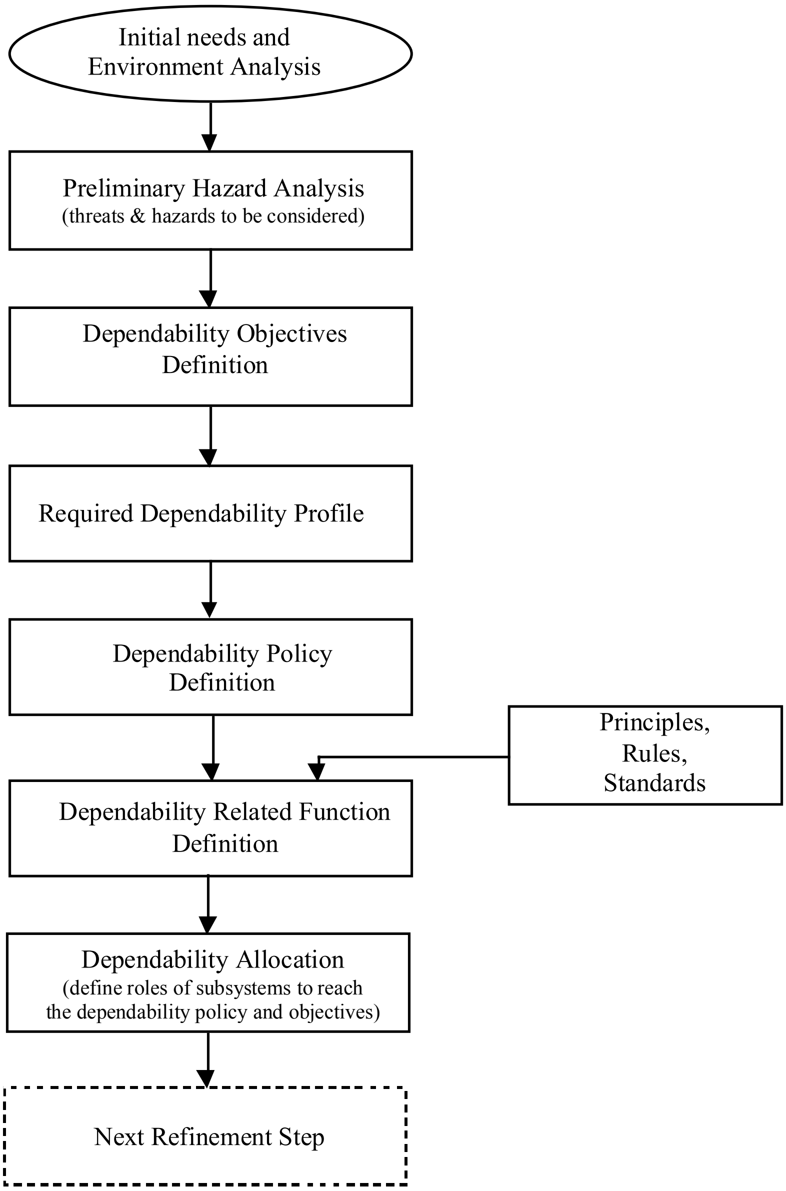
Initial needs and Environment Analysis
Preliminary Hazard Analysis
(threats & hazards to be considered)
Dependability Objectives Definition
Required Dependability Profile
Dependability Policy Definition
Principles, Rules, Standards
Dependability Related Function Definition
Dependability Allocation
(define roles of subsystems to reach the dependability policy and objectives)
Next Refinement Step

Figure 3: Dependability Construction Framework

3.3.2.3 Dependability Objectives Definition

The purpose of the Dependability Objective step is to define the requirements that will allow the system to meet the dependability needs expressed through the threats and hazards identified during the hazard analysis process. These objectives shall be stated in a general way and shall not prescribe a specific implementation or the use of a specific mechanism.

The input for this activity is the list of threats and hazards resulting from the hazard analysis. Dependability objectives definition activity produces a set of objectives for each dependability attribute (safety, availability, reliability, confidentiality, integrity, maintainability). These objectives cover threats and hazards considered for the system and its environment through a ranking specific for each attribute.

The threats and hazards considered are linked to the Dependability Objectives through a rating of hazards and threats. The activity consists of comparing the level of risk (associated to the hazards and threats identified) with what is acceptable for the system. Objectives are then defined as system attributes which are required to reduce the risks to an acceptable level.

Some Objectives may be stated in form of a system attribute which can not be directly mapped to a system function. For example an objective may describe a state or set of states the system shall never reach. These types of objectives are called "negative objectives", since they can not be directly transformed into directly implementable functional requirements. Checking that those objectives are met by the system is therefore much harder since one has to check the whole system (or at least a whole component) for compliance with this objective.

If the TDA is a part of a larger system, the Dependability Objectives of the TDA shall be related to the wider Dependability Objectives of the larger system.

3.3.2.4 Required Dependability Profile

For each function or component the required dependability profile is specified, indicating for each dependability attribute (safety, security, availability…) the confidence level that should be achieved. For complex functions or components there may be a sub-profile, i. e. within the architecture of this component there may be a requirement for different dependability profiles for functions or sub-components.

Arguments have to be presented which justify the selection of the individual dependability profiles and demonstrate that the overall dependability profile for the Target of Dependability Assessment (TDA) related to the Dependability Objectives is achieved. As an example, some of these arguments and rules can be found in DO 178B through classes A, B, C, D, E.

The dependability profile determines the choice of activities to be performed to provide the evidence for a particular assessment, but it does not impose a list of fixed activities to be done. This is left for each industry sector to determine the activities appropriate for achieving adequate confidence in the system development process.

3.3.2.5 Dependability Policy Definition

The dependability policy defines the properties and rules that the system should maintain to fulfil its Dependability Objectives. An example of such properties is the "fail-safe" property: whatever a failure of any part of the system happens, the system is put in a safe state. This may be achieved by implementing the "fail-halt" principle (see A.6.7). Another example is the Bell-LaPadula security policy: to prevent confidential information leakage (Dependability Objective), clearance and classification levels are assigned to users and data, and rules are defined to indicate which clearance level users can read which classified data (no read-up) and which clearance level users can write which classified data (no write-down).

Usually, several policies are possible to achieve the same Dependability Objectives. For instance, a very high reliability may satisfy the safety objectives, or another security policy (e.g. the Brewer-Nash policy) may prevent the dreaded information leakage. But the choice of policy may influence strongly the design of the system, as well as its cost.

3.3.2.6 Dependability Related Function Definition

This section shall describe the overall architecture of the TDA identifying the main functions related to dependability and provide the dependability requirements for those functions. A mapping shall show how the individual statements of the Dependability Policy are broken down to the functions or components of the Target of Dependability Assessment and how they are reflected in the dependability requirements for those functions or components.

In order to develop a dependable system, a number of dependability principles may be used to achieve the desired dependability attributes. Having established the dependability profile of a system, an appropriate set of dependability principles will need to be used which provide the required dependability attributes. The list of principles to be used is defined in Annexe A, Section A.6, and consists of:

- Identification and Authentication
- Access Control
- Accountability and Audit
- Object Re-Use
- Levels of Redundancy
- Design Diversity
- Fail-Halt System
- Graceful Degradation
- Defence in Depth
- Defensive Programming
- Partitioning and Separation

The principles may be further divided into more specific functions and techniques which will provide greater or lesser confidence. Thus, redundancy could be sub-divided into dual redundant and triple redundant systems. Triple redundant systems may then be deemed as suitable for higher confidence levels than dual redundant systems.

It is not the intention here to define the method for identifying which principles should be used for a particular dependability profile, as this is left for each industry sector to determine.

3.3.2.7 Dependability Allocation

The Dependability Allocation derives the Dependability Objectives and Policy according to the system refinement. Its purpose is to specify each subsystem role to enforce the Dependability Policy and Objectives.

Each subsystem can be human, software or hardware. The allocation will define and justify the choice of the TDA and the associated confidence level.

The choice of the confidence level and the associated justification is closely linked to the risk ranking and associated objectives/policy. Since high confidence levels induce high costs, technical justification for the choice of these levels has to be found. Then, the dependability allocation activity shall show that chosen confidence levels meet the Dependability Objectives.

The dependability allocation is performed on the basis of the Dependability Policy, principles from the application area and rules established for the dependability allocation. Principles are constituted of formal guidance and informal experience, state of the art and codes of practices applicable for the considered system. Rules for the dependability allocation are the algorithms chosen before the system development to establish a mapping between confidence level and Dependability Objectives.

The output of the dependability allocation process is composed of some identified components that have to enforce measures described in the Dependability Policy. These components are associated to a dependability profile. They are the targets of the dependability assessment.

3.3.2.8 Realisation

Realisation must be understood as the process of creating a system component at every stage of system decomposition (unit, sub-system, etc..) and is not a task necessarily performed just immediately after dependability allocation. Moreover, except for some simple processes such as software unit generation by compiling the corresponding source code, realisation of components generally does not derive directly from its specification requirements and needs complementary studies.

These studies which address the definition of materials, substructures, lay-out, assembly/mounting procedures and verifications, etc. are included by some company in a manufacturing process under the responsibility of a specialised department or another organisation (sub-contractor for instance), although a further decomposition of the unit under consideration may result that the limit between design process and manufacturing process may vary.

The realisation process may introduce hazards which should be taken into account in the Dependability Target at the relevant level. These hazards are pointed out from technology (EMI, radioactive emanation), lay-out (interference, noise, breaks), mounting (screwing, riveting, connecting), etc. and may lead to perform verification activities (soldering inspection, welding radiography, continuity testing, screw torque verification, etc..). Others may be located inside the generation process itself, known to be imperfect but not replaceable (unavailability or lack of a better process) and may lead to define complementary countermeasures (to assume the required security level for example) or functions and components (improved shielding and isolation, for compiler bug by-pass).

3.3.3 Confidence Providing Processes

The Confidence Providing Processes (CPPs) are detailed here roughly in order to place them relatively to the overall dependability construction framework. To perform the confidence providing processes, a great number of activities taken from safety and security sector, called Confidence Providing Activities (CPAs) are available. Chapter 4 will present these activities in detail arranged by confidence providing processes. The CPPs are generally performed by the developer, whilst the assessor evaluates the results of the confidence providing activities. The

CPPs are the scope of the dependability assessment. However the assessor can repeat some confidence providing activities or do complementary confidence providing activities. The role of the developer is to check that his system is developed in conformance with the criteria and he must provide the proof to the assessor that the system developed is compliant with the criteria.

Figure 4 positions these processes relatively to the development phase of a system.

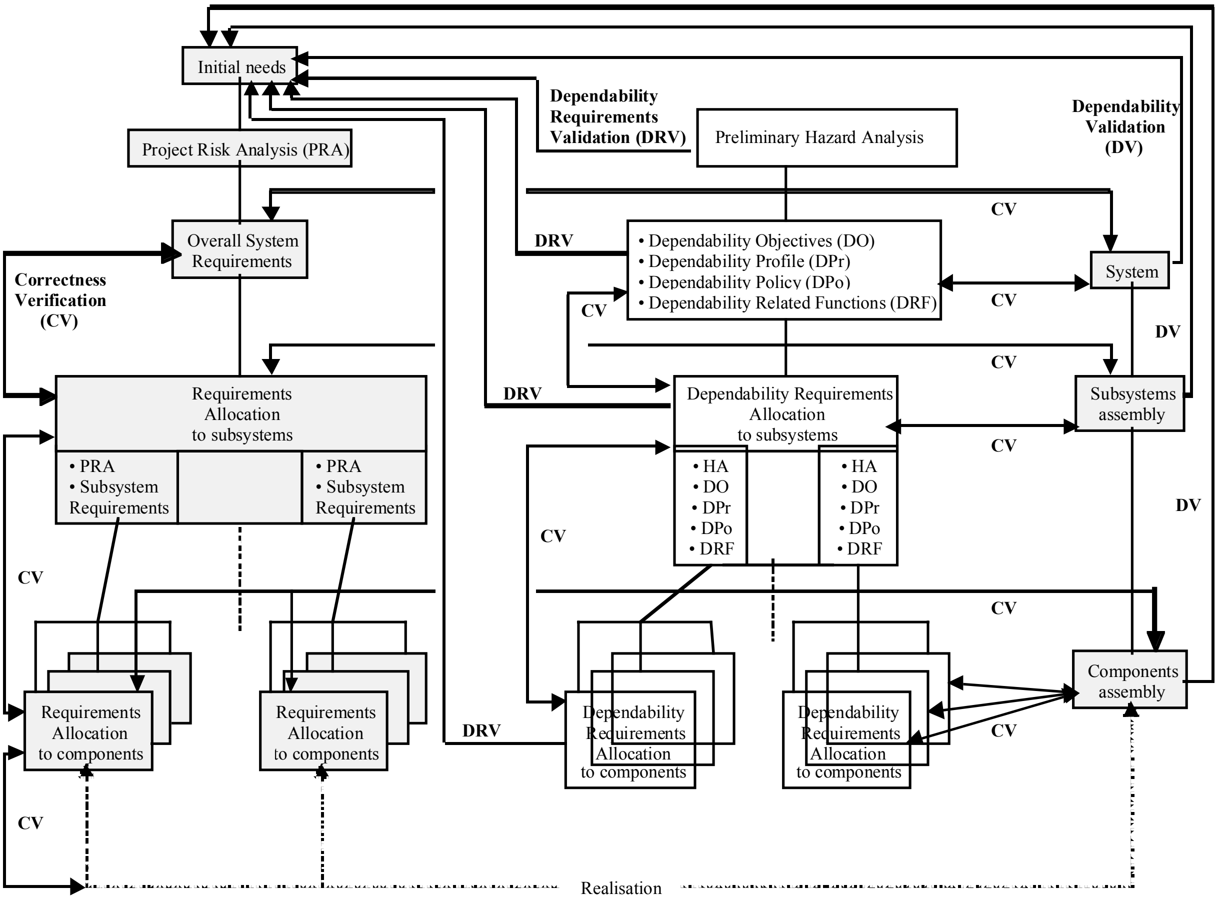
Initial needs
Project Risk Analysis (PRA)
Dependability Requirements Validation (DRV)
Preliminary Hazard Analysis
Dependability Validation (DV)
Overall System Requirements
Correctness Verification (CV)
DRV
• Dependability Objectives (DO)
• Dependability Profile (DPr)
• Dependability Policy (DPo)
• Dependability Related Functions (DRF)
CV
System
DV
Requirements Allocation to subsystems
• PRA
• Subsystem Requirements
Dependability Requirements Allocation to subsystems
• HA
• DO
• DPr
• DPo
• DRF
Subsystems assembly
Requirements Allocation to components
Dependability Requirements Allocation to components
Components assembly
Realisation

Figure 4: Major development steps

3.3.3.1 Dependability Requirement Validation

The Dependability Requirement Validation aims to ensure that the Dependability Objectives specified in the Dependability Target cover properly the hazards and the threats identified at higher level.

This includes checks for completeness and consistency of the hazard analysis performed as well as an analysis if the dependability policy derived and specific functional requirements specified are sufficient to cover all aspects of the overall Dependability Objectives.

The first assessment activity that has to be performed for a dependable system is a thorough analysis if the steps that lead to a dependability target have been performed appropriately. This shall ensure that the Dependability Objectives are as complete as possible, that all potential hazards are identified and correctly rated and that the dependability policy and the dependability requirements for the individual components and functions consistently and completely reflect the objectives as well as the identified hazards.

The Dependability Requirement Validation is close to the Dependability Validation because both activities verify that dependability requirements are correctly derived from higher level dependability needs. The only difference between both lies on the fact that Dependability Requirement Validation is applied before the TDA has been established while the Dependability Validation is carried out on the basis of the TDA.

3.3.3.2 Correctness Verification

The Correctness Verification aims to ensure that the steps of the development process have been carried out correctly from the functional requirements to the system exploitation. Correctness verification is therefore concerned with the assessment, that a lower level of design (including the implementation) is consistent with the higher design levels. This activity does not address threat, hazard or Dependability Objective coverage but only that proper development has been done. It is near to a quality verification.

Correctness verification is the process to check, if the system is compliant with the specification and if a low level design and specification is compliant with the higher levels of design. This includes also the check for compliance with the requirements specification as long as the requirements are stated in a way, which allows to check the compliance directly. Various CPAs have been identified, that can be used for Correctness Verification. This includes mostly testing procedures as well as informal or formal design analysis and verification techniques.

The rigour that can be applied for Correctness Verification is dependent on the precise and unambiguous representation of the different design levels. Formal analysis and verification techniques require a higher level of precision and unambiguity in the design representation, which limits the methods that can be used for the description of the design. Especially for high levels of confidence in the correctness of a system, the design must not be subject to ambiguity.

Details of this concept can be found in Section 4.3.

3.3.3.3 Dependability Validation

The Dependability Validation checks that the Dependability Objectives are well covered with effectiveness by the TDA. This activity typically includes penetration testing in the security sector, fault assumption validation in the safety sector and hardness testing.

The design of a system always starts with some high level objectives which are not always directly translatable to precise requirements, whose implementation can be shown using Correctness Verification arguments. Indeed, dependable systems often have objectives or requirements stated in a form that compliance with them cannot be shown by usual Correctness Verification activities.

The main problem of a system design is to identify the requirements for the system. It is well known that those requirements can never be completely related to the objectives. The purpose of the Dependability Validation activities is to identify critical problems, where the system is compliant with the specification but does not fulfil the objectives. The main task of all those activities is therefore, to identify side effects of the system or non covered/suitable aspects, which may lead to critical situations.

3.3.3.4 Process Quality

Process Quality is a prerequisite to develop a dependable system since it is not possible to trust a system which has not been realised following a well defined process. The quality assurance shall be extended to include the new processes introduced by the dependability activities.

This includes all activities related to the assessment of processes and procedures (e.g. for development, operation and maintenance). Those activities shall ensure that processes and procedures are well defined and applied during all stages of the system lifecycle.

3.4 Dependability Profiles

3.4.1 Introduction

The level of risk associated with a system is linked with the level of confidence that is required that the system will behave in a dependable manner. Therefore, if a failure of a system may result in the loss of many lives, the level of confidence required in that system would be greater than that where a failure only results in minor injury. Confidence in a system is achieved in a number of ways including :

- choices made in the system design,
- analysis of the system,
- the appropriateness of the system development process,
- independent assessment.

The level of detail and rigour applied in each of these areas will vary according to the required level of confidence.

3.4.2 Selection of Confidence Levels

The confidence level required for a system should be determined during hazard analysis and the subsequent allocation of dependability requirements to the various system components. It is not the intention here to define the precise method for doing this as this is left for each industry sector to determine. However, suffice to say that an approach can be defined for linking risks to required confidence levels.

3.4.3 Confidence levels in existing Security and Safety Standards

The concept of confidence levels is not new and has been used in a number of standards in different fields. The ITSEC defines 6 levels of confidence (E1-E6) and a number of safety standards (including IEC 1508, EN50128, ISA SP84) define 4 (SIL1-SIL4). These standards do not provide a generic structure which can be applied to all aspects of dependability and so the concepts need to be harmonised.

The immediate problem with this is that the different standards define differing numbers of confidence levels. This makes the mapping between levels more difficult and requires a detailed comparison to be conducted. However, it is possible to identify how the various confidence levels relate to each other and this process can be used when defining the detailed criteria.

3.4.4 Dependability Attributes and Confidence Levels

Dependability has a number of recognised attributes:

- Availability,
- Confidentiality,
- Reliability,
- Integrity,
- Safety,
- Maintainability.

A system may have requirements to possess some of these attributes and the importance of the attributes in a particular application may not be uniform, e.g. a safety system may have safety requirements and maintainability requirements but the safety requirements are more significant than the maintainability requirements. Therefore, each attribute of dependability will have associated with it a confidence level:

Availability	A0-A4
Confidentiality	C0-C4
Reliability	R0-R4
Integrity	I0-I4
Safety	S0-S4
Maintainability	M0-M4

Table 1: Confidence Level Scale

A confidence level for an attribute is used in one of two ways:

- as a requirement to be achieved by properly applying the appropriate procedures, controls and techniques (as defined by these Criteria) during the system lifecycle

- as the actual confidence level achieved as determined by an assessment according to these Criteria.

For each attribute, level 1 is the lowest confidence level and level 4 is the highest. Level 0 means that there is no requirement for that dependability attribute.

So, for example, a system may be deemed to have A1, C0, R3, I3, S3, M2 dependability requirements. This is termed the system **dependability profile**.

3.4.5 Confidence Levels and the Criteria

Chapter 4 defines in detail the Confidence Providing Activities (CPAs) and the assessor's activities.

The choice of CPAs needed to provide the necessary evidence for an assessment is determined by the dependability profile. Table 16 (page 117) shows which CPAs are required for each confidence level of all dependability attributes. Chapter 4 then specifies how the levels of rigour, detail and independence are interpreted for each of the four confidence levels 1 to 4. (There is no requirement neither on CPPs nor on assessment for level 0.)

The level of rigour determines how the CPA has to be done (e.g. from informally to formally, from manually to automatically, etc.), the degree of justification to be provided to the evaluator (e.g. suitability of methods, evaluation of tools, etc.), and the kind of evidence required (e.g. test coverage).

The level of detail defines the scope of the CPA such as whether it addresses:

- parts or all of the system,
- one or many refinement levels,
- all the properties or only a subset.

The level of independence specifies the organizational links between those carrying out the CPA and the developers. Three levels of independence are distinguished: independent persons, independent departments, and independent organizations.

If a particular CPA is required for a particular attribute then that CPA must be applied (with the appropriate level of rigour, detail and independence) to all components of the system that implement that attribute. Thus, there will be benefits (in simplicity of the system, efficiency of implementation etc.) to be gained if the system architecture can demonstrate separation of functions so that the higher confidence levels for at least some of the attributes are confined to certain parts of the system. The CPAs appropriate for those higher levels of confidence then need to be applied only to those parts of the system.

Assessment activities are always carried out at the same level for all confidence levels as defined in Chapter 4.

3.5 The Dependability Target

The Dependability Target is a document which is the initial input for a dependability assessment. It is the major dependability specification document.

The document is written on the basis of the criteria in order to be compliant with them, it compiles most of the information from the framework described in Section 3.3.2: TDA identification, Dependability Objectives and associated policy.

The dependability target at the system level has to be developed before the confidence providing processes start. It gives an overview on the system architecture and the system environment. In addition it also defines the appropriate dependability profile for the system. The Dependability Target is improved by the addition of new hazards appearing at each step of the refinement of the system into components and at each step of the components assembly.

Figure 5 illustrates the structure of the Dependability Target.

3.5.1 Content of a Dependability Target

The following section summarises the content of a Dependability Target. Most of the chapters are mandatory, but may contain references to other documents which describe the aspects relevant to that chapter.

3.5.1.1 Introduction

The introduction section is always needed in a dependability target. It contains the following parts:

- a precise identification of the Target of the Dependability Assessment (TDA),
- a short overview of the TDA (or a reference, where this overview can be found),
- a short description of the overall goals of the assessment as well as a reference to other standards that shall be met.

3.5.1.2 Description of the Target of Dependability Assessment

This part shall describe:

- the overall purpose of the TDA (and especially its dependability attributes),
- the relation of the TDA to other systems, the task of the TDA within an organisation,
- specific non-technical aspects that have to be considered during the dependability assessment.

All or parts of this section may be satisfied by precise references to other documents.

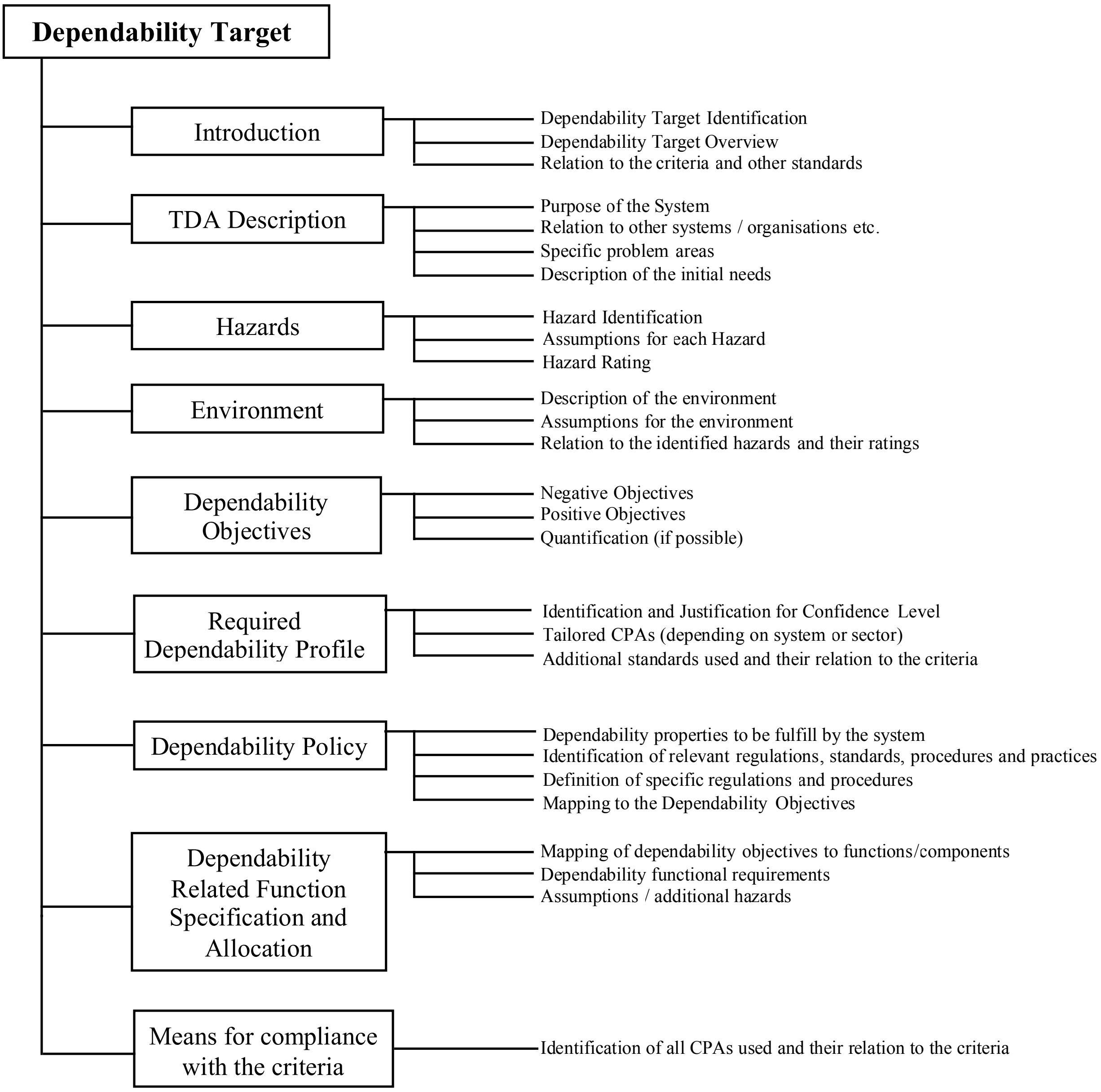
Dependability Target
Introduction
Dependability Target Identification
Dependability Target Overview
Relation to the criteria and other standards
TDA Description
Purpose of the System
Relation to other systems / organisations etc.
Specific problem areas
Description of the initial needs
Hazards
Hazard Identification
Assumptions for each Hazard
Hazard Rating
Environment
Description of the environment
Assumptions for the environment
Relation to the identified hazards and their ratings
Dependability Objectives
Negative Objectives
Positive Objectives
Quantification (if possible)
Required Dependability Profile
Identification and Justification for Confidence Level
Tailored CPAs (depending on system or sector)
Additional standards used and their relation to the criteria
Dependability Policy
Dependability properties to be fulfill by the system
Identification of relevant regulations, standards, procedures and practices
Definition of specific regulations and procedures
Mapping to the Dependability Objectives
Dependability Related Function Specification and Allocation
Mapping of dependability objectives to functions/components
Dependability functional requirements
Assumptions / additional hazards
Means for compliance with the criteria
Identification of all CPAs used and their relation to the criteria

Figure 5: Dependability Target Structure

3.5.1.3 Initial needs and Environment Analysis

This section shall describe the system and the environment of the TDA as specified in Section 3.3.2.1.

3.5.1.4 Hazards

This section shall contain the results of the Preliminary Hazard Analysis as described in Section 3.3.2.2.

3.5.1.5 Dependability Objectives

This section describes the Dependability Objectives as identified in Section 3.3.2.3.

3.5.1.6 Required Dependability Profile

This section describes the Dependability Profile as identified in Section 3.3.2.4.

3.5.1.7 Dependability Policy

This section contains the results of the corresponding Section 3.3.2.5.

3.5.1.8 Dependability Function Specification and Allocation

This section includes the results of the appropriate Sections 3.3.2.6 and 3.3.2.7

3.5.2 Use of the Dependability Target

The Dependability Target at system level is normally written in an early stage of the system development. The purpose of this document is to serve as a guideline for the dependability assessment of the system. Assessors will use this document to identify the overall Dependability Objectives, the hazards to the system and their ratings, the dependability policy and the allocation of dependability functions to the parts of the system..

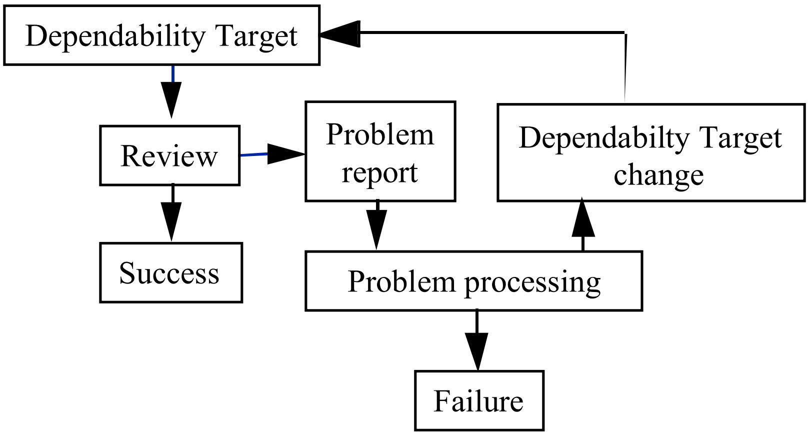

Figure 6: Dependability Target Review

The assessors may review the Dependability Target in an iterative manner until it provides a sound basis for the assessment. The sponsor is responsible of the preparation of the Dependability Target which is intended to serve as a basis for the dependability assessment. The Dependability Target should be specified in terms defined in Section 3.5.1, drawing on the guidance given there on the presentation and content.

The developer shall be notified of any problems (Problem Report) in order to the Dependability Target be amended prior assessment starting.

The findings of the reviews shall be documented in an assessment technical report for reuse during the conduct phase. The assessors shall confirm that the Dependability Target is acceptable.

The criteria provide a list of CPAs related to specific dependability attributes and to specific phases of the system lifecycle as described in Chapter 4.

The Dependability Target is an input for the developer to select the CPAs that seem appropriate for the TDA with its specific dependability.

3.6 Complex Systems

Complex Systems are normally broken down into several subsystems, which themselves may be broken down further into sub-subsystems, and so on. Such a decomposition is necessary to:

- control the complexity of the system
- to isolate certain types of functionality (e.g. confidentiality) in a few components rather than spread it thinly throughout the system
- to allow different types of component (e.g. hardware or software) to be developed by different teams
- to allow some components to be subcontracted out
- to allow some components to be implemented using Commercial Off The Shelf (COTS) products.

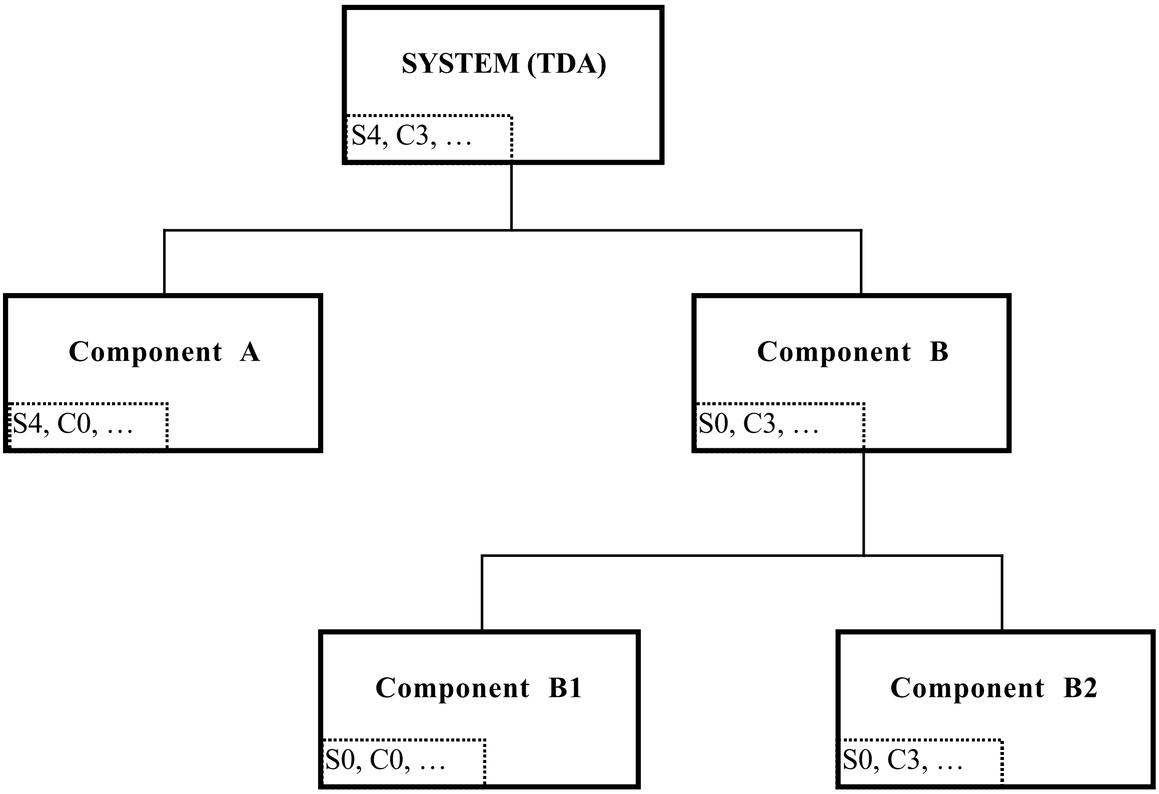

Figure 7: Example of System Decomposition

3.6.1 Dependability Construction Framework

As a system is decomposed into components and those components are further decomposed, decisions will be taken about the allocation of requirements to each component and the design of each component. As an integral part of this process, the Dependability Construction Framework will be applied recursively to each component. Thus, in accordance with Section 3.5, the Dependability Target will be updated with the Dependability Target information for each component.

The application of the Dependability Construction Framework will involve the determination of the Dependability Profile for each component. These Dependability Profiles can be (and should be) different from the system Dependability Profile. It is good practice to isolate critical functionality in a part of the system, so that its implementation can be separated from other parts of the system.

Figure 7 shows an example of a system (with a partial Dependability Profile of S4, C3, …) decomposed into 2 major components, A and B. The safety related functionality has been confined to A, and the confidentiality functionality has been confined to B, as demonstrated by their respective partial Dependability Profiles of S4, C0, … and S0, C3, … Component B has been further decomposed into components B1 and B2. B1 is safety and confidentiality irrelevant (partial Dependability Profile of S0, C0, …) and B2 implements all the confidentiality functionality (partial Dependability Profile of S0, C3, …).

In an example like this, strong arguments must be provided to justify the separation of the components and show that:

- B cannot affect the safety functionality of A
- A cannot affect the confidentiality functionality of B
- B1 cannot affect the confidentiality functionality of B2
- etc.

If these arguments can be presented then consideration of the critical functionality can be confined to the relevant parts of the system. In the example shown, the activities needed for the development of a confidentiality related component can be confined to B, B2, ignoring A and B1.

3.6.2 Subcontractors

The development of a component may be subcontracted by the developer of the system (prime contractor) to another organisation. That organisation can be either a separate company with a formal contractual relationship or a different department or team within the same company. In the former case there will be a formal contractual relationship and in the latter case at least an implicit contract. In either case, we can regard the organisation implementing the component as a subcontractor.

Irrespective of whether there is a formal contract, it is crucial that the requirements for the component are correctly and completely described. The dependability requirements will be expressed in the Dependability Target for the component.

It may or may not be possible to perform Dependability Requirements Validation and Dependability Validation against the system Initial Needs etc. depending on the prime contractor's relationship with the subcontractor. Figure 8 illustrates the two possibilities.

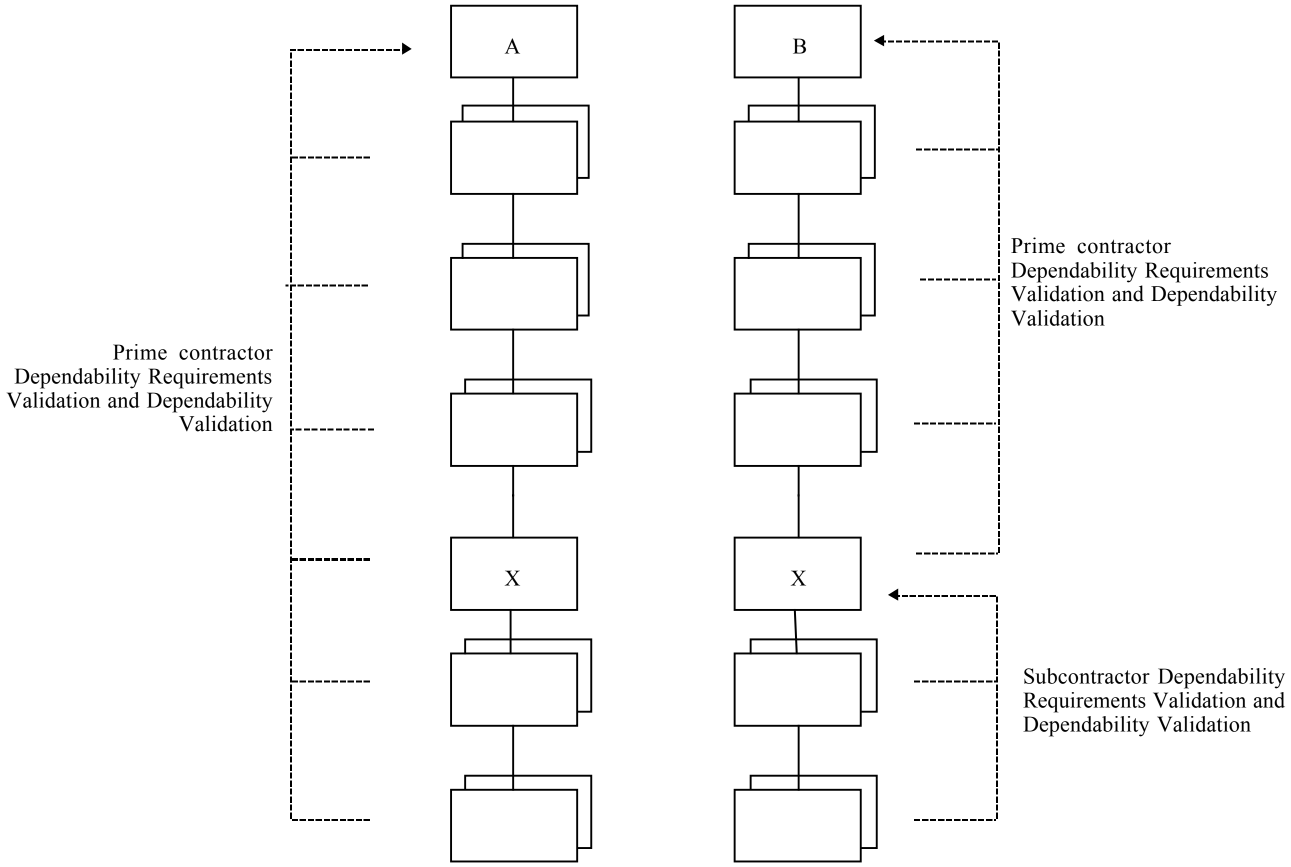
A
B
X
X
Prime contractor Dependability Requirements Validation and Dependability Validation
Prime contractor Dependability Requirements Validation and Dependability Validation
Subcontractor Dependability Requirements Validation and Dependability Validation

Figure 8: Subcontracted Component

For system A, component X has been subcontracted but the prime contractor and subcontractor both have access to all the relevant information about the system. In this case, Dependability Requirements Validation and Dependability Validation can be performed for parts of the system (including X) against the Initial Needs etc. for the system.

For system B, again component X has been subcontracted but the relationship between the prime contractor and the subcontractor is such that:

- the prime contractor does not have access to the design etc. of X
- the subcontractor does not have access to the design etc. of the system outside component X.

This is a less desirable situation than with system A. Wherever possible, the prime contractor should try to ensure that he has sufficient access to the design etc. of X.

In this case the prime contractor can perform the Dependability Requirements Validation and Dependability Validation only on the components for which it has information. Specifically, the subcontractor can perform Dependability Requirements Validation and Dependability Validation only on the components for which it has information. Specifically, the Dependability Requirements Validation and Dependability Validation for the components of X can be performed only against the Initial Needs etc. of X, not of the system. It is most important to ensure that the requirements for X (which are imposed on the subcontractor) are correct and complete with respect to the system Initial Needs etc. Any errors or omissions will result in a component X that is not suitable for use in the system.

In each case, system A or system B, a Dependability Target is required for component X.

3.6.3 COTS Products

There is often a need (or wish) to include COTS products in a system, for instance as component B2 in Figure 7. In order for this to happen a number of conditions must be fulfilled:

- the COTS product must provide the dependability (and other) functionality that is required of component B2
- the COTS product must not provide too much unwanted functionality that could provide additional hazards
- there must be sufficient confidence that the COTS product does provide just the necessary dependability functionality in the environment in which it will be used within the system
- it must be possible to check that the previous conditions are fulfilled.

There must be a description of the COTS product identifying the functionality provided by the product and the environment(s) within which it will provide that functionality. Ideally this description will be in a Dependability Target, but it must exist, otherwise it will not be possible to check whether the product's functionality matches the requirements of the component it is to replace.

If the system component (B2 in our example) has some dependability requirements then it will have a Dependability Profile. This shows that there are some confidence requirements that have to be satisfied by the product and these confidence requirements can be obtained only by some form of assessment. There are a number of possibilities:

- the product has been subjected to an assessment against the Criteria - there will be a certificate and assessment reports giving the results of the assessment and these results (achieved Dependability Profile, assessed functionality and environment) can be checked against the requirements for the component B2
- the product has been assessed against some other criteria - this is similar to the previous case but it is also necessary the decide to what extent the other assessment method is equivalent to a Criteria assessment
- the product has not been assessed - the product cannot be used until it has been assessed either as a separate product or as part of the system assessment.

4 Criteria Framework

In Chapter 3 the processes needed to specify an adequate Dependability Profile and to build and assess a dependable system have been introduced. The roles which are relevant to generate confidence in such a system have been characterised and the Confidence Providing Processes have been correlated to the processes of regular development, to the life cycle phases and to the roles.

This chapter describes the Confidence Providing Processes in adequate detail. It defines and describes the set of Confidence Providing Activities, that possibly constitute the Confidence Providing Processes and it provides guidance to select the appropriate activities and tailor their application with respect to the Dependability Profile of a TDA.

4.1 Structure of the Presentation of Processes and Activities

The Confidence Providing Processes that have been introduced in Chapter 3 are described in detail in the following sections:

- 4.2 Dependability Requirements Validation
- 4.3 Correctness Verification
- 4.4 Dependability Validation
- 4.5 Process Quality

The presentation of these processes has two principal directions:

- It gives a general description of the process under consideration covering its objectives, scope and whether particular aspects have to be considered at the various stages of the development or for sector specific applications.
- It specifies the Confidence Providing Activities, which are required to achieve the process objectives.

The Confidence Providing Activities are presented in a uniform schema:

Synopsis: presents the principal characteristics of the CPA, which are an extract of the following sections, in the form of a tabular overview.

Objectives: describes, how and to which degree the CPA covers the objectives of the Dependability Related Process.

Inputs: describes the prerequisites for the Confidence Providing Activity.

Description of the CPA: points out the characteristics of the CPA and what the developer has to do.

Levels: describes, how the levels of Rigour, Detail and Independence have to be interpreted for the CPA and how they map to the Confidence Levels.

Methods and tools: gives a characterisation of the methods and tools that are currently available and possibly a judgement on their applicability.

Output: specifies requirements on content and form of the results.

Use of results: indicates, the development is affected by the output of the activity.

Relation to the life cycle: points out the phases of the life cycle, where the CPA should be applied.

Assessors' activities: specifies, what the assessor has to do with respect to the CPA. While the CPA has to be carried out in the first place by the developer, it may be required, that the assessor analyses the results of the CPA or that he repeats a CPA, etc.

Where a CPA is used in more than one Dependability Related Process, it is described once at its first occurrence and referenced at later occurrences.

Each CPA has to be selected and tailored according to the applicable dependability profile of the TDA. Chapter 4.6 describes the procedure to treat the varying Confidence Levels of a Dependability Profile and to select the right Dependability Profile in those cases, when a CPA relates components of a system that have different Dependability Profiles. It also presents a mapping table, that specifies which CPAs should be selected and which levels of detail, rigour and independence apply for a given level of confidence.

The CPAs recommended in the Criteria and discussed in this Chapter are:

- Preliminary Hazard Analysis
- Hazard Analysis
- Probabilistic Quantitative Evaluation
- Common Cause Analysis
- Static Analysis
- Behavioural Analysis
- Formal Methods & Proofs
- Testing
- Traceability Analysis
- Penetration Analysis
- Covert Channel Analysis
- Experimental Evaluation
- Quality Assurance

4.2 Dependability Requirements Validation

4.2.1 Motivation and Scope

Dependability requirements validation aims to ensure that the set of requirements defined during the different stages of the design process and the assumptions established at each stage are valid, complete, consistent and comply with the initial needs. Dependability requirements validation is to be performed along side the design and before the implementation starts whereas validation of the implementation itself is covered by dependability validation activities. Moreover, Dependability requirements validation and Correctness Verification are complementary. The main difference is that the latter checks the specifications established at a given level of system refinement against the specifications of the higher level, whereas the former ensures that the specifications at a given level comply with the initial needs. However, Correctness Verification activities can serve the purpose of dependability requirements validation.

Requirements and assumptions should be validated at each hierarchical level of requirements definitions. All relevant engineering disciplines, environmental factors, human factors, administrative, organisational constraints, … should be considered during the validation process. Moreover, all the dependability objectives should be addressed globally to analyse potential interdependencies between dependability attributes and define an optimal design that ensures the best tradeoffs.

Different kinds of assumptions are established during the design process: e.g. operational and environmental assumptions, fault and hazard assumptions, design related assumptions, independence assumptions, etc. All these assumptions should be checked for their validity. Especially, fault assumptions constitute a key point in the development of dependable systems. At each system description abstraction level, associated fault assumptions should be defined taking into account the fault and hazard assumptions established at the higher levels. This leads to a hierarchy of fault models. Ensuring the consistency of these fault models is a difficult task that requires a thorough examination and in-depth analysis. In particular, it is essential to study how the faults that occur at the lower levels manifest and propagate to the higher levels, and how these faults might affect the system level stated objectives and initial needs. During the dependability requirements validation process, it is particularly important to ensure that the fault and hazard assumptions established during the design are representative of those likely to occur during operation. Also, the validation should check the validity of the rating of the likelihood and consequences of the hazards.

The activities defined in this section try to address specific aspects of the problem stated above. They can never guarantee that all problems are identified, but they nevertheless increase the confidence in the system. Most of those activities can be used to address all dependability attributes. When a given activity is specific to one dependability attribute, this is explicitly mentioned.

4.2.2 Dependability requirements validation confidence providing activities

This section presents the following dependability requirements validation confidence providing activities:

- Preliminary Hazard Analysis
- Hazard Analysis
- Common Cause Analysis
- Probabilistic Quantitative Evaluation

The first three activities can be used to address all dependability attributes. Probabilistic quantitative evaluation is recommended for all dependability attributes, however there is little evidence about the applicability of this technique to confidentiality.

4.2.2.1 Preliminary Hazard Analysis (PHA)

<table>
<tr><td colspan="2"><b><i>Name of the CPA</i></b>
Preliminary Hazard Analysis</td></tr>
<tr><td colspan="2"><b><i>Objectives</i></b>
Determine the criticality of the system by finding hazards, rating them and deriving a Dependability Profile, Dependability Objectives and a Dependability Policy.</td></tr>
<tr><td><b><i>Inputs</i></b>
• Knowledge of the application field, its policies, standards and constraints
• Knowledge of Dependability analysis
• Needs for the system
• Set of rules for the acceptability of risks</td><td><b><i>Outputs</i></b>
• List of hazards with their rating
• Dependability Profile
• Dependability Objectives
• Dependability Policy
• Report of Preliminary Hazard Analysis</td></tr>
<tr><td colspan="2"><b><i>Mapping Confidence Levels to Dependability Profile</i></b>
Same level of detail, rigour and independence
(Remark to independence: A Dependability team (person) together with a team consisting of members of the developer and/or the sponsor should do the PHA. It should not be only one person)</td></tr>
<tr><td colspan="2"><b><i>Methods and Tools</i></b>
HAZOPS, ETA, FMECA (functional level), Checklists
(See also CPA “Hazard Analysis”)</td></tr>
</table>

Objective

The Preliminary Hazard Analysis provides results, which will determine the level of criticality of the system in terms of a Dependability Profile and Dependability Objectives, which will be used as guidelines for the system design.

A Hazard is defined as an event that potentially has unacceptable consequences. The Preliminary Hazard Analysis consists of the following steps:

- The identification of potential hazards arising from the system itself and from its environment, the identification of unacceptable consequences (failures) of these hazards and the identification of their causes (faults).
- The estimation of the likelihood and rate occurrence of the causes and consequences and of the severity of the consequences, often known as Hazard Rating. This should be done for different attributes (damages for persons, equipment, finance, etc.). This estimation results

first in the definition of the criticality of the functions to be accomplished by the system and the assignment of Confidence Levels for every Dependability Attribute for the system, which are collected in the Dependability Profile.

- From the Dependability Profile, the Dependability Objectives are derived for the reduction of the occurrence of identified causes and/or the reduction of the severity of consequences. These are general dependability properties which the system must have or may not have, e. g. the leakage of information.
- From these the Dependability Policy with high level measures and principles is derived, e. g. multilevel security, fail safe concept. In addition, Dependability related functions are specified at this stage, e. g. access control mechanisms.
- A validation of whether the Dependability Objectives and the Dependability related functions reduces the occurrence of identified causes and/or the severity of consequences. This validation will also include the Dependability principles that are to be used in the system to achieve the Dependability Objectives, together with rationale for their choice showing how they will meet the Dependability Objectives.

Inputs

For the identification of the hazards the analyst(s), i.e. the Dependability team, needs either direct experience of, or access to persons with knowledge of the application field, and the history of dependable system developments in this field. The system needs are used to scope the analysis.

The standards, policies and constraints to be used in that application sector for the choice of dependability principles must be agreed upon.

In order to perform Hazard Rating, a set of rules on the acceptability of risks, if such exists for the application field or the expertise of a set of persons with sufficient sector and application knowledge to make decisions on risk issues is needed.

Description of the CPA

i. Identification of hazards

The range of hazards to be considered in the analysis must be defined. This definition must include both hazards which have their causes in the environment and those which have their causes in the system. Also the field of investigation should be limited by making assumptions about the anticipated faults to limit the need to consider all possible faults in the subsequent analysis. The analysis will link these faults to hazards and all anticipated failures of the system, concentrating on identifying those that may prove to be unacceptable.

On these conditions the first step of this CPA is to identify all possible hazards for the system to be developed, the corresponding faults and failures of these hazards. The scope of the analysis of hazards will be defined through the provision of the system needs, and the description of system boundaries, its interfaces to the environment and the interactions (through the interfaces) which can occur.

This identification of hazards is typically done workshops using application sector experts, with experience of previous or similar systems, and experts in the field of dependability. The activity to identify hazards must be carried out in a methodical way, using for example HAZOPS, checklists or a Step-by-Step-Method like the one in the [IT-SHB].

ii. Rating of hazards

The next step of the Preliminary Hazard Analysis is the rating of the identified hazards. For that the frequencies of the occurrence of the found faults have to be estimated. For the Preliminary Hazard Analysis it seems sufficient to relate these frequencies to classes, e. g. once a day, once a month. Also the probabilities that the hazards result in unacceptable consequences have to be estimated, again it is sufficient to do that in classes. There should also be a relation to the severity of damages, e. g. minor, major, critical, catastrophic, and to the aspects of damages, e.g. damages to persons, damages to equipment, financial loss.

With these estimations the risk of an identified hazard could be defined as the probable frequency of occurrence of this hazard causing a failure and the severity class of this failure. Often it should be sufficient to consider the failure with the highest severity caused by this hazard.

This grading aims to distinguish between acceptable and unacceptable hazards. The higher the rating of the consequences of the hazardous failure, and the higher the probability of its occurrence, the lower is its level of acceptability. Each application sector will typically have its own method for rating the acceptability of hazards, which will depend on laws, regulations or an assessment of public perception.

The unacceptable hazards are those which must be eliminated (prevented, avoided) completely by the system design or the system design must guarantee that the Dependability related functions reduces the consequences of these hazards. The acceptable hazards will each be allocated an acceptable level of occurrence (e.g. improbable, remote, frequent).

With this list of unacceptable hazards the Dependability Profile could be selected for the system according to the highest risk for each Dependability Attribute.

Remark: The combination of Hazard Identification and Hazard Rating is often called Risk Analysis. Many methods for risk analysis exist in the safety as well as the security sector, but there are great differences between the safety and the security sectors in the risk analysis approaches as well as the methods used. To avoid any confusion with this term, Hazard Identification and Hazard Rating is used in this document.

iii. Dependability Objectives

From the Dependability Profile the Dependability Objectives are determined which need to be satisfied by the system to remove the identified unacceptable hazards or reduce their consequences to an acceptable level.

These Dependability Objectives are defined in terms of general properties which the system must have, or may not have, e. g. the leakage of information.

iv. Dependability Policy and Dependability related functions

The Dependability Policy is derived from the Dependability Profile and the Dependability Objectives together with the standards, regulations and constraints of the system. Principles and high level measures are set up by which the previous found properties are achieved, e. g. use multilevel security.

Furthermore, the Dependability Objectives and the Dependability Policy result in the definition of dependability related functions and requirements, which have to be integrated in the system, e. g. access control mechanisms with further detailed specifications.

v. Validation

The last step of the Preliminary Hazard Analysis is to validate the results of the previous steps.

It must be assessed whether the principles of the Dependability Policy and the Dependability related functions satisfy the Dependability Objectives and the Dependability Profile established during the Hazard Rating.

It should also be validated that the Dependability Principles, i.e. the design guidelines and approaches that are generally accepted for that specific sector, which will be used in the system, leads to a reduction of the risks

If it was decided in the Hazard Rating to allocate the Dependability Objectives and the statements of the principles of the Dependability Policy to anticipated parts of the system (physical, personal, IT, organisation, subsystem, component), these decisions should be validated, again whether they lead to a reduction of risks.

The identification of the Dependability Related Functions, the definition of their functional requirements and the description of the relationship between each function and the Dependability Objectives should be considered.

Levels of Rigour, detail and independence

The Preliminary Hazard Analysis is performed with the same high level of rigour and detail whenever it is performed, as it provides results, which will determine the level of criticality of systems or sub-systems.

A Dependability team (person) together with a team consisting of members of the sponsor and the developer should do the PHA. It should not be only one person.

Methods and Tools

Data collection and analysis for PHA are in some cases supported by tools (e.g. Hypertext). The better known preliminary hazard analysis techniques, such as HAZOPS or the Step-by-Step-Method in the [IT-SHB], have some supporting tools, although care must be taken when using them that restrictive assumptions about the range of dependability attributes have not been made.

In fields where the techniques are well developed, some tool support for Hazard Rating is available. These techniques and tools are generally specific for the safety or security area and may even differ from sector to sector. Care has to be taken when selecting a technique and tool to ensure that all dependability attributes relevant for the system are covered.

For additional information see CPA "Hazard Analysis" (0).

Output

The results of the Hazard Identification will be a list of hazards, which are relevant for the TDA, together with an indication of their anticipated consequences.

The results of the Hazard Rating include:

- A set of rated hazards and Dependability Objectives for the system, consisting of objectives for the elimination of unacceptable hazards, objectives for the reduction of the frequency or consequences of acceptable hazards and a Dependability Profile for the system.
- A description of the Dependability Policy specifying the high level measures and principles by which the required Dependability Objectives will be achieved.
- A specification of the Dependability Related Functions to be implemented in the system.

The results of the Preliminary Hazard Analysis will be presented in a report including a detailed description of the analysis performed to validate whether the requirements in the Dependability Profile are fulfilled.

Use of the Results

The results of this analysis are used to provide an initial assessment of whether a system is subject to hazards, which will need special attention in its development or operational use. It may be possible theoretically to conclude at this stage the system is not a dependable system, but it seems unlikely that this will happen for a system where a Preliminary Hazard Analysis was decided to be done.

The Dependability Objectives and the Dependability Profile should be used to guide the design of the system. The Dependability Profile will also be used to define the assessment activities to be performed.

The validation of the Dependability Objectives together with the results of Hazard Identification and Hazard Rating, the Dependability Profile, the Dependability Policy and the Dependability related functions provide part of the Dependability Target of the system.

The validation of the Preliminary Hazard Identification and Hazard Rating activities may result in new hazards or differences in the rating of already identified hazards. As a consequence of this also the Dependability Profile may need to be adapted to these results.

Relation to System Lifecycle

This activity should be applied at the beginning of the Project, preferably in the concept exploration and definition phase but not later than the phase of demonstration and validation of concepts.

Usually, it is necessary to repeat a Hazard Analysis during the demonstration and validation phase.

A Hazard Analysis can be applied in any other phase when a re-examination of the basic assumptions about the dependability characteristics of the system is required.

For additional information see CPA "Hazard Analysis".

Assessor's activities

The assessor of the Preliminary Hazard Analysis has to check thoroughly whether the Analysis has performed adequately. Therefore he has to check the results and check the justification.

He has also to repeat parts of the analysis to validate whether the Dependability Profile is appropriately derived.

His result report should include a statement of whether the Dependability Objectives, the principles of the derived Dependability Policy and the Dependability related functions meet the requirements or not.

4.2.2.2 Hazard analysis

Name of the CPA

Hazard Analysis

Objectives

Hazard Analysis is to be applied in an iterative way at different system abstraction levels (function, subsystem, component, etc.). At each level, the objective is to identify potential failures and hazards likely to have an impact on the dependability related requirements assigned to that level, to analyse their causes and to estimate their consequences with respect to each dependability objective. For each hazard, the potential effects are determined and the criticality of the effect is rated.

Inputs

- Specification of the dependability related functions at the system decomposition level considered
- System boundaries and interfaces at that level
- Description of system architecture
- Results of hazard analyses performed at the higher levels

Outputs

- List of hazards with their rating for each item included in the boundary of the analysis
- Justification of the validity and representativity of assumptions
- Means for eliminating or reducing hazard effects
- Report of Hazard Analysis

Mapping to Confidence Levels

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 2	IL 3

Methods and Tools

HAZOPS, FMECA, FTA, ETA
(See also CPA “Preliminary Hazard Analysis”)

Objective

Hazard Analysis is to be applied in an iterative way at different system abstraction levels (function, subsystem, component, etc.), whereas the Preliminary Hazard Analysis CPA is to be applied at the initial stage of system development to define dependability objectives to be satisfied by the TDA. At each level, the objective of hazard analysis is to identify potential failures and hazards likely to have an impact on the dependability related requirements assigned to that level, to analyse their causes and to estimate their consequences with respect to each dependability objective. For each hazard, the potential effects are determined and the criticality of the effect is rated. As the design process progresses, the hazard analysis should be refined and updated. The impact of the new identified hazards on the hazard analysis performed at the higher level should be investigated. In particular, error propagation in both directions, from higher levels to lower levels and vice-versa, should be addressed. The assumptions considered during the hazard analysis process should be clearly stated to allow the validity of these assumptions to be analysed and assessed during further stages of the development. The application of the hazard analysis at different decomposition levels of the design leads to a hierarchy of fault and failure models. At each level of the decomposition, associated fault assumptions should be defined taking into account the fault assumptions established at the higher levels. Ensuring the consistency of these fault models is a difficult task that requires a thorough examination and in-depth analysis. In particular, it is essential to study how the faults that occur at the lower levels manifest and propagate to the higher levels. In particular, error propagation analysis is important for the specification and the design of error containment barriers and protection mechanisms.

Hazard analysis should cover the whole lifecycle, taking into account all the hazards that might occur during the development, installation, commissioning, operation & maintenance and decommissioning. It should be updated as the development of the system progresses. The results of hazard analysis should be re-examined each time the system requirements or its environment change. Several methods and techniques that can be used for hazard analysis are presented in the sequel.

Inputs

The inputs needed to perform hazard analyses include:

- the Dependability Target,
- the specification of the Dependability Related Functions to be accomplished at the system decomposition level considered,
- the description of the system boundaries and interfaces at that level,
- a detailed description of the architecture,
- and the results of the hazard analyses performed at the higher levels.

Description of the CPA

A hazard analysis activity is performed in several steps:

- Define the goals and the boundaries of the hazard analysis.

- Define a list of undesired hazard events for each item included in the boundary of the analysis and identify their potential local and global impacts on dependability.
- Identify the causes leading to each undesired hazard, specify the assumptions considered with respect to the classes of faults considered (nature, phase of occurrence, persistence, etc.), and justify these assumptions.
- Estimate the rate of occurrence of the undesired hazard and its level of acceptability.
- Analyse how the identified hazards affect the design of higher levels and how these hazards are handled by lower level design.
- Summarise the results of the hazard analysis activity.

Levels

Rigour

RL 1

Hazard analysis shall be performed informally. The results shall also be described informally.

RL 2

Hazard analysis shall be performed using well defined hazard analysis plan and methodology. The results shall be strongly justified.

RL 3

Hazard analysis shall be performed using a formally defined hazard analysis plan and methodology assisted by an appropriate tool. The results shall be described in detail and be justified by examples from similar systems.

Detail

DL 1

Hazard analysis shall be applied in the architectural design phase. It must also be done if there is a significant change in the system requirements/environment, etc.

DL 2

Hazard analysis shall be applied in all design stages down to the detailed design. Any significant change in the system requirements, environment should lead to the re-examination of hazard analysis results at each step of the development process.

Independence

IL 1

Hazard analysis shall be performed by the developers within the development team.

IL 2

Hazard analysis shall be performed by a team not directly involved in the development of the system to be assessed.

IL 3

Hazard analysis shall be performed by a team independent from the organisation(s) involved in the development of the system to be assessed. All members of the team shall have significant experience with systems similar to the one that is assessed.

Definition of Confidence Levels for Hazard Analysis

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 2	IL 3

Table 2: Definition of Confidence Levels for Hazard Analysis

Output

A Hazard Analysis report should be provided to summarise the scope and the results of the activity. In particular, a list of hazards for each item included in the boundary of the analysis, with a description of their consequences, with respect to each dependability related requirement and to the system dependability objectives, should be detailed. The fault and failure assumptions established during the analysis should be summarised and justified. A summary of design related considerations should be given to explain how the identified hazards are handled by the system.

Use of the Results

The results of the hazard analysis should be used to update the design and to refine the dependability requirements as the design progresses. The results should also be used to define verification and testing scenarios to analyse how the identified hazards are handled by the implementation of the system. The results of the hazard analysis should be used for the development of operation, maintenance, and decommissioning procedures.

Relation to the System Life Cycle

Hazard analysis is an essential activity that can be applied during all design stages of TDA. The previous paragraphs, under levels of Detail, identify when the activity must be applied but this does not prevent the developer from using it at additional points in the development.

Assessor's activities

The primary role of the assessor is to check the accuracy, correctness and completeness of the hazard analysis through thorough examination and in depth analysis of the justifications and evidence provided. The assessor should make his own hazard analysis for the most critical parts of the system to ensure that some known hazards have not been overlooked or ignored. The assessor should check the validity of the assumptions established during the hazard analysis to ensure that these assumptions correspond to what might really happen in operation and that the system as implemented provides a good coverage of the faults and errors assumed in the design process. Moreover, due to the antagonism that exists between some dependability attributes, e.g. safety and availability, confidentiality and availability, the assessor should check that the system as implemented provide a right balance and a satisfactory trade-off between the these attributes.

Methods and Tools

Several methods and techniques can be used to perform hazard analyses. The most popular ones are presented in the sequel.

Failure Modes, Effects and Criticality Analysis (FMECA)

A Failure Modes, Effects and Criticality Analysis (FMECA) is a systematic method aimed at identifying the failure modes of a function, system, component, etc. and determining their effects on the system, and the criticality of these effects. The FMECA can be applied in all stages of the system construction process. Usually, FMECA relies on the functional structure of the system and makes it possible to highlight design weaknesses "if any" in the system with respect to dependability. It can address all design stages or even an operational system but it is often preferable to apply it as soon as possible during the life cycle to minimise the costs of design modifications (if any). However, its use at an early stage in the development process is rather delicate, as the lack of precise knowledge of the real structure of the system and of the failure modes, etc. becomes a major hurdle. Sometimes FMECA is applied without rating the criticality of the effects. Then the method is called "Failure Mode and Effects Analysis" (FMEA).

The general principle used for the application of the method consists in including the following items in a table — after listing the various failure modes based on the functional or structural description of the system—, and for each failure mode of each component:

- its possible causes,
- its effect on the behaviour of the component itself (local effect), or at system level (global effect),
- the means of detection that can be employed,

- corrective actions to be implemented, more particularly, when dealing with catastrophic failure modes,
- criticality of the failure mode: this aspect is not always taken into account.

When iterating the application of the method on a sub-component, the omission of certain failure modes of the component to which it belongs may be revealed. Indeed, the failure modes of the higher level component appear as combinations of sub-components' failure modes. In particular, the latter having global effects must necessarily correspond to failure modes of the higher level component.

The tables summarising FMECA results are useful for the design, as they support the selection of dependability related mechanisms and permit the early detection (and modification) of several design shortcoming. They equally provide a valuable support for system verification and testing, as they highlight the critical components requiring in depth analysis and validation. Moreover, FMECA provides valuable inputs for other hazard analysis and dependability validation activities, e.g., fault tree analyses, quantitative evaluation method. Nevertheless, the FMECA approach features some limitations:

- it does not consider multiple failures;
- it calls for a great deal of information to be handled, particularly in the case of complex systems.

Fault Tree Analysis (FTA)

Fault Tree Analysis (FTA) is a deductive failure analysis, which focuses on one particular undesired event and provides a method for determining causes of this event. The analyst begins with an undesired top-level hazard event and systematically determines all single faults and failure combinations of the system components at the next lower level, which could cause this event. The analysis proceeds down through successively more detailed levels until a basic elementary event is uncovered. An event can be regarded as elementary when it is either independent of the others or when its probability of occurrence can be estimated, or simply when one does not intend to, or cannot, break it down further. An elementary event may be internal or external to the system analysed. It may correspond to a hardware component failure, a human error, etc.

A Fault Tree is generally represented graphically as a logical diagram with “AND” and “OR” gates connecting the set of events described in the tree. Fault trees can also be used to calculate the probability of occurrence of a hazard. The evaluation of the probability associated with the minimal cut-set derived from the tree allows the identification of the critical scenarios of events leading to the undesired hazard and the selection of appropriate design decisions to handle these events. Some simple tools exist, which help to perform the analysis in a systematic way. But expert knowledge is always needed to perform a fault tree analysis.

The Fault Tree analysis typically complements the FMEA. Thus, combinations of failures, which had been overlooked by FMEA can be considered. Fault Tree Analyses can be applied during all stages of the system development process. In the early phases, they provide useful support for the definition of the dependability-related functions and mechanisms that need to be implement to fulfil the dependability objectives. As the design proceeds and new hazard are identified, the fault tree analysis should updated and applied at a level of detail corresponding to the level of refinement of the design.

Fault tree analysis has the following advantages:

- It can reveal multiple failure sequences involving different parts of the system
- Fault trees can examine the entire system
- A partial fault tree is still a useful result
- It is a systematic approach

The disadvantages are:

- It is dependent on the completeness of the hazard analysis
- It is not practical for systems with a large number of identified hazards
- Depends on the knowledge of the analyser
- Fault trees may become very large and complex

The output of the fault tree analysis shall be used for hazard rating as well as for the identification of critical components. If the fault tree analysis reveals an unacceptable high rating for an identified hazard, changes to the system design have to be made to either reduce the probability of a hazard or limit its effects.

Event Tree Analysis (ETA)

Event Tree Analysis consists in describing, generally in a diagrammatic form, the sequence of events that can develop in a system after an initiating event occurs, and thereby indicate how serious consequences can occur. The initiating event could be a human error, a hardware component failure, etc. In contrast to Fault Tree Analysis, Event Tree Analysis starts from an initiating event and then identifies a sequence of system faults and errors that define an accident scenario. The initiating event is the root of the event tree. Starting from this event two lines are drawn: one with a positive consequence, the other with the negative consequence. This is then done for each subsequent consequence until all consequences are considered. Similarly to Fault Trees, probabilistic quantitative evaluations can be performed based on Event Trees. For each consequence, a probability value can be computed. Thus, the overall probability of each accident or failure scenario resulting from an initial event can be determined.

Event Tree Analysis can be applied in all phases of the system lifecycle.

Hazard and Operability Studies (HAZOPS)

The aim of Hazard and Operability Studies (HAZOPS) is to establish, through a series of systematic examinations of system components and their operation, failure modes likely to result in potentially hazardous situations in the system environment. HAZOPS have been traditionally used to analyse control command systems used in the nuclear and chemical fields. Typical hazardous events on the controlled systems are fire, explosion, release of toxic material or serious economic loss. However, HAZOPS can be extended and used in other application sectors to address security as well as safety related objectives.

HAZOPS are usually performed, in a series of scheduled meetings, by a team of persons with multidisciplinary engineering competencies, led by a trained specialist in hazard analysis. The success of HAZOPS is highly dependent on the knowledge of the assessors on the type of system to be assessed and on the application field. When forming a team for HAZOPS a good mix of knowledge on the system itself and its environment as well as on potential hazards and their consequences for similar systems should be accomplished.

Prior to the study, agreed checklists are compiled for the systematic examination of each system component to be analysed. Questions such as: What if it happens? How can it happen? When can it happen? Does it matter? Are asked. When positive answers are given further questions are asked, such as: What can be done about it? When must it done? Is there an alternative? etc. The results are usually written down in form of a table or a diagram. Clearly, HAZOPS are similar, to some extent, to FMEA. While FMEA is generally oriented on the failure modes of components, HAZOPS are oriented on the functions of process sections, which are systematically investigated using the checklists.

The result of HAZOPS is a list of potential problems which either will be removed by changes in the design or which have to be analysed in more detail during the later implementation and operation phases.

HAZOPS can be applied during all the life cycle stages. At the early design stages, HAZOPS are particularly useful to perform for preliminary hazard analyses leading to the definition of dependability related requirements.

4.2.2.3 Common Cause Analysis

Name of the CPA

Common Cause Analysis

Objectives

Identify single points of failure that may cause several critical components to fail. Such failures, which bypass or invalidate independence assumptions, are called “Common Mode Failures”.

Inputs

- Dependability Target, Dependability Objectives and Dependability Related Functions
- Independence assumptions
- Description of TDA architecture
- System components hazard analyses
- Deep knowledge of development, operation, maintenance, and decommissioning processes

Outputs

- List of independent parts of the system and their interfaces
- List of failures and common causes that may have impacts on several parts of the system
- Validation of independence assumptions
- Report of Common Cause Analysis

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 2	DL 2	IL 2
Level 4	RL 2	DL 2	IL 3

Methods and Tools

Zonal analysis, Environment related common cause analysis and process-related common mode analysis rely on engineering judgement and are supported by other hazard analysis methods (FMECA, FTA, etc.)

(See “Hazard Analysis” (4.2.2.2) and “Probabilistic Quantitative Evaluation” (0) CPAs)

Objective

System Architectures that rely on redundancy or that use components or software in common with other systems suffer from the potential of single failures that may cause several critical components to fail. Such failures, which bypass or invalidate redundancy or independence assumptions, are called “Common Mode Failures”. It is the objective of the Common Cause Analysis to identify these types of failures and their causes.

Common Cause Analysis shall be used for all systems where a single point of failure has to be avoided or where redundancy is used to reduce the probability of some critical failures. Also, Common Cause Analysis aims to identify Common Mode Failures caused by external events from the environment, and those related to design faults and to the engineering processes used for system development, operation, maintenance and decommissioning (e.g. failures that may result from a common manufacturing process of different parts of the system).

Inputs

Common Cause Analysis requires a detailed description of the architecture of TDA and associated dependability requirements. Inputs from hazard analysis methods (e.g. FMECA, FTA) are also needed to perform the common cause analysis. Identification of process related common mode failures requires a deep knowledge of the development, operation, maintenance, installation and system disposal processes. A detailed description of the system environment is also needed to be able to identify the external events that might lead to common mode failures.

Description of the CPA

The Common Cause Analysis can be sub-divided into three sub-activities:

a) Zonal Analysis
b) Environment-related Common Cause Analysis
c) Process related Common Mode Analysis

Zonal Analysis (generally called “Zonal Safety Analysis” in avionics) examines each physical zone of the system to ensure that equipment installation and potential physical interference with adjacent systems do not violate the independence requirements. An important aspect of the zonal analysis is the identification of interfaces and interference with other parts of the system. Failures identified for the part have to be assessed for their potential impacts on all other parts of the system.

Environment Related Causes Analysis identifies events occurring outside the system boundaries, which may violate the independence claims made for the parts of the system to be analysed. Environment Related Common Cause Analysis may have impacts on several parts of the system and may therefore not be identified by the Zonal Analysis. Examples of such events are: 1) fire, flood and other natural disasters, 2) high intensity radiation, 3) loss of power supply, 4) loss of network connections, etc.

Process related Common Mode Analysis is concerned with verification of the independence claims made in the system design. It is aimed at identifying problem areas, where (sometimes hidden) dependencies may lead to common failures in critical devices. This includes but is not limited to: 1) common type of equipment or technologies, 2) common development, manufacturing or installation process, 3) common maintenance procedures or personnel, 4) common assessment activities or procedures

Having identified those problems areas, a checklist of potential sources for common modes is derived and potential failure lists are developed and considered for their potential impact on the system.

Levels

Rigour

RL 1

The Common Cause Analysis is performed informally on the basis of the system design, its environment and the anticipated knowledge of engineering process. This analysis should be supported by checklists and based on experiences from similar systems and projects. The Common Cause Analysis should confirm that the system partitioning is compliant with the dependability objectives and the Dependability Profile.

RL 2

The Common Cause Analysis is performed based on a detailed analysis plan. The interfaces between the different parts (zones) are systematically analysed. The Environment-related Causes Analysis is performed based on a carefully derived list of external events. Arguments are given which justify the completeness of this list. Failure models are used to describe the impact of a failure and the parts affected by a failure are identified completely. The Common Mode Analysis is based on detailed checklists for Common Mode failure types and failure sources. The parts of the system affected by a Common Mode failure are completely identified and justification is given, why the other parts are not affected. The Common Cause Analysis should confirm that the system partitioning into fault independence and error containment regions is compliant with the dependability objectives and the Dependability Profile. Detailed information should be provided to describe how design faults are addressed.

Detail

DL 1

The activities of the Common Cause Analysis are performed during the requirements and architectural design phases. The identification of Common Mode failures should also account for operation, maintenance and decommissioning procedures.

DL 2

The activities of the Common Cause Analysis are performed during the requirements, architectural design and detailed design phases. They should cover the whole life cycle of the systems including installation, operation, maintenance and decommissioning phases. At each stage of the design process, detailed analysis should be performed to check the validity of the independence assumptions and to ensure that design decisions are compliant with the Dependability objectives and the Dependability Profile.

Independence

IL 1

The Common Cause Analysis shall be performed by the developers within the development team.

IL 2

The Common Cause Analysis shall be performed by a team not directly involved in the development of the system to be assessed.

IL 3

The Common Cause Analysis shall be performed by a team independent from the organisation(s) involved in the development of the system to be assessed. All members of the team shall have significant experience with systems similar to the one that is assessed.

Definition of Confidence Levels for Common Cause Analysis

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 2	DL 2	IL 2
Level 4	RL 2	DL 2	IL 3

Table 3: Definition of Confidence Levels for Common Cause Analysis

Methods and Tools

Common Cause Analysis activities rely engineering judgement and on the use of other basic hazard analysis methods, which are also listed in this chapter (see "Hazard Analysis" CPA, 4.2.2.2).

Output

The result of the Common Cause Analysis is the sum of the results from the three sub-activities. The result of the Zonal Analysis consists of:

a) A list of widely independent parts (zones) of the system.
b) A list of interfaces and remaining dependencies between the parts.

c) A list of failures of the individual parts that may have impacts on other parts of the system. The failure modes and effects are also described.

The results of the Environment related Causes Analysis consists of:

a) A description of the analysed environment related hazards.

b) A list of the parts of the system affected by these hazards.

c) A description of the failure modes caused by these hazards as well as a description of its effect.

d) A description of the deviation to the initial assumptions and the implication of this deviation.

The results of the Process related Common Mode Analysis consists of a list of Common Mode failures and their effects.

Use of the results

The results of the Common Cause Analysis are used to change the system design in a way that either eliminates the Common Mode Failures detected by the analysis or reduces the effect of such a failure to an acceptable level. The failures identified by this analysis method have to be either rated as acceptable or have to be eliminated.

Relation to System Lifecycle

Common Cause Analysis can be applied in all stages of the system lifecycle but is most useful and cost-effective when applied early in the design process because of the potential influence on system architecture. Therefore, appropriate specification or design changes can be implemented early to correct the problems identified by the analysis method. Note that this CPA can also be use as a Dependability Validation CPA when applied to the system implementation.

Assessor's activities

The assessor should thoroughly examine the correctness of Common Cause Analyses performed and analyse their completeness. He should ensure that the results of Common Cause Analyses have been taken into account in the design. Particularly, he should examine to what extent the measures used to prevent process related common mode failures are satisfactory.

4.2.2.4 Probabilistic Quantitative Evaluation

Name of the CPA Probabilistic Quantitative Evaluation	
Objectives Quantify the impact of faults, errors and failure on the behaviour of the system and support the selection of potential architectures that allow the dependability objectives to be satisfied. This is done by building a stochastic model describing the system behaviour in the presence of faults and evaluating quantitative measures from the model.	
Inputs • Specification of system behaviour to be described in the model • Quantitative measures to be evaluated • Results of hazard analyses • Reliability data to specify the values of the model parameters	***Outputs*** ▪ Quantitative measures evaluated from the model and sensitivity analysis results ▪ List of most critical parameters to be monitored during the system life cycle for data collection and system analysis ▪ Feedback on the design ▪ Report of Probabilistic Quantitative Evaluation
Mapping to Confidence Level	
Methods and Tools Markov models, Reliability diagrams, Fault Trees , Stochastic Petri nets and their offspring, Monte-Carlo Simulation	

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 3	IL 3

Objective

This activity consists in building a stochastic model (generally a Markov model) describing the occurrence of hardware and software components failures and repairs and their consequences on the behaviour of the global system. To some extent, the impact of human errors on the system behaviour can also be included in probabilistic evaluations. Quantitative measures are evaluated from the model and compared to the dependability objectives. This activity supports, in particular, the selection of potential architectures that allow the dependability objectives to be satisfied. Verification of the validity of the predictions derived from the models with respect to the behaviour of the real system will be performed within Dependability Validation process activities (see “Experimental Evaluation” CPA). Moreover, preventive maintenance actions and periodic tests can be planned based on quantitative evaluations. However, quantitative evaluations can not be applied for the assessment of all the dependability attributes, and the confidence in the results obtained depends on the validity of the assumptions and also on the dependability levels to be achieved. In fact, quantitative evaluation of highly critical software is unfeasible in the current state of the art. Moreover, quantitative evaluation of operational security is not yet mature and it is still a research topic.

Inputs

The development of the stochastic model used to evaluate the specified quantitative measures (mean time to failure, failure rate, availability, etc.) is based on the set of hazards and failures derived from the hazard analysis. To be able to define detailed models of the architecture, information from the specification and design documents are needed. Moreover, to specify the components failure and repair rates used as parameters in the model, reliability data should be collected from similar systems, or be derived from reliability handbooks such as MIL-HDBK handbook [DoD 65] or CNET Handbook [CNET 93] for electronic components,.

Description of the CPA

Three major steps can be distinguished for performing a Probabilistic Quantitative evaluation:

a) Choice of the quantitative measures to be evaluated and definition of the assumptions to be considered.

b) Construction of a stochastic model describing the system behaviour at a level of detail allowing the selected quantitative measures to be evaluated in accordance with the assumptions considered.

c) Processing of the model and analysis of the results. This last step includes an assessment of the sensitivity of the results to the values of the parameters and to the assumptions considered in the models.

Levels

Rigour

RL 1

The objectives and the quantitative evaluation results of analysis shall be stated and the methods and tools used to achieve these objectives shall be described.

RL 2

The objectives of the probabilistic quantitative evaluation shall be described together with the models used. The assumptions made shall be explained and justified based on the results of the hazard analysis. Finally, the methods and tools used to achieve the objectives shall be justified and the results obtained shall be explained and correspondence between the objectives and the results shall be demonstrated.

RL 3

Besides the requirements corresponding to RL 2, the probabilistic quantitative evaluation shall address the impact of different classes of faults (permanent faults, transient faults, etc.) and the impact of the fault tolerance coverage on the dependability objectives. The techniques used to obtain coverage estimates shall be described and justified. Sensitivity studies shall be performed to analyse the impact of the parameters and the distributions used in the models on the dependability objectives. Service experience and experimental evaluation shall be used, when applicable, to justify the values assigned to the parameters.

Finally the validity of the evaluation tools used shall be assessed.

Detail

DL 1

Probabilistic quantitative evaluation shall be used during the high-level design stage to support design decisions.

DL 2

Probabilistic quantitative evaluation shall be used during the high-level design stage and detailed design stages to support design decisions. The evaluation shall be continuously updated.

DL 3

Probabilistic quantitative evaluation shall be used during the high-level design stage and detailed design stages to support design decisions. The evaluation shall be continuously updated. Evaluation of the efficiency of fault tolerance mechanisms and error recovery procedures should be considered using simulation-based fault injection experiments.

Independence

IL 1

Probabilistic quantitative evaluation shall be performed by independent persons.

IL 2

Probabilistic quantitative evaluation shall be performed by an independent department.

IL 3

Probabilistic quantitative evaluation shall be performed an independent organisation.

Definition of Confidence Levels for Probabilistic Quantitative Evaluation

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 3	IL 3

Table 4: Confidence Levels for Probabilistic Quantitative Evaluation

Methods and Tools

The methods and tools used for deriving dependability quantitative measures depend on the dependability objectives to be quantified, the modelling assumptions considered, and the life cycle stage where the evaluations are performed. Two classes of techniques can be distinguished:

- *Modelling based evaluations*: when an abstract model is built and analytical or Monte-Carlo simulation is used to process the model and evaluate the dependability measures.
- *Experimental evaluation*: when the system is exercised with real inputs, data is collected and processed by statistical techniques to evaluate the dependability measures. This data can be collected during fault injection experiments performed on a real prototype, or during system testing in its operational environment, or during system operation.

These two classes of techniques are complementary. In fact, modelling leads to the definition of parameters characterising the stochastic processes (failures, repairs, coverage, etc.) described in the model. These parameters can be estimated based on the data collected during the experimental evaluation. On the other hand, modelling can be used to define the most important parameters for which detailed experiments are needed to provide realistic estimations (for instance coverage, latency, etc.).

Experimental evaluation is based on the implementation of the system. Therefore, experimental evaluation methods and techniques are discussed in Dependability Validation Process CPAs (see Section 4.4.2.3).

Analytical modelling and Monte-Carlo simulations can be used during the early stages of the design to evaluate several alternative architectures and select the one best fitting the dependability objectives. Several evaluation techniques are available including: combinatorial model types such as Reliability Block Diagrams, fault trees, reliability graphs, etc. and Markov model types such as continuous-time Markov chains and their extensions, Generalised stochastic Petri nets, Markov reward models, etc.

The choice between these methods depends on several criteria: the kind of system behaviour to be modelled, the measures to be computed, the modelling power, etc. For instance, Markov models can handle some dependencies, in a system, which combinatorial models cannot.

Several modelling and evaluation tools are commercially available for analysing fault tolerant computing architectures, for instance SURF2 [Béounes et al. 1993], SHARP [Sahner et al. 1996], ULTRASAN [Sanders et al. 1995], SPNP [Ciardo et al. 1989], etc.

Output

The results obtained from probabilistic quantitative evaluation activities should be documented and the assumptions considered during the modelling should be justified and supported by data collected from previous experience with similar systems. Sensitivity analyses should be performed to show the impact of various parameters on the dependability objectives

Use of the Results

The outputs of probabilistic quantitative evaluation are helpful for guiding the design process and controlling the dependability level achieved by the system. The verification of the results obtained is therefore important to analyse the validity of the design decisions, and to ensure that the dependability level achieved before operation is satisfactory. If the results are not satisfactory, modification of the system design is required.

Relation to System Lifecycle

Probabilistic quantitative evaluation is to be applied during the early design stages to support architecture design decisions. The evaluations should be updated when data is collected during testing (see “Experimental Evaluation” Dependability Validation CPA, Section 4.4.2.3).

Assessor’s activities

The main role of the assessors is to examine:

a) the representativity and the validity of the assumptions considered,

b) the validity of the methods used and the models derived for conducting the evaluations,

c) the results of evaluation to check whether the objectives are satisfied.

4.2.2.5 Dependability Requirements Validation Activities: Summary

Table 5 summarises the mapping between dependability requirements validation activities and the confidence levels.

Dependability Requirements Validation Activities	Level 1	Level 2	Level 3	Level 4
Preliminary Hazard Analysis	Same level of rigour, detail and independence			
Hazard Analysis	RL1, DL1, IL1	RL2, DL1, IL2	RL3, DL2, IL2	RL3, DL3, IL3
Common Cause Analysis	RL1, DL1, IL1	RL2, DL1, IL2	RL2, DL2, IL2	RL2, DL2, IL3
Probabilistic Quantitative Evaluation	RL1, DL1, IL1	RL2, DL1, IL1	RL3, DL2, IL2	RL3, DL3, IL3

Table 5: Relation between Dependability Requirements Validation Activities and Confidence Levels

Among these activities, Preliminary Hazard Analysis should be applied for any system irrespective of the confidence level required. Indeed, the determination of the confidence level of the TDA and the corresponding dependability objectives is the result of this activity. Therefore, Preliminary Hazard Analysis must be performed with the highest level of rigour and detail to ensure that the confidence level assigned to the TDA satisfies the initial needs. The selection among the other dependability requirements validation confidence providing activities and methods depends on the dependability attributes of the system, the life cycle phases that have to be addressed by the evaluation and the target confidence level. In addition, the selection is influenced by sector specific or system specific needs. In particular, some factors such as the system complexity, the novelty of the design, the types of technologies used, and the availability of experimental data on similar products and systems, should be considered to find a right mix of the validation activities to be used, and corresponding levels of rigour, detail and independence. Moreover, alternative validation methods, which are not included in the Criteria, can be used, provided that convincing arguments and justifications are given to demonstrate the suitability and efficiency of these methods. When alternative techniques or alternative combination of levels of rigour, detail and independence are specified, the dependability target document shall specify, which one of the alternative techniques shall be used and which alternative has to be taken in the definition of the rigour, detail and independence levels. The choice may be restricted by the demand to be compliant with a sector specific standard.

Besides the activities reported in Table 5, Penetration Analysis and Covert Channel Analysis categorised as Dependability Validation CPAs (see Sections 4.4.2.1 and 4.4.2.2) can also be applied during the design stages for Dependability Requirements Validation.

4.3 Correctness Verification

4.3.1 Motivation and scope

The primary purpose of Correctness Verification confidence providing activities is to ensure that the system has been constructed correctly based on the validated requirements. These activities should address the way the system is constructed, how it will be operated and maintained, and (for some systems) how it will be disposed from operation.

Correctness Verification confidence providing activities correspond to the traditional verification activities carried out during the construction process to check that each system decomposition level (from the definition of the high-level requirements up to the realisation of the physical components and the integration of the final architecture) has been correctly implemented and meets its specified requirements. These activities can also be performed during system installation, operation, maintenance, and disposal to ensure that the procedures performed during these phases are compliant with the ones that have been specified and validated, and satisfy the system requirements and dependability objectives.

During the requirements and design refinement process, the objectives to be satisfied by Correctness Verification confidence providing activities at each level of this process are:

- Compliance with high-level requirements
- Accuracy (i.e. absence of ambiguity) and consistency (i.e., the corresponding requirements do not conflict with each other)
- Traceability to high-level requirements
- Verifiability (i.e., each stated requirement should be verifiable).

During the implementation and integration phases, the objective of the Correctness Verification confidence providing activities is to ensure that each level of the implementation satisfies its validated requirements.

Correctness Verification activities should particularly ensure that the faults and errors assumptions are correctly handled by the implementation of the dependability related functions and mechanisms. Special attention should be put on the verification of the correct refinement of faults and error assumptions and the correct integration of all dependability related functions and mechanisms implemented in the system.

Correctness Verification activities are to be applied also to check the inputs and the results of the requirements validation and system validation confidence providing processes, as well as those of the verification process itself.

Correctness Verification objectives are satisfied through a combination of inspections, reviews, analyses, model checking, the execution of test cases and procedures, … applied in accordance with a Verification Plan. The verification assumptions for each verification technique used should be clearly stated and justified. The verification scenarios and results should be also checked for accuracy, completeness and traceability to the system requirements.

For Correctness Verification to be practical, every stated requirement should be verifiable. Verifiability is a function of the way the system is designed and implemented. The verifiability constraint imposes that the developer specifies how the system requirement will be verified and provides evidence that the requirements have been satisfied once the actual system is delivered for evaluation.

Correctness Verification activities must be synchronised with the system life cycle phases. The Criteria do not prescribe a specific flow of activities within these phases. Therefore, the Criteria should be applicable for almost any lifecycle model used for construction, operation and maintenance or system decommissioning. The outputs of each life cycle phase must be verified for completeness, consistency and conformance to the standards. During these phases, the way the system will be operated, maintained and disposed from operation should be addressed and verified.

4.3.2 Correctness Verification Objectives and Lifecycle stages

For each life cycle stage, we define a set of objectives to be covered by Correctness Verification confidence providing activities.

4.3.2.1 Requirements Verification

At this stage, verification activities focus on the completeness, accuracy and consistency of the high level requirements. Clearly, there are close interactions between Correctness Verification activities and requirement validation activities. The latter focus on dependability related requirements whereas Correctness Verification activities have a larger scope as they address the whole set of requirements defined for the system to be evaluated.

The objectives to be satisfied are the following:

- Dependability related functions requirements comply with dependability objectives, are unambiguous and consistent.
- Dependability related functions requirements are traceable to dependability objectives.
- Dependability Related functions requirements are verifiable.
- Dependability Related functions requirements conform to standards.

4.3.2.2 Design Verification

At this stage, verification activities focus on the completeness, accuracy and consistency of the low-level requirements of dependability related functions and the corresponding architecture. The traceability of Low Level Requirements to High Level Requirements is an important issue that shall be addressed by verification activities. The objectives to be satisfied are the following:

- Low Level Requirements comply with High Level Requirements: Low Level Requirements satisfy High Level Requirements and the design basis for their existence, are correctly defined.
- Low Level Requirements are unambiguous and consistent: each Low Level Requirement is unambiguous and Low Level Requirements do not conflict with each other.
- Low Level Requirements are traceable to High Level Requirements: each Low Level Requirement is traceable to High Level Requirements.
- The definition of error containment and fault handling boundaries is correct: the partitioning is consistent with the fault independence assumptions.
- Low Level Requirements are verifiable.

- Low Level Requirements conform to standards.
- Architecture is compatible with High Level Requirements and is consistent.
- Architecture is verifiable.
- Architecture conforms to standards.

4.3.2.3 Implementation Verification

At this stage, verification activities address the implementation of each component of the architecture. The objectives to be satisfied are the following:

- Component implementation complies with Low Level Requirements and the architecture
- Implementation of protection and fault tolerance mechanisms is accurate and consistent
- Component implementation is verifiable
- Component implementation conforms to standards

4.3.2.4 Integration Verification

At this stage, verification activities address the integration of system components in relation to the architecture and the integration of the system into its operational environment. The objectives to be satisfied are the following:

- Integration complies with the architecture
- Integrated system complies with High Level Requirements
- Integrated system complies with Low Level Requirements
- Protection and fault tolerance mechanisms are compatible

4.3.2.5 Operation, Maintenance and Decommissioning Procedures Verification

The objectives to be satisfied are the following:

- The procedures comply with High Level Requirements
- The procedures are consistent and unambiguous

4.3.3 Correctness Verification Planning

The purpose of Correctness Verification planning is to define the processes and criteria to be applied for the verification of each item to achieve the verification objectives. The verification plan shall include the following information:

- Identification of the system or item configuration to be verified and the corresponding requirements
- Definition of the specific verification methods, tools and environments, to be employed to show compliance with the verification objectives, based on the confidence level.
- Definition of the criteria to be used to assess the evidence resulting from each verification method.

- Identification of roles, responsibilities, and independence of persons performing the verification activities

4.3.4 Assessors Activities

Correctness Verification confidence providing activities should be assessed by the assessors for adequacy, completeness, accuracy, and efficiency. Therefore, the assessors must analyse and check the evidence and justification provided about the verification means (methods and tools) used, the verification cases and procedures applied, and the results obtained from the verification activities. At their discretion, the assessors may also complement these activities by performing additional verifications, or for instance by submitting test cases to the developer who will have to run them and produce the proof of their success.

The assessors shall ensure that the verifications performed meet the requirements of the standards, are consistent with the verification plan, and are justifiable for the dependability confidence level.

4.3.5 Correctness Verification Confidence Providing Activities

This section lists a set of Correctness Verification confidence providing activities to be performed to satisfy the verification objectives. These are classified as follows:

- Static analyses including inspections, walkthroughs, and reviews
- Testing
- Formal methods and proof-of-correctness
- Behavioural analysis
- Traceability analyses

The inputs required for these verification activities will depend on the lifecycle stage, the level of detail, the level of rigor, and the level of independence.

4.3.5.1 Static analyses: inspections, reviews, walkthroughs

Name of the CPA

Static Analyses: inspections, reviews, walkthroughs

Objectives

Static analysis involves the verification of the system without executing it. Static analysis can be performed at any stage of the development process and can be applied to every working product dealing with the construction, operation and maintenance of the system. The purpose of static analyses is to find the faults that may have been introduced in the specification, design or implementation of the system by a thorough analysis of these documents.

Inputs	***Outputs***
The inputs required for static analyses are the products that have to be scrutinised and the corresponding specifications	A verification report should be provided including a summary of the verifications performed, the list faults or critical issues identified, and suggestions for improvements or corrections.

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL 1
Level 2	RL 2	DL 2	IL 1
Level 3	RL 2	DL 3	IL 2
Level 4	RL 3	DL 3	IL 3

Methods and Tools

Inspections, reviews, walkthroughs

Objective

Static analysis techniques involve the verification of the system without executing it. Static analysis can be performed at any stage of the development process and can be applied to every working product dealing with the construction, operation and maintenance of the system. The purpose of static analyses is to find the faults that may have been introduced in the specification, design or implementation of the system by a thorough analysis of these documents.

The effectiveness of static analysis has been verified through several real case studies. Inspections, reviews and walkthroughs have a wide range applicability, they can be used to check hardware, software and any other related workproducts produced during the development, including test cases and procedures and verification results. However, static analyses are not appropriate for the verification of the dynamic behaviour of the system, including synchronisation, concurrency, real-time constraints, etc. These aspects have to be addressed by other verification techniques.

Inputs

The inputs required for static analyses are the products that have to be scrutinised. These products could be a specification document, a design document, an implementation, a verification plan, tests cases, etc. Static verification consists in checking that the corresponding product satisfies its objectives and requirements. These requirements should be clearly stated for each static analysis activity.

Description of the CPA

Static analysis can be done manually or can be supported by automatic tools. Static analysis techniques include inspections, reviews and walkthroughs of the system products produced at different stages of the development process. The verification process is generally guided by checklists. Inspections are conducted by small groups of peers who examine manually the developing product, a piece at a time, to ensure that it is correct, and conforms to products specifications and requirements. The purpose of these inspections is the detection (and subsequent correction) of faults. A checklist of likely mistakes may be used to guide the fault finding process. Unlike, reviews and walkthroughs, inspections are closely controlled by a predefined formal process. Reviews and walkthroughs are typically peer group discussion activities without much focus on fault identification. Walkthroughs are generally a training process, and focus on learning about a single document. Reviews focus more on consensus and buy-in to a particular document [Gilb and Graham 1993].

Besides manual static analyses supported by checklists, some limited analyses can be applied automatically by static analysis tools. This process can be applied to documents written in a specific language, for instance, a software program, a formally specified design document, etc. In this case, syntactic errors, violation of coding rules, etc. can be easily detected.

Levels

Rigour

RL 1

Static analyses can be limited to informal reviews and walkthroughs. The results of the reviews shall be documented. All planned reviews defined in the verification plan have to be done.

RL 2

Static analyses shall follow a systematic formal inspection approach. The inspection process shall follow a specified series of steps that define what is inspected, who inspects it, what preparation is needed for the inspection, how the inspection was conducted, what data was collected, and what the follow-up to the inspection was. The members of the team shall bring different engineering disciplines and different insights to identifying faults. The results of inspections shall be explained.

The faults identified during this process must be corrected and the corrections shall be reviewed to ensure that the discovered faults were eliminated. If some corrections are not implemented, justifications must be given.

RL 3

Besides the requirements corresponding to RL2, automatic static analysis tools shall be used.

The inspection process shall be guided by a statistical quality improvement process, which allows the control of the inspection process efficiency. The inspection meeting duration and the number of documents reviewed shall be limited [Dyer 1992, Humphrey 1989]

Detail

DL 1

Static analyses shall be used during the requirements phase

DL 2

Static analyses shall be applied in all design stages down to the detailed design.

DL 3

Static analyses shall be applied in all design stages as well as in the implementation phase and integration phase.

Independence

IL 1

Static analyses shall be performed by independent persons.

IL 2

Static analyses shall be performed by an independent department.

IL 3

Static analyses shall be performed by an independent organisation.

Definition of Confidence Levels for Static Analysis

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL 1
Level 2	RL 2	DL 2	IL 1
Level 3	RL 2	DL 3	IL 2
Level 4	RL 3	DL 3	IL 3

Table 6: Definition of Confidence Levels for Static Analysis

Outputs

The review process should be documented. A verification report should be provided including a summary of the verifications performed, the list faults or critical issues identified during the review process, and suggestions for improvements or corrections. Evidence should be provided about how these faults have been addressed and corrected.

Use of the Results

The outputs of the static analyses process should be used to follow up the quality of the products and to increase the confidence in the correctness of the reviewed products.

Relation to System Lifecycle

Static analyses can be applied at each stage of the system life cycle process. They may take place for instance, between requirements definition and architectural design, between architectural and detailed design, and between detailed design and implementation. Besides covering development issues, static analyses should be applied for the verification of the operation, maintenance procedures and system disposal procedures.

Assessors activities

The role of the assessor is to check that all the planned reviews and static analyses are done and examine the process used to perform these analyses. He has to ensure that static verification activities have been performed according to the levels of rigor, detail and independence specified in the Criteria. The assessor should examine the results related to the critical issues of the system, and analyse their accuracy and completeness. He should ensure that the dependability related requirements they can be analysed with static analysis techniques have not been overlooked.

4.3.5.2 Testing

Name of the CPA

Testing

Objectives

Testing is a dynamic verification technique that involves exercising the system (or component) implementation by supplying it with input values.

Inputs	***Outputs***
• The inputs depend on the testing method used and the system level considered for testing. • The test cases for the verification of the dependability related functions should be based on hazard analysis results.	• Description of the test cases and corresponding results • Discrepancies between expected and actual results should be explained • Test coverage • Report on Testing

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL1	DL1	IL 1
Level 2	RL1	DL 2	IL 1
Level 3	RL2	DL 3	IL 2
Level 4	RL3	DL 3	IL 3

Methods and Tools

Functional testing, structural testing, Boundary-value testing, etc. (See Appendix A.5.3)

Objective

Testing is a dynamic verification technique that involves exercising the system (or component) implementation by supplying it with input values. Testing has the following two objectives:

- to demonstrate that the system implementation performs its intended functions,
- to provide confidence that the implemented system does not perform unintended functions that may affect the dependability objectives. It should be noted that the complete absence of unintended function can never be established by test.

In general, testing is “partial”, as it is not feasible to exercise the system with each possible data item from the input domain. Therefore, there arise the major issue of selecting a (small) subset of the input domain that is well suited for revealing the actual but unknown faults.

The main advantage of testing is that it involves the execution of the actual system. Therefore, it contributes to identification of faults introduced in the requirements definition as well as in the implementation of the system. Nevertheless the aim of testing is to reveal the presence of faults but it can not prove their absence. Testing is more suited to check positive requirements than negative requirements. Moreover, in order to be efficient the test cases applied should be realistic with respect to the real environment. Also, the criteria for test case generation should be defined carefully to allow a satisfactory coverage of critical requirements.

There is no universal testing method that can ensure that all residual faults are revealed and fixed. Therefore, a combination of several testing methods is recommended to achieve a high level of confidence in the correctness of the system.

Correctness of testing results relies on the reliability of the tools used. Hence, the assessment of these tools should also be considered.

Inputs

The inputs required for testing depend on the testing method used, the system level considered for testing, and the target of the test (hardware, software, etc.). For each testing method, we should provide a detailed definition of the tests cases applied to the system, the definition of pass/fail criteria to analyse the outputs of the testing process, and the expected results and the tolerances associates with those results.

The test cases for the verification of the dependability related functions should be identified using information from the various hazard analyses that have been performed.

Description of the CPA

Levels

Rigour

At least, system-level functional testing under operational conditions shall be applied complemented with system testing with extreme/special boundary values.

A test report should be provided including a description of the high-level requirements tested together with the testing methods, the success/fail test criteria, the test scenarios, the test results, and a test coverage summary (relating the test cases and results to the requirements). The test report should state what requirements are not tested and justify that the non-tested requirements can not lead to unacceptable behaviour according to the definition of the corresponding confidence level.

RL 2

Besides the requirements of RL1, the test cases should be defined to demonstrate the compliance of the implementation with high-level requirements as well as with low-level requirements. Deterministic test cases focused on the verification of some specific points in the input domain (extreme/special boundary values) shall be applied.

The test methods and the test criteria used shall be explained and justified. It shall be proved that the test patterns generation methods used are adequate to satisfy the test criteria, and that these criteria are satisfied by the test results.

The test environment used for the generation of the test cases and the analysis of test results shall be described.

The test cases and the test results shall be provided and explained.

The correspondence between test cases and the requirements shall be stated and justified. Requirements coverage analysis shall determine what requirements are not tested and how these requirements are covered by other verification techniques.

Evidence of re-tests after the discovery and correction of errors relevant to dependability is obligatory to demonstrate that the errors have been eliminated and no new errors have been introduced.

All classes of faults identified by hazard analysis shall be considered and evidence should be provided to show that these faults are properly handled by the system. Justification shall be given to show that the test methods used are adequate and efficient to address these faults. Those classes of faults that are not considered shall be identified and a justification about not considering them shall be provided. Fault injection should be used wherever possible to show the correct functioning of the fault tolerance mechanisms implemented in the system.

Evidence shall be given to justify that the test coverage achieved is satisfactory. When quantitative targets for the test coverage are specified, evidence should be provided to ensure that these targets are achieved.

RL 3

Besides the requirements corresponding to RL2, whatever the level of application of testing (unit level, integration level, system level), a mixed strategy for the generation of the test patterns shall be used: statistical test patterns generation complemented with deterministic test cases focused on the verification of some specific points in the input domain (extreme/special boundary values), which are difficult to cover with random patterns.

The system shall be tested in its operational environment, before its delivery (when this is possible).

Automatic test tools shall be used wherever possible.

The test tools used shall be validated and evidence should be provided to demonstrate that the level of validation of the test tools is compliant with the system confidence level.

Detail

DL 1

System-level testing

DL 2

System-level testing and integration testing

DL 3

System-level testing, integration testing and unit testing

Independence

The independence requirements specified below apply to system level testing.

IL 1

Testing shall be performed by independent persons.

IL 2

Testing shall be performed by an independent department.

IL 3

Testing shall be performed by an independent organisation.

Definition of Confidence Levels for Test Methods

	Rigour	Detail	Independence
Level 1	RL1	DL1	IL 1
Level 2	RL1	DL 2	IL 1
Level 3	RL2	DL 3	IL 2
Level 4	RL3	DL 3	IL 3

Table 7: Definition of Confidence Levels for Test Methods

Methods and Tools

Methods for the determination of the test patterns can be classed according to two viewpoints: 1) criteria for selecting the test inputs, and 2) generation of the test inputs. The presentation of the different criteria for test pattern selection (conformance or fault finding testing, functional or structural testing, fault-based or criteria-based) and generation (deterministic or probabilistic) are presented in Annexe A.5.3. Fault-based testing is particularly useful for the verification of the error and fault handling mechanisms that are critical in the context of dependable systems. The selection of test methods also depends on classes of faults considered: test methods used to reveal design faults are different from those aimed at uncovering hardware faults.

Output

The testing activities shall be documented in a testing report that includes, for each component, the following information:

- the test methods and tools used and justification of these choices
- the test cases applied and expected results
- the results of each test case
- the discrepancies between expected and actual results
- the test coverage achieved

When discrepancies occur, between expected and actual results, the analysis made and the decisions taken shall be documented.

Use of the Results

Test coverage analysis shall determine what are the criteria elements (requirements, fault assumptions, etc.) covered by testing. This analysis shall lead to the identification of additional verification effort needed if the coverage achieved is unsatisfactory. Testing results may lead to the re-examination of the fault assumptions and the modification of the system design or implementation.

When discrepancies occur, between expected and actual results, and corrections are implemented in the system regression testing should be done to ensure that the corrections introduced are successful and also that no additional errors are introduced in the system.

Relation to System Lifecycle

Testing can be applied once an implementation of the system or its components is available. Three levels can be distinguished for testing:

- unit testing: each component is tested with respect to its specification
- integration testing: check components integration with respect to the architecture and identify the faults that are due to interactions between system components. In particular, special attention should be put on the integration of the various hierarchical fault and error processing mechanisms.

- system testing: testing the global system.

A combination of different testing methods at each level of testing is recommended to increase confidence in the system.

Assessor's activities

Different aspects should be covered during the assessment of testing activities by the assessors. Assessment must address:

- the test plan which includes the definition of the test methods and the test criteria used, tests cases applied and test environment and tools, etc.
- the test report: which details the test outputs and the test coverage achieved, records the identified faults and failures, and states to what extent the test coverage and criteria are satisfied.

The primary role of the assessor is to check that the information provided meets all requirements of the criteria. They must examine the correctness and completeness of the test procedures and results. The assessors shall ensure that the test coverage achieved is sufficient in accordance with the confidence level required. They can check by sampling the results of the tests or ask for supplementary tests to be performed.

4.3.5.3 Formal methods and Proof-of-correctness

Name of the CPA

Formal methods and Proof-of-correctness

Objectives

Formal methods can help in the development and presentation of clear, precise, and unambiguous statements of requirements, assumptions, specifications and designs. They can be used to verify some properties related to the specification itself, or to show the consistency between the successive refinement levels of the design

Inputs	***Outputs***
Formal verification requires a formal description of the requirements and a statement of the properties that could lead to unacceptable failure conditions.	• Description of the properties checked • List of specification faults and inconsistencies discovered and suggestions for their correction • Report on Formal Methods & Proofs

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	-	-	-
Level 2	RL 1	DL 1	IL 1
Level 3	RL 2	DL 1	IL 2
Level 4	RL 3	DL 2	IL 3

Methods and Tools

See [Rushby 1993]

Objective

A mathematical proof is a finite series of inferences in a formal system, i.e., in a formalism involving a syntax and a semantics based on a mathematical background associated with manipulation rules which enable to perform transformations and verifications. Such proofs may be used to verify some properties related to the specification itself, and in this case the corner stone is the determination of the properties to be satisfied by the specification. They may also be used to show the consistency between the successive refinement levels of the design (from the requirement down to the implementation).

Formal verification has the following advantages:

- It is well suited to demonstrate the correctness of the basic mechanisms and algorithms concerned with complex behaviours, including those concerned with concurrency and synchrony (e.g. interrupt handling and co-ordination of multiple processes), those concerned with real-time constraints, those concerned with redundancy management and fault tolerance, and those concerned with the manipulation of low-level hardware features (e.g. the use of memory management and protection hardware to provide partitioning and fault containment across tasks sharing the same processor).

- Formal methods can help requirements authors to state what the component concerned is intended to do, what it should not do, what constraints it must satisfy, and what assumptions may be made about its environment.
- Formal verification is well suited to prove negative properties: if the properties that could lead to unacceptable behaviour are stated precisely, then formal methods can be used to show that the system can not have these kinds of behaviours.

The following critical issues have to be considered when using formal methods and proofs:

- Formal methods deal with models of the system not the system itself. The formal model is generally derived from an informal specification. Consequently some testing is required to validate the system with respect to this specification, and so, to complement formal verification activities. Moreover, dependability validation criteria shall contribute in assessing the validity of the models and the assumptions used in formal specification and verification.
- The validity of formal verification results depends on the validity of the proofs and the tools used to support the verification process. In particular, the formal proofs, including those checked by machine, may be incorrect. Note that tools validation is a more general issue that should be addressed for any validation and verification activity supported by tools.

Inputs

Formal verification requires a formal description of the requirements and a statement of the properties that could lead to unacceptable failure conditions.

Description of the CPA

At each design level where formal verification is used, the analyst must identify the critical component whose behaviour will be proved, derive a list of potential unacceptable behaviour and prove that these behaviours can not occur. The use of formal methods leads to consider the following issues: *what* the component concerned is intended to do, what it should *not* do, what *constraints* it must satisfy, and what *assumptions* may be made about its environment.

Levels

Rigour

The following classification of formal methods according to their level of rigour is based on [Rushby 1993].

RL 1

Use of concepts and notation from discrete mathematics. This consists of replacing some of the natural language used in requirements statements and specifications with notation and concepts derived from logic and discrete mathematics. This formalised description can help in the development and presentation of clear, precise, and systematic statements of requirements, assumptions, specifications and designs. Moreover, it provides a compact notation of these statements to be written down and communicated with less ambiguity than natural languages, and to systematise and guide the elaboration and the refinement of the specifications and design assumptions. Verification activities are generally performed in the informal style.

RL 2

Use of formalised specification languages with some mechanised support tools (such as syntax checkers, typecheckers, etc.). In addition to the benefits of the methods corresponding to RL1, these methods allow specifications to be structured into units with explicitly specified interfaces. Mechanised tools allow accurate detection of certain types of faults, or may make it simple to trace dependencies through large specifications. It may be possible also to quickly generate a simulation, or prototype implementation from a specification either automatically or with some human guidance. Proofs at this level are usually performed informally by hand, except for some methods which provide explicit formal rules of deduction.

RL 3

Use of fully formal specification languages with comprehensive support environments including mechanised theorem proving or proof checking. These methods employ a specification language with a very direct interpretation in logic, and with correspondingly formal proof methods. Proof mechanisation can take the form of a proof checker (i.e., a computer program that checks the steps of a proof proposed by a the human user) or of a theorem prover (i.e. a computer program that attempts to discover proofs without human guidance) or, most commonly of something in between. The reason for using mechanised theorem proving is not to relieve human users of responsibility, but to augment their ability to conduct arguments of great length and in extreme detail.

Detail

The use of formal methods is strongly encouraged to provide complete analysis and exploration of those aspects of design that seem least well covered by other correctness verification techniques. These arise in redundancy management, partitioning and the synchronisation and co-ordination of distributed components, and primarily concern fault tolerance, timing, concurrency and non determinism. Only two levels of detail are considered for the use of formal methods.

DL 1

Formal methods should be used to support the correct refinement of the specification of some selected critical dependability related mechanisms and algorithms concerned with complex behaviours (e.g. redundancy management, partitioning, synchronisation, state consistency, etc.). The use of formal methods should be focused on proving that these mechanisms do not exhibit certain *undesired* properties.

DL 2

Besides the requirements defined for DL1, it should be ensured that the implementation of algorithms and mechanisms are faithful to their verified specifications. This level of detail shall be required only if the complexity of these mechanisms can be handled by currently available methods and tools.

Independence

The independence levels defined in the sequel apply to the verification of the formalised specification and not to the development of the specification itself.

IL 1

The verification of the formal specification and of the validity of the proofs generated by hand shall be done by an independent person.

IL 2

The verification of the formal specification and of the validity of the proofs generated by hand shall be done by an independent department.

IL 3

The verification of the formal specification and of the validity of the proofs generated by hand shall be done by an independent organisation.

Definition of Confidence Levels for Formal Methods

	Rigour	Detail	Independence
Level 1	-	-	-
Level 2	RL 1	DL 1	IL 1
Level 3	RL 2	DL 1	IL 2
Level 4	RL 3	DL 2	IL 3

Table 8: Definition of Confidence Levels for Formal Methods

Methods and Tools

Several formal specification and verification methods and tools are currently available. The methods differ by the level of rigor, the availability of some mechanised support tools for theorem proving or proof checking, the kind of behaviour they are more suited to address, and their applicability in the life cycle stages of the development.

Before applying formal methods to a system development, it is important to consider carefully a number of choices and decisions: for what purpose are formal methods to be applied, to what components and properties, and with what level of rigor?

A presentation of several formal methods and tools, and a discussion about selection criteria can be found in [Rushby 1993].

Output

The results of formal verification shall be documented in a report, which must include the following information:

- the objectives of formal verification: what are the components concerned, what properties are verified, what are the assumptions stated,
- the methods and tools used and justification of these choices,
- the results of the verification process.

Use of the Results

If an unacceptable behaviour is identified by this method, changes in the system design have to be made, which either eliminate the identified undesired behaviour or limit its effect to an acceptable level.

Relation to System Lifecycle

In examining where in the lifecycle formal methods may be best applied, we need to consider the effectiveness and the cost of formal methods relative to other methods. In both these dimensions, there is a strong case against applying formal methods late in the lifecycle due to the complexity of the system representation at this stage [Rushby 1993].

Assessors' activities

The main role of the assessors is to analyse the information provided about the scope of the formal method used, the properties analysed, and the adequacy of the methods and tools used to meet the objectives.

4.3.5.4 Behavioural analysis

Name of the CPA

Behavioural analyses

Objectives

Behavioural analysis consists of verification performed on a behaviour model deduced from either its specification, or its design, or the implementation. The models allow the verification of general properties such as consistency, completeness, absence of deadlocks, etc. and also specific properties deduced from the requirements (for instance, that a specific sequence of events should not occur)

Inputs

The model used for behavioural analysis is deduced from the specification or the structure of the component analysed. Verification properties should also be specified. Some properties should be deduced from the hazard analyses.

Outputs

- Description of the properties checked
- List of design faults discovered and suggestions for their correction

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 3	IL 3

Methods and Tools

Graphs, Decision tables, Finite-State machines, State-charts, Petri nets, etc.

Objective

Behavioural analysis consists of verification performed on a behaviour model deduced from either its specification, or its design, or the implementation. The models allow the verification of general properties such as consistency, completeness, absence of deadlocks, etc. and also specific properties deduced from the requirements (for instance, that a specific sequence of events should not occur).

Behaviour analysis shall focus on system behaviour in presence of faults to ensure that fault assumptions are properly covered by protection and fault tolerance mechanisms.

The popularity of behavioural analyses is increasing since they can be used along the development process for specifying, designing as well as for verification purposes. These techniques are well suited to analyse complex behaviour of the system such as concurrency, synchronisation, real-time constraints, etc. Moreover, the reference models can be used for the design of functional tests cases. Several behavioural models have to be constructed to catch several aspects of the system behaviour. Some available methods allow the construction of hierarchical models that facilitate the analysis of the system. Nevertheless, verification based on behavioural analyses is limited by the complexity of the models constructed, and the validity of the results depend on the accuracy of the models. Exhaustive state exploration or exhaustive path searching methods are generally unfeasible, verification is therefore limited based of a partial analysis. This limitation applies to all verification techniques.

Inputs

The model used for behavioural analysis is deduced from the specification or the structure of the component analysed. Verification properties should also be specified. Some properties should be deduced from the hazard analyses.

Description of the CPA

Levels

Rigour

RL 1

Behavioural analysis shall focus on a selected subset of system properties, which are significant with respect to system dependability objectives. The objectives and the results of the behavioural analysis shall be stated and the methods and tools used to achieve these objectives shall be identified.

RL 2

Behavioural analysis shall focus on a selected subset of system properties, which are significant with respect to system dependability objectives. The objectives of behavioural analysis shall be described together with the models used. The assumptions made shall be explained and justified based on the results of the hazard analysis. Finally, the methods and tools used to achieve the objectives shall be justified and the results obtained shall be explained and correspondence between the objectives and the results shall be demonstrated.

RL 3

Besides the requirements corresponding to RL2, behavioural analyses shall be supported by computer aided simulation tools. Justification should be given about the adequacy of the tools to meet the objectives and about the level of validation of these tools.

Detail

DL 1

Behavioural analysis shall be used to verify some properties derived from high level requirements.

DL 2

Behavioural analysis shall be used to verify some properties derived from high level requirements and low level requirements.

DL 3

Behavioural analysis shall be used to verify some properties derived from high level requirements and low level requirements. Additional behavioural analyses shall be based on a model derived from the implementation of the system.

Independence

IL 1

Behavioural analyses shall be performed by independent persons.

IL 2

Behavioural analyses shall be performed by an independent department.

IL 3

Behavioural analyses shall be performed by an independent organisation.

Definition of Confidence Levels for Behavioural Analyses

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 3	IL 3

Table 9: Definition of Confidence Levels for Behavioural Analysis

Methods and Tools

Techniques available for performing behavioural analyses include graphs, decision tables, finite state machines, Petri nets, statecharts, etc. These techniques are supported by automatic tools that provide valuable help in building and validating the models (see [Davis 1993]).

Output

Behavioural analysis contributes to the identification of faults in different design stages of the system. They also contribute to the increase the confidence about the correct behaviour of the system with respect to some critical properties. The output of behaviour analysis shall record the properties checked by behavioural analyses and the faults identified that should lead to some modification of the system.

Moreover, the models used for performing behavioural analyses can be used as reference models for the generation of functional test cases to test the final implementation of the system.

Use of the Results

Inconsistencies or incompleteness in the specifications identified at this stage should be solved: either the design should be modified, or evidence should be provided to show that the identified vulnerabilities cannot have critical consequences on the system behaviour.

Relation to System Lifecycle

Behavioural analyses can be used along all the design stages of the dependability life cycle.

Assessor's activities

The role of the assessor is to check the results of the behavioural analysis. He has to assess if the analyses performed are representative of the expected behaviour of the system and to ensure that the critical issues of the design are covered by the analyses performed. The assessor verification activities should be guided by the information provided by the hazard analyses.

4.3.5.5 Traceability analysis

Name of the CPA

Traceability Analysis

Objectives

The purpose of traceability analysis are: 1) check that each high level requirement is satisfied by low level requirements (top-down traceability), 2) check that each low level requirement correspond to a specified high level requirements (bottom-up traceability), 3) understand which requirements are verified by each test case and verification procedure.

Inputs

To perform the traceability analysis, the dependability requirements and the detailed description of the system at each stage when the traceability analysis is performed, are needed.

Outputs

- List of requirements not covered by the design, by the implementation of the system, or by the operation, maintenance and disposal procedures, and justifications
- List of design elements which do not correspond to specified requirements, and justifications
- Report on Traceability Analysis

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL1	DL1	IL 1
Level 2	RL1	DL1	IL 1
Level 3	RL2	DL2	IL 2
Level 4	RL2	DL 2	IL 2

Methods and Tools

Cross-references, matrices, axiomatic, RTM, etc. (see [Davis 1993, Pearson 1998])

Objective

The traceability of the requirements through the lifecycle is particularly important for critical systems: we must be sure that every requirement (either functional or non functional) is satisfied by the implementation, and that the implementation does not provide functions other than those required.

The purpose of traceability analysis are: 1) ensure that each high level requirement is satisfied by low level requirements (top-down traceability), 2) ensure that each low level requirement correspond to a specified high level requirements (bottom-up traceability), 3) understand which requirements are verified by each test case and verification procedure.

Confidence in the system is built up by checking that the dependability related functions of the target of evaluation are correctly refined throughout the different steps of the dependability lifecycle. In particular, it is important to analyse the traceability of the fault and error assumptions and ensure that these assumptions are handled correctly by dependability related functions and mechanisms.

Traceability analysis is a basic technique for ensuring the correct refinement of the dependability requirements through the design up to implementation. This technique should be combined with the other verification techniques since the traceability of some requirements can only be ensured by the use of those techniques.

Inputs

To perform the traceability analysis, the dependability requirements and the detailed description of the system at each stage when the traceability analysis is performed, are needed.

Description of the CPA

Levels

Rigour

RL1

The traceability analysis shall demonstrate that each high-level requirement is taken into account by the implementation. The traceability analysis should also show the correspondence between the test procedures and the requirements.

RL2

The traceability analysis shall demonstrate that each high-level requirement is taken into account by the implementation, and each low level requirement traces to a high-level requirement. All functions implemented in the system should be traceable to a parent requirement. There should be no implemented functions that do not correspond to a specified and validated requirement.

The traceability analysis should also show the correspondence between the test cases and verification procedures and the requirements.

Detail

DL 1

The traceability analysis shall be focused on dependability related functions and requirements.

DL 2

Both functional and dependability related requirements should be covered by traceability analysis.

Independence

IL 1

The traceability analysis can be done by the developers.

IL 2

The traceability analysis shall be done by an independent person.

Definition of Confidence Levels for Traceability Analysis

	Rigour	Detail	Independence
Level 1	RL1	DL1	IL 1
Level 2	RL1	DL1	IL 1
Level 3	RL2	DL2	IL 2
Level 4	RL2	DL 2	IL 2

Table 10: Definition of Confidence Levels for Traceability Analysis

Methods and Tools

Several frameworks and methods for traceability are reported in the literature [Pearson et al. 1998], e.g. cross-references, matrices, relational, axiomatic. In [Davis 1993], a Requirements traceability matrix (RTM) is proposed to support traceability analysis. In its simplest form, an RTM is a two dimensional table with one row for every requirement and one column for each design item. An *X* is placed in each entry where the design item helps to satisfy the indicated requirement. When completed, a row with no *X*s indicates that there exists a requirement that is not met, i.e., no part of the design contributes to its satisfaction. A column with no *X*s indicates an extraneous design item.

Output

The traceability analysis helps to identify:

- the requirements which are not covered by the design, by the implementation of the system, or by the operation, maintenance and disposal procedures,
- the design elements which do not correspond to specified requirements.

Use of the Results

Based on the results of the traceability analysis, the system design must be updated and modified to meet the requirements.

Relation to System Lifecycle

Traceability analysis can be applied at each stage of the development lifecycle.

Assessors' Activities

The role of the assessor is to check that the traceability analysis is complete and is achieved correctly. The completeness of a requirement traceability analysis will depend on the complexity of the system and the confidence level.

The assessor shall check that the traceability analysis has been done and the results obtained are satisfactory with respect to the confidence level. He shall ensure that all critical requirements (including those related to system installation, maintenance and disposal) are covered by the traceability analysis.

4.3.5.6 Recommended verification techniques: Summary

None of the verification techniques mentioned in the previous section can ensure that all residual faults have been revealed and fixed. Therefore, they should be complementary for at least two reasons: first they do not always apply to the same stages of the development process, and second, in essence they are quite different and each verification technique is more suited to reveal a specific class of faults. As illustrated by several experimental studies, the combined use of several verification techniques is required to ensure a satisfactory confidence level in the system correctness. As a result, a proper verification process calls for the combined utilisation of several techniques.

The choice of the techniques to be used is highly dependent on the complexity and the criticality of the component to be assessed:

For all confidence levels, the combined utilisation of static analysis and testing techniques should be a suitable and cost effective verification process for any system:

- Static analyses: they can be applied at each stage of the development process and to each development working product (specification and design documents, verification plans, etc.). A large proportion of faults can be revealed early after their introduction: the efficiency of this low cost technique has been illustrated on several case studies and it is very easy to use; however, in order to be efficient, static analyses should be done rigorously.
- Testing techniques: they allow, in particular, the coverage of the dynamic aspects of the system behaviour that can hardly be addressed by static analyses techniques. Whatever the level of application of testing (unit level, integration level, system level), it is recommended to use a mixed strategy for the generation of the test patterns: probabilistic test patterns complemented with deterministic test cases which are focused on the verification of some specific points in the input domain which are difficult to cover with random patterns.

For Levels 3 and 4, behavioural analyses as well as formal methods should be required in addition.

The following table summarises the mapping between Correctness Verification activities and confidence levels (combination of CPAs is addressed in Section 4.6).

Correctness Verification Methods	Level 1	Level 2	Level 3	Level 4
Testing	RL1, DL1, IL1	RL1, DL2, IL1	RL2, DL3, IL2	RL3, DL3, IL3
Static analysis	RL1, DL1, IL1	RL2, DL2, IL1	RL2, DL3, IL2	RL3, DL3, IL3
Traceability analysis	RL1, DL1, IL1	RL1, DL1, IL1	RL2, DL2, IL2	RL2, DL2, IL2
Behavioural analysis	RL1, DL1, IL1	RL2, DL1, IL1	RL3, DL2, IL2	RL3, DL3, IL3
Formal methods	-	RL1, DL1, IL1	RL2, DL1, IL2	RL3, DL2, IL3

Table 11: Mapping between Correctness Verification activities and Confidence Levels

4.4 Dependability Validation

4.4.1 Motivation and scope

The purpose of Dependability Validation activities is, to ensure that a system is compliant with the stated objectives. The main problem is that many generally stated objectives can not be directly translated into specific requirements or directly implementable specifications for the system, where compliance to those requirements or the specification can be demonstrated by more easily verifiable correctness arguments. So the main objective of the validation activities is to ensure that a system does not have undesirable side-effects, which do not contradict the system specification but result in a system behaviour that may have intolerable consequences (i.e. may not satisfy the dependability objectives).

The Dependability Validation activities address the problem that

- any specification of a system is incomplete with respect to the overall objectives,
- the set of assumptions made during the specification and design process can not be documented completely
- and that some of the assumptions made during system construction, operation or disposal may be wrong.

The activities defined in this section try to address specific aspects of the problem stated above. They can never guarantee that all flaws are identified, but they nevertheless increase the confidence in the system.

There are different reasons why a system may fail to fulfil its objectives but is still compliant with the derived requirements and specifications. Those reasons are:

- Wrong or incomplete assumptions have been made in the process of deriving requirements and specifications from the stated objectives,

- The requirements and specifications derived from the objectives are incomplete or wrong.

Several areas may be affected by those two aspects:

- Some functions may not be well suited for specific situations (Suitability),
- Some functions may be conflicting with one or more other functions in specific situations. This may lead to cases where the objectives are not fulfilled (Binding),
- The system is too difficult to use and usage errors therefore have a high probability (Ease of Use),
- The mechanisms/algorithms used in the implementation have weaknesses which may be in contradiction with the objectives (Strength of Mechanism),
- The implementation may have additional side effects, which are in contradiction with some aspects of the objectives (Vulnerabilities).

The confidence providing Dependability Validation activities and techniques listed in Section 4.4.2 deal with the problems mentioned above. When a given activity is specific to one dependability attribute, this is explicitly mentioned.

4.4.2 Dependability Validation Confidence Providing Activities

This section presents the following dependability validation confidence providing activities:

- Penetration Analysis
- Covert Channel Analysis
- Experimental Evaluation

Covert Channel Analysis is specific to confidentiality.

4.4.2.1 Penetration Analysis

Name of the CPA

Penetration Analysis

Objectives

This activity tries in a systematic way to identify faults that may lead to a critical situation. To do this, the negative objectives for the system are turned into “positive” objectives for the penetration analyser. For example, if the objective is in the form “The system shall never do x”, it is turned into the task for the penetration tester: “Try to identify a situation, where the system does x”.

Inputs	***Outputs***
• Dependability target • TDA design documentation • TDA implementation • System operation procedure	List of potential faults, with a description of their exploitability and suggestions for removal or for protection against them

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL3	IL 3

Methods and Tools

Fault Hypothesis Method

Objective

This activity tries in a systematic way to identify faults that may lead to a critical situation. To do this, the negative objectives for the system are turned into “positive” objectives for the penetration analyser. For example, if the objective is in the form “The system shall never do x”, it is turned into the task for the penetration tester: “Try to identify a situation, where the system does x”.

This activity is usually performed when an implementation of the system, subsystem or component being studied is available. However, it could be used also during the design stages. In that case, the penetration analysis becomes similar to static verification techniques. Thorough inspections and reviews of the design documents and modelling and simulation of system behaviour in the presence of faults might lead to the identification of scenarios where some known vulnerabilities are not properly handled.

Penetration analysis is mainly a Dependability Validation CPA. However, it can be also used as a Dependability Requirements Validation CPA if it is applied during the design stages.

Penetration analysis is a main activity in the security sector to achieve a high level of confidence. Although the method initially starts with the view of a "penetrator", who deliberately tries to break into a system, it can also be applied generally for dependable systems. In this case the objective is to identify a combination of valid input values that may bring the system into a potentially critical situation. This may be achieved by a hypothesis for those critical situations (i.e., negating the objective or requirement) and then trying to identify scenarios (combination of valid input values) which will (or may) lead to this situation. With this interpretation, penetration testing can be applied for all dependability objectives.

Inputs

To perform the penetration analysis, the penetration tester needs the system Dependability Target (containing the system dependability objectives, the system dependability policy and the assumptions on the environment of the system), the system design documentation, the system implementation, the system operational procedures (for operation phase).

Description of the CPA

The penetration tester shall use a systematic approach to identify the parts of the system that may contain faults. The following steps can be followed:

1. Define the scope and the boundaries of penetration analysis.
2. Define a list of known faults and vulnerabilities to be checked.
3. For each fault and vulnerability, try to identify a scenario where dependability related requirements are not satisfied.
4. Summarise the results and identify improvement areas of the system design to address the problems identified during the penetration analysis activity.

Levels

Rigour

RL 1

The penetration tester shall make informal inspections and generate and conduct a small number of penetration test cases from the observations made during the inspections.

RL 2

The penetration tester shall apply the Fault Hypothesis Method (see Methods and Tools) to generate penetration test cases. The generation as well as the results of the verification shall be documented in a penetration test plan.

RL 3

The penetration tester shall apply the Fault Hypothesis Method to generate penetration test cases. All verification cases shall be described in a penetration test plan, which documents all four stages of the Fault Hypothesis method.

The following issues can provide guidelines to generate penetration analysis verification cases:

- Past experience with flaws in other similar systems.
- Ambiguous, unclear architecture and design.
- Circumvention of "omniscient" dependability controls.
- Incomplete design of interfaces and implicit sharing.
- Deviations from the protection policy and model.
- Deviations from initial conditions and assumptions.
- System anomalies and special precautions.
- Operational practices, prohibitions, and spoofs.
- Development environment, practices, and prohibitions.
- Implementation errors.

Whenever a fault could not be confirmed, a detailed description shall be given explaining what has been done to try to confirm the fault and justifying, why the fault could not be confirmed.

Some members of the penetration analysis team shall have significant experience in penetration analysis of systems similar to the one that is analysed.

Detail

DL 1

The penetration analysis shall use the implementation documentation down to the architectural and detailed design documentation of the TDA

DL 2

The penetration analysis shall use the implementation documentation down to the architectural and detailed design documentation of the TDA as well as the software implementation.

DL 3

The penetration analysis shall use the implementation documentation down to the architectural and detailed design documentation of the TDA as well as the software implementation and the hardware and firmware design and implementation of the system.

Independence

IL 1

The penetration analysis shall be performed by the developers within the development team.

IL 2

The penetration analysis shall be performed by a team not directly involved in the development of the system to be assessed.

IL 3

The penetration analysis shall be performed by a team independent from the organisation(s) involved in the development of the system to be assessed. All members of the team shall have significant experience with systems similar to the one that is assessed.

Definition of Confidence Levels for Penetration Analysis

	Rigour	Detail	Independence
Level 1	RL 1	DL 1	IL 1
Level 2	RL 2	DL 1	IL 2
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL3	IL 3

Table 12: Definition of Confidence Levels for Penetration Analysis

Methods and Tools

The central method for penetration analysis is the Fault Hypothesis Method (FHM) (in the security area called "Flaw Hypothesis Method").

This method consists of four stages:

1. Fault Generation develops an inventory of suspected faults.
2. Fault Confirmation assesses each fault hypothesis as true, false, or unchecked.

3. Fault Generalisation analyses the generality of the underlying dependability weakness represented by each confirmed fault.
4. Fault Elimination recommends fault repair, or the use of external controls to manage risks associated with residual faults.

A description of the Fault Hypothesis Method and a preliminary list of areas where faults can be expected are given in [NRL-Handbook].

Output

The outputs of a penetration analysis consists of a list of potential faults, with a description of their exploitability and suggestions for removal or for protection against the fault through measures in the system environment.

Use of the Results

The faults identified in the penetration analysis are rated according to the hazard rating method used. Hazards rated with an unacceptable high value have to be reduced or eliminated in the same way as those identified in the initial hazard analysis. All documentation affected by the changes to the system or the environment must be updated.

Relation to System Lifecycle

Penetration Analysis can be applied during design, implementation, integration, operation and maintenance.

Assessor's activities

The penetration analyses done by the developers should also be complemented by penetration tests performed by the assessors with levels of detail and rigour defined in accordance with the required confidence level. The main objective is to confirm or disprove the claimed minimum strength of dependability mechanisms implemented in the system and to ensure that known vulnerabilities are properly handled by the system.

4.4.2.2 Covert Channel Analysis

<table>
<tr><td colspan="2"><b><i>Name of the CPA</i></b>
Covert Channel Analysis</td></tr>
<tr><td colspan="2"><b><i>Objectives</i></b>
A covert channel analysis can be applied for systems with the objective to prohibit the flow of information to specific users or to external or internal entities (mainly to ensure confidentiality). This objective may be introduced during system design through the use of specific mechanisms that require the prohibition of the flow of specific information held in the system (e. g. passwords or keys).</td></tr>
<tr><td><b><i>Inputs</i></b>
A formal or semi-formal specification of the design to be analysed has to be provided and the flows of information that the system has to prohibit have to be specified</td><td><b><i>Outputs</i></b>
List of potential covert channels, with a description of the exploitability and an estimation of the bandwidth of the channel</td></tr>
<tr><td colspan="2"><b><i>Mapping to Confidence Level</i></b>
<table>
<tr><th></th><th>Rigour</th><th>Detail</th><th>Independence</th></tr>
<tr><td>Level 1*
*does not apply for level 1</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Level 2</td><td>RL 1</td><td>DL 1</td><td>IL 1</td></tr>
<tr><td>Level 3</td><td>RL 2</td><td>DL 2</td><td>IL 2</td></tr>
<tr><td>Level 4</td><td>RL 3</td><td>DL 3</td><td>IL 3</td></tr>
</table>
</td></tr>
<tr><td colspan="2"><b><i>Methods and Tools</i></b>
Shared Resource Matrix Method, Information flow Method, Covert Flow Tree Method</td></tr>
</table>

Objective

A covert channel analysis can be applied for systems with the objective to prohibit the flow of information to specific users or to external or internal entities (mainly to ensure confidentiality). This objective may be introduced during system design through the use of specific mechanisms that require the prohibition of the flow of specific information held in the system (e. g. passwords or keys). This activity shall be performed during the system construction phase. It is usually performed at the end of the construction phase based on the system implementation. However, it can also be used during the detailed design stage. Therefore, covert channel analysis is mainly a Dependability Validation CPA, but it can also be used for Dependability Requirements Validation.

Although a lot of research work has been spent in this area, there is still no systematic way to detect covert timing channels. But also for the analysis of covert storage channels the existing methods are all over pessimistic in the sense that they discover also those potential covert channels, which can not be exploited in practice and therefore do not result in a serious fault.

Inputs

To perform a covert channel analysis a formal or semi-formal specification of the design to be analysed has to be provided and the flows of information that the system has to prohibit have to be specified. In the case of a covert channel analysis for the implementation, this implementation has to be provided in form of source code and/or complete formal or semi-formal hardware descriptions.

Description of the CPA

The flow of information between all entities defined in the design or implementation is analysed and it is checked if this flow does not contradict the specified information flow requirements. Usually one of the methods listed in the sequel is used to perform this analysis.

Levels

Rigour

RL 1

The Covert Channel Analysis is performed by an informal analysis using one of the described analysis methods. The analysis may not include a systematic search for covert timing channels. A short argument is given why and how an identified channel can be exploited. The bandwidth of exploitable covert channels is estimated in an informal way.

RL 2

The Covert Channel Analysis is performed in a formal way which ensures the completeness of the analysis of storage channels with respect to the underlying method and the design level analysed. The analysis also includes a systematic search for timing channels. For each identified potential covert channel, a convincing argument should be given to state if the channel is exploitable or not. The bandwidth of exploitable covert channels is estimated by a formal calculation using well defined input parameters.

RL3

The Covert Channel Analysis is performed in a formal way assisted by a tool suitable to identify covert storage channels. This tool shall ensure the completeness of the analysis with respect to the underlying method and the design level analysed. The analysis also includes a systematic search for covert timing channels. For each identified covert channel rated as unexploitable a convincing argument is given for this rating. For each exploitable covert channel a description is given detailing how it can be exploited. In addition a test case is provided demonstrating the exploitation of the channel. The maximum bandwidth of each exploitable covert channel is calculated based on verified assumptions and input parameters.

Detail

DL 1

The Covert Channel Analysis is applied to the architectural and detailed design of the system. The analysis is limited to covert storage channels.

DL 2

The Covert Channel Analysis is applied to all design and implementation phases and includes also a search for covert channels at the source code level.

DL3

The Covert Channel Analysis is applied to all design and implementation phases and includes a search for covert channels at the source code level. In addition the system hardware architecture is carefully analysed for potential covert channels.

Independence

IL 1

The Covert Channel Analysis shall be performed by the developers.

IL 2

The Covert Channel Analysis shall be performed by a team not directly involved in the development of the system to be assessed.

IL 3

The Covert Channel Analysis shall be performed by a team independent from the organisation(s) involved in the development of the system to be assessed. All members of the team shall have significant experience with systems similar to the one that is assessed.

Definition of Confidence Levels for Covert Channel Analysis

	Rigour	Detail	Independence
Level 1* *does not apply for level 1	-	-	-
Level 2	RL 1	DL 1	IL 1
Level 3	RL 2	DL 2	IL 2
Level 4	RL 3	DL 3	IL 3

Table 13: Definition of Confidence Levels for Covert Channel Analysis

Method and Tools

The following methods for covert channel analysis exist:

- Shared Resource Matrix Method
- Information Flow Method
- Covert Flow Tree Method

Some other methods are listed in research oriented papers and may be considered for specific applications.

All methods try to identify and describe the information flow between system entities in a systematic way, thereby identifying potential covert channels. An informal analysis has to be performed thereafter to check, if the potential covert channel identified by the method can be exploited in the real system.

Several proprietary tools exist for covert channel analysis, which are related to formal specification and verification methods endorsed by the National Computer Security Center in the US. In addition some software analysis tools which check the dynamic behaviour of programs may be used to assist in a covert channel analysis. All those tool are limited to the detection of covert storage channels.

Output

The output of a covert channel analysis consists of a list of potential covert channels with a description of the exploitability and an estimation of the bandwidth of the channel.

Use of the Results

The output of the covert channel analysis should lead to a modification of the design (if the channel is rated as a fault with unacceptable consequences). The results should also be communicated to system integrators/system accreditors or system operators to inform them about the potential danger associated with a specific channel and how those problems may be overcome.

Relation to System Lifecycle

Covert Channel Analysis is applied in the detailed design and the implementation and integration phase of the system lifecycle.

Assessor’s activities

The role of the assessor is to examine thoroughly the results of the covert channel analysis and possibly to complete it by performing its own analyses.

4.4.2.3 Experimental Evaluation

Name of the CPA

Experimental Evaluation

Objectives

This activity consists in evaluating system dependability measures, or partial measures describing the behaviour of the most critical parts of the system (e.g. fault tolerance mechanisms), based on data collected during the execution of the real implementation. The main objective is to ensure that the quantitative dependability objectives are met by the final implementation of the system before its release to operation and during the operation phase.

Inputs

- Specification of Dependability Objectives and Dependability Related Functions
- Results of Hazard Analyses

Outputs

- Description of experimental evaluation experiments and results
- Dependability measures evaluated and comparison with Dependability Objectives
- Report of Experimental Evaluation

Mapping to Confidence Level

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 2	IL 3

Methods and Tools

Fault injection, field testing, statistical evaluation

Objective

This activity consists in evaluating system dependability measures, or partial measures describing the behaviour of the most critical parts of the system (e.g. fault tolerance mechanisms), based on data collected during the execution of the real implementation. The main objective is to ensure that the quantitative dependability objectives are met by the final implementation of the system before its release to operation and during the operation phase. When the system is exercised with real inputs, data is collected and processed by statistical techniques to evaluate the dependability measures. This data can be collected during fault injection experiments performed on a real prototype, or during system testing in its operational environment (i.e., during field testing), or during system operation.

It is noteworthy that experimental evaluation and probabilistic quantitative evaluation (see Section 4.2.3) are complementary. Indeed, modelling leads to the definition of parameters characterising the stochastic processes (failures, repairs, coverage, etc.) described in the model. These parameters can be estimated based on the data collected during the experimental evaluation. On the other hand, modelling can be used to define the most important parameters for which detailed experiments are needed to provide realistic estimations (for instance coverage, latency, etc.).

Inputs

The Dependability Target and the results of dependability requirements validation CPAs (especially, hazard analyses, common causes analyses and probabilistic quantitative evaluation) are needed to support and define experimental evaluation experiments.

Description of the CPA

Experimental evaluation involves the specification of the quantitative measures to be evaluated, the definition and execution of controlled experiments for the evaluation of these measures, and finally the statistical processing of the data collected from the experiments and the evaluation the specified measures.

Levels

Rigour

RL 1

The objectives and the experimental evaluation results of analysis shall be stated and the methods and tools used to achieve these objectives shall be described.

RL 2

The objectives of the experimental evaluation shall be described together with the experimental environment used. The assumptions made shall be explained and justified. It should be shown that the experimental evaluation experiments are performed according to realistic conditions close to the real operational environment.

RL 3

Besides the requirements corresponding to RL2, the experimental evaluation shall address the impact of different classes of faults (permanent faults, transient faults, etc.) and lead to the evaluation of the fault tolerance coverage achieved by the fault tolerance mechanisms. The statistical techniques used to obtain coverage estimates shall be described and justified. Finally the validity of the evaluation tools used shall be assessed.

Detail

DL 1

Experimental evaluation shall be based on data collected during system testing and field testing.

DL 2

Controlled experiments should also be defined to analyse the most critical parts of the system, especially, to evaluate the efficiency of fault tolerance and protection mechanisms.

Independence

IL 1

Experimental evaluation shall be performed by independent persons.

IL 2

Experimental evaluation shall be performed by an independent department.

IL 3

Experimental evaluation shall be performed an independent organisation.

Definition of Confidence Levels for Experimental Evaluation

	Rigour	Detail	Independence
Level 1	RL1	DL 1	IL1
Level 2	RL 2	DL 1	IL 1
Level 3	RL 3	DL 2	IL 2
Level 4	RL 3	DL 2	IL 3

Table 14: Confidence Levels for Experimental Evaluation

Methods and Tools

Experimental evaluation is based on data collected when the system is exercised with real inputs. This data can be collected during fault injection experiments performed on a real prototype, or during system testing in its operational environment, or during system operation. The data collected is processed by statistical techniques to estimate the values of the parameters used in the probabilistic quantitative evaluation models to evaluate the dependability measures. These measures can also be evaluated directly form the collected data.

Output

The outputs of experimental evaluation activities and the experiments performed should be described in detail to allow the assessors to check the validity of the evaluations and to asses their completeness with respect to specified objectives.

Use of the Results

The outputs of experimental evaluation are needed to evaluate the dependability level achieved by the system. The verification of the results obtained is therefore important to analyse the validity of the design decisions, and to ensure that the dependability level achieved before operation is satisfactory. If the results are not satisfactory, modification of the system design or additional testing of the system is required to remove residual faults.

Relation to System Lifecycle

Experimental evaluation is to be applied once a real implementation of the system or its components is available. The evaluations should be updated when data is collected during testing. In particular, the data collected during the testing of the overall system under operational conditions should be used to evaluate the dependability measures of the final system before its delivery. Evaluations are to be carried out also during the operation of the system for maintenance planning and to provide feedback for future developments.

Assessor’s activities

The main role of the assessors is to examine:

- the representativity and the validity of the assumptions considered,
- the validity of the methods used and the experiments performed,
- the results of evaluation to check whether the objectives are satisfied.

4.4.2.4 Dependability Validation Activities: Summary

Table 15 summarises the mapping between dependability validation activities and the confidence levels.

Dependability Validation Activities	Level 1	Level 2	Level 3	Level 4
Penetration Analysis	RL1, DL1, IL1	RL2, DL1, IL2	RL3, DL2, IL2	RL3, DL3, IL3
Covert Channel Analysis	-	RL1, DL1, IL1	RL2, DL2, IL2	RL3, DL3, IL3
Experimental Evaluation	RL1, DL1, IL1	RL2, DL1, IL1	RL3, DL2, IL2	RL3, DL2, IL3

Table 15: Relation between Dependability Validation Activities and Confidence Levels

The selection of dependability validation confidence providing activities and methods depends on the dependability attributes of the system, the life cycle phases that have to be addressed by the evaluation and the target confidence level. In addition, the selection is influenced by sector specific or system specific needs. In particular, some factors such as the system complexity, the novelty of the design, the types of technologies used, and the availability of experimental data on similar products and systems, should be considered to find a right mix of the validation activities to be used, and corresponding levels of rigor, detail and independence. Moreover, alternative validation methods, which are not included in the Criteria, can be used, provided that convincing arguments and justifications are given to demonstrate the suitability and efficiency of these methods. When alternative techniques or alternative combination of levels of rigour, detail and independence are specified, the dependability target document shall specify, which of the alternative techniques shall be used and which alternative has to be taken in the definition of the rigour, detail and independence levels. The choice may be restricted by the demand to be compliant with a sector specific standard.

4.5 Process Quality

4.5.1 Objective

4.5.1.1 Introduction

The objective of Process Quality (PQ) is to ensure that the system development is properly structured, planned and controlled, in particular that:

- there is a defined system lifecycle (for the whole life of the system from conception to last decommissioning)
- there are appropriate plans and procedures to control the development, operation, maintenance, and decommissioning of the system
- there is adequate evidence that the plans and procedures are followed in practice.

It should be noted that the quality checks defined in these Criteria are not the same as those defined in quality assurance standards such as ISO 9000, which are concerned with the controls implemented by an organisation. These Criteria address only those quality checks which are specific to a particular dependable system, which is the subject of an assessment.

4.5.1.2 Lifecycle

Confidence in a dependable system may be increased if its passage through all stages of its life are properly controlled.

Firstly there should be a defined and structured lifecycle which identifies the stages through which the system can pass. In general the lifecycle will define the activities that can be applied to the system at each stage, the inputs to these activities, the outputs from these activities and the documentation that will be produced.

4.5.1.3 Plans and Procedures

Within the structure of the lifecycle, all activities must be planned with adequate resources. Different organisations may be responsible for different parts of the system lifecycle, for instance, a company may be responsible for the development and installation of a system but not its, maintenance, operation or decommissioning. In the following text, the term project is used to mean those parts of the system lifecycle that a particular organisation is responsible for and can be expected to control. The plans will identify such things as:

- lifecycle
- the project organisation and roles
- interfacing with clients, subcontractors, regulatory authorities and other organisations
- justification of competence for key roles
- dependability and other requirements imposed from outside
- project activities
- project schedules
- project resources
- progress reporting
- review and approval
- project risk management
- documentation to be produced
- quality management
- configuration management
- change management
- procedures to be followed for all activities.

There must be defined procedures for all the project activities. These will define how each activity is to be carried out.

4.5.1.4 Evidence

Evidence of the adequacy of the plans and procedures and the compliance with these is provided in two ways: through quality control and quality assurance (QA).

The quality control provides the development team control activities (such as review and walkthroughs) that ensure that activities have been performed properly. For instance, a plan can be reviewed for completeness, consistency, compliance with contractual and other requirements and compliance with documentation standards. The quality assurance is provided by a person or persons who are independent of the project team. They will:

- provide guidance on the appropriate controls and procedures to be used
- check that the plans and procedures are appropriate and are being applied correctly.

Thus there is one CPA, viz. Quality Assurance (QA).

4.5.2 Quality Assurance

<table>
<tr><td colspan="2"><b><i>Name of the CPA</i></b>
Quality Assurance</td></tr>
<tr><td colspan="2"><b><i>Objectives</i></b>
To provide assurance that the project is properly structured, planned and controlled</td></tr>
<tr><td><b><i>Inputs</i></b>
• project inputs such as contract, company procedures etc.
• plans
• procedures
• outputs from development activities</td><td><b><i>Outputs</i></b>
• guidance
• review comments
• audit reports
• approvals</td></tr>
<tr><td colspan="2"><b><i>Mapping Confidence Levels to Dependability Profile</i></b>
Same level of rigour, detail and independence
The QA personnel are independent of the development team</td></tr>
<tr><td colspan="2"><b><i>Methods and Tools</i></b>
checking, reviewing, witnessing, auditing</td></tr>
</table>

4.5.2.1 Objectives

The objective of QA is to provide assurance that the project is properly structured, planned and controlled. In general, the QA staff will also provide guidance to the development team on such things as the appropriateness on procedures or how to tailor the organisation's standards and procedures to the particular needs of and constraints of the project.

4.5.2.2 Inputs

The inputs to the QA activities are:

- all the inputs to the project (such as the contract, externally imposed requirements, organisation standards and procedures, external standards etc.)
- project plans and procedures
- other project outputs
- the project activities.

4.5.2.3 Description of the CPA

Two distinct groups of people are involved in QA – the development team and the QA staff.

The development team provide the bulk of the inputs to the QA process by:

- producing appropriate plans and procedures to control all aspects of its work on the project
- implementing these plans and procedures correctly.

The QA staff provide assurance that the appropriate plans and procedures are in place and that they are implemented correctly. They will do this by

- providing guidance to the development team on the appropriateness of organisation or other standards and procedures
- checking in various ways that the development team is structuring, planning and controlling the project properly.

4.5.2.4 Levels of Rigour, Detail and Independence

The same level of rigour, detail and independence will be used regardless of the Dependability Profile.

The QA staff shall be independent of the development team (including the project manager).

4.5.2.5 Methods and Tools

The QA staff will use a number of techniques that will generally include:

- participating in reviews etc.
- witnessing reviews
- checking the level and type of review comments from development team reviewers
- being part of the approval process
- witnessing testing
- auditing the development team against some or all of the plans and procedures
- checking the knowledge of the staff involved to ensure that there is general familiarity with the processes being used and that they are therefore being effective

- providing guidance to the development team on the appropriateness of organisation or other standards and procedures.

The QA staff are not necessarily expected to have all the technical expertise of the development team. For instance, they might well review a document for compliance with documentation standards but would not necessarily review it technically. Rather, they would be expected to obtain assurance that appropriate staff have performed the technical review properly.

The details of the QA activities will be defined in the Quality Plan and other quality related plans and procedures.

4.5.2.6 Output

The output from the QA activities will be defined in such documents as the Quality Plan and Quality Procedures but will typically include such things as:

- guidance to the development team on the appropriateness of procedures etc.
- review comments
- approval signatures
- audit reports.

4.5.2.7 Use of the Results

The results of the QA activities will improve the management and control of the development team work. For instance:

- review comments will can a item to be modified
- notification of non-adherence to a procedure will cause the development staff to operate differently or cause the procedure to be changed
- withholding an approval signature may prevent the project progressing to the next stage (e.g. from design to manufacture) until something is rectified.

Normally, any QA comments on the project will be directed to the project staff who will normally agree a response with the QA staff. Exceptionally, if the project staff's response is not acceptable, then the QA staff will use their independent reporting line to senior management who can then exert suitable pressure on the project.

4.5.2.8 Relation to the System Lifecycle

The QA activities run in parallel with the development team activities, that is, throughout the whole lifecycle.

4.5.2.9 Assessor Activities

The assessor will do sufficient checking of the QA and development team activities to gain adequate confidence that all the Criteria requirements have been met.

It recommended that, where possible, the assessor QA activities are carried out at the same time as other assessor activities. For instance it may be simple to check that the right review and approval actions have been carried out on a hazard analysis at the same time as checking its technical accuracy and completeness.

The assessor can use any appropriate technique. They can include:

- checking documents, conformance certificates etc. for appropriate signatures
- auditing the QA staff
- repeating QA activities such as reviewing documents, auditing the development team, witnessing testing.

4.6 Mapping between Attributes, Levels and Actions

4.6.1 The Mapping Table

Table 16 gives the mapping from the Dependability Profile (Attributes and Confidence Levels) to CPAs and levels of Rigour, Detail and Independence. The mapping is independent of the Dependability Attribute except for Confidentiality.

Each entry in the tables gives:

- the recommendation of whether the CPA should be used
- the required levels of Rigour, Detail and Independence if the CPA is used.

The recommendations are indicated as follows:

- There is no recommendation for or against the use of this CPA.

R The technique is recommended and the CPA should be used with the specified levels of Rigour, Detail and Independence. However, it might be possible to justify using a lower level of Rigour, Detail and/or Independence if this is compensated by using another CPA from the same CPP with a higher level of Rigour, Detail and/or Independence. For instance, the use of formal methods with proofs can compensate for a reduced level of testing (possibly at the module level) but it cannot replace testing entirely.

HR The CPA is highly recommended. A very strong justification must be provided if the CPA is used at a level below the required levels of Rigour, Detail and Independence.

The required levels of Rigour, Detail and Independence are given as:

RLn, DLn, ILn	when different levels are possible for the CPA (see the descriptions of the CPAs in the earlier sections of this chapter for details)
one level	when only one level is possible
-	Since there is no recommendation for or against the use of the CPA, no level of Rigour, Detail and Independence are identified

4.6.2 Selecting the Appropriate Dependability Profile

In general, different components of the system will have different Dependability Profiles. Figure 9 shows an example of a system decomposed into components with different Dependability Profiles.

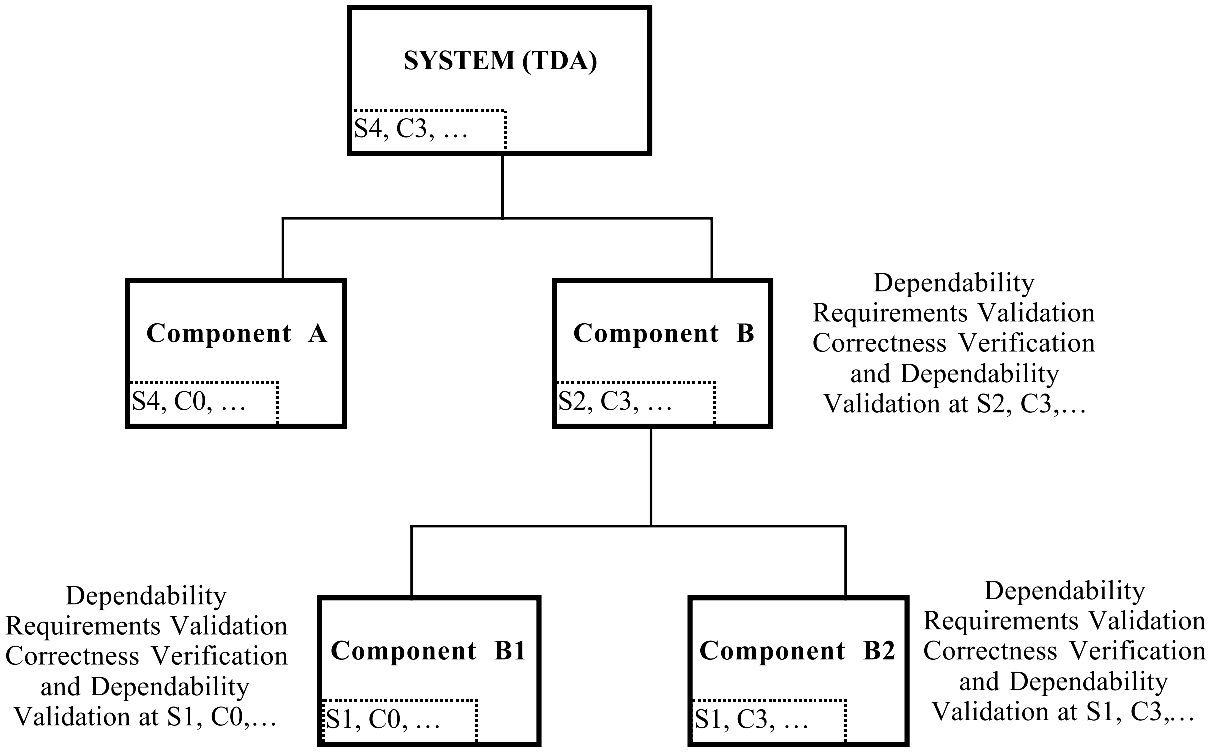
SYSTEM (TDA)
S4, C3, …
Component A
S4, C0, …
Component B
S2, C3, …
Dependability Requirements Validation Correctness Verification and Dependability Validation at S2, C3,…
Dependability Requirements Validation Correctness Verification and Dependability Validation at S1, C0,…
Component B1
S1, C0, …
Component B2
S1, C3, …
Dependability Requirements Validation Correctness Verification and Dependability Validation at S1, C3,…

Figure 9: Example of Different Dependability Profiles

For each component, the Dependability Requirements Validation, Correctness Verification and Dependability Validation CPAs should be applied with levels of Rigour, Detail and Independence in accordance with the associated Dependability Profile. During the system decomposition process, a hierarchy of Dependability Profiles might be defined with a Dependability Profile associated with each component. The Dependability profiles of the sub-components corresponding to a given hierarchical level, together with the associated

Dependability Objectives and Dependability Related Functions should be validated taking into account the Dependability Profile, Dependability Objectives and Dependability Related Functions of the corresponding component belonging to the immediately superior hierarchical level. The validation of the dependability refinement and allocation will be based on the CPAs defined in Table 16 with levels of Rigour, Detail and Independence corresponding to the dependability profile of the latter component.

If we take the example of components B, B1 and B2, the Confidence Providing activities should be applied as follows:

- Use CPAs with levels of Rigour, Detail and Independence corresponding to (S2, C3) to ensure that the decomposition of (S2, C3) into (S1, C3) for B2 and (S1, C0) for B1 is correct and valid (i.e., satisfies the requirements of B).
- Use CPAs with levels of Rigour, Detail and Independence corresponding to (S1, C3) for B2 and (S1, C0) for B1, respectively to ensure that each component satisfies its allocated requirements.

In all cases, the CPAs must be applied with respect to each Dependability Attribute at the confidence level for that attribute. For instance, in the above example, Correctness Verification testing of component B2 needs to be performed at Confidence Level 3 for confidentiality functionality and at Confidence Level 1 for safety functionality.

The developer would not be prevented from performing a CPA at a higher Confidence level than that required by the Criteria if it was thought to be more efficient. For example, in the case of component B2, the developer might decide to perform one hazard analysis at Confidence Level 3 and address both safety and confidentiality.

CPP/CPA	Attributes	Confidence Level			
		1	2	3	4
Dependability Requirements Validation					
PHA	All	HR one level	HR one level	HR one level	HR one level
Probabilistic Quantitative Evaluation	All but confidentiality	R RL1, DL1, IL1	HR RL2, DL1, IL1	HR RL3, DL2, IL2	HR RL3, DL3, IL3
Common Cause Analysis	All	R RL1, DL1, IL1	HR RL2, DL1, IL2	HR RL2, DL2, IL2	HR RL2, DL2, IL3
Hazard Analysis	All	HR RL1, DL1, IL1	HR RL2, DL1, IL2	HR RL3, DL2, IL2	HR RL3, DL2, IL3
Correctness Verification					
Static Analysis	All	HR RL1, DL1, IL1	HR RL2, DL2, IL1	HR RL2, DL3, IL2	HR RL3, DL3, IL3
Behavioural Analysis	All	R RL1, DL1, IL1	R RL2, DL1, IL1	HR RL3, DL2, IL2	HR RL3, DL3, IL3
Formal Methods & Proofs	All	- -	R RL1, DL1, IL1	R RL2, DL1, IL2	HR RL3, DL2, IL3
Testing	All	HR RL1, DL1, IL1	HR RL1, DL2, IL1	HR RL2, DL3, IL2	HR RL3, DL3, IL3
Traceability Analysis	All	HR RL1, DL1, IL1	HR RL1, DL1, IL1	HR RL2, DL2, IL2	HR RL2, DL2, IL2
Dependability Validation					
Penetration Analysis	All	R RL1, DL1, IL1	HR RL2, DL1, IL2	HR RL3, DL2, IL2	HR RL3, DL3, IL3
Covert Channel Analysis	Confidentiality	- -	HR RL1, DL1, IL1	HR RL2, DL2, IL2	HR RL3, DL3, IL3
Experimental Evaluation	All	HR RL1, DL1, IL1	HR RL2, DL1, IL1	HR RL3, DL2, IL2	HR RL3, DL2, IL3
Process Quality					
Quality Assurance	All	HR one level	HR one level	HR one level	HR one level

Table 16: Selection of CPAs by Dependability Confidence Profile

5 Mapping to existing standards

The SQUALE Criteria have been designed to assess dependable systems for any industrial sector. But in most industrial sectors, critical computing systems must be certified with respect to existing standards, or will have to be compliant with future standards. Even if the SQUALE Criteria present some advantages on other standards (e.g., by covering all dependability attributes, rather than only safety or security), it would be very costly to realise independent assessments, for the SQUALE Criteria, and for the sector-specific standard. That is why the SQUALE Criteria have been designed to be flexible enough to adopt confidence providing techniques that may be required in specific standards. This flexibility should facilitate the assessment, with respect to a given standard, of a system previously assessed by using SQUALE Criteria. The aim of this chapter is to show how the SQUALE Criteria may be adapted to facilitate the assessment of a system according to existing standards (ITSEC, IEC 1508, IEC 880, EN 50126, EN 50128, DO 178B). In each case, are indicated which complementary activities required by these standards are not covered by the SQUALE Criteria.

On the other hand, this flexibility should enable certification bodies to define more precise instantiations of the SQUALE Criteria, adapted to a particular sector or class of applications. This could lead to new sector-specific standards, that would cover all the dependability requirements which can be defined in this sector.

5.1 ITSEC

5.1.1 Introduction

The “Information Technology Security Evaluation Criteria”, commonly known as "ITSEC", is a widely known and accepted basis for the security evaluation of IT-systems and products in Europe.

It was derived as a harmonised approach from the schemes that had been defined in the UK, France, Germany and the Netherlands. The ITSEC was published in 1991 by the European Commission [CEC 91] and is a major input document for the Common Criteria, where the IT Security Evaluation approaches of North America and Europe are harmonised.

5.1.2 Overview and Scope of ITSEC

The ITSEC defines Security as “the combination of confidentiality, integrity and availability”, so it focuses on these three attributes of dependability. Nevertheless, in practice, most ITSEC evaluations have been done principally against confidentiality requirements with attention being paid to integrity or availability.

The evaluation approach of the ITSEC can be summarised as follows:

- For each product or system that is going to be evaluated, a Security Target document must be written.

- The ITSEC then define 7 evaluation assurance levels (E0 to E6) where E0 is the lowest and E6 the highest level.
- The assurance levels define the inputs and actions needed to assess the correctness of the "Target of Evaluation" with increasing rigour and detail. In addition the ITSEC define actions to assess the "Effectiveness" of the Target of Evaluation. The following areas are covered by those Effectiveness criteria:
    - Suitability
    - Ease of Use
    - Strength of Mechanism
    - Binding
    - Construction Vulnerabilities
    - Operational Vulnerabilities

The ITSEC are accompanied by the IT Security Evaluation Manual (ITSEM) which describes the evaluation methodology in detail. In addition, country specific schemes define the process how to get an ITSEC certificate.

5.1.3 Similarities

The ITSEC defines the Target of Evaluation (TOE) as "an IT system or product which is subjected to security evaluation". This is the equivalent to the SQUALE criteria's Target of Dependability Assessment (TDA) with respect to the corresponding Dependability Attributes of security.

Also the Security Target of the ITSEC is very similar to the Dependability Target of the SQUALE criteria (see Section 3.5).

The ITSEC does not prescribe specific functional requirements but define generic headings that should be used to describe functional requirements in the Security Target document. This is nearly the same as in the criteria where no specific Dependability related functions are required to be integrated into the system.

5.1.4 General differences

The ITSEC consider the attributes integrity, confidentiality and availability of security whereas the SQUALE Criteria expand these to the concept of Dependability with the additional attributes safety, reliability and maintainability. This leads to a precise and subtle description of the criticality of a system with the Dependability Profile.

In practice, integrity and availability often have not been treated properly. The main emphasis in Security Targets has been on confidentiality with few or no Security Enforcing Functions for availability and integrity.

The ITSEC deal mainly with the assessment of products and system prior to their use. They do not deal much with the development process or the system operation.

The SQUALE Criteria give more guidelines for the development process, especially for the concept exploration phase and the definition phase.

In addition to the SQUALE Criteria, the ITSEC describe some functionality classes in the annex which can guide the design of Dependability related functions, especially those supporting the attributes of security.

5.1.5 Confidence Levels

5.1.5.1 Introduction

The ITSEC then define 7 evaluation assurance levels (E0 to E6) where E0 is the lowest and E6 the highest level. For each level the document describes:

- The requirements for the sponsor of the evaluation
- The requirements for evidence
- The actions that the assessors have to perform.

The ITSEC has a E0 to mean there is no confidence in the security of the TOE. A TOE either passes its evaluation and achieves the required E level (say E3) or it fails and achieves E0. The Criteria should allow an expected Dependability Profile of say A4, C2, R3, I2, S3, M3 and an achieved Dependability Profile of A3, C1, R3, I1, S3, M2.

There is (only) one overall assurance level Ex, x $\in$ {1, 2, 3, 4, 5, 6}, for a product or system in the ITSEC.[2] This may lead to an assurance level Ey which is the lowest common denominator of the capabilities of the product or system according to security. The Dependability Profile describes precise the different capabilities of a system in terms of the attributes of Dependability[3].

As there is not a single overall level in the SQUALE Criteria, there can be no direct relation between an E-level (ITSEC) and a Dependability Profile (SQUALE) or a Confidence Level. Paragraph 1.2 of the ITSEC allows for different E levels for different security functions and allows different Security Targets for the different levels. This has often been interpreted by allowing an assurance profile where different components of the TOE can have different E levels but there must be sufficient separation between the components at the different levels.

5.1.5.2 Confidence Providing Activities (CPAs)

The Confidence Providing Activities in the SQUALE Criteria help the acquirer to get appropriate requirements for the system, the developer to build a system with the required properties and the supplier to provide a system which fulfils demonstrably the required properties.

[2] A system in the ITSEC is a "specific IT-installation, with a particular purpose and operational environment".

[3] A system in the SQUALE criteria is "an integrated composite (people, products and processes) that provide a capability to satisfy a stated need or objective, bound together to interact".

In this section these CPAs are treated which are mentioned in the ITSEC and in the SQUALE Criteria. The CPAs from the SQUALE Criteria which do not appear in this section, are not mentioned in the ITSEC.

Dependability Requirements Validation Activities

For every E-level the ITSEC requires a "vulnerability analysis" which corresponds to the SQUALE PHA. With increasing level number the vulnerability analysis is done with increasing level of rigour and detail. It is done by the assessor independently from the developer.

Correctness Verification Activities

For every E-level the ITSEC requires testing and tracing. At level E1 there is no independence between the assessor and the developer required. From level E3 "re-testing" is required to be done by the assessor.

From level E4 "formal models of the security enforcing functions" are required to be done by the developer/sponsor. The assessor has to check them.

Dependability Validation Activities

From level E2 penetration testing is required to be done by the assessor.

Process Quality Assessment

Is not mentioned explicitly in the ITSEC.

5.1.5.3 Level of Rigour, Detail and Independence

It is required by the ITSEC that the assessor is an independent person or organisation.

5.1.6 Dependability Target

For each product or system that is going to be evaluated, the ITSEC requires a Security Target to be written, which contains:

- a description of the security functionality of the product or system,
- the target evaluation level,
- a description of the system security policy or a product Rationale,
- a description of the threats the product or system is going to counter,
- a description of the environment (in case of a system) or the assumptions on the environment (in case of a product),
- a description of the security objectives,
- an (optional) high level description of the mechanisms that implement the security functions.

This Security Target is used as a baseline for the evaluation.

This approach is nearly the same as in the SQUALE Criteria. In addition and with respect to all Dependability Attributes the SQUALE Criteria requires the following:

- the initial needs of the system,
- an (initial) description of the system and its environment,
- the Dependability Profile,
- the Dependability related functions.

5.1.7 Lifecycle

The ITSEC treats the Construction Phase with the Development Process and the Development Environment, and the Operation Phase of the system to be evaluated. This means:

During the Development Process the ITSEC focus mainly on the required deliverables, e.g. a formal model, and not on the process itself. For the Development Environment especially the Configuration Control is evaluated. For E-levels Higher than E2 there are requirements for the programming languages.

For the Operations Process only the documentation for the user and the Administration documentation is evaluated together with information about the Configuration and the Delivery Procedures together with the Start-up procedures.

5.2 IEC 1508

5.2.1 Introduction

This section presents the mapping between IEC 1508 and the Criteria.

The mapping uses the 1995 draft of IEC 1508. At the time of writing (Summer 1998) IEC 1508 is undergoing revision and is now called IEC 61508.

5.2.2 Overview

IEC 1508 sets out a generic approach for all safety lifecycle activities for achieving functional safety in electrical/electronic/programmable electronic systems (E/E/PESs). Assessment is addressed as part of the approach.

Although parts of the standard suggest (particularly when describing the relationship of the E/E/PES to the Equipment Under Control) that it is aimed at control systems of varying complexity, other parts of the standard and discussions with the authors of the standard make it clear that it applies to all safety related E/E/PESs. For instance, the standard would apply to a standalone computer system with a medical database that decides (or strongly influences) the drug dosages for a patient.

The standard also provides an overall framework that is applicable to any safety related system regardless of the technology used. It gives only partial information about the activities at this level.

5.2.3 General Similarities

Like the Criteria, IEC 1508 is generic in that it provides a unified approach that can be adopted when generating application sector standards.

Both IEC 1508 and the Criteria require the developer to perform the activities that will provide confidence in the safety/dependability of the system.

5.2.4 General Differences

IEC 1508 addresses only safety whilst the Criteria address 6 attributes of dependability, of which safety is one.

IEC 1508 addresses the E/E/PES and software safety lifecycles within the whole system safety lifecycle (from Concept to Decommissioning). The Criteria address only the development of a system.

IEC 1508 addresses all the activities needed to develop the E/E/PES including those (both development and assessment ones) that will provide confidence in its safety. The Criteria concentrates on the developer's CPAs and the associated assessment activities.

IEC 1508 does not require a Dependability Target although the contents of the Dependability Target are distributed among several documents that are required by the standard.

5.2.5 Dependability Target

IEC 1508 does not define a Dependability Target because the primary thrust of the standard is aimed at the safety management of the whole lifecycle of the safety related E/E/PES - assessment is only one activity within the lifecycle.

The information in the Criteria Dependability Target is contained in a number of IEC 1508 documents including the:

- Safety Plan
- Overall Concept Description
- Overall Scope Definition Description
- Hazard and Risk Management Description
- Overall Safety Requirements Specification
- Safety Requirements Allocation Description
- Software Safety Requirements Specification
- E/E/PES Safety Requirements Specification.

5.2.6 Confidence Levels

5.2.6.1 Introduction

In IEC 1508 there are four safety integrity levels (SILs), SIL 1 (lowest) to SIL 4 (highest). All activities are mandatory for all SILs. The activities are essentially equivalent to the CPAs within the Criteria but functional safety assessment is included as one of the activities. The SIL affects:

- the level of independence of the personnel carrying out the activity
- the techniques used for an activity
- the allowable architecture of the E/E/PES.

5.2.6.2 CPAs

Table 17 shows how the IEC 1508 activities map to the Criteria CPPs and CPAs. Some notes are also provided to give more information about the nature of the mapping.

The table is organised by the Chapter headings within Parts 1 to 3 of IEC 1508. The parts address the following:

- Part1 - General Requirements. System level aspects are addressed in this part so that the requirements for the E/E/PES and software safety lifecycles can be put into context.
- Part 2 - E/E/PES Requirements
- Part 3 - Software Requirements.

IEC 1508 Requirements	**Criteria CPA**
System Level Requirements (IEC 1508 Part 1)	
Competence of Persons	part of Process Quality, (*IEC 1508 provides more detailed requirements*)
Safety Management	part of Process Quality, *(IEC 1508 provides more detailed requirements)*
Overall Safety Lifecycle	part of Process Quality
Concept	part of Dependability Target Construction
Overall Scope Definition	part of Dependability Target construction
Hazard Analysis	Preliminary Hazard Analysis and Hazard Analysis
Overall Safety Requirements	part of Dependability Target construction
Safety Requirements Allocation	part of Dependability Target construction
Overall Operation and Maintenance Planning	*not addressed because the Criteria address only development*
Overall Validation Planning	part of Process Quality

IEC 1508 Requirements	**Criteria CPA**
Overall Installation and Commissioning Planning	*not addressed because the Criteria address only development*
Realisation : E/E/PES	see E/E/PES Requirements below
Realisation : Other Technology *(identified in IEC 1508 but no requirements provided)*	-
Realisation : External Risk Reduction Facilities *(identified in IEC 1508 but no requirements provided)*	-
Overall Installation and Commissioning	*not addressed because the Criteria address only development*
Overall Safety Validation *(IEC 1508 does not suggest any techniques beyond test and analysis)*	Dependability Validation
Overall Operation and Maintenance	*not addressed because the Criteria address only development*
Overall Modification and Retrofit	Change Control within Process Quality
Decommissioning	*not addressed because the Criteria address only development*
Verification	Verification Planning is part of Process Quality Verification is covered in Correctness Verification
Functional Safety Assessment	Assessment activities *(IEC 1508 gives independence levels)*
Documentation	part of Process Quality
E/E/PES Requirements (IEC 1508 Part 2)	
Personnel Competence *IEC 1508 refers back to Part 1*	part of Process Quality
Safety Management *IEC 1508 refers back to Part 1*	part of Process Quality
E/E/PES Safety Lifecycle	part of Process Quality
E/E/PES Safety Requirements Specification	Dependability Target construction
E/E/PES Validation Planning	part of Process Quality
E/E/PES Design and Development *IEC 1508 provides a lot of detail on allowable architectures and design techniques*	the decomposition into components is implicit in Dependability Target construction but the design techniques etc. are not addressed.

IEC 1508 Requirements	**Criteria CPA**
E/E/PES Integration *consists of integration and testing*	integration is implicit in Correctness Verification and Dependability Validation integration testing is addressed in Correctness Verification
E/E/PES Operation and Maintenance Procedures	*not addressed because the Criteria address only development*
E/E/PES Safety Validation	Dependability Requirements Validation and Dependability Validation
E/E/PES Modification	Change Control within Process Quality
E/E/PES Verification *includes verification planning*	Verification Planning is part of Process Quality Verification is covered in Correctness Verification
Functional Safety Assessment *includes independence levels* *the form of the activities is not clear*	Assessment activities
Documentation	part of Process Quality
Software Requirements (IEC 1508 Part 3)	
Competence of Persons *refers back to Part 1*	part of Process Quality
Safety Management *refers back to Part 1*	part of Process Quality
Software Safety Lifecycle	part of Process Quality
Software Safety Requirements Specification	part of Dependability Target construction
Software Validation Planning	part of Process Quality
Software Design and Development *includes module testing and software integration and testing*	Software decomposition is implicit in Dependability Target construction Testing is part of Correctness Verification
Programmable Electronics (PE) Integration (H/W & S/W) *integration and testing of software onto the hardware*	Integration is implicit in Correctness Verification and Dependability Validation Integration Testing is part of Correctness Verification
Software Operation and Maintenance Procedures	*not addressed because the Criteria address only development*
Software Safety Validation *includes independence levels* *only testing techniques identified*	Testing within Dependability Validation
Software Modification *includes independence levels for modification approvals*	Change Control within Process Quality

IEC 1508 Requirements	**Criteria CPA**
Software Verification *includes verification planning*	Verification Planning is part of Process Quality Verification is covered in Correctness Verification
Functional Safety Assessment *includes independence levels* *the form of the activities is not clear*	Assessment activities
Documentation	part of Process Quality

Table 17: Mapping between IEC 1508 Activities and Criteria CPPs and CPAs

5.2.6.3 Level of Rigour

IEC 1508 allows different levels of rigour at the activity level (e.g. design) by recommending different techniques for different SILs. Degrees of rigour are not specified within a technique.

5.2.6.4 Level of Detail

Levels of detail are not specified in IEC 1508. All activities are required for all SILs and there are no levels of detail specified for the various techniques.

5.2.6.5 Level of Independence

IEC 1508 recommends different levels of independence (independent person, department or organisation) for:

- performing Functional Safety Assessment
- performing Software Safety Validation
- authorising Software Modification.

5.2.7 Lifecycle

IEC 1508 defines:

- the System Lifecycle, the System Safety Lifecycle within the System Lifecycle, the E/E/PES Safety Lifecycle within the System Safety Lifecycle, the Software Safety Lifecycle within the E/E/PES Safety Lifecycle,
- the activities needed for each stage of the E/E/PES Safety Lifecycle and the Software Safety Lifecycle (and some activities for the System Safety Lifecycle).

5.2.8 Assessment activities

IEC 1508 requires a Functional Safety Assessment activity. The independence of the assessor depends on the SIL.

Assessment actions are not defined in the standard - they will be defined in the Functional Safety Assessment Plan. However:

- IEC 1508, Part 1, Clause 8, describes the Functional Safety Assessment activities in general terms
- IEC 1508, Part2, Clause 8, E/E/PES Functional Safety Assessment, refers back to Part 1
- IEC 1508, Part 3, Clause 8, Software Functional Safety Assessment, refers back to Part 1 and additionally defines:
    - the software activities/deliverables that have to be assessed
    - the level of independence for the assessor
- IEC 1508, Part 3, Annex A, Table C.8 gives a list of recommended techniques for Software Functional Safety Assessment.

The list of recommended assessment techniques includes checklists, software complexity metrics, event tree analysis, FMECA, HAZOPS, Monte-Carlo simulation. This suggests that the assessor is expected to do some analysis rather than just check the project outputs.

5.3 EN 50126

5.3.1 Introduction

This section presents the mapping between SQUALE Criteria and EN 50126 standard. It points out the differences and suggests how SQUALE Criteria have to be completed, if necessary, to be compliant with this standard.

Notice that this standard is the top one of some other standards which are deduced from it: EN 50128 for the software, EN 50129 for electronic systems and EN 50159-1/2 for communication in (closed/open transmission systems).

REF 01	EN 50126: Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS). (final draft – may 1998)
REF 02	EN 50128: Railway applications – Software for railway control and protection systems.
REF 03	EN 50129: Railway applications – Safety related electronic systems for signalling
REF 04	EN 50159_1: Railway applications – Communication, signalling and processing systems. Part 1: Safety-related communication in closed transmission systems.
REF 05	EN 50159_2: Railway applications – Communication, signalling and processing systems. Part 2: Safety-related communication in open transmission systems.

5.3.2 Overview

The EN 50126 standard provides a general process to manage RAMS requirements for railway applications. EN 50126 is addressed to the manufacturer (design, development, verification and validation) and end-user. It doesn't give explicitly objectives to the assessor or legal authority. Nevertheless, some assessment requirements have to be checked for the system acceptance (§6.10).

This standard concerns all levels of railway application (from system level down to software or hardware components) (§1.2) and covers all the lifecycle

EN 50126 defines RAMS objectives but introduce the new notion of "quality of service" which is the users (passengers for a railway system) point of view (§4.2).

- for RAM objectives, 3 classes are defined : significant, major, minor (§4.5.2.2);
- for safety objectives, probability of occurrence and gravity are defined to build a risk evaluation and acceptance matrix.

Six frequency categories are defined: incredible, improbable, remote, occasional, probable, frequent.

Four safety integrity levels (S.I.L.) are defined (§4.6):

Level 1:	insignificant
Level 2:	marginal
Level 3:	critical
Level 4:	catastrophic

It is assumed that Level 0 is "no safety-related".

The risk is evaluated depending on the SIL and the frequency of occurrence of hazardous events.

A frequency crossed SIL table gives the risk level evaluation and acceptance (§4.6.3.4 tab. 6) depending on the safety objectives (contract, standard, legal rules,….). Four categories are proposed: intolerable, undesirable, tolerable, negligible.

EN 50126 doesn't give any quantitative objectives (§1.1).

EN 50126 is convenient to new systems or to old systems to be updated (§1.2).

EN 50126 supposes the manufacturer provide an assurance quality process compliant with ISO 9000 standards (§5.3.5 d).

EN 50126 takes into account different points of view to accept a risk: ALARP, GAMAB, MEM. (§4.6.3.3.).

EN 50126 takes into account cost effectiveness of FDMS activities.(it is compliant with the above principle) (§5.2.3.).

5.3.3 General similarities

EN 50126 is based on a traditional system development lifecycle split in several phases from the system concept (§6.1) to the system disposal (§5.2, §6.14).

EN 50126 requirements are very close to SQUALE requirements for RAM and Safety.

5.3.4 General differences

EN 50126 is restricted to railway applications.

EN 50126 doesn't take into account (and so, all the other standards deduced from it):

- security aspects (§1.1., §4.3.4.)
- assessment process (§1.1)
- entry into service authorisation by the concerned authority (§1.1).

EN 50126 takes into account the quality of service (linked to FDM attributes): it's the user point of view.

EN 50126 takes into account "human factors" (§4.4.2.4.-§4.4.2.11)

EN 50126 doesn't assign a level of detail/rigour/independence to each criterion but gives general principles depending on SIL.

EN 50 126 takes into account the "influencing factors" depending on the system phase: system conditions, operating condition, maintenance.

EN 50126 doesn't describe the "how" but only the "what" (and "Why").

5.3.5 System lifecycle

SQUALE lifecycle is split into three main phases (Construction phase, Implementation phase and decommissioning phase) themselves split in other sub-phases which are not explicitly described (cf. CPA to identify some of them). EN 50126 proposes another decomposition (§5.2 and fig. 8). For each phase, tasks are detailed (§5.2, fig.9).

5.3.5.1 Construction phase

SQUALE phase	**EN 50126 phase**
Concept exploration and Design	Concept
Design	System definition and application conditions
Design	Risk analysis
Design	System requirements
Design	Apportionment of system requirements
Detailed Design Implementation	Design and implementation
	Manufacture

Integration and test	Installation
	System validation
	System acceptance

5.3.5.2 Operation & maintenance phase

SQUALE phase	**EN 50126 phase**
Operation • Start of operation • Operational control	Operation and maintenance
Maintenance • maintenance procedures	Operation and maintenance cf. §4.3.5 b, c; §4.3.6 c, d, e

Check procedures are performed to warrant the same SIL when a component is modified.

5.3.5.3 Decommissioning phase

SQUALE phase	**EN 50126 phase**	**Comments**
Decommissioning	Decommissioning and disposal	Suggested but not described.

5.3.5.4 Deliverables list

For each system phase (cf. §6.i.4), EN 50126 gives the deliverables but doesn't explicitly describe documents or products. Only, the contents of deliverables is suggested. For each CPA Squale gives the content of deliverables in the output section.

5.3.6 Confidence levels used in EN 50126

Safety integrity level is applied depending on the system specification and risk: §5.2.2., 5.2.3.

Safety integrity level are compliant with SQUALE levels for safety (S1 to S4). But EN 50126 doesn't describe detail/rigor/independence for each activity. In fact, principles are general (cf. §4.7.2, §4.7.5) and applicable to each activity chosen by the developer/designer depending on its SIL (§5.3.6) and with a formal agreement of the end user.

5.3.6.1 Level of rigor

EN 50126 doesn't use the notion of "level of rigor" to manage Techniques and Methods. It is assumed that activities are led in accordance with the SIL (cf. §5.3.6). EN 50126 doesn't describe task or activities. It specifies the needs but not how (by which techniques or methods) to reach them.

5.3.6.2 Level of independence

Depending on the safety integrity level, EN 50126 defines the level of independence between design, verification, validation and assessment (§4.4.2.11):

5.3.6.3 Level of detail

In EN 50126, the level of detail is not explicit. In fact, it is depending on the SIL (cf. §5.3.6).

5.3.6.4 Competence of persons

EN 50126 defines the need about competence of person (§5.3.5 b).

5.3.7 Dependability target

EN 50126 mentions qualitative dependability objectives. It suggests typical RAMS parameters in the informative annex C.

5.3.8 Confidence providing activities

EN 50126 doesn't describe RAMS activities. It recommends, for lifecycle phases in the informative annex B, some tasks and refers to standards for usable methods and tools. Confidence is reached when all the defined activities are correctly performed.

5.3.8.1 Dependability Requirements Validation assessment

EN 50126 proposes RAM and safety activities but does not take into account security, integrity, confidentiality attributes. In its scope, activities (cf. informative annex B) are defined to justify and verify all RAMS requirements.

5.3.8.2 Correctness verification assessment

In EN 50126, verification is performed at each phase (§5.2.9, §5.2.10). It is true for RAMS activities as well as development activities. RAMS process can be seen as a second point of view on the development process. Verification are defined as a supplier responsibility.

5.3.8.3 Dependability validation assessment

In EN 50126, validation is performed at each phase (§5.2.9, §5.2.10). RAMS activities allow to ensure that development process fulfilled the dependability objectives.

Validation is defined as a supplier responsibility.

5.3.8.4 Process quality assessment

EN 50126 assumed (§5.3.5.d.) that process development is compliant with EN ISO 9001, EN ISO 9002 and EN 29000-3 standards. The developer has to write an assurance quality plan.

EN 50126 requires a configuration management system (§5.3.5 e).

5.3.9 Criteria applicability

The SQUALE Criteria seem compliant with EN 50126 standard requirements. Two omissions are identified:

- a link to the "quality of service" from RAM attributes,
- the lack of the hazard log.

5.4 EN 50128

5.4.1 Introduction

The EN 50128 standard covers software for railway control and protection systems. It is a part of a set of railway standards: EN 50126, EN 50129 and EN 50159-1/2. This standard specifies rules and techniques to be applied during the software development and validation depending on its safety integrity level as defined in EN 50126.

REF 01 EN 50126 Railway applications – The specification and demonstration of reliability, availability, maintainability and safety (RAMS)

REF 02 EN 50129 Railway applications – Software for railway control and protection systems.

5.4.2 Overview

EN 50128 describes the goal of the software assessment (chapter 14) and gives a set of general criteria to be satisfied. However, this standard is addressed to the development or verification and validation teams more than the assessment team.

Software Safety Integrity Level (SWSIL) are defined taking into account the risk (probability of occurrence and gravity defined at the system level). According to EN 50126, five safety integrity levels are defined:

Level 0:	not safety-related
Level 1:	low
Level 2:	medium
Level 3:	high
Level 4:	very high

SWSIL are used to class software components. The constraints about independence to be applied to the software are depending on its SIL level.

There is no explicit RAM objective for software.

5.4.3 General similarities

EN 50128 is based on a traditional software development lifecycle split into several phases. Unfortunately, it does not take into account a development process entirely based on formal method and tools.

EN 50128 distinguishes two kinds of activities:

- Verification: to check that the phase products are compliant with their inputs (and then, with their superior phase)
- Validation: at each phase, to check conformity between the products and their specifications.

5.4.4 General differences

EN 50128 does not take into account:

- security aspects
- assessment process
- entry into service authorisation by the concerned authority (§1.1).

So, EN 50128 is focused only on safety aspects. However, described activities or methods can also meet other RAM objectives.

EN 50128 does not describe the decommissioning phase.

EN 50128 considers the possibility of re-using software (already validated and assessed).

EN 50128 considers the use of generic software configured by specific data or COTS software.

EN 50128 does not assign a level of detail/rigour/independence to each criterion but gives general principles depending on the software SIL.

5.4.5 Software lifecycle

SQUALE lifecycle is split into three main phases (Construction phase, Implementation phase and decommissioning phase) themselves split in other sub-phases. Applied to software, it is easy to cross-reference with EN 50128 sub-phases.

5.4.5.1 Construction phase

SQUALE phase	EN 50128 Phase	EN 50128 Phase
Design phase	S/W requirements specification Software architecture	Software assessment
Implementation	S/W design and implementation S/W verification and testing Software/hardware integration	Software assessment
Integration and test	Software validation	Software assessment

Software development is performed in a top-down approach up to the smaller software component (which can be described).

5.4.5.2 Operation and maintenance phase

SQUALE phase	EN 50128 Phase
Start of operation	not described (out of software)
Operational control	not described (out of software)
Maintenance procedures	Software maintenance

5.4.5.3 Decommissioning phase

Not described in EN 50128.

5.4.5.4 Deliverables list

EN 50128 gives a list of software documentation or product for each phase.

EN 50128	Phase
SW quality assurance plan	SW plans
SW configuration management plan	SW plans
SW verification plan	SW plans
SW integration tests plan	SW plans
SW/HW integration tests plan	SW plans
SW maintenance plan	SW plans
SW requirements specification	SW requirements
SW requirements test specification	SW requirements

SW requirements verification report	SW requirements
SW architecture specification	SW design
SW design specification	SW design
SW architecture and design verification	SW design
SW module design specification	SW module design
SW module test specification	SW module design
SW module verification report	SW module design
SW source code	SW code
SW source code verification	SW code
SW module test report	SW testing
SW integration test report	SW testing
SW/HW integration test report	SW/HW integration
SW validation report	SW validation
SW assessment report	SW assessment
SW change records	SW maintenance
SW maintenance records	SW maintenance

5.4.6 Confidence levels used in EN 50128

Software safety integrity level is produced in accordance with EN 50126: §5. It is depending on the system specification and risk: §5.2.2, 5.2.3.

Safety integrity levels are compliant with SQUALE levels for safety (S1 to S4). But EN 50128 doesn't describe detail/rigor/independence for each activity. In fact, principles are general and applicable to each activity chosen by the developer/designer depending on its SIL.

5.4.6.1 Level of rigour

Techniques and level of rigor are used in accordance with the software SIL: annex A (normative). EN 50128 distinguishes four classes to recommend or mandate technique:

M	mandatory (the technique has to be used)
HR	highly recommended
R	recommended
-	no request

5.4.6.2 Level of independence

Depending on the safety integrity level of the software, EN 50128 defines the level of independence between design, verification, validation and assessment: §6.2.6., 6.2.8., 6.2.10, 14.4.1.

5.4.6.3 Level of detail

The level of detail is not explicit. In fact, it is depending on the software SIL: §14.4.1, 14.4.6, 14.4.8, and the kind of activity.

5.4.6.4 Competence of persons

EN 50128 defines criteria about competence of persons: §6.2.3, 6.2.4, 6.2.5.

5.4.7 Dependability target

EN 50128 does not mention dependability objectives. It takes into account only the software SIL (based on safety objectives).

5.4.8 Confidence providing activities

EN 50128 describes several activities and, if necessary, gives possible combination of activities. The confidence is reached when all the defined activities (described in the software safety plan) are correctly performed.

Activities and their constraints (mandatory, recommended…) are described in annex A (normative).

5.4.8.1 Dependability requirements validation assessment

EN 50128 standard is related to software, at a step where all dependability and safety needs are already defined from the upper phases. Therefore, there is no such activities dedicated to software.

5.4.8.2 Correctness verification assessment

EN 50128 describes verification activities at each step of the software development cycle. No activities are mandatory but highly recommended taking into account the software SIL.

5.4.8.3 Dependability validation assessment

EN 50128 doesn’t mention dependability objectives. It takes into account only the software SIL (based on safety objectives). Validation software activities are proposed and some of them are mandatory for SIL 3 or SIL 4 software. In fact, it can be implicit that dependability objectives are taking into account by the means of the software specifications.

5.4.8.4 Process quality assessment

EN 50128 gives specific criteria in §15. It is assumed that software development is compliant with EN ISO 9001 (highly recommend) and EN 29000-3 standards. The developer has to write a software assurance quality plan.

5.4.9 Criteria applicability

SQUALE criteria concern a larger scope than the CENELEC EN 50128 standard applicable to railway safety software. Limited to the same scope, they are compliant with this standard because criteria are generic but standard is more accurate because it is specific to software process. EN 50128 standard has to be completed by specific activities in case of dependability or security attributes.

5.5 IEC 880

5.5.1 Introduction

The standard covers software for safety systems of nuclear power stations and provides extensive guidance for all the lifecycle phases of highly reliable software development and operation and for the corresponding verification and validation activities.

5.5.2 Overview

IEC 880 is not a standard focused on assessment methods description. However it specifies safety functions to be use in designing nuclear power stations instrumentation control software and gives recommendations for the use of techniques and tools during software production and operation.

5.5.3 General Similarities

Software lifecycle and documentation to be produced at the end of each phase are similar to SQUALE development documentation and correctness documentation.

IEC 880 verification activities are quite similar to SQUALE correctness verification activities, except for behavioural analysis which is missing. Generally IEC 880 gives more importance to testing activities at each level of software decomposition than to other activities.

Level of independence although limited to two levels in IEC 880 is introduced for testing activities.

5.5.4 General differences

IEC 880 validation corresponds to SQUALE testing activity (correctness verification) and must not be confused with SQUALE validation activities. SQUALE validation activities do not have corresponding IEC activities. Hazard analysis is not included as a required study, however techniques to avoid errors are prescribed with functions to meet the single failure criteria.

Notions of Confidence level, Rigour level, level of detail and competence of persons are unknown in this standard.

5.5.5 Confidence levels used in IEC 880

5.5.5.1 Introduction

IEC 880 used the concept of function classes according to IEC publications 643 and 639. The standard applies only to high reliable software which are used to generate safety functions (Class 1 functions defined in the IEC publication 643), non-safety functions are defined by publication 639.

There is no direct links between these two function levels with the five confidence level defined in SQUALE, however class 1 functions could be mapped to SQUALE level 3 and above, while the second class could be mapped to SQUALE level 2 and below.

5.5.5.2 Confidence Providing Activities

In IEC 880 the description of Confidence Providing Activities are scattered from chapter 4 to chapter 8. Following paragraphs state the correspondence with SQUALE activities, only for the construction phase.

Correctness verification assessment

Correctness verification activities of IEC 880 covers quite well SQUALE activities. Verification of Dependability requirements, design, implementation, integration, operation and maintenance are realised by a set of techniques described in IEC 880 annexes and covers SQUALE items:

- static analyses;
- formal methods and proof of correctness;
- testing.

Traceability analysis is not treated as rigorously as in SQUALE criteria and behavioural analysis is missing.

The following IEC paragraphs are relative to correctness verification activities:

- §4.9 are requirements for software functional quality requirements that are included in SQUALE correctness verification criteria;
- §6.1 describes the verifications process which is equivalent to SQUALE correctness verification process. Emphasis is given on testing activities;
- § 6.2 fixes verifications activities.
- § 6.2.1 requires a verification plan specifying activities to be done by the verification team;
- § 6.2.2 describes verification activities for design phases. It corresponds to SQUALE correctness verification activities excluding behavioural analysis. Phase concludes with a verification report (see annex F.M5) and a review;

- § 6.2.3 describes verification of the coding phase focused on testing verification (see SQUALE correctness verification, testing activity). §6.2.3 specifies the contents of verification:
    - the software test specifications (with software test procedures);
    - the software test report;
    - the hardware and software integration process and testing.

Chapter 7 links integration of the software to system integration (hardware and software combined) and requires a system integration plan (§7.1 and §7.2) and system integration procedures (§7.4), both including software integration and testing (§7.5). §7.7 specifies system integration testing report.

Chapter 8 is relative to software validation. The IEC 880 validation meaning is equivalent to SQUALE testing activity at system level.

The validation activities consist in validating hardware and software as a system in accordance with the safety system requirements. The validation is conducted in accordance with a formal validation plan which identifies the validation for static and dynamic cases. At the end of the validation a validation report development is produced. This report summarises the results of the hardware+software validation and assesses the system compliance with all requirements.

Dependability validation assessment

Missing in IEC 880.

Process quality assessment

Most of Confidence Providing Activities of Process Quality are missing in IEC 880 for the construction phases. The standard just mentioned the diffusion process of software requirements and requires control of system configuration and problem resolution procedure. However, emphasis is given to modification control during operational phase.

IEC 880 items are:

- chapter 3 defines the requirements project structure which are given in terms of requirements on software lifecycle and Quality Assurance Plan;
- §4.11 define the diffusion process of functional requirements that should be included in the Quality plan;
- system configuration control treated in §7.3 and error resolution procedures (§7.6) are a subset of SQUALE process quality activities.

Chapter 9 describes the procedures to be applied for a modification or during the maintenance phase. A formal modification control procedure is established including verification and validation:

- a modification request is generated;

- the modification is evaluated independently and the decision is made (accept/execute or reject/justify the modification);
- after implementation of the modification, the whole verification and validation process is performed again according to the modification analysis;
- two reports are produced :
    - a modification report resuming all actions made for the modification purposes;
    - a modification field document recording the date of the implementation and the result of the specified observations.

5.5.5.3 Level of Rigour

The standard does not define levels of rigour and techniques to be used during software lifecycle are recommended correspondingly. Level of rigour in technique use should be determined by an analysis of the standards relative to the two classes of functions (safety related and non-safety related).

5.5.5.4 Level of Detail of Assessment

IEC 880 does not define required level of detail for any assessment that is carried out.

5.5.5.5 Level of Independence

IEC880 is intended for use by development organisation and does not consider confidence providing activities and assessment independently from the development activities. Just for verification activities, and specifically only for software testing (see §6.2.1) independence of the team should be assumed, but the standard does not make difference between independent department and organisation. §6.2.1 states that independent implies verification either by an individual or organisation which is separate from the developing individual or organisation and specifies that the management of the verification team shall be separate and independent from the management of the development team.

5.5.5.6 Competence of persons

The IEC 880 has no requirements regarding the competence of persons involved in the development or of those that conduct the assessment.

However, special attention is prescribed for training of operators and instrumentation and control specialists. Training of the personal shall be provided using a training system that react as close as possible as the nuclear plant (simulator recommended including normal and abnormal reactor conditions) and defined by a training plan.

5.5.6 Dependability Target

Most of the contents of the Dependability Target (DT) come from IEC 880 software requirements described in its chapter 4.

Software requirements contains the safety functions requirements and dependability objectives and shall describe the functions, their interfaces and how they are designed. Chapter 4 and its corresponding annexes gives also guidance for functional design and requirements for functions that shall be provided in order to meet the main dependability objectives.

- §4.1 DT chapter Dependability Related Function Specification;
- §4.2 (see also A.2.2.) DT chapter Dependability Objectives;
- §4.3 Safety guide 50-SG-D3/1980 should appear in DT chapter Dependability Policy;
- §4.4 DTD chapter Initial need and Environment Analysis and IEC publication 639 should be indicated in chapter Dependability Policy. §4.4 describes the environment of the software and its interfaces. Interfaces specification should be detailed in DT § Dependability Related Function Specification and Allocation;
- §4.5 could be included in DT Chapter Dependability Related Function Specification and Allocation. This part specifies functions dedicated to the prevention of radioactive release. Prevention of radioactive release should appear in the DT chapter Dependability Objectives;
- §4.6 The constraints should be described in DT chapter Dependability Related Function Specification and Allocation;
- §4.7 describes special operating conditions and should be splitted in DT chapter Dependability Objectives, DT chapter Dependability Related Function Specification and Allocation, and some studies in Hazard analysis;
- §4.8 describes the requirements for the self-supervision (§4.8.1), requirements for design of safety functions (§4.8.2), (§4.8.3), (§4.8.4), (§4.8.5). §4.8 should appear in DT chapter Dependability Policy;
- §4.10 recommends the use of formal specification.

For different level of the dependability target, improved during the construction phase, IEC 880 chapter 5 contents provide material that could be included in dependability policy. This chapter consists of recommendations mainly for software design and coding and ends with a documentation list.

- §5.2.5 and 5.2.6 recommends the use of automatic tools;
- §5.3 addresses recommendations for software design and coding that should be included in DT chapter Dependability Policy;
- §5.4 dresses the list of the documentation to be produced during the software lifecycle.

5.5.7 *Software lifecycle*

In SQUALE project the generic lifecycle is composed of three major phases, the construction phase, the operation phase and the maintenance phase.

5.5.7.1 Construction phase

The system development lifecycle described in the IEC 880 fits perfectly into the SQUALE lifecycle construction phase.

The system development lifecycle is decomposed into three branches:

- the hardware requirements;
- the integration requirements;
- the software requirements.

Figure 10 below shows the adequacy of the software requirements branch with the SQUALE lifecycle concerning the construction phase. In the same way, we could show the similarity of the two other branches with SQUALE lifecycle construction phase.

In this figure, verification activities correspond to the SQUALE correctness verification activities, whereas validation activities are equivalent to SQUALE testing activity at system level.

The development method (see B.1.a) is a top down approach (see B.1.b) leading to a software decomposed into modules, sub-modules, etc.

The recommended techniques are:

- formal and semi-formal methods (see B.1.b);
- automatic test tools when possible.

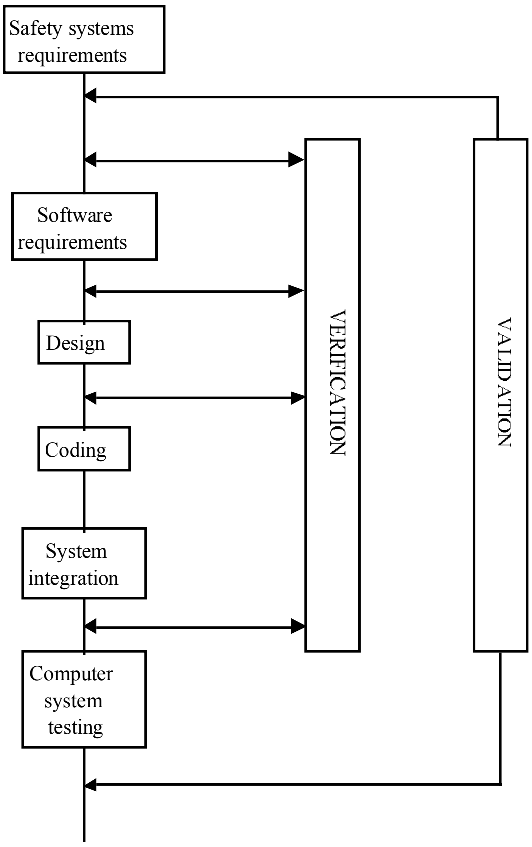

Figure 10: Software requirements branch of the system development lifecycle

5.5.7.2 Operation and maintenance phases

In IEC 880 operation includes periodic testing which is not developed in SQUALE project. Periodic testing must be viewed in SQUALE sense as a function to meet dependability objectives for the operation phase. Emphasis is given on modification procedures.

5.5.7.3 Deliverables list

Except for software performance specifications, the contents of IEC 880 deliverables are not clearly stated and so lead to the same flexibility as SQUALE criteria. However, IEC 880 links tightly the lifecycle with deliverables in locating the documentation production phase. Some documents in IEC 880 have no equivalent in SQUALE. Table 18 below shows these differences.

IEC 880	**Phase**	**Correspondence with SQUALE**
Quality Assurance plan	Software requirements	Quality plan
Design verification report	Design	Design verification
Software test specification	Design	Dependability Related Function definition
Specification of periodic test procedures	Design	/
Software test report	Coding	Testing
Software integration plan	Software requirements	Integration verification
Integration system verification test report	Hard/Soft integration	Integration verification
Computer system validation plan	Hard/Soft integration	Testing
Computer system validation report development	Computer system validation	Testing
Training plan	Computer system validation	/
User manual	Computer system validation	Development documentation
Commission test plan	Computer system validation	Testing
Commission test report	Commissioning	Testing
Anomaly report	Exploitation/Maintenance	Process Quality
Software modification request	Hard/Soft integration and Exploitation/Maintenance	Process Quality
Software modification report	Hard/Soft integration and Exploitation/Maintenance	Process Quality
Software modification “ field ” document	Exploitation/Maintenance	Process Quality

Table 18: Comparison of IEC documents and SQUALE documents

5.5.8 Miscellaneous

IEC 880 gives some importance to system installation and operation. Chapter 10 requires confidence providing activities regarding the commissioning and the operation of the system, in addition to change control (see above). These activities concern computer system and operating personnel and may be divided into three major activities:

- commissioning tests;
- operator training;
- periodic testing.

Commissioning tests consist in providing a test program to verify the integrity of the installed computer based safety system. A commissioning test plan including acceptance criteria and a commissioning test report presenting test results and conformance to acceptance criteria are established.

In order to achieve safe plant operation, operator behaviour is as important as equipment reliability. Therefore a training program is provided containing a training plan and a user manual. Operator training is realised on a training system which must be equivalent to the actual hardware/software system.In addition to commissioning tests, periodic test procedures are defined in a test programme. These periodic tests allow to verify the availability and the reliability of the safety system.

In SQUALE project all the aspects concerning the operating personnel are not taken into account.

Recommendation number	IEC 880	SQUALE
A.1	Define system lifecycle	Generic life cycle
A.2.1.1	Computer system specification :	TDA Description and Dependability Objectives
A.2.1.2	Function specifications	Dependability Related Function Specification and Allocation
A.2.2	Computer system configuration	Dependability Objectives
A.2.3	Principles for man-machine dialogue	Dependability Policy
A.2.4	Items relating to interfaces	Dependability Policy
A.2.5	Description of system function	Dependability Related Function Specification
A.2.6	Description of constraints between hardware and software	Dependability Related Function Specification
A.2.7	Description of special operating conditions	Dependability Related Function Specification
A.2.8	Self-supervision	Dependability Policy

A.2.9	Presentation of software functional requirements	Process Quality (Quality plan)
B.1.a	Maintainability objective relative to changeability	Dependability Objective
B.1.b	Top down approach for lifecycle definition and software decomposition into modules, sub modules, etc.. Recommends the use of semi-formal and formal methods (B.1.bh). Automatic design tools are recommended (B.1.bi).	Generic lifecycle and construction framework, Dependability criteria (levels 3 & 4) Dependability criteria (all levels)
B.1.c	Defines verification activities for each stage of the lifecycle.	Correctness verification activities for design and coding phases
B.1.d	Defines the modification process during development and verification activities following software change.	Process Quality
B.1.e	Define modification process in operation.	Process Quality.
B.2.a B.2.b	Guidance for software architecture (decomposition into modules)	Dependability Policy
B.2.c	Rules for operating system use.	Dependability Policy
B.2.d	Rules relative to execution time.	Dependability Policy
B.2.e	Rules relative to interrupts use.	Dependability Policy
B.2.f	Coding rule for arithmetic expressions	Dependability Policy
B.3.a	defensive programming rules	Dependability Policy
B.3.b	rules for safe output (error recovery)	Dependability Policy
B.3.c	software integrity rules	Dependability Policy
B.3.d	error checking rules	Dependability Policy
B.4.a,b,c,d,e,f	coding rules	Dependability Policy
B.4.g	Defines testing activities at each level of design	Correctness verification activities
B.5.a,b,c,d	coding rules (language dependent)	Dependability Policy
C	Outline for the software performance specification	No correspondence
D.1	Language, translator, linkage editor, etc..	rules for the choice of tools are not stated in SQUALE criteria
E.1,2,3,	Software testing activities	Correctness verification activities

		(all activities defined by criteria except behavioural analysis)
F.1	Documents relating to software production	Development documentation
F.2	Documents relating to software modifications	Process Quality
F.3	Documentation organisation relating to the software realisation	Lifecycle and construction framework.

Table 19: Mapping of IEC 880 detailed recommendations and SQUALE criteria

5.6 ETR 367

This standard consists of guidelines on the relevance of security evaluation to Telecommunication Security. It details the aspects of the security policy concerned with security criteria, evaluation, testing and accreditation and describes the standards and recommendations which should be used during the specification of an ETSI standard for telecommunication system evaluation.

No evaluation standard in the Telecommunication field is actually available. The guidelines refer mainly to ITSEC criteria. Consequently, the mapping of SQUALE criteria with ETR 367 is totally identical with the mapping with ITSEC criteria.

5.7 Avionics Standards: RTCA DO178B

5.7.1 Introduction

This section covers certification and standardisation issues in the avionics industry. Figure 11 presents some of the standardisation documents describing Aerospace Recommended Practice (ARP) that were developed in the context of Federal Aviation Regulations (FAR) and Joint Airworthiness Requirements (JAR). These documents represent certification guidance documents covering system, safety, software and hardware processes. As a whole, they constitute a comprehensive framework addressing the total life cycle for systems that implement aircraft-level functions. These documents are focused on safety related requirements and they are intended to be used in conjunction, as shown in Figure 11.

The ARP 4754 document includes system-level guidelines that are intended to provide designers, manufacturers, installers and certification authorities with a common international basis for demonstrating compliance with safety airworthiness requirements applicable to highly integrated or complex systems. The certification guidelines for the development of software systems are given in the DO-178B standard, whereas guidelines for the development of hardware systems are under definition. Safety requirements allocated to software subsystems and hardware subsystems are derived from the system level safety requirements. Guidelines and methods for conducting the safety assessment process are developed in ARP 4761 document.

In the following, we analyse the mapping between the DO-178B standard and the SQUALE Criteria.

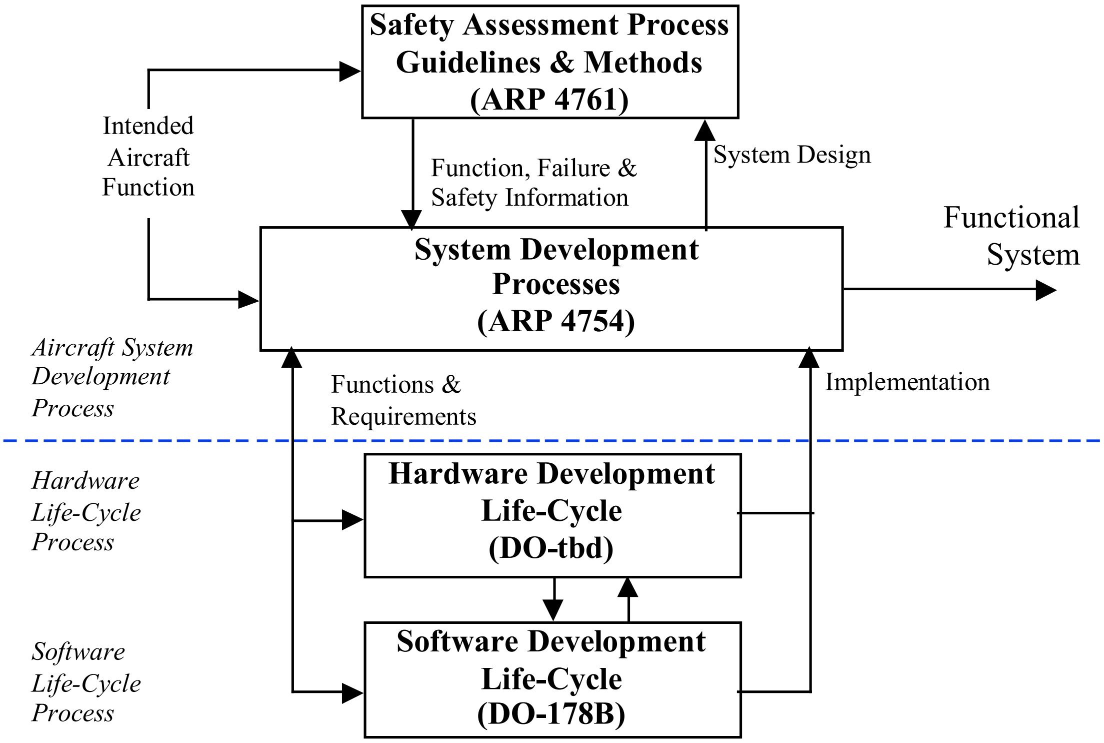
Safety Assessment Process Guidelines & Methods (ARP 4761)
Intended Aircraft Function
Function, Failure & Safety Information
System Design
System Development Processes (ARP 4754)
Functional System
Aircraft System Development Process
Functions & Requirements
Implementation
Hardware Life-Cycle Process
Hardware Development Life-Cycle (DO-tbd)
Software Life-Cycle Process
Software Development Life-Cycle (DO-178B)

Figure 11: Certification guidance documents covering System, Safety, Software and Hardware processes

5.7.2 Overview of DO-178B

The DO 178-B is a joint RTCA (Requirements and Technical Concepts for Aviation) and EUROCAE (European Organisation for Civil Aviation Equipment) standard providing guidelines for the production of airborne software that performs its intended function with a level of confidence in safety that complies with airworthiness requirements.

DO-178B is a process-oriented document. The standard defines three categories of processes that must be included in each software development project, no matter which software life cycle is chosen. For each process, the standard describes: 1) the objectives to be fulfilled, 2) the activities to be performed and design recommendations to achieve those objectives, and 3) the software life cycle data and evidence to be provided to show that the objectives have been satisfied.

In avionics, the certification authority considers the software as part of the airborne system installed on the aircraft or engine; that is it does not approve the software as a unique, standalone product. The certification authority establishes the certification basis for the aircraft or engine in consultation with the sponsor. The certification basis defines the particular regulations together with any special conditions, which may supplement the published regulations.

The objective of the certification liaison process is to establish communication between the sponsor and the certification authority throughout the software life cycle to assist the certification process. The sponsor submits a Plan for Software Aspects for Certification that serves as the primary means for communicating the proposed development methods for the certification authority for agreement, and defines the means of compliance with DO-178B standard. Prior to certification, the sponsor submits a Software Accomplishment Summary and evidence showing compliance of the final product with the Plan for Software Aspects for Certification that was initially agreed. This data is reviewed by the certification authority. The sponsor must resolve all issues raised as a result of these reviews before the system can be certified.

5.7.3 General Similarities

The dependability assessment framework and the roles defined in the Criteria fit the general avionics certification scheme. The contents of the Plan for Software Aspects for Certification are similar to those corresponding to the dependability target defined in the SQUALE Criteria.

Both DO-178B and the Criteria define four levels of confidence based on the severity of the hazard and failure conditions with respect to their impact on dependability objectives (for DO-178B only safety is addressed), and one additional level, which corresponds to systems that do not have any impact on dependability (safety for DO-178B).

DO-178B and the Criteria adopt a process-oriented approach for the definition of criteria and guidelines used for the development and assessment of dependability related aspects.

5.7.4 General Differences

DO-178B addresses only safety whilst the Criteria cover all dependability attributes, including safety. DO-178B addresses only on software systems. System level guidelines are discussed in the ARP 4754 standard.

DO-178B focuses on verification activities. Unlike the Criteria, DO-178B does not distinguish between Correctness Verification and validation activities as we do in the Criteria. Considerations related to Process Quality Assurance are covered within three main processes: software planning process, software quality assurance and software configuration management process. The verification activities discussed in DO-178B are more detailed than in the Criteria. This is related to the fact that DO-178B is software oriented whereas the Criteria aims to be generic and independent of the technology considered.

The concepts of levels of rigour and detail are not explicitly distinguished in DO-178B, but they are implicit in the definition of the objectives to be satisfied for each process and the mapping between objectives and software levels.

Independence requirements defined in DO-178B concern verification and software quality processes only. For all software levels, independence of quality assurance activities from development activities is required. As regards verification activities, the independence requirement concerns Levels A and B only. However, independence is not required for all objectives that must be achieved by these levels (even for Level A), but only for some of them.

Compared to DO-178B, the SQUALE Criteria are more stringent with respect to independence requirements to be fulfilled for verification activities. For quality assurance process activities, both require the independence of quality assurance staff and the developers.

5.7.5 Confidence levels

DO-178B defines five confidence levels for the software (called software levels) based upon the contribution of software to potential failure conditions as determined by the system safety assessment process. Five failure condition categories are defined: catastrophic, hazardous/severe-major, major, minor, and no effect. Accordingly, the five software levels are defined as follows:

- Level A: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a *catastrophic failure* condition for the aircraft.
- Level B: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a *hazardous/severe-major failure* condition for the aircraft.
- Level C: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a *major failure* condition for the aircraft.
- Level D: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a *minor failure* condition for the aircraft.
- Level E: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function with *no effect* on aircraft operational capability or pilot workload. Once software has been confirmed as level E by the certification authority, no further guidelines of this document apply.

For each software level, the objectives to be satisfied by each process are defined together with the outputs of each process.

The software levels defined in the DO-178B fit the confidence levels defined in the SQUALE evaluation framework.

5.7.6 Dependability Target

The Plan for Software Aspects for Certification includes contents similar to those defined in the dependability target:

- an overview of the system, including a description of its functions and their allocation to the hardware and software, the architecture, processors used, hardware/software interfaces, and safety features;
- a brief description of the software functions with emphasis on the proposed safety and partitioning concepts (e.g., redundancy, diversification, fault tolerance, etc.);

- a summary of the proposed means of compliance, with the statement and justification of the target software confidence level (called software level);
- a description of the software life cycle processes to be used and the objectives that will be satisfied by each process;
- a description of software life cycle data that will be produced during the life cycle and the data to be submitted to the certification authority;
- a scheduling of the review meetings with the certification authority;
- additional considerations, such as alternative methods for compliance, tool qualification, COTS, etc.

5.7.7 Confidence providing processes and activities

DO-178B defines three categories of processes that must be included in each software development project, no matter which software life cycle is chosen:

- Software development processes: Requirements, Design, Coding and Integration
- Integral processes that ensure the correctness, the control and confidence of the software life cycle processes and their outputs: Verification process, Configuration management process, Quality assurance process, and Certification liaison process.
- Software planning process which defines and orchestrates the activities of the software development processes and the integral processes.

The objectives and activities of the software verification process include both the verification of the outputs of the software development processes and the verification of the results of the verification process itself. Verification process objectives are satisfied through a combination of reviews, analyses, the development of test cases and procedures, and the execution of those test procedures.

The dependability related requirements allocated to the software are derived from system level dependability requirements. Allocation of the requirements to software is performed within the system safety assessment process, which is described in ARP 4754 standard.

Correctness Verification activities and objectives defined in the SQUALE Criteria are similar to those defined in DO-178B. However, DO-178B does not distinguish between Correctness Verification activities, dependability requirements validation and dependability validation. This distinction is considered in the ARP 4754 document which focuses on system level aspects and particularly on the validation of safety related requirements.

5.7.8 Levels of rigour, detail and independence

The philosophy adopted in DO-178B consists in defining objectives to be satisfied by each process and specifying, for each software level, the applicability and independence requirements for each objective. The concepts of levels of rigour and detail are not explicitly distinguished in this standard, but they are implicit in the definition of the objectives to be satisfied and the definition of the mapping with the software levels.

For all software levels, it is required that a structured development approach is used leading to the definition of high-level requirements, software architecture, low-level requirements, source code, and executable object code. The same objectives are also defined for all levels for the Certification-liaison process. As regards the software planning process, differences between levels with respect to the objectives to be satisfied concern Level D only. However, for some objectives, there is a difference in the level of rigour of software planning outputs control management, for Levels C and D, as compared to Levels A and D.

As regards the software quality assurance process, the only difference concerns Levels C and D for which it is not required to provide assurance that the transition criteria between life cycle processes are satisfied.

The main difference between software levels with respect to the objectives to be fulfilled (i.e. levels of rigour and detail) concerns the verification process, especially for levels C and D compared to Levels A and B. The only difference between Level B and Level A is that it is not required to provide test coverage of software structure with respect to modified condition and decisions. Nevertheless, other types of test coverage analysis are required.

Independence requirements concern verification and software quality processes only. Independence is defined as the separation of responsibilities, which ensures the accomplishment of objective evaluation. For software verification process objectives, independence is achieved when the verification activity is performed by a person other than the developer of the item being verified, and a tool(s) may be used to achieve an equivalence to the human verification activity. For the software quality assurance process, independence also includes the authority to ensure corrective action.

For all software levels, independence of quality assurance activities from development activities is required. As regards verification activities, the independence requirement concerns Levels A and B only. However, independence is not required for all objectives that must be achieved by these levels (even for Level A), but only for some of them.

Compared to DO-178B, the SQUALE Criteria are more stringent with respect to independence requirements to be fulfilled for verification activities. For quality assurance process activities, both require the independence of quality assurance staff and the developers.

5.7.9 Lifecycle

The DO-178 main objective is to provide guidelines for the development of software systems that satisfy airworthiness requirements. Operation related aspects are not covered in this standard, they are addressed at the system level when hardware and software subsystems are integrated. System level considerations are addressed in the ARP 4754 standard.

Four major phases are distinguished during the development process: Requirements, Design, Coding and Integration.

5.7.10Mapping between DO-178B requirements and the Criteria

Table 20 below shows how the DO-178B requirements map to the criteria. For each process defined in DO-178B, and for each objective to be achieved for that process, we indicate the correspondence with the Criteria CPPs.

DO-178B Requirements	Criteria CPA
Software Planning Process	
Software development and integral processes are defined	Covered by Process Quality
Transition criteria, interrelationships and sequencing among processes are defined	Covered by Process Quality
Software lifecycle environment is defined	Covered by Process Quality
Software development standards are defined	Covered by Process Quality
Software plans comply with DO-178B standard	Covered by Process Quality
Software Plans are coordinated	Covered by Process Quality
Software Development Processes	
High-level requirements are defined	Implicit in the Criteria and checked by Process Quality
Derived High-level requirements are defined	Implicit in the Criteria and checked by Process Quality
Software architecture is developed	Implicit in the Criteria and checked by Process Quality
Low-level requirements are developed	Implicit in the Criteria and checked by Process Quality
Derived low-level requirements are developed	Implicit in the Criteria and checked by Process Quality
Source code is developed	Implicit in the Criteria and checked by Process Quality
Executable object-code is produced and integrated in the target computer	Implicit in the Criteria and checked by Process Quality
Verification of outputs of Software Requirements Process	
Software high-level requirements comply with system requirements	Covered by Correctness Verification and dependability requirements validation
High-level requirements are accurate and consistent	Covered by Correctness Verification
High-level requirements are compatible with target computer	Covered by Correctness Verification
High-level requirements are verifiable	Covered by Correctness Verification
High-level requirements conform to standards	Covered by Correctness Verification and Process Quality
High-level requirements are traceable to system requirements	Covered by Correctness Verification
Algorithms are accurate	Covered by Correctness Verification

DO-178B Requirements	**Criteria CPA**
Verification of outputs of Software Design Process	
Low-level requirements comply with high-level requirements	Covered by Correctness Verification and dependability requirements validation
Low-level requirements are accurate and consistent	Covered by Correctness Verification
Low-level requirements are compatible with target computer	Covered by Correctness Verification
Low-level requirements are verifiable	Covered by Correctness Verification
Low-level requirements conform to standards	Covered by Correctness Verification and Process Quality
Low-level requirements are traceable to high-level requirements	Covered by Correctness Verification
Algorithms are accurate	Covered by Correctness Verification
Software architecture is compatible with high-level requirements	Covered by Correctness Verification
Software architecture is compatible with target computer	Covered by Correctness Verification
Software architecture is verifiable	Covered by Correctness Verification
Software architecture conforms to standards	Covered by Correctness Verification and Process Quality
Software partitioning integrity is confirmed	Covered by Correctness Verification and dependability requirements validation
Verification of Software Coding & Integration Processes	
Source code complies with low-level requirements	Covered by Correctness Verification
Source code complies with software architecture	Covered by Correctness Verification
Source code is verifiable	Covered by Correctness Verification
Source code conforms to standards	Covered by Correctness Verification and Process Quality
Source code is traceable to low-level requirements	Covered by Correctness Verification
Source code is accurate and consistent	Covered by Correctness Verification
Output of software integration process is complete and correct	Covered by Correctness Verification
Testing of outputs of integration process	
Executable Object code complies with high-level requirements	Covered by Correctness Verification and dependability validation
Executable Object code is robust with high-level requirements	Covered by Correctness Verification and dependability validation

DO-178B Requirements	**Criteria CPA**
Executable Object code complies with low-level requirements	Covered by Correctness Verification
Executable Object code is robust with low-level requirements	Covered by Correctness Verification
Executable Object code is compatible with target computer	Covered by Correctness Verification
Verification of Verification Process Results	
Test procedures are correct	Covered by Correctness Verification
Test results are correct and discrepancies explained	Covered by Correctness Verification
Test coverage of high-level requirements is achieved	Covered by Correctness Verification but not detailed
Test coverage of low-level requirements is achieved	Covered by Correctness Verification but not detailed
Test coverage of software structure is achieved *Modified condition/decision, decision coverage, statement coverage, data coupling and control coupling*	Covered by Correctness Verification but not detailed
Software Configuration Management Process	
Configuration items are identified	part of Process Quality
Baselines and traceability are established	part of Process Quality
Problem reporting, change control, change review, and configuration status accounting are established	part of Process Quality
Archive, retrieval, and release are established	part of Process Quality
Software load control is established	part of Process Quality
Software life cycle control is established	part of Process Quality
Software Quality Assurance Process	
Assurance is obtained that software development and integral processes comply with approved software plans and standards	part of Process Quality
Assurance is obtained that transition criteria for the software life cycle processes are satisfied	part of Process Quality
Software conformity review is conducted	part of Process Quality
Certification Liaison Process	
Communication and understanding between the applicant and the certification authority is established	Roles and activities in the dependability assessment process

DO-178B Requirements	**Criteria CPA**
The means of compliance is proposed and agreement with the plan for software Aspects of Certification is obtained	Roles and activities in the dependability assessment process
Compliance substantiation is provided	Roles and activities in the dependability assessment process

Table 20: Mapping of DO-178B requirements to the Criteria

Annexe A - Dependability Concepts and Terminology

The terminology presented here has been developed from the mid-seventies to the early nineties by the Fault-Tolerant Computing community, and especially the IFIP Working Group 10.4. [Carter 82, Laprie & Costes 82, Laprie 85, Avizienis & Laprie 86, Laprie 92, Laprie 95].

A.1 Basic definitions

Dependability is defined as that property of a system such that *reliance can justifiably be placed on the service* it delivers. The service delivered by a system is its behaviour *as it is perceptible* by its user(s); a user is another system (human or physical) which *interacts* with the former.

Depending on the application(s) intended for the system, different emphasis may be put on different facets of dependability, i.e., dependability may be viewed according to different, but complementary, *properties,* which enable the *attributes* of dependability to be defined:

- the *readiness for usage* leads to **availability**,
- the *continuity of service* leads to **reliability**,
- the *non-occurrence of catastrophic consequences on the environment* leads to **safety**,
- the *non-occurrence of unauthorised disclosure of information* leads to **confidentiality**,
- the *non-occurrence of improper alterations of the system* leads to **integrity**,
- the *ability to undergo repairs and evolution* leads to **maintainability**.

Associating integrity and availability with respect to authorised actions, together with confidentiality, leads to **security**.

A system **failure** occurs when the delivered service deviates from fulfilling the system **function**, the latter being what the system *is aimed at*. An **error** is that part of the system state which is *liable to lead to subsequent failure*: an error affecting the service is an indication that a failure occurs or has occurred. The *adjudged or hypothesised cause* of an error is a **fault**.

The development of a dependable computing system calls for the *combined* utilisation of a set of methods and techniques which can be classed into:

- **fault prevention**: how to prevent fault occurrence or introduction,
- **fault tolerance**: how to ensure a service up to fulfilling the system's function in the presence of faults,
- **fault removal**: how to reduce the presence (number, seriousness) of faults,
- **fault forecasting**: how to estimate the present number, the future incidence, and the consequences of faults.

The notions introduced here can be grouped into three classes and are summarised by Figure 12:

- the **impairments** to dependability: faults, errors, failures; they are undesired — but not in principle unexpected — circumstances causing or resulting from un-dependability (whose definition is very simply derived from the definition of dependability: reliance cannot, or will not any longer, be placed on the service);

- the **means** for dependability: fault prevention, fault tolerance, fault removal, fault forecasting; these are the methods and techniques enabling one a) to provide the ability to deliver a service on which reliance can be placed, and b) to reach confidence in this ability;
- the **attributes** of dependability: availability, reliability, safety, confidentiality, integrity, maintainability; these:
  - enable the properties which are expected from the system to be expressed, and
  - allow the system quality resulting from the impairments and the means opposing to them to be assessed.

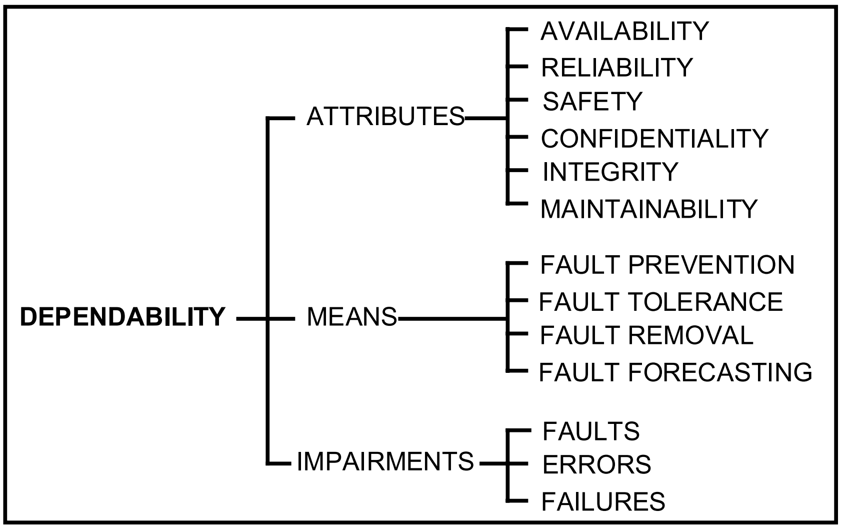

Figure 12: Dependability Framework

A.2 Dependability Concept Rationale

A major strength of the dependability concept, as it is formulated here, is its integrative nature, which enables to put into perspective the more classical notions of reliability, availability, safety, security, maintainability, which are then seen as attributes of dependability. The fault-error-failure model is central to the understanding and mastering of the various impairments which may affect a system, and it enables a unified presentation of these impairments, though preserving their specificities via the various fault classes which can be defined (see A.4.2). The model provided for the means for dependability is extremely useful, as those means are much more orthogonal to each other than the usual classification according to the attributes of dependability, with respect to which the design of any real system has to perform trade-offs, due to the fact that these attributes tend to be in conflict with each other.

The definitions of *dependability* which exist in several current standards differ from the definition we have given. Two such differing definitions are :

- "The collective term used to describe the availability performance and its influencing factors: reliability performance, maintainability performance and maintenance support performance" [ISO 91].
- "The extent to which the system can be relied upon to perform exclusively and correctly the system task(s) under defined operational and environmental conditions over a defined period of time, or at a given instant of time" [CEI 92].

The ISO definition is clearly centred upon availability. This is no surprise as this definition can be traced back to the definition given by the international organisation for telephony, the CCITT [CCITT 84], although it could be argued that it seems strange to take a definition formulated in a given context, and to use it as it is in a broader context. However, the willingness to grant dependability with a generic character is noteworthy, so going beyond availability as it was usually defined, and thus restituting the role of reliability and maintainability. In this respect, the ISO/CCITT definition is consistent with the definition given in [Hosford 60] for dependability: "the probability that a system will operate when needed".

The second definition, from [CEI 92], introduces the notion of reliance, and as such is much closer to our definition. This CEI definition stays however fundamentally in the spirit of the extension of availability.

A.3 The Attributes of Dependability

The attributes of dependability have been defined in Section A.1 according to different properties, which may be more or less emphasised depending on the application intended for the computer system under consideration. Especially, reliability, safety, confidentiality may or may not be required according to the application.

Integrity is a pre-requisite for availability, reliability and safety, but may not be so for confidentiality (for instance when considering attacks via covert channels or passive listening). The definition given for integrity — absence of improper alterations of information — generalises the usual definitions, which relate to the notion of authorised actions only (e.g., prevention of the unauthorised amendment or deletion of information [CEC 91], assurance of approved data alterations [Jacob 91]; naturally, when a system implements an authorisation policy, "improper" encompasses "unauthorised".

Whether a system holds the properties which have enabled the attributes of dependability to be defined should be interpreted in a relative, probabilistic, sense, and not in an absolute, deterministic sense: due to the unavoidable presence or occurrence of faults, systems are never one hundred percent available, reliable, safe, or secure.

The definition given for maintainability goes deliberately beyond **corrective maintenance**, aimed at preserving or improving the system's ability to deliver a service fulfilling its function (relating to reparability only), and encompasses via evolvability the other forms of maintenance: **adaptive maintenance**, which adjusts the system to environmental changes (e.g. change of operating systems or system data-bases), and **perfective maintenance**, which improves the system's function by responding to customer — and designer — defined changes, which may involve removal of specification faults [Ramamoorthy 84].

Security has not been introduced as a single attribute of dependability, in agreement with the usual definitions of security, which view it as a *composite* notion, namely the combination of confidentiality, integrity, and availability [CEC 91].

The variations in the emphasis to be put on the attributes of dependability have a direct influence on the appropriate balance of the techniques addressed in the previous section to be employed in order that the resulting system be dependable. This is an all the more difficult problem as some of the attributes are antagonistic (e.g. availability and safety, availability and security), necessitating that trade-offs be performed. Considering the three main design dimensions of a computer system, i.e. cost, performance and dependability, the problem is further exacerbated by the fact that the dependability dimension is less understood than the cost-performance design space [Siewiorek & Johnson 82].

A.4 Dependability impairments

The creation and manifestation mechanisms of faults, errors, and failures may be summarised as follows:

- A fault is **active** when it produces an error. An active fault is either a) an internal fault which was previously **dormant** and which has been activated by the computation process, or b) an external fault. Most internal faults cycle between their dormant and active states. Physical faults can directly affect the hardware components only, whereas human-made faults may affect any component.
- An error may be latent or detected. An error is **latent** when it has not been recognised as such; an error is **detected** by a detection algorithm or mechanism. An error may disappear before being detected. An error may, and in general does, propagate; by propagating, an error creates other — new — error(s). During operation, the presence of active faults is determined only by the detection of errors.
- A failure occurs when an error "passes through" the system-user interface and affects the service delivered by the system. A component failure results in a fault:
    - for the system which contains the component, and
    - as viewed by the other component(s) with which it interacts; the failure modes of the failed component then become fault types for the components interacting with it.

These mechanisms enable the "fundamental chain" to be completed:

• • • → failure → fault → error → failure → fault → • • •

The arrows in this chain express the causality relationship between faults, errors and failures. They should not be interpreted in a restrictive manner:

a) by propagation, several errors can be generated before a failure occurs, and

b) an error can lead to a fault without a failure to be observed, if the observation is not performed, as a failure is an event occurring at the interface between two components.

A.4.1 Failures and Failure Modes

Failure occurrence has been defined with respect to the function of a system, not with respect to its specification. Indeed, if an unacceptable behaviour is generally identified as a failure due to a deviation from the compliance with the specification, it may happen that such a behaviour complies with the specification, and be however unacceptable for the system user(s), thus uncovering a specification fault. In the latter, recognising that the event is undesired (and is in fact a failure) can only be performed after its occurrence, for instance via its consequences.

A system may not, and generally does not, always fail in the same way. The ways a system can fail are its *failure modes*, which may be characterised according to three viewpoints: domain, perception by the system users, and consequences on the environment, as indicated by Figure 13.

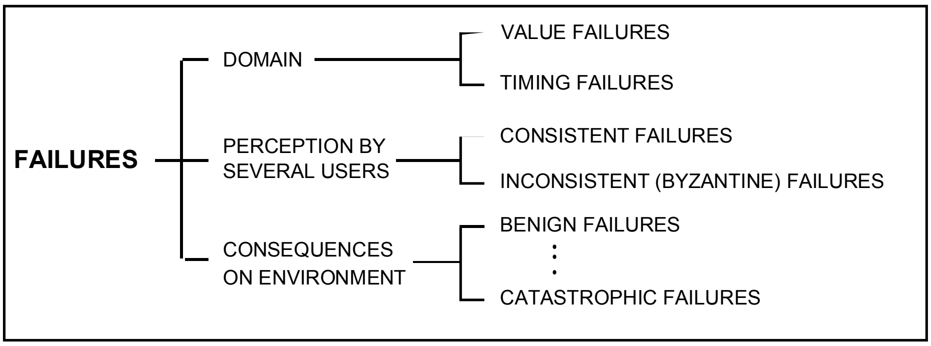

Figure 13: The Failure Modes

A class of failures relating to both value and timing are the **halting failures**: system activity, if any, is no longer perceptible to the users. According to how the system interacts with its user(s), such an absence of activity may take the form of a) frozen outputs (a constant value service is delivered; the constant value delivered may vary according to the application, e.g. last correct value, some predetermined value, etc.), or of b) a silence (no message sent in a distributed system). A system whose failures can be — or more generally are to an acceptable extent — only halting failures, is a **fail-halt system**; the situations of frozen outputs and of silence lead respectively to **fail-passive systems** and to **fail-silent systems** [Powell et al. 88].

A system whose failures can only be — or more generally are to an acceptable extent — benign failures is a **fail-safe system**. The notion of failure severity, resulting from grading the consequences of failures upon the system environment, enables the notion of criticality to be defined: the **criticality** of a system is the highest severity of its (possible) failure modes. The relation between failure modes and failure severity is highly application-dependent. However, there exist a broad class of applications where inoperation is considered as being a naturally safe position (e.g. ground transportation, energy production), whence the direct correspondence which is often made between fail-halt and fail-safe [Mine & Koga 67, Nicolaidis et al. 89]. Fail-halt systems (either fail-passive or fail-silent) and fail-safe systems are however examples of **fail-controlled systems**, i.e. systems which are designed and realised in order that they may only fail — or may fail to an acceptable extent — according to restrictive modes of failure, e.g. frozen output as opposed to delivering erratic values, silence as opposed to babbling, consistent failures as opposed to inconsistent ones; fail-controlled systems may in addition be defined via imposing some internal state condition or accessibility, as in the so-called *fail-stop* systems [Schlichting & Schneider 83].

A.4.2 Fault Classes and Fault Assumptions

When designing a dependable system, it is very important to identify which fault classes are to be taken into account because different means are to be used to deal with different fault classes. Thus, fault assumptions influence directly the design choices, and also the level of dependability that can be achieved [Powell 92].

The following classification of faults is performed in two steps:

- elementary fault classes are first defined, according to various viewpoints, as indicated in Figure 14;
- the combination of these elementary faults leads to the combined fault classes given by Figure 15. The labels associated to those combined fault classes are consistent with the labels commonly used in order to point in a condensed manner at one or several fault classes.

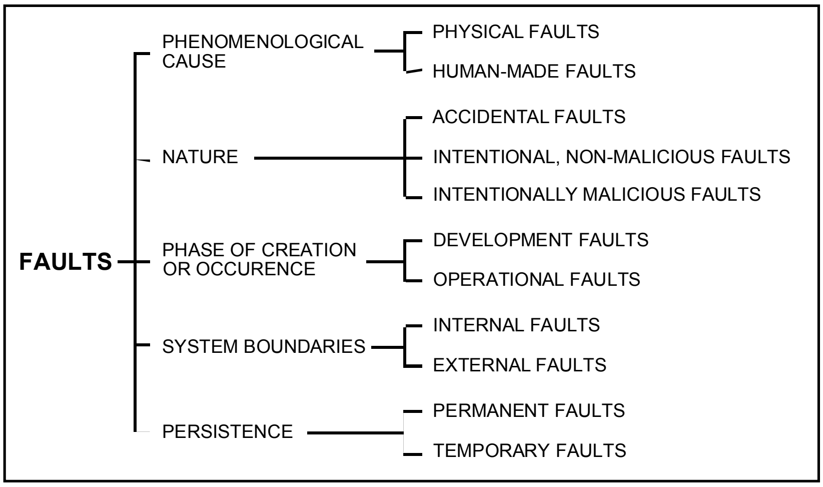

Figure 14: The elementary fault classes

Two comments regarding the human-made fault classes:

- Intentional, non-malicious, design faults result generally from trade-offs, either
  - aimed at preserving acceptable performances, at facilitating the system utilisation, or
  - induced by economic considerations. Intentional, non-malicious interaction faults may result from the action of an operator either aimed at overcoming an unforeseen situation, or deliberately violating an operating procedure without having developed the consciousness of the possibly damaging consequences of his or her action. These classes of intentional non-malicious faults share the characteristic that, often, it is realised they were faults only after an unacceptable system behaviour, thus a failure, has occurred.

- Malicious logic encompass development faults such as Trojan horses, logic or timing bombs, trapdoors, as well as operational faults (for the considered system) such as viruses or worms [Landwehr et al. 93].

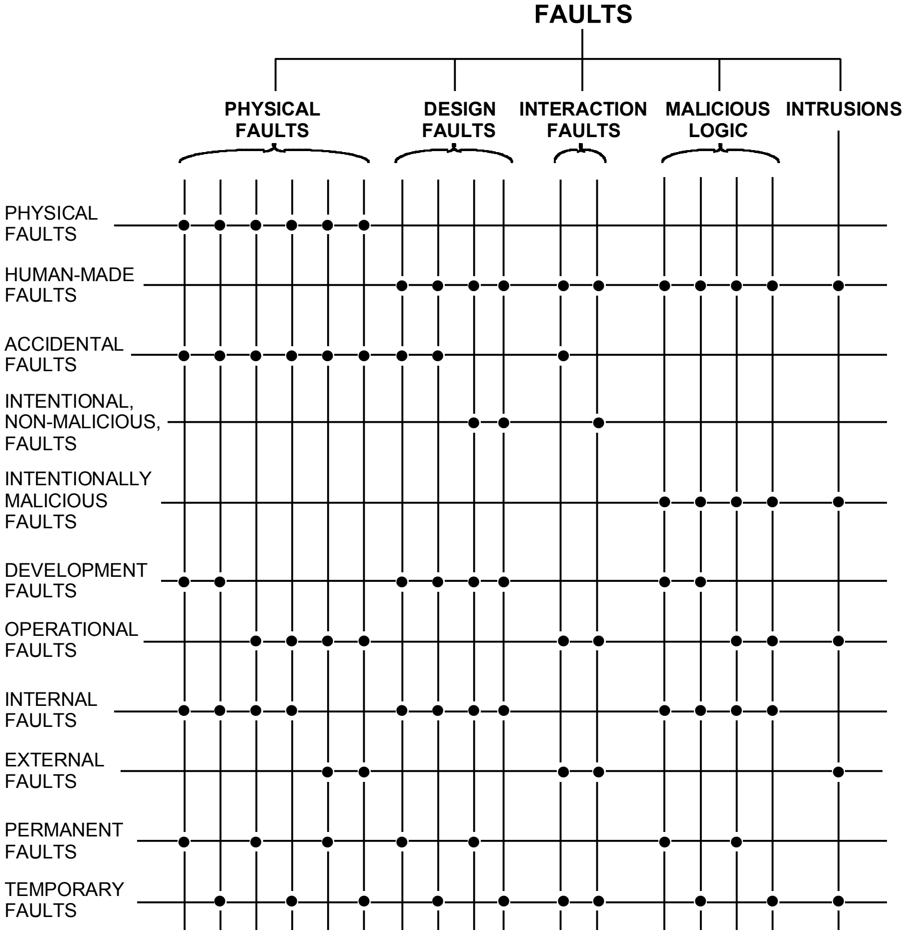

Figure 15: The combined fault classes

The complexity of the creation, activation and manifestation of faults leads to a multiplicity of possible causes of failures. This complexity leads to the definition of fault classes which are more abstract than the classes which have been considered so far, be it:

- for classifying faults uncovered during operation, or
- for stating fault assumptions when designing a system.

Two such fault classes are:

- **configuration change faults** [Wood 94]: the service delivered by the system is affected, subsequently to a maintenance action, either evolutive or perfective (e.g. introduction of a new software version on a network server),
- **timing faults**, leading to timing failures, and **omission faults**, such that the system does not respond to a request, thus leading to halting failures [Cristian et al. 85].

A.5 The means for dependability

In this section, we briefly examine in turns the means for dependability : fault prevention, fault tolerance, fault removal and fault forecasting.

A.5.1 Fault prevention

Fault prevention aims at preventing the introduction of development faults and the occurrence of operational faults. As such, fault prevention encompasses all the techniques used to improve the quality of design, development and manufacturing (e.g. formal methods, CASE tools, quality assurance), and also the organisational aspects related to the development process as well as to the operation of the system.

A.5.2 Fault tolerance

Fault tolerance [Avizienis 67] is carried out by error processing and by fault treatment [Anderson & Lee 81]. **Error processing** is aimed at removing errors from the computational state, if possible before failure occurrence; **fault treatment** is aimed at preventing faults from being activated — again.

Error processing can be carried out via three primitives:

- **error detection**, which enables an erroneous state to be identified as such;
- **error diagnosis**, which enables to assess the damages caused by the detected error, or by errors propagated before detection;
- **error recovery**, where an error-free state is substituted for the erroneous state; this substitution may take on three forms:
  - **backward recovery**, where the erroneous state transformation consists of bringing the system back to a state already occupied prior to error occurrence; this involves the establishment of **recovery points**, which are points in time during the execution of a process for which the then current state may subsequently need to be restored;
  - **forward recovery**, where the erroneous state transformation consists of finding a new state, from which the system can operate (frequently in a degraded mode);
  - **compensation**, where the erroneous state contains enough redundancy to enable its transformation into an error-free state.

When backward or forward recovery are utilised, it is necessary that error detection precedes error recovery. Backward and forward recovery are not exclusive: backward recovery may be attempted first; if the error persists, forward recovery may then be attempted. In forward recovery, it is necessary to *assess the damage* caused by the detected error, or by errors propagated before detection; damage assessment can — in principle — be ignored in the case of backward recovery, provided that the mechanisms enabling the transformation of the erroneous state into an error-free state have not been affected [Anderson & Lee 81].

The association into a component of its functional processing capability together with error detection mechanisms leads to the notion of **self-checking component**, either in hardware [Carter & Schneider 68, Wakerly 78, Nicolaidis et al. 89] or in software [Yau & Cheung 75, Laprie et al. 90a]; one of the important benefits of the self-checking component approach is the ability to give a clear definition of *error confinement areas* [Siewiorek & Johnson 82]. When error compensation is performed in a system made up of self-checking components partitioned into classes executing the same tasks, then state transformation is nothing else than switching within a class from a failed component to a non-failed one. On the other hand, compensation may be applied systematically, even in the absence of errors, then providing **fault masking** (e.g. in majority vote). However, this can at the same time correspond to a redundancy decrease which is not known. So, practical implementations of masking generally involve error detection, which may then be performed *after* the state transformation. As opposed to fault-masking, implementing error processing via error recovery *after* error detection has taken place, is generally referred to as **error detection and recovery**.

The first step in *fault treatment* is **fault diagnosis**, which consists of determining the cause(s) of error(s), in terms of both location and nature. Then come the actions aimed at fulfilling the main purpose of fault treatment: preventing the fault(s) from being activated again, thus aimed at making it(them) passive, i.e. **fault passivation**. This is carried out by preventing the component(s) identified as being faulty from being invoked in further executions. If the system is no longer capable of delivering the same service as before, then a **reconfiguration** may take place, which consists in modifying the system structure in order that the non-failed components enable the delivery of an acceptable service, although degraded; a reconfiguration may involve some tasks to be given up, or re-assigning tasks among non-failed components.

The preceding definitions apply to physical faults as well as to design faults: the class(es) of faults which can actually be tolerated depend(s) on the fault hypothesis which is being considered in the design process, and thus relies on the *independence* of redundancies with respect to the process of fault creation and activation. An example is provided by considering tolerance of physical faults and tolerance of design faults. A (widely-used) method to attain fault tolerance is to perform multiple computations through multiple channels. When tolerance of physical faults is foreseen, the channels may be identical, based on the assumption that hardware components fail independently; such an approach is not suitable for the tolerance to design faults where the channels have to provide *identical services* through *separate designs and implementations* [Elmendorf 72, Randell 75, Avizienis 78], i.e. through **design diversity** [Avizienis & Kelly 84].

An important aspect in the co-ordination of the activity of multiple components is that of preventing error propagation from affecting the operation of non-failed components. This aspect becomes particularly important when a given component needs to communicate some information to other components that is private to that component. Typical examples of such *single-source information* are local sensor data, the value of a local clock, the local view of the status of other components, etc. The consequence of this need to communicate single-source information from one component to other components is that non-failed components must reach an *agreement* as to how the information they obtain should be employed in a mutually consistent way. Specific attention has been devoted to this problem in the field of distributed systems (see e.g. atomic broadcast [Cristian et al. 85a], clock synchronisation [Lamport & Melliar-Smith 85, Kopetz & Ochsenreiter 87] or membership protocols [Cristian 88]). It is important to realise, however, that the inevitable presence of structural redundancy in any fault-tolerant system implies distribution at one level or another, and that the agreement problem therefore remains in existence. Geographically localised fault-tolerant systems may employ solutions to the agreement problem that would be deemed too costly in a "classical" distributed system of components communicating by messages (e.g. inter-stages [Lala 86], multiple stages for interactive consistency [Frison & Wensley 82]).

A.5.3 Fault removal

Fault removal is composed of three steps: verification, diagnosis, correction. **Verification** is the process of checking whether the system adheres to properties, termed the *verification conditions* [Cheheyl et al. 81]. The verification techniques can be classed according to whether or not they involve exercising the system.

Verifying a system without actual execution is **static verification**. The verification can be conducted:

- on the system itself, in the form of a) *static analysis* (e.g. inspections or walk-through [Myers 79], data flow analysis [Osterweil et al. 76], complexity analysis [McCabe 76], compiler checks, etc.) or b) *proof-of-correctness* (inductive assertions [Hoare 69]);
- on a model of the system behaviour (e.g. Petri nets, finite state automata), leading to *behaviour analysis* [Diaz 82].

Verifying a system through exercising it constitutes **dynamic verification**; the inputs supplied to the system can be either symbolic in the case of **symbolic execution**, or valued in the case of verification testing, usually simply termed **testing**.

Testing exhaustively a system with respect to all its possible inputs is generally impractical. The methods for the determination of the test patterns can be classed according to two viewpoints: criteria for selecting the test inputs, and generation of the test inputs.

The techniques for selecting the test inputs may in turn be classed according to three viewpoints:

- the purpose of the testing: checking whether the system satisfies its specification is **conformance testing**, whereas testing aimed at revealing faults is **fault-finding testing**;
- the system model: depending on whether the system model relates to the function or the structure of the system, leads respectively to **functional testing** and **structural testing**;
- fault model: the existence of a fault model leads to **fault-based testing** [Morell 90], aimed at revealing specific classes of faults (e.g. stuck-at-faults in hardware production [Roth et al. 67], physical faults affecting the instruction set of a microprocessor [Thatte & Abraham 78], design faults in software [Goodenough & Gerhart 75, DeMillo et al. 78, Howden 87]); if there is no fault model, one is then led to **criteria-based testing**, where the criteria may relate, for software, to path sensitisation [Howden 76], to utilisation of the program variables [Rapps & Weyuker 85], to input boundary values [Myers 79], etc.

Combining the various viewpoints leads to the testing approaches, where the distinction between hardware and software is important since hardware testing is mainly aimed at removing production faults, whereas software testing is concerned only with design faults:

- structural testing when applied to hardware generally means fault-finding, fault-based, whereas it generally means fault-finding, non-fault-based testing when applied to software;
- functional testing when applied to hardware generally means fault-finding, fault-based, whereas it generally means either conformance or fault-finding, criteria-based when applied to software;
- *mutation testing* of software [DeMillo et al. 78] is fault-finding, structural, fault-based, testing.

The *generation* of the test inputs may be deterministic or probabilistic:

- in **deterministic testing**, test patterns are predetermined by a selective choice according to the adopted criteria,
- in **random**, or **statistical, testing**, test patterns are selected according to a defined probability distribution on the input domain; the distribution and the number of input data are determined according to the adopted criteria [David & Thévenod-Fosse 81, Duran & Ntafos 84, Thévenod-Fosse & Waeselynck 91].

A.5.4 Fault forecasting

Fault forecasting is conducted by performing an *evaluation* of the system behaviour with respect to fault occurrence or activation. Evaluation has two aspects :

- *ordinal evaluation*, aimed at identifying, classifying and ordering the failure modes, or the methods and techniques aimed at avoiding them;
- *probabilistic evaluation*, aimed at evaluating in terms of probabilities some of the attributes of dependability.

The methods and tools enabling ordinal and probabilistic evaluations are either specific (e.g., failure mode and effect analysis for qualitative evaluation, or Markov chains for quantitative evaluation), or can be used for performing both forms of evaluation (e.g. reliability block diagrams, fault-trees).

The definition of the measures of dependability necessitates first the notions of proper and improper services to be defined :

- **proper service**, where the delivered service fulfils the system function;
- **improper service**, where the delivered service does not fulfil the system function.

A failure is thus a transition from proper to improper service, and the transition from improper to proper service is a **restoration**. Quantifying the alternation of proper-improper service delivery enables then reliability and availability to be defined as measures of dependability :

- reliability: a measure of the continuous delivery of proper service — or, equivalently, of the time to failure;
- availability: a measure of the delivery of proper service with respect to the alternation of proper and improper service.

A third measure, **maintainability**, is usually considered, which may be defined as a measure of the time to restoration from the last experienced failure, or equivalently, of the continuous delivery of improper service.

As a measure, safety can be seen as an extension of reliability. Let us group the state of proper service together with the state of improper service subsequent to benign failures into a safe state (in the sense of being free from catastrophic damage, not from danger); **safety** is then a measure of continuous safeness, or equivalently, of the time to catastrophic failure. Safety can thus be considered as reliability with respect to the catastrophic failures.

In the case of multi-performing systems, several services can be distinguished, as well as several modes of service delivery, ranging from full capacity to complete disruption, which can be seen as distinguishing less and less proper service deliveries. Performance-related measures of dependability for such systems are usually grouped under the notion of **performability** [Meyer 78, Smith & Trivedi 88].

When performing an evaluation, the approaches differ significantly according to whether the system is considered as being in stable reliability or in reliability growth, which may be defined as follows [Laprie et al. 90b]:

- **stable reliability**: the system's ability to deliver proper service is preserved (stochastic identity of the successive times to failure);
- **reliability growth**: the system's ability to deliver proper service is improved (stochastic increase of the successive times to failure).

When evaluating fault-tolerant systems, the coverage of error processing and fault treatment mechanisms has a very significant influence [Bouricius et al. 69, Arnold 73]; its evaluation can be performed either through modelling [Dugan & Trivedi 89] or through testing, then called *fault-injection* [Arlat et al. 90, Gunneflo et al. 89].

A.6 Dependability Principles

In dependable system engineering, some design guidelines and approaches have proved to be useful and are accepted principles in specific application sectors. Some of these principles are even highly recommended by these sectors standards. Some examples of such principles are presented here.

A.6.1 Identification and Authentication

This principle is mostly recommended for sectors where security is a major concern. The definition given here is adapted from the ITSEC [CEC 91].

In many systems there are requirements to determine and control the users who are permitted access to resources controlled by the system. This involves not only establishing the claimed identity of a user, but also verifying that the user is indeed the user claimed. This is done by the user providing the system with some information that is known by the system to be associated with the user in question.. Identification and authentication shall cover any functions intended to establish and verify a claimed identity.

Identification and authentication shall include any functions to enable new user identities to be added, and old user identities to be removed or invalidated. Similarly, it shall include any functions to generate, change, or allow authorised users to inspect, the authentication information required to verify the identity of particular users. It shall also include functions to assure the integrity of, or prevent the unauthorised use of, authentication information. It shall include any functions to limit the opportunity for repeated attempts to establish a false identity.

A.6.2 Access Control

Access control is common practice in sectors where security is a major concern. The definition given here is adapted from the ITSEC [CEC 91].

In many systems there are requirements to ensure that users and processes acting on their behalf are prevented from gaining access to information or resources that they are not authorised to access or have no need to access. Similarly, there are requirements concerning the unauthorised creation or amendment (including deletion) of information.

Access control principle shall cover any functions intended to control the flow of information between, and the use of resources by, users, processes and objects. This includes the administration (i.e. the granting and revocation) of access rights and their verification.

Access control shall include any functions to set up and maintain any lists or rules governing the rights to perform different types of access. It shall include any functions concerned with temporarily restricting access to objects that are simultaneously accessible by several users or processes and are needed to maintain the consistency and accuracy of such objects. It shall include any functions to ensure that upon creation, default access lists or access rules apply to objects. It shall include any functions to control the propagation of access rights to objects. It shall also include any functions to control the inference of information by the aggregation of data from otherwise legitimate accesses.

According to this definition, the main purpose of access control is to prevent unauthorised (presumably malicious) actions. But access control mechanisms, and more generally any protection means, can be useful to detect errors and prevent their propagation., be these errors due to malicious actions or accidental faults. Thus, access control can be a way to implement error confinement regions (see A.5.2 and A.6.11) and can have much larger application than just security concerns.

A.6.3 Accountability and Audit

Accountability and audit are also common practices in sectors where security is a major concern. The definitions given here are adapted from the ITSEC [CEC 91].

In many systems there are requirements to ensure that relevant information is recorded about actions performed by users or processes acting on their behalf so that the consequences of those actions can later be linked to the user in question, and the user held accountable for his actions. Accountability shall cover any functions intended to record the exercising of rights which are relevant to security.

Moreover, in many systems there are requirements to ensure that sufficient information is recorded about both routine and exceptional events that later investigations can determine if security violations have actually occurred, and if so what information or other resources were compromised. Audit shall cover any functions intended to detect and investigate events that might represent a threat to security.

Accountability and audit shall include functions related to the collection, protection and analysis of such information.

A.6.4 Object Re-use

Object reuse is an important topic for sectors where confidentiality is a major concern. The definition given here is adapted from the ITSEC [CEC 91].

In many systems there will be requirements to ensure that resources such as main memory and areas of disk storage can be reused while preserving security. Object reuse shall cover any functions intended to control the reuse of data objects. It shall include functions to initialise or clear unallocated or reallocated data objects. It shall include any functions to initialise or clear reusable media such as magnetic tapes, or to clear output devices such as display screens when not in use.

A.6.5 Levels of Redundancy

As indicated in A.5.2, a widely-used method to attain fault tolerance is to perform multiple (equivalent) computations through multiple channels. When dealing with physical faults, the channels may be identical, based on the assumption that hardware components fail independently. Then the outputs of the various channels are identical in absence of faults, if the computations can be made deterministic (i.e. if computations can be made independent from the inherent discrepancies between the various channels: different local time values for the same events, local information such as host-ids or addresses, etc.).

The level of redundancy, i.e. the channel multiplicity, has to be chosen according to the way errors are to be detected and the faulty channels diagnosed, according to the expected system behaviour in presence of faults The level of redundancy, i.e. the channel multiplicity, has to be chosen according to the way errors are to be and according to the number of faults to be tolerated :

- If the errors are detected by specific error detection means (watch-dogs, error detecting codes, likelihood checking, etc.), these means allow also to identify and isolate faulty channels and channel multiplicity is used for continuous operation despite faults affecting some channels. In that case, *n* channels are used to tolerate *n*-1 faults.
- On the contrary, if errors are detected by comparing the outputs produced by the various channels, at least two channels are necessary to detect errors and at least three different

channels are necessary to identify one faulty channel. Moreover, if an agreement must be guaranteed between the non faulty channels and if the faulty channels can exhibit inconsistent ("Byzantine") failures, some agreement protocols may require higher levels of redundancy (e.g. $3f+1$ independent channels to tolerate f Byzantine channel failures).

A.6.6 Design Diversity

Multiple channels can be used to tolerate faults if the corresponding redundancies are independent with respect to the process of fault creation and activation. While identical channels can be used when considering only physical faults, this is not true if you have to consider faults affecting the design process: software faults, faults in the design of the hardware or of the hardware components, faults in the development or manufacturing environment (compilers, test units, etc.), in the software distribution, etc. Such faults may affect identically several identical channels.

In that case, the various channels have to be designed and developed independently. Design diversity can be imposed on different parts of the system development: application software, operating system, languages, hardware, etc. But diversely designed channels can exhibit non-identical behaviours when running the same computations, and provision has to be made against discrepancies between non-faulty channels.

A.6.7 Fail-Halt System

A system can be designed so that, when an error is detected, its outputs are frozen (fail-passive system) or silent (fail-silent system). To do so, the error detection mechanisms can isolate the outputs, stop the clock, switch the power supply off, etc. Or the computing system can be forced to reach a stable, predefined state. The coverage of the fail-halt property is defined as the probability that the system halts when a failure occurs.

This property can be very useful in some safety-critical systems where halting the computer system prevents the occurrence of catastrophic failures, i.e. where the controlled system reaches systematically a safe state when the computer halts.

In some other cases, the fail-halt property is not sufficient for the controlled system to reach a safe state. For instance, in railway systems, a stopped train is in a safe state only if this stop is signalled to the following trains on the same track. A system which, when failing, is able to signal its failure and then halts, is usually called "fail-stop".

For safety critical systems whose safety depends on fail-halt or fail-stop properties, the property coverage has to be proven to be sufficiently high to satisfy the safety objectives.

Fail-halt components are also often used as building blocks to construct fail-operational systems, since the knowledge of the failing component behaviour helps to simplify the design of fault tolerant mechanisms at the system level.

A.6.8 Graceful Degradation

On the contrary to fail-halt systems, if a fault tolerant system has to stay operational after the failure of one or several of its components, the fault treatment may cause a reduction of the processing capacity of the system by passivation of the faulty unit(s). This processing capacity reduction may last as long as needed by the maintenance to reconfigure the system and bring it back to its nominal configuration. The system should be designed to minimise the consequences of this capacity reduction, i.e., to minimise the service degradation. Three kinds of degradation can be considered:

- Redundancy degradation: the processing capacity reduction results in a reduction of the system redundancy, and thus reduces the ability of the system to tolerate further faults, or at least reduce its ability to tolerate as many faults as the nominal configuration. This can be acceptable if the system is not safety critical or if the mean duration of the degradation is short enough for the probability of occurrence of further faults be low enough.
- Performance degradation: the processing capacity reduction results in an increase of the tasks execution time (time performance degradation) or in accuracy (accuracy performance degradation). This can be accepted if the real time requirements or accuracy requirements are not too stringent.
- Task degradation: the reduced processing capacity is devoted to the most critical tasks, and the least critical tasks are aborted.

The system should be designed to exhibit a behaviour mixing these three classes of degradation to satisfy as much as possible the system requirements despite the occurrence of one or several successive component failures.

A.6.9 Defence in Depth

To cope with the expected risks and threats, protection and fault tolerance mechanisms have to be implemented in the system. But in most cases, no single mechanism presents a high enough coverage to satisfy the dependability requirements of the system. Thus, several mechanisms have to be integrated in such a way that even if one mechanism fails, other mechanisms will prevent the system failure. For instance, an intruder would have to by-pass or break several protection mechanisms (authentication, access controls) to perform an intrusion. Or conversely, an error which has not been detected by an error detection mechanism, has to be likely to be detected by others before the error propagates to the boundaries of the system, thus provoking a system failure.

A.6.10 Defensive Programming

Defensive programming aims at limiting error propagation, by inserting into functional program modules some non-functional components (usually called “executable assertions”) which verify certain properties on module input data, output data or intermediary results [Rabéjac 95]. Defensive programming gives program modules a robust behaviour against errors, whether these errors are due to incorrect interactions with other modules or to internal faults. When an error is detected by an assertion, an exception is raised and another component (called exception handler) is run in order to prevent error propagation. Being able to access internal software states, assertions can provide early error detection. Formal methods, can be used to develop and validate executable assertions.

A.6.11 Partitioning and Separation

In order to limit the consequences of component failures, the system has to be partitioned in several units and these units are to be separated and isolated to reduce the risks of common mode faults (i.e., single faults affecting several units) and of error propagation between units (error confinement regions). This principle can be used to limit the extent of intrusions (related with defence-in-depth principle, see A.6.9), and to organise redundant units to build fault tolerant systems.

Annexe B - Glossary

No.	Term	Definition
1.	Assessment	The process of analysis to determine whether the design authority and the validator have achieved a system that meets the specified requirements and to form a judgement as to whether the system is fit for its intended purpose.
2.	Authenticity	The validity of a claimed identity (Remark: The definition of integrity includes this property)
3.	Availability	a) Dependability with respect to readiness for usage. b) A measure of the delivery of proper service with respect to the alternation of proper and improper service.
4.	Confidence Level	Level of confidence concerning one dependability attribute. The Confidence Level can be a requirement (i.e. part of the dependability profile) or the result of the assessment with respect to this attribute of the resulting system. The rating with respect to one dependability attribute which quantifies the level of confidence in the system having that attribute is given in levels from 0 to 4. Level 0 means no requirements or no confidence level achieved. Level 4 is the highest level.
5.	Confidence Providing Activity	Those activities which constitute a Dependability Related Process.
6.	Confidence Providing Process	The processes • Dependability Requirements Validation • Correctness Verification • Dependability Validation and • Process Quality which comprise the Confidence Providing Activities that have to be carried out during system development.
7.	Confidentiality	Non occurrence of unauthorised disclosure of information.
8.	Correctness Verification	Verify that the functions implemented conform with their specifications. Correctness Verification is a Dependability Related Process.

No.	Term	Definition
9.	Criteria	Description of the means by which confidence in the dependability of a system can be established. (term specific to SQUALE)
10.	Dependability Target	A description of what will be assessed including the identified hazards and ratings, the dependability profile, the dependability objectives, the dependability policy, an overview of the system architecture and the system environment.
11.	Dependability	That property of a system, such that reliance can justifiably be placed on the service it delivers.
12.	Dependability Allocation	Assignment of dependability objectives and the statements of the dependability policy to the defined parts of the system (physical, personal, IT, organisation, subsystem, component) using dependability principles.
13.	Dependability Attribute	is one of the following: • Availability • Reliability • Safety • Confidentiality • Integrity • Maintainability
14.	Dependability Objective	Dependability properties that need to be satisfied by a system or subsystem/component *(to remove identified hazards or reduce their effect to an acceptable level)*.
15.	Dependability Policy	Specifies the set of regulations, standards, practices, principles and procedures that should be used to achieve the dependability objectives.
16.	Dependability Profile	Combination of confidence levels for each of the Dependability attributes.
17.	Dependability Principle	Design guidelines and approaches that are generally accepted in a specific sector. Examples are: Fail Stop, Defensive Programming, Redundancy, Partitioning, Defence in Depth, ITSEC Generic Headings.
18.	Dependability Related Functions	Those that contribute or influence the satisfaction of the dependability objectives.

No.	Term	Definition
19.	Dependability Requirement Validation	Dependability Requirement Validation has to ensure, that at each level of system decomposition the hazards and threats of a TDA have been properly identified, that they are covered by the respective dependability objectives, the dependability Policy and the dependability related functions and that they comply with the initial needs of the dependable system. Dependability Requirement Validation is a Dependability Related Process.
20.	Dependability Validation	The dependability Validation checks that the Dependability Objectives are completely and effectively covered by the TDA. Dependability Validation is a Dependability Related Process.
21.	Detail	Defines the scope of the CPA such as whether it addresses: • parts or all of the system, • one or many refinement levels, • all the properties or only a subset.
22.	Hazard	A hazard is an event with potentially unacceptable consequences.
23.	Hazard Analysis	Identification of causes for hazards and definition of which hazards are expected to be reduced or removed by the system. • Identify the causes for hazards in the environment that the system should protect against • Identify the causes for hazards introduced by failures of the system. • Identify the causes for hazards on the system caused by the environment.
24.	Hazard Rating	Rating the acceptability of hazards to establish dependability objectives.

No.	Term	Definition
25.	Independence	Separation of responsibilities which ensures the accomplishment of objective verification, validation or assessment • Independent person: A person who is independent and distinct from the developers, and who does not have direct responsibility for development activities. • Independent department: A department which is separate and distinct from the departments responsible for the main development of the system. • Independent organisation: An organisation which is separate and distinct, by ways of management and other resources, from the organisations responsible for the main development of the system.
26.	Integrity	Non-occurrence of improper alterations of the system
27.	Maintainability	Ability to undergo repairs and evolution
28.	Preliminary Hazard Analysis	The preliminary hazard analysis is not to be confused with hazard analysis. The aim of a Preliminary Hazard Analysis is to determine the criticality of the System by finding hazards, rating them and deriving a Dependability Profile and Dependability Objectives. A preliminary hazard analysis consists of three steps: 1. The identification of hazards, the identification of unacceptable consequences (failures) of hazards and the identification of causes (faults) for hazards. 2. The estimation of the occurrence of the causes and consequences and of the severity of the consequences, often known as hazard rating. This should be done for different attributes (damages for persons, equipment, finance, ...). This estimation results in Confidence Levels for every Dependability Attribute and in Dependability Objectives for the reduction of the occurrence of identified causes and/or the reduction of the severity of consequences. 3. A validation of whether the Dependability Objectives or related functions reduces the occurrence of identified causes and/or the severity of consequences.

No.	Term	Definition
29.	Process Quality	Process Quality ensures, that the system development is properly structured, planned and controlled. Process Quality is a Dependability Related Process. The Criteria address only those quality checks that are specific to a dependable system.
30.	Quality Assurance	All the planned and systematic activities implemented within the quality system and demonstrated as needed, to provide adequate confidence that the entity will fulfil requirements for quality.
31.	Reliability	A measure of the continuous delivery of proper service (or of the time to failure) Dependability with respect to the continuity of service.
32.	Required Dependability Property	Condition or capability that must be met or possessed by a system or system component to satisfy the dependability objectives.
33.	Rigour	The level of rigour for the application of a CPA indicates how the activity has to be done, the degree of justification to be provided to the assessor and the kind of evidence required.
34.	Risk	The probable rate of occurrence of a hazard causing harm and the degree of severity of the harm.
35.	Safety	Non occurrence of catastrophic consequences for the environment (in terms of human injury, deaths, financial loss, etc.)
36.	Security	Confidentiality and integrity together with availability
37.	Service	System behaviour as perceived by the system user.
38.	System	An integrated composite (people, products and processes) that provide a capability to satisfy a stated need or objective, bound together to interact.

No.	Term	Definition
39.	Target of Dependability Assessment (TDA)	The system (or part of a system), identified within the Dependability Target, that will be subject of an assessment according to these Criteria. It will include all the components that: • enforce the dependability requirements claimed for the TDA (dependability enforcing components) • are depended upon to operate correctly while supporting the dependability enforcing components (dependability related components) • are neither dependability enforcing nor dependability related but cannot be shown to be separate and independent from the dependability enforcing and dependability related components The TDA corresponds exactly to the Target of Evaluation (TOE) in ITSEC evaluations.

Annexe C - Bibliography

Arlat et al. 90 J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins and D. Powell, “Fault Injection for Dependability Validation — A Methodology and Some Applications”, *IEEE Trans. on Software Engineering*, 16 (2), pp.166-182, February 1990.

ARP 4754 Certification Considerations for Highly-Integrated or Complex Aircraft Systems, SAE/ARP 4754, SAE, September 1, 1995

ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, SAE/ARP 4761, SAE, December 28, 1994

Anderson & Lee 81 T. Anderson, P.A. Lee, *Fault Tolerance — Principles and Practice,* Prentice Hall, 1981.

Arnold 73 T.F. Arnold, "The concept of coverage and its effect on the reliability model of repairable systems", *IEEE Trans. on Computers,* vol. C-22, June 1973, pp. 251-254.

Avizienis 67 A. Avizienis, "Design of fault-tolerant computers", in *Proc. Fall Joint Computer Conf.*, 1967, pp. 733-743.

Avizienis 78 A. Avizienis, "Fault tolerance, the survival attribute of digital systems", *Proceedings of the IEEE,* vol. 66, no. 10, Oct. 1978, pp. 1109-1125.

Avizienis & Kelly 84 A. Avizienis, J.P.J. Kelly, "Fault tolerance by design diversity: concepts and experiments", *Computer,* vol. 17, no. 8, Aug. 1984, pp. 67-80.

Avizienis & Laprie 86 A. Avizienis, J.C. Laprie, "Dependable computing: from concepts to design diversity", *Proceedings of the IEEE,* vol. 74, no. 5, May 1986, pp. 629-638.

Béounes *et al.* 93 C. Béounes, M. Aguéra, J. Arlat, C. Bourdeau, J.-E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell and P. Spiesser, “SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems”, in *Proc. 23rd Int. Symp. on Fault-Tolerant Computing (FTCS-23),* pp.668-673, IEEE Computer Society Press, Toulouse, France, June 1993

Bouricius et al. 69 W.G. Bouricius, W.C. Carter, P.R. Schneider, "Reliability modeling techniques for self-repairing computer systems", in *Proc. 24th ACM National Conf., 1969*, pp. 295-309.

Carter 82 W.C. Carter, "A time for reflection", in *Proc. 12th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12),* Santa Monica, California, June 1982, p. 41.

Carter & Schneider 68 W.C. Carter, P.R. Schneider, "Design of dynamically checked computers", in *Proc. IFIP'68 Cong.,* Amsterdam, 1968, pp. 878-883.

CCITT 84 *Termes et définitions concernant la qualité de service, la disponibilité et la fiabilité*, Recommandation G 106, CCITT, 1984; in French.

CEC 91 *Information Technology Security Evaluation Criteria,* Harmonized Criteria of France, Germany, the Netherlands, the United Kingdom, Commission of the European Communities, 1991.

CEI 92 *Industrial-process measurement and control — Evaluation of system properties for the purpose of system assessment. Part 5: Assessment of system dependability*, Draft, Publication 1069-5, CEI Secretariat, Feb. 1992.

Cheheyl et al. 81 M.H. Cheheyl, M. Gasser, G.A. Huff, J.K. Miller, "Verifying security", *Computing Surveys,* vol. 13, no. 3, Sep. 1981, pp. 279-339.

CNET 93 Centre National Des Télécommunications, “Recueil de données de fiabilité de composants électroniques”, 1993

Ciardo *et al.* 89 G. Ciardo, J. Muppala and K. Trivedi, “SPNP: Stochastic Petri Net Package”, in *Petri Nets and Performance Models (PNPM89),* pp.142-151, IEEE Computer Society Press, Kyoto, Japan, December 1989.

Common Criteria Common Criteria for Information Technology Security Evaluation, Common Criteria Implementation Board, Version 2.0, CCIB-98-026, CCIB-98-027, CCIB-98-027A, CCIB-98-028, May 1998

Cristian et al. 85 F. Cristian, H. Aghili, R. Strong, D. Dolev, "Atomic broadcast: from simple message diffusion to Byzantine agreement", in *Proc. 15th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-15),* Ann Arbor, Michigan, June 1985, pp. 200-206.

Cristian 88 F. Cristian, "Agreeing on who is present and who is absent in a synchronous distributed system", in *Proc. 18th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-18),* Tokyo, June 1988, pp. 206-211.

CSE-SRM A Guide to Security Risk Management for Information Technology Systems, Interim Version 1.0, Communications Security Establishment, Government of Canada, Aug. 1995

CSE-RA A Guide to Risk Assessment and Safeguard Selection for Information Technology Systems, Interim Version 1.0, Communications Security Establishment, Government of Canada, Aug. 1995

CSE-CA A Guide to Certification and Accreditation for Information Technology Systems, Interim Version 1.0, Communications Security Establishment, Government of Canada, Aug. 1995

David & Thévenod-Fosse 81 R. David, P. Thévenod-Fosse, "Random testing of integrated circuits", *IEEE Trans. on Instrumentation and Measurement,* vol. IM-30, no. 1, March 1981, pp. 20-25.

Davis 93 A. M. Davis, *Software Requirements,* ISBN N°0-13-805763-X, PTR Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

DeMillo et al. 78 R.A. DeMillo, R.J. Lipton, F.G. Sayward, "Hints on test data selection: help for the practicing programmer", *Computer*, April 1978, pp. 34-41.

Diaz 82 M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models", *Computer Networks,* vol. 6, no. 6, Dec. 1982, pp. 419-441.

DoD-65 Military Standardization Handbook: Reliability Prediction of Electronic Equipment
MIL-HDBS-217, US Department of Defense, 1965

DoD-882C System Safety Program Requirements
MIL-STD-882C, US Department of Defense, Washington D.C., Jan. 1993

Dugan & Trivedi 89 J.B. Dugan, K.S. Trivedi, "Coverage modeling for dependability analysis of fault-tolerant systems", *IEEE Trans. on Computers,* vol. 38, no. 6, June 1989, pp. 775-787.

Duran & Ntafos 84 J.W. Duran, S.C. Ntafos, "An evaluation of random testing", *IEEE Trans. on Software Engineering,* vol. SE-10, no. 4, July 1984, pp. 438-444.

Dyer 92 M. Dyer, *The Cleanroom Approach to Quality Software Development,* Wiley Series in Software Engineering Practice, 198p., John Wiley & Sons, Inc., New York, 1992.

Elmendorf 72 W.R. Elmendorf, "Fault-tolerant programming", in *Proc. 2nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-2),* Newton, Massachusetts, June 1972, pp. 79-83.

EN 50126 Railway applications – The specification and demonstration of reliability, availability, maintainability and safety (RAMS), European Committee for Electrotechnical Standardization (CENELEC), June 1997

EN 50128 Railway Applications: Software for Railway Control and Protection System*s*, European Committee for Electrotechnical Standardization (CENELEC), January 1997.

EN 50129 Railway Applications: Safety Related Electronic Systems for Signalling, European Committee for Electrotechnical Standardization (CENELEC), April 1997.

Frison & Wensley 82 S.G. Frison, J.H. Wensley, "Interactive consistency and its impact on the design of TMR systems", *Proc. 12th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12),* Santa Monica, California, June 1982, pp. 228-233.

Gilb and Graham 93 T. Gilb and D. Graham, *Software Inspection,* ISBN N°0-201-63181-4, Addison-Wesley, UK, 1993

Goodenough & Gerhart 75 J.B. Goodenough, S.L. Gerhart, "Toward a theory of test data selection", *IEEE Trans. on Software Engineering,* vol. SE-1, no. 2, June 1975, pp. 156-173.

Gunneflo et al. 89 U. Gunneflo, J. Karlsson, J. Torin, "Evaluation of error detection schemes using fault injection by heavy-ion radiation", in *Proc. 19th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-19),* Chicago, June 1989, pp. 340-347.

Hawes 95 A. Hawes, "Extending a Security Evaluation Standard (the ITSEC) to Dependability", *Achievement and Assurance of Safety*, F. Redmill, T. Anderson, eds, Springer Verlag, 1995, pp. 117-130.

Hoare 69 C.A.R. Hoare, "An axiomatic basis for computer programming", *Communications of the ACM*, vol. 12, no. 10, Oct. 1969, pp. 576-583.

Hosford 60 J.E. Hosford, "Measures of dependability", *Operations Research,* vol. 8, no. 1, 1960, pp. 204-206.

Howden 76 W.E. Howden, "Reliability of the path analysis testing strategy", *IEEE Trans. on Software Engineering,* vol. SE-2, no. 3, Sep. 1976, pp. 208-215.

Howden 87 W.E. Howden, *Functional Program Testing and Analysis*, McGraw-Hill, 1987.

Humphrey 89 W. S. Humphrey, *Managing the Software Process,* ISBN N°0-201-18095-2, Addison-Wesley, 1989.

IEC 880 Software for Computers in the Safety Systems of Nuclear Power Systems, IEC, 1986

IEC 1508 Draft IEC 1508 - Functional safety: safety-related systems, Part 1 to 7, IEC CD, June 1995

Ippolito & Wallace L. M. Ippolito, D. R. Wallace, "A Study on Hazard Analysis in High Integrity Software Standards and Guidelines", NISTIR 5589, National Institute for Standards and Technology, Gaithersburg, Jan. 1995

ISO 91 *Quality Concepts and Terminology, Part one: Generic Terms and Definitions*, Document ISO/TC 176/SC 1 N 93, Feb. 1992.

IT-SHB IT-Sicherheitshandbuch, *Handbuch für die sichere Anwendung der Informationstechnik*, Version 1.0 - März 1992, BSI 7105, Bundesamt für Sicherheit in der Informationstechnik.

Jacob 1991 J. Jacob. "The Basic Integrity Theorem," *Proc. Int. Symp. on Security and Privacy,* pp. 89-97, Oakland, CA, USA, 1991.

Kanoun *et al.* 96 K. Kanoun, J.-C. Laprie, P. Thevenod, D. A. Wolff, A. J. Campbell and P. Anders, *Means to Assess the Dependability of Core Modules*, BRITE_EURAM IMAGES 2000 Contract, LAAS Report 96010, 1996

Kopetz & Ochsenreiter 87 H. Kopetz, W. Ochsenreiter, "Clock synchronization in distributed real-time systems, *IEEE Trans. on Computers*, vol. C-36, no. 8, Aug. 1987, pp. 933-940.

Lamport & Melliar-Smith 85 L. Lamport, P.M. Melliar-Smith, "Synchronizing clocks in the presence of faults", *Journal of the ACM,* vol. 32, no.1, Jan. 1985, pp. 52-78.

Lala 86 J.H. Lala, "A byzantine resilient fault tolerant computer for nuclear power plant applications", *Proc. 16th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-16),* Vienna, Austria, June 1986, pp. 338-343.

Landwehr et al. 93 C.E. Landwehr, A.R. Bull, J.P. McDermott, W.S. Choi, "A taxonomy of computer security flaws, with examples", Naval Research Laboratory, Report no. NRL/FR/5542-93-9591, Nov. 1991.

Laprie & Costes 82 J.C. Laprie, A. Costes, "Dependability: a unifying concept for reliable computing", *Proc. 12th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12),* Santa Monica, California, June 1982, pp. 18-21.

Laprie 85 J.C. Laprie, "Dependable computing and fault tolerance: concepts and terminology", in *Proc. 15th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-15),* Ann Arbor, Michigan, June 1985, pp. 2-11.

Laprie et al. 90a J.C. Laprie, J. Arlat, C. Béounes, K. Kanoun, "Definition and analysis of hardware- and software-fault-tolerant architectures", *IEEE Computer,* vol. 23, no. 7, July 1990, pp. 39- 51.

Laprie et al. 90b J.C. Laprie, C. Béounes, M. Kaâniche, K. Kanoun, "The KAT (knowledge-action-transformation) approach to the modeling and evaluation of reliability and availability growth", *IEEE Trans. on Software Engineering*, vol. 17, no. 4, April 1991, pp. 370-382.

Laprie 92 J.C. Laprie (Ed.), *Dependability: Basic Concepts and Terminology*, Springer-Verlag, Vienna, 1992.

Laprie 95 J.C. Laprie, "Dependability - Its attributes, impairments, and means", in *Predictably Dependable Computing Systems*, B.Randell, J.C. Laprie, H. Kopetz, B. Littlewood, eds, Springer-Verlag, 1995, pp. 3-18.

Lyu 1995 M. R. Lyu (Ed.), *Handbook of Software Reliability Engineering,* ISBN N°0-07-039400-8, McGraw-Hill, 1995.

McCabe 76 T.J. McCabe, "A complexity measure", *IEEE Trans. on Software Engineering,* vol. SE-2, no. 4, Dec. 1976, pp. 308-320.

Meyer 78 J.F. Meyer, "On evaluating the performability of degradable computing systems", in *Proc. 8th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-8),* Toulouse, France, June 1978, pp. 44-49.

Mine & Koga 67 H. Mine, Y. Koga, "Basic properties and a construction method for fail-safe logical systems", *IEEE Trans. on Electron. Computers,* vol. EC-16, no. 6, June 1967, pp. 282-289.

Morell 90 L.J. Morell, "A theory of fault-based testing", *IEEE Trans. on Software Engineering*, vol. 16, no. 8, Aug. 1990, pp. 844-857.

Myers 79 G.J. Myers, *The Art of Software Testing*, John Wiley & Sons, 1979

Nicolaidis et al. 89 M. Nicolaidis, S. Noraz, B. Courtois, "A generalized theory of fail-safe systems", in *Proc. 19th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-19),* Chicago, USA, June 1989, pp. 398-406.

NRL-Handbook Handbook for the Computer Security Certification of Trusted Systems Naval Research Laboratory, Code 5540, Washington, D.C.

Osterweil et al. 76 L.J. Osterweil, L.D. Fodsick, "DAVE — A validation error detection and documentation system for Fortran programs", *Software Practice and Experience*, Oct.-Dec. 1976, pp. 473-486.

Pearson et al. 98 S. Pearson, S. Riddle, A. Saeed, "Traceability for the development and assessment of safe avionic systems", in *Proc. 8th Int. Symp. of the International Council ems Engineering (INCOSE-98),* Vancouver, Canada, July 1998, pp. 445-452.

Powell et al. 88 D. Powell, G. Bonn, D. Seaton, P. Verissimo, F. Waeselynck, "The Delta-4 approach to dependability in open distributed computing systems", in *Proc. 18th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-18),* Tokyo, Japan, June 1988, pp. 246-251.

Powell 92 D. Powell, "Failure Mode Assumptions and Assumption Coverage", *Proc. 22nd IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-22),* Boston, July 1992, pp.386-395.

Ramamoorthy 84 C.V. Ramamoorthy, A. Prakash, W.T. Tsai, Y. Usuda, "Software engineering: problems and perspectives", *IEEE Computer*, Oct. 1984, pp. 191-209.

Randell 75 B. Randell, "System structure for software fault tolerance", *IEEE Trans. on Software Engineering,* vol. SE-1, no. 2, June 1975, pp. 220-232.

Rapps & Weyuker 85 S. Rapps, E.J. Weyuker, "Selecting software test data using data flow information", *IEEE Trans. on Software Engineering,* vol. SE-11, no. 4, April 1985, pp. 367-375.

Rabéjac 95 C. Rabéjac, *Auto-surveillance logicielle pour applications critiques : méthodes et mécanismes*, Doctorate Thesis, Institut National Polytechnique de Toulouse, 27 November 1995, Tech. Report LAAS-CNRS n.95449, in French.

Roth et al. 67 J.P. Roth, W.G. Bourricius, P.R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits", *IEEE Trans. on Electronic Computers*, vol. EC-16, Oct. 1967, pp. 567-579.

RTCA-DO178B Software Considerations in Airborne Systems and Equipment Certification RTCA/DO-178B/ED-12B, RTCA, December 1, 1992

Rushby 1993 J. Rushby, *Formal Methods and the Certification of Critical Systems*, SRI International, CSL Technical Report, N°SRI-CS-93-07, November 1993

Sahner *et al* 1996 S. A. Sahner, K.S. Trivedi, and A. Puliafito *Performance and Reliability Analysis of Computer Systems,* ISBN N°0-7923-9650-2, Kluwer Academic Publishers, boston, USA, 1996

Sanders *et al.* 1995 W. H. Sanders, W. D. Obal II, M. A. Qureshi and F. K. Widjanarko, "The *UltraSAN* Modeling Environment", *Performance Evaluation,* 21 (special "Performance Evaluation Tools") 1995

Schlichting & Schneider 83 R.D. Schlichting, F.B. Schneider, "Fail-stop processors: an approach to designing fault-tolerant computing systems", *ACM Trans. on Computing Systems,* vol. 1, no. 3, Aug. 1983, pp. 222-238.

Siewiorek & Johnson 82 D.P. Siewiorek, D. Johnson, "A design methodology for high reliability systems: the Intel 432", in D.P. Siewiorek, R.S. Swarz, *The Theory and Practice of Reliable System Design,* Digital Press, 1982, pp. 737-767.

Siewiorek & Swarz 92 D.P. Siewiorek, R.S. Swarz, *Reliable Computer Systems, Design and Evaluation*, Digital Press, 1992, pp. 737-767.

Smith & Trivedi 88 R.M. Smith, K.S. Trivedi, A.V. Ramesh, "Performability analysis: measures, an algorithm, and a case study", *IEEE Trans. on Computers*, vol. 37, no. 4, April 1988, pp. 406-417.

Thatte & Abraham 78 S.M. Thatte, J.A. Abraham, "A methodology for functional level testing of microprocessors", in *Proc. 8th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-8),* Toulouse, France, June 1978, pp. 90-95.

Thévenod & Waeselynck 91 P. Thévenod-Fosse, H. Waeselynck, "An investigation of statistical software testing", *Journal of Software Testing, Verification and Reliability*, vol. 1, no. 2, 1991, pp. 5-25.

Wakerly 78 J.F. Wakerly, *Error-Detecting Codes, Self-Checking Circuits, and Applications*, New York: Elsevier North-Holland, 1978

Wood 94 A. Wood, “NonStop availability in a client/server environment”, Tandem Technical Report 94.1, March 1994.

Yau & Cheung 75 S.S. Yau, R.C. Cheung, "Design of self-checking software", in *Proc. 1975 Int. Conf. on Reliable Software*