

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Khalid AL-SARAYREH

IDENTIFICATION, SPECIFICATION AND MEASUREMENT,
USING INTERNATIONAL STANDARDS,
OF THE SYSTEM NON FUNCTIONAL REQUIREMENTS ALLOCATED TO REAL-
TIME EMBEDDED SOFTWARE

MONTREAL, AUGUST 23, 2011

© Copyright 2011 Khalid Al-Sarayreh

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain Abran, Ph.D, Thesis Supervisor
Software Engineering and Information Technology Department, École de technologie
supérieure

Mrs. Christine Tremblay, Ph.D, President of the Board of Examiners
Electrical Engineering Department, École de technologie supérieure

Mr. Pierre Bourque, Ph.D, Examiner
Software Engineering and Information Technology Department, École de technologie
supérieure

Mr. Yann-Gael Guéhéneuc, Ph.D, External Examiner
Computer Engineering and Software Engineering Department, École Polytechnique de
Montréal

THIS THESIS WAS PRESENTED AND DEFENDED

BEFORE A BOARD OF EXAMINERS AND THE PUBLIC

AUGUST 23, 2011

AT ECOLE DE TECHNOLOGIE SUPERIEURE

ACKNOWLEDGMENTS

First and foremost I would like to express my gratitude to Professor Alain Abran, my thesis supervisor at École de technologie supérieure, for his continuous support, time, advice and patience throughout this thesis. Without his patient guidance, this work would never have been carried out.

I am deeply grateful to my committee members Prof. Christine Tremblay, Pierre Bourque, and Yann-Gael Guéhéneuc for their time and effort in reviewing this work.

I would like to express my thanks to Prof. Charles Simon, (the President of the COSMIC group), who has provided me extensive feedback during my research program.

I would like to thank everyone in the Software Engineering Research Laboratory (GÉLOG), the Department of Software Engineering and IT. Finally, I wish to express my thanks to my family and my wife for their support and understanding throughout this long process.

**IDENTIFICATION, SPÉCIFICATION ET MESURES,
À L'AIDE DE STANDARDS INTERNATIONAUX,
DES BESOINS NON FONCTIONNELS DES SYSTÈMES ALLOUÉS AUX
LOGICIELS EMBARQUÉS EN TEMPS RÉEL**

Khalid Al-SARAYREH

RÉSUMÉ

Au cours de la phase de l'analyse des besoins pour le développement d'un système, l'accent est souvent porté sur les besoins fonctionnels, tandis que les besoins non fonctionnels (Non Functional Requirements - NFR) sont capturés par les analystes systèmes seulement à un niveau très global : au cours de cette phase, les NFR sont décrits typiquement au niveau du système et non au niveau du logiciel. Le détail de ces besoins non fonctionnels est souvent précisé (c'est à dire défini au niveau de détail nécessaire) par les concepteurs du système à la phase de l'architecture et à la phase de conception du système.

Pour le moment, il n'y a pas de consensus sur la manière de décrire et de mesurer les besoins non fonctionnels des systèmes (system-NFR) : c'est donc un défi de les prendre en compte dans l'estimation des efforts pour le développement des logiciels qui feront partie de ces systèmes et dans l'évaluation de la productivité des projets de développement de ces logiciels.

Durant la phase de l'analyse des besoins pour les logiciel, les besoins non fonctionnels des systèmes peuvent être décrits et spécifiés comme étant les besoins fonctionnels alloués au logiciel : ceci permet alors aux ingénieurs logiciels de développer, tester et configurer les livrables finaux aux utilisateurs du système.

La motivation de cette recherche est de contribuer à l'effort d'amélioration des modèles d'estimation des projets de développement logiciel en introduisant les exigences non fonctionnelles des systèmes au sein du processus d'estimation du projet logiciel et ce au travers d'une vue quantitative.

Le but de cette recherche est d'aider les chefs de projets, les organisations ainsi que les chercheurs, à prendre des décisions éclairées sur les plannings des projets et sur le développement des logiciels et ce pendant la phase initiale d'identification des besoins, des spécifications et d'estimation des besoins non fonctionnels d'un système incluant du logiciel embarqué. Plus précisément, cette étude a comme but de contribuer à une meilleure définition, description et estimation de certains entrants, qui sont les besoins non fonctionnels du système, nécessaire pour réaliser une estimation préalable des couts.

Dans les standards internationaux, un certain nombre de concepts sont utilisés afin de décrire différents types de besoins non fonctionnels des systèmes, que ces besoins soient au niveau

du système, du logiciel ou du matériel. L'objectif de cette recherche est d'effectuer, le plus en amont possible, les spécifications et la quantification des besoins fonctionnel pour le logiciel, dérivés des besoins non fonctionnels au niveau système, en utilisant comme base les standards de l'ingénierie du logiciel.

Pour atteindre cet objectif de recherche les deux sous-objectifs de recherche spécifiques suivants doivent être atteints:

- Conceptions de modèles standards pour l'identification et la spécification des besoins fonctionnels de l'utilisateur (FUR) alloués au logiciel pour répondre aux besoins non fonctionnels du système (system-NFR).
- Mesure de la taille fonctionnelle, en utilisant le standard COSMIC ISO 19761.

des besoins fonctionnels alloués au logiciel pour répondre aux besoins non fonctionnels du système les résultats de cette recherche sont à un ensemble de quatorze (14) modèles de spécifications et de mesure, basés sur des standards, pour les besoins non-fonctionnels du système qui peuvent être alloués à du logiciel embarqué temps réel.

La contribution principale de cette recherche est cet ensemble de quatorze modèles des besoins fonctionnels des utilisateurs du logiciel basés sur des standards pour l'identification, la spécification et la mesure des besoins non fonctionnels du système.

Mots clés: Génie logiciel, Besoins non fonctionnels (NFR), Standards internationaux ECSS, ISO 9126 et IEEE-830, Mesure des besoins fonctionnels de l'utilisateur du logiciel, COSMIC – ISO 19761.

**IDENTIFICATION, SPECIFICATION AND MEASUREMENT,
USING INTERNATIONAL STANDARDS,
OF THE SYSTEM NON FUNCTIONAL REQUIREMENTS ALLOCATED TO
REAL-TIME EMBEDDED SOFTWARE**

Khalid Al-SARAYREH

ABSTRACT

During the system requirements phase, the focus is often on the functional requirements of the system, while non-functional requirements (NFR) are captured by system analysts at a very global level only: in this system analysis phase, these NFR are typically described at the system level and not at the software level. Detailing these NFR is typically left to be handled (i.e., defined at the necessary level of detail) much later by system designers in the system architecture and design phases.

As yet, there is no consensus on how to describe and measure the system non-functional requirements (system-NFR); it is therefore challenging to take them into account in software project estimation and software project productivity benchmarking.

In the software requirements engineering step, the system-NFR can be detailed and specified as software functional user requirements (software-FUR), to allow a software engineer to develop, test, and configure the final deliverables to the system users.

The research project motivation is to contribute to the improvement of the estimation models of software development effort by including the system-NFR in the software estimation process through a quantitative view of such NFR.

The goal for this research project is to help project managers, organizations, and researchers to make informed decisions on project planning and software development projects in the early identification, specification, and measurement of the system-NFR for the embedded software. More specifically, this research project aims at contributing to better define, describe, and measure the system-NFR allocated to software-FUR for real time and embedded software..

The research objective is the early specification and measurement of software-FUR derived from system-NFR, using as a basis the systems and software engineering standards.

To achieve this research objective the following two specific research sub-objectives must be reached:

- Designs of standard-based generic models for the identification and specification of software-FUR for system-NFR;

- Measurement of the functional size of software-FUR for system-NFR using the COSMIC ISO 19761 standard.

The results of this research will be a set of standard-based specification and measurement models for system-NFR for real-time embedded software.

The main outcome of this research study is the set of fourteen (14) standard-based models of software-FUR for the early identification, specification, and measurement of system non-functional requirements allocated to software.

Keywords: Software Engineering, Non functional requirement (NFR), ECSS, ISO 9126 and IEEE830 International Standards, Software-FUR Measurement, COSMIC – ISO 19761.

TABLE OF CONTENTS

	Page
INTRODUCTION	1
CHAPTER 1 LITERATURE REVIEW	7
1.1 Introduction.....	7
1.2 NFR in the academic literature	8
1.3 NFR in international standards	11
1.3.1 European international standards (ECSS).....	11
1.3.2 IEEE 830 standard	12
1.3.3 ISO 9126 standard series	13
1.3.4 ISO 19759 (SWEBOK guide).....	14
1.3.5 ISO 19761 (COSMIC method).....	15
1.3.6 Software Functional Size Measurement (FSM).....	17
1.3.7 COSMIC guideline for sizing service oriented software.....	18
1.4 NFR ontologies	20
1.5 Summary	20
CHAPTER 2 RESEARCH GOAL, OBJECTIVES, AND METHODOLOGY	23
2.1 Introduction.....	23
2.2 Research motivation.....	23
2.3 Research goal.....	23
2.4 Research objectives.....	24
2.5 Users of research.....	24
2.6 Research input.....	24
2.7 Overview of the research methodology	25
2.8 Detailed research methodology.....	26
CHAPTER 3 IDENTIFICATION OF NFR CONCEPTS, AND VIEWS	30
3.1 Introduction.....	30
3.2 Reliability systems requirements	31
3.2.1 ECSS: views and concepts for reliability.....	31
3.2.2 IEEE: views and concepts for reliability	32
3.2.3 ISO views and concepts for reliability.....	32
3.3 Maintainability systems requirements	34
3.3.1 ECSS: views and concepts for maintainability.....	34
3.3.2 IEEE: views and concepts for maintainability.....	34
3.3.3 ISO: views and concepts for maintainability	35
3.4 Interface systems requirements.....	36
3.4.1 ECSS: views and concepts for interfaces.....	36

3.4.2	IEEE: view and concepts for interfaces	37
3.5	Portability systems requirements	39
3.5.1	ECSS: view and concepts for portability	39
3.5.2	IEEE: view and concepts for portability	39
3.5.3	ISO standards: views and concepts for portability	40
3.6	Operations systems requirements	41
3.6.1	ECSS: views and concepts for operations	42
3.6.2	IEEE: views and concepts for Operations	43
3.7	Configuration systems requirements.....	43
3.7.1	ECSS: views and concepts for configuration.....	43
3.7.2	ISO 19759 (SWEBOK Guide): views and concepts for configuration	44
3.8	Data definitions and database systems requirements.....	45
3.8.1	ECSS views and concepts for data definition and database	45
3.9	Adaptation and installation: systems requirements.....	47
3.9.1	ECSS: views and concepts for adaptation and installation.....	47
3.9.2	IEEE: views and concepts for adaptation and installation.....	48
3.9.3	ISO: views and concepts for adaptation and installation.....	49
3.10	Design and implementation constraints (D&I) systems requirements	50
3.10.1	ECSS: views and concepts for D&I constraints	50
3.10.2	ISO 19759 (SWEBOK Guide): views and concepts for D&I.....	52
3.11	Performance systems requirements.....	53
3.11.1	ECSS: views and concepts for performance.....	54
3.11.2	IEEE: views and concepts for performance.....	55
3.12	Security systems requirements.....	57
3.12.1	ECSS: views and concepts for security	57
3.13	Safety systems requirements.....	59
3.13.1	ECSS: views and concepts for safety.....	59
3.13.2	ISO: views and concepts for safety.....	61
3.13.3	IEEE: views and concepts for safety	61
3.14	Resources systems requirements.....	62
3.14.1	ECSS: views and concepts for resources	63
3.14.2	IEEE: views and concepts for resources	64
3.14.3	ISO: views and concepts for resources	64
3.15	Human factors system requirements.....	65
3.15.1	ECSS: views and concepts for human factors	66
3.16	Discussion and observation.....	68
3.16.1	ECSS standards.....	68
3.16.2	IEEE standards.....	69
3.16.3	ISO 9126 standards.....	69
3.16.4	ISO 19759 (SWEBOK guide).....	69
3.17	Summary.....	70
CHAPTER 4 RELIABILITY: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.....		71

4.1 Introduction.....71

4.2 A standard-based model of software-FUR for system reliability NFR73

 4.2.1 Mapping reliability views and vocabulary from standards.....73

 4.2.2 Identification of the system reliability functional types allocated to software-FUR.....74

 4.2.3 Model of the functions types relationships based on system views.....83

 4.2.4 Model of the functional types relationships based on COSMIC views.....83

4.3 A standard-based model of software-FUR for system reliability NFR using an COSMIC-SOA.....87

4.4 Sizing of the standard-based model of software-FUR for system reliability NFR.....89

 4.4.1 Measurement of exchange messages for system reliability89

 4.4.2 Measurement of intermediary services for system reliability91

 4.4.3 Measurement of the direct and indirect data movements for system reliability.....92

4.5 A measurement example.....94

4.6 Summary95

CHAPTER 5 MAINTAINABILITY: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.....97

5.1 Introduction.....97

5.2 A standard-based model of software-FUR for system maintainability-NFR99

 5.2.1 Mapping maintainability views, concepts, and terms from standards.....99

 5.2.2 Identification of system maintainability functions types allocated software-FUR.....100

 5.2.3 Model of the functions types relationships based on system views.....113

5.3 A standard-based model of software-FUR for system maintainability using SOA...117

 5.3.1 Measurements of exchange messages for system maintainability.....117

 5.3.2 Measurement of intermediary services for system maintainability124

 5.3.3 Measurements of data movements between Functional processes.....136

 5.3.4 Indirect data movements for all function types.....138

5.4 Sizing of the standard-based model for system maintainability-NFR.....140

5.5 A measurement example.....142

 5.5.1 Measurement of the exchange messages143

 5.5.2 Measurement of the intermediary services143

 5.5.3 Measurement of data movements (Function Level)144

5.6 Summary145

CHAPTER 6 INTERFACES: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.....147

6.1 Introduction.....147

6.2 A standard-based model of software-FUR for system interfaces NFR148

6.2.1	Mapping system interface views and concepts and terms from standards ..	149
6.2.2	Interface functions to be specified	150
6.2.3	Identification of the system interface function types allocated to software-FUR.....	150
6.3	A standard-based model of software-FUR for system interfaces NFR using SOA...	156
6.4	Sizing of the standard-based model of software-FUR for system interfaces NFR....	157
6.4.1	Measurement of exchange messages for system interface	158
6.4.2	Measurement of intermediary services for system interface	159
6.4.3	Measurement of the direct and indirect data movements for system interface	160
6.5	A Measurement Example.....	161
6.5.1	Measurement of exchange messages	162
6.5.2	Measurement of intermediary services	163
6.5.3	Measurement of data movements	163
6.6	Summary	164
CHAPTER 7	THE OTHER ELEVEN TYPES OF SYSTEM-NFR IN ECSS: SPECIFICATION AND MEASUREMENT MODELS	166
7.1	Introduction.....	166
7.2	Portability system requirements.....	166
7.2.1	Mapping views and concepts for portability from ECSS, ISO, and IEEE ..	167
7.2.2	Software portability functions to be specified	168
7.2.3	Identification of the function types in software portability	169
7.2.4	A standard-based model of software-FUR for system portability using SOA.....	169
7.3	Operations system requirements.....	171
7.3.1	Mapping system operations views and concepts from ECSS and IEEE standards	171
7.3.2	A standard-based model of software-FUR for system operation-NFR using SOA.....	173
7.4	Configuration System requirements	174
7.4.1	Mapping system configuration views and concepts from ECCS standards	174
7.4.2	Configuration function types and functions to be specified	175
7.4.3	A standard-based model of software-FUR for system configuration-NFR using an SOA	176
7.5	Data definitions and database system requirements	177
7.5.1	Mapping data definition views and concepts from ECCS standards.....	177
7.5.2	A standard-based model of software-FUR for system data definition and database -NFR using an SOA	178
7.6	Adaptation and installation system requirements	180
7.6.1	Mapping the adaptation and installation views and concepts from standards.....	180
7.6.2	Software adaptation and installation functions and function types to be specified	181

7.6.3	A standard-based model of software-FUR for system adaptation and installation-NFR using an SOA	182
7.7	Design and implementation (D&I) constraints system requirements	183
7.7.1	D&I constraints requirements and functions to be specified	183
7.7.2	Model of function types relationships.....	184
7.7.3	A model of D&I constraints services.....	185
7.7.4	A model of D&I constraints of data movements in Software-FUR view....	186
7.8	Performance system requirements	186
7.8.1	Mapping views and concepts for performance from ECSS and IEEE standards	187
7.8.2	Software system performance functions to be specified.....	187
7.8.3	Identification of the function types in the performance system requirements.....	188
7.8.4	A standard-based model of software-FUR for system performance-NFR using an SOA	189
7.9	Security system requirements	191
7.9.1	Mapping views and concepts for security from ECSS, ISO, and IEEE standards	191
7.9.2	Software system security functions to be specified	191
7.9.3	Identification of the function types in the security	192
7.9.4	A standard-based model of software-FUR for system security-NFR using an SOA.....	193
7.10	Safety system requirements	195
7.10.1	Mapping views and concepts for safety from ECSS, ISO, and IEEE standards	195
7.10.2	Software system safety functions to be specified	195
7.10.3	Identification of the function types in the software safety systems requirements.....	196
7.10.4	A standard-based model of software-FUR for system safety-NFR using an SOA.....	197
7.11	Resources system requirements	198
7.11.1	Mapping views and concepts for resources from ECSS, ISO, and IEEE standards	198
7.11.2	Software system resources functions to be specified.....	198
7.11.3	Identification of the function types in the resources systems requirements	199
7.11.4	A standard-based model of software-FUR for system resources-NFR using an SOA.....	200
7.12	Human factors system requirements.....	202
7.12.1	Software system human factors functions to be specified.....	202
7.12.2	Identification of the function types in the human factors	202
7.12.3	A standard-based model of software-FUR for human factors-NFR using an SOA.....	203
7.13	Summary.....	205
CHAPTER 8	A CASE STUDY USING THE STANDARD-BASED MODEL OF SOFTWARE-FUR FOR SYSTEM RELIABILITY-NFR	206

8.1 Introduction.....206

8.2 The Valve Control System (VCS) Case Study206

8.3 Step 1: Addition of reliability requirements at the system level.....208

8.4 Step 2: Allocate system reliability-FUR to software functions to be added to VCS.209

8.5 Step 3: The specification of the ECSS-based reliability functions allocated to software for the VCS components.....211

8.6 Step 4: Measurement of the software-FUR for the system reliability-NFR.....212

 8.6.1 Measurement strategy phase.....212

 8.6.2 COSMIC mapping phase.....212

 8.6.3 COSMIC measurement phase.....213

8.7 Summary.....214

CHAPTER 9 TRACEABILITY MODEL AND OPERATIONS PROCEDURES.....215

9.1 Introduction.....215

9.2 System Requirement traceability matrix (RTM) in ECSS standards.....215

9.3 System functional & NFR traceability matrix218

9.4 Tractability of standard-based models of software-FUR for system-NFR.....220

9.5 Tractability to the standard-based model of software-FUR for system reliability-NFR in ECSS221

9.6 Summary.....223

CONCLUSION.....225

ANNEX I THE NFR TERMS, CONCEPTS AND VOCABULARY AS DEFINED IN THE LITERATURE.

ANNEX II THE DETAILS OF 11 STANDARD-BASED MODELS OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.

ANNEX III THE DETAILED TRACEABILITY TO THE ECSS STANDARDS SERIES OF THE STANDARD-BASED MODELS OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.

ANNEX IV THE PUBLISHED WORKS IN INTERNATIONAL CONFERENCES FOR SEVEN (7) STANDARD-BASED MODELS OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR.

BIBLIOGRAPHY.....231

LIST OF TABLES

		Page
Table 1.1	List of the NFR in ECSS Standards.....	12
Table 1.2	List of the NFR in IEEE-Std 830 - 1998.....	13
Table 1.3	List of quality characteristics in ISO 9126	13
Table 1.4	COSMIC guideline offers three types of data movement's architecture...19	
Table 3.1	Reliability views, concepts, and terms in ECSS and ISO.....	33
Table 3.2	Maintainability: views concepts and terms in the ECSS and ISO	35
Table 3.3	Interface: views, concepts, and terms in ECSS and IEEE	38
Table 3.4	Portability: views, concepts, and terms in the standards	40
Table 3.5	Operations: view, concepts, and terms in the ECSS standards.....	43
Table 3.6	Configuration: views, concepts, and terms in ECSS and ISO 19759	44
Table 3.7	Data definitions and database: views, concepts, and terms in ECSS.....	47
Table 3.8	Adaptation and installation: views, concepts, and terms in the standards.49	
Table 3.9	D&I constraints: views, concepts, and terms in ECSS and ISO	53
Table 3.10	Performance: views, concepts, and terms in ECSS	56
Table 3.11	Security: views, concepts, and terms in standards.....	58
Table 3.12	Safety: views, concepts, and terms in the standards	62
Table 3.13	Resources: views, concepts, and terms in standards.....	64
Table 3.14	Human factors: views, concepts, and terms in ECSS	67
Table 4.1	Reliability: functions in ECSS, IEEE & ISO 9126.....	73
Table 4.2	System reliability functions types and related software functions.....	74
Table 4.3	Measurement of the exchange messages for the proposed model	90
Table 4.4	Measurement example for the interactions between one application functional process and one service functional process	90
Table 4.5	Measurement of the intermediary services for the proposed model	91
Table 4.6	COSMIC-SOA measurement example for the intermediary services between functional Services	92
Table 4.7	Measurements results for direct and indirect data groups	93

Table 4.8	Measurements results for direct and indirect data movements.....	95
Table 5.1	Maintainability requirements in ECSS & ISO 9126.....	99
Table 5.2	System maintainability requirements and related software functions	100
Table 5.3	Measurement of the exchange messages of the application, sub application, and services for SMFP	118
Table 5.4	Measurement of the exchange messages of the application, sub application, and services for SRFP	120
Table 5.5	Measurement of the exchange messages of the application, sub application, and services for SMP	121
Table 5.6	Measurement of the exchange messages of the application, sub application, and services for SSP	122
Table 5.7	Measurement of the exchange messages of the application, sub application, and services STP	124
Table 5.8	Measurement of the intermediary services for SMFP	126
Table 5.9	Measurement of the intermediary services for SRFP	128
Table 5.10	Measurement of the intermediary services for SMP.....	129
Table 5.11	Measurement of the intermediary services for SSP.....	130
Table 5.12	Measurement of the intermediary services for SSP and STP	132
Table 5.13	Measurement of the intermediary services for SMFP, SRFP, SMP and STP.....	135
Table 5.14	Measurement of the direct data movements for SMFP	136
Table 5.15	COSMIC-SOA measurement of the direct data movements for SRFP ...	137
Table 5.16	COSMIC-SOA measurement of the direct data movements for SMP	138
Table 5.17	Measurement of the indirect data movements for the model.....	139
Table 5.18	Measurement of the maintainability model (Function Level)	141
Table 5.19	Measurement of the maintainability model (Service level).....	141
Table 5.1	Interface requirements in ECSS and IEEE	149
Table 6.2	System interface functions that may be allocated to software-FUR.....	150
Table 6.3	COSMIC-SOA measurement example for the interactions between a functional process and its own functional service process	158

Table 6.4 Measurement for exchange messages.....159

Table 6.5 COSMIC-SOA measurement example for the intermediary service.....160

Table 6.6 COSMIC-SOA measurement for intermediary services.....160

Table 6.7 Measurements of direct and indirect data groups for system interfaces..161

Table 6.8 Measurement results for the interactions of three functional processes ..162

Table 6.9 Measurement results of intermediary services.....163

Table 6.10 Measurements of direct and indirect data movements for system interfaces.163

Table 7.1 Portability requirements in ECSS, ISO, and IEEE167

Table 7.2 Portability types by environment.....168

Table 7.3 Portability functions that may be allocated to software.....168

Table 7.4 Function types for portability that may be allocated to software.....169

Table 7.5 System operations FUR in the ECSS standards series.....171

Table 7.6 Software-FUR for system configuration NFR.....174

Table 7.7 Configuration functions that may be allocated to software175

Table 7.8 Functions to address system data definition and database requirements.177

Table 7.9 System data definition requirements and related software functions178

Table 7.10 Adaptation and installation in ECSS & ISO 9126.....180

Table 7.11 System adaptation and installation requirements related software.....181

Table 7.12 System adaptation and installation functions and functions types.....181

Table 7.13 Software D&I functions to be specified.....183

Table 7.14 Performance requirements in ECSS and IEEE187

Table 7.15 System performance functions that may be allocated to software.....188

Table 7.16 Function types for performance functions that may be allocated to software.....189

Table 7.17 Security requirements in ECSS, ISO, and IEEE.....191

Table 7.18 System security functions that may be allocated to software192

Table 7.19 Function types for security functions that may be allocated to software.193

Table 7.20 Safety requirements in ECSS, ISO, and IEEE195

Table 7.21 System safety functions that may be allocated to software196

Table 7.22	Function types for safety functions that may be allocated to software.....	196
Table 7.23	Resources requirements in ECSS, ISO, and IEEE.....	198
Table 7.24	System resources functions that may be allocated to software.....	199
Table 7.25	Function types for the resources functions that may be allocated to software.....	199
Table 7.26	Function types for the resources functions that may be allocated to software.....	200
Table 7.27	Human factors functions that may be allocated to software.....	202
Table 7.28	Function types for human factors functions that may be allocated to software.....	203
Table 8.1	Alignment of system reliability requirements with the standards-based.....	208
Table 8.2	Allocation of Reliability-FUR to the VCS Components	211
Table 8.3	The measurement details for the system reliability requirements allocated to software functions.....	213
Table 9.1	Traceability of the standard-based models of software-FUR for system-NFR.....	221
Table 9.2	Traceability results from ECSS for the standard-based model of software-FUR for system reliability-NFR from ECSS	222
Table 9.3	Traceability results from ECSS for the standard-based model of software-FUR for system reliability-NFR from ECSS	223

LISTE OF FIGURES

		Page
Figure 1.1	Mapping system-FUR and NFR to software-FUR	7
Figure 1.2	Generic flow of data groups for a functional perspective in COSMIC – ISO 19761	16
Figure 2.1	Research methodology – overview of phases	29
Figure 4.1	Mapping system requirements into software-FUR for reliability	72
Figure 4.2	System Reliability Prediction (SRP): system modelling view	75
Figure 4.3	System Reliability Prediction (SRP): COSMIC modelling view.	76
Figure 4.4	System Reliability Prediction Failures (SRPF): system modelling view ..	77
Figure 4.5	System Reliability Prediction Failures (SRPF): COSMIC modelling	78
Figure 4.6	System Reliability Prediction Faults (SRPF1): system modelling view. ..	79
Figure 4.7	System Reliability Prediction Faults (SRPF1): COSMIC modelling	80
Figure 4.8	System Reliability Prediction Errors (SRPE): system modelling view	81
Figure 4.9	System Reliability Prediction Errors (SRPE): COSMIC modelling view ..	82
Figure 4.10	System modelling view for system reliability requirements	85
Figure 4.11	A standard-based model of software-FUR for system reliability NFR	86
Figure 4.12	A standard-based model of software-FUR for system reliability NFR COSMIC modelling view (Function and Service Levels)	88
Figure 5.1	Mapping system-NFR to the maintainability FUR allocated to software ..	98
Figure 5.2	System Modeling View of a System Maintainability Failure Procedure (SMFP)	102
Figure 5.3	COSMIC View of a Maintainability Failures Procedure (SMFP)	103
Figure 5.4	System Modeling Maintainability of the Registered Failures Procedure (SRFP)	105
Figure 5.5	COSMIC Modeling View of the Maintainability Registered Failures Procedure (SRFP) (i.e., with COSMIC data movements)	106
Figure 5.6	System Modeling View of a System Malfunction Procedure (SMP)	107
Figure 5.7	COSMIC Modeling View of a System Malfunction Procedure (SMP) ..	108

Figure 5.8 System Modeling View of System Stability Procedure (SSP)109

Figure 5.9 COSMIC Modeling View of a System Stability Procedure (SSP).....110

Figure 5.10 System Modeling View of the System Testability Procedure (ST).....111

Figure 5.11 COSMIC Modeling View of a System Testability Procedure (ST)112

Figure 5.12 System Modeling View for System Maintainability Requirements115

Figure 5.13 Standard-based model of software-FUR for system maintainability-NFR
(Functional Level).....116

Figure 5.14 Exchange of data messages for sub applications (A, B, C, D)118

Figure 5.15 Interactions sub applications (E and F) with their services for SRFP119

Figure 5.16 Interactions sub applications (G and H) with their services for SMP.....121

Figure 5.17 Interactions sub applications (K and L) with their services for SSP122

Figure 5.18 Interactions sub applications (I and J) with their services for STP.....123

Figure 5.19 The intermediary services between sub application services (SA, SB, SC,
and SD) for SMFP125

Figure 5.20 The intermediary services for sub application (SE and SF) for SRFP127

Figure 5.21 The intermediary services between sub application for SMP.....129

Figure 5.22 The intermediary services between sub application for SSP.....130

Figure 5.23 The intermediary services between sub application services (SI and SK),
(SI and SL) and (SJ and SK), (SJ and SL) for (SSP and STP)131

Figure 5.24 The intermediary services between sub application services SK134

Figure 5.25 The intermediary services between sub application services SL.....134

Figure 5.26 Direct data movements between the application (SDF) and sub
applications (A, B, C, D) for SMFP.....136

Figure 5.27 Direct data movements between sub applications (A, B, C, D) and sub
applications (E and F) for SRFP137

Figure 5.28 Direct data movements between sub applications (E and F) and sub
applications (G and H) for SMP138

Figure 5.29 Indirect data movements between all sub applications in all functional
types139

Figure 6.1 Mapping system requirements to software-FUR for an interface.....148

Figure 6.2	System interface components (SIC): a system modeling view.....	152
Figure 6.3	System interface components (SIC): COSMIC modeling view	152
Figure 6.4	System interface specifications (SIS): a system modeling view	153
Figure 6.5	System interface specifications (SIS): COSMIC modeling view	154
Figure 6.6	System modeling view for the system interface requirements	155
Figure 6.7	A standard-based model of software-FUR for system interfaces NFR ...	156
Figure 6.8	A standard-based model of software-FUR for system interfaces NFR ...	157
Figure 7.1	Standard-based model of software-FUR for system portability-NFR	170
Figure 7.2	System operations functions and function types.....	172
Figure 7.3	A standard-based model of software-FUR for system operations-NFR..	173
Figure 7.4	A standard-based model of software-FUR for system.....	176
Figure 7.5	A standard-based model of software-FUR for system.....	179
Figure 7.6	A standard-based model of software-FUR for system adaptation and installation-NFR using an SOA	182
Figure 7.7	A standard-based model of software-FUR for system D&I constraints- NFR.....	184
Figure 7.8	A model of D&I constraints requirements allocated to software	185
Figure 7.9	Direct Data Movements	186
Figure 7.10	Indirect Data Movements.....	186
Figure 7.11	A standard-based model of software-FUR for.....	190
Figure 7.12	A standard-based model of software-FUR for system security-NFR.....	194
Figure 7.13	A standard-based model of software-FUR for system safety-NFR using an SOA.....	197
Figure 7.14	A standard-based model of software-FUR for system resources-NFR using SOA.....	201
Figure 7.15	A standard-based model of software-FUR for human factors-NFR using SOA.....	204
Figure 8.1	VCS blocks diagram with its hardware and software components.....	207
Figure 8.2	A standard-based model of FUR for system reliability-NFR	210

Figure 9.1 The requirement traceability matrix (RTM) in ECSS standards and cycle
life217

Figure 9.2 A modified requirement traceability matrix (M-RTM) in219

LIST OF ABBREVIATIONS

ACRF	Access Control Role Function
AEF	Accuracy Errors Function
AF	Activity Function
AF	Authentication Function
AF	Antivirus Function
ANSI	American National Standards Institute
ARF	Automatic Restart Function
BBRMAF	Block of Bus Relative Memory Addresses Function
BF	Bandwidth Function
BTF	Backup Type Function
CCFF	Configuration Control Flow Function
CDFE	Configuration Data Flow Function
CDFE	Correct Data Faults Function
CDTF	Control Data Transfer Function
CE	Cognitive Ergonomics
CF	Concurrency Function
CIF	Communication Interface Function
CM	Configuration Management
COSMIC	Common Software Measurement International Consortium
CPU	Central Processing Unit
CSDF	Correct System Defects Function
CTF	Complex Type Function
D&I	Design and Implementation Constraints
DDBMSF	Distributed Data Base Management System Function
ECLSS	Environment Control and Life Support Systems Design
ECSS	European Cooperation on Space Standardization
EDF	Encryption and Decryption Function
EDTF	Error Data Tolerance Function

EE	Environmental Ergonomics
EF	Event Function
EHIF	Error to Handle Input Function
EPCOF	Error to Produce Correct Output Function
EPKIF	External PKI Function
EPOF	Error to Produce Output Function
EPS	Evaluation Processing Speed
ESA	European Space Agency
ESF	Electrical Safety Function
ESF	Environmental Safety Function
FATF	Fault Allocation Time Function
FCA	Functional Configuration Audit
FDCF	Failure Data Control Function
FDf	Fault Detection Function
FDf	Failure Detection Function
FDMF	Failure Data Monitoring Function
FDOF	Failure Data Operation Function
FF	Firewall Function
FIF	Failure Isolation Function
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode, Effects and Criticality Analysis
FMF	Failure Mechanism Function
FOF	Failure Operation Function
FP	Function Point
FPDCF	Fault Prevention of Data Control Function
FPF	Fault Prevention Function
FPSF	Fault Prevention of System Function
FR	Functional Requirements
FRF	Fault Removal Function
FRTF	Fault Recovery Tolerance Function

FSM	Functional Size Measurement
FSTF	Failure System Tolerance Function
FUR	Functional User Requirements
GSC	General System Characterization
HCF	Human Capabilities Function
HIF	Hardware Interface Function
HIF	Human Interface Factors
HR	Hardware Resources
HTPF	Host-Target Platform Function
I/O	Input/Output
IBF	Independence of the Browser Function
ICD	Interface Control Document
ICF	Independence of the Client Function
ICF	Interface Customization Function
ICTPF	Interface Characteristics and Task Performance Function
IDF	Independence of the Database Function
IEEE	Institute of Electronics Engineers
IFPUG	International Function Point Users Groups
IMF	Independence of the Middleware Function
INF	Independence of the Network Function
IOPF	Inter-Operational Function
IORF	Localizing I/O Resources Function
IOSF	Independence of the Operating System Function
IOTAF	I/O Transmission Addresses Function
IPLVMF	Independence of the Programming Language Virtual Machine Function
ISC	Isolating System Calls
ISF	Interface Specification Function
ISF	Independence of the Server Function
ISLF	Interface Specification Link Function
ISO	International Organization for Standardization

ISRCF	Identification of Safety Related Controls Function
ISSCF	Isolating Software System Calls Function
ISTF	Independence of the Storage Function
IVF	Interrupt Vectors Function
KA	Knowledge Area
MCSIF	Memory Capacity for Software Item Function
MMTF	Main Memory Time Function
MRF	Memory Resources Function
MSF	Mechanical Safety Function
NFR	Non Functional Requirements
OO	Object oriented
OPCIF	Operational Control Interface Function
OPDIF	Operational Data Interface Function
OPEF	Operational Environment Function
OPF	Operational Functions
OPFE	Operational Function Event
OSF	Operational Safety Function
PCA	Physical Configuration Audit
PCSIF	Processor Capacity for Software Item Function
PF	Parameter Function
PIEF	Processor Instruction Execution Function
PSF	Personal Selection Function
PSF	Psychology and Physiological Safety Function
RC	Resource Consumption
RDF	Reporting Data Function
RDF	Redundant Data Function
RDTF	Register Data Transfer Function
RDTF	Registered Data Transfer Function
RPNF	Redundant Power and Network Function
RRS	Response to Reference Signals

RTF	Response Time Function
RVF	Record Value Function
SA	System Availability
SC	System Confidentiality
SCDF	Configuration Data Function
SCDF1	Control Data Function
SCRf	Software Configuration Risk Function
SDC	System Data Components
SDCSIF	Storage Device Capacity for Software Item Function
SDD	Software Design Document
SDF	System Diagnostic Functions
SDI	System Data Items
SDRF	Software Design Risk Function
SDSF	Software Data Structure Function
SDT	System Data Types
SDTF	Set Data Transfer with System Resources Function
SDTF	Storage Device Time Function
SEF	System Element Function
SEF	Stability Errors Function
SEF	Software Elements Function
SET	System Entity Types
SF	Staffing Function
SFDF	System Failure Detection Function
SFIF	System Failure Isolation Function
SFTF	System Failure Tasks Function
SHC	System Hardware Components
SHE	System Hardware Environment
SI	Security Integrity
SIC	System Interface Components
SIE	System Integrated Environment

SIF	Software Interface Function
SIOR	System I/O Resources
SIS	System Interface Specifications
SLF	Security Login Function
SLOF	System Loss Operation Function
SMDF	Monitoring Data Function
SMFP	System Maintainability Failure Procedure
SMP	System Malfunction Procedure
SOA	Service Oriented Architecture
SOPC	System Operational Control
SOPD	System Operational Data
SPC	System Program Calls
SPDS	System Product Data Schema
SR	Software Resources
SRF	Storage Resources Function
SRFP	System Registered Failures Procedure
SROF	Software Operation Risk Function
SRP	System Reliability Prediction
SRPE	System Reliability Prediction Error
SRPF	System Reliability Prediction Failure
SRPF1	System Reliability Prediction Fault
SRSF	System Redundancy Status Function
SSAF	System Safety Audit Function
SSC	System Software Components
SSE	System Software Environment
SSF	System Scalability Function
SSM	System Safety Mechanism
SSP	System Stability Procedure
SSR	System Safety Risk
SSRI	Safety Switching of Redundant Information

STF	System Time Function
STF	Simple Type Function
STF	Settling Time Function
STOSF	Specific Real Time Operating System Function
STP	System Testability Procedure
SVF	Simple Value Function
SVT	System Value Types
SWEBOK	Software Engineering Body of Knowledge
TECPF	Tracking Error for Command Profiles Function
TF	Training Function
TRF	Transmission Resources Function
TT	Throughput Time
UIF	User Interface Function
UML	Unified Modeling Language
VCS	Valve Control System
VCS	Valve Control System
WF	Workload Function

INTRODUCTION

The terminology adopted in this thesis is closely aligned with the system and software engineering terminology adopted jointly by the ISO and IEEE standards organizations, and in particular in (ISO 15288 2008, ISO 12207 2008 and ISO 15979 2002).

Non functional requirements (NFR) play a critical role in system development. They may have a considerable impact on project effort (Chung and do Prado Leite 2009) and should be taken into account for estimation purposes and for comparing project productivity. In current practice, NFR may be viewed, defined, interpreted, and evaluated differently by different people in the later phases of the project (Chung, Nixon et al. 2000), particularly when they are stated vaguely and only briefly at the system requirements phase.

NFR have received less attention in the software engineering literature and are definitely less well understood than other cost factors (Mylopoulos, Chung et al. 1992). Furthermore, measurement is essential if NFR are to be taken as quantitative inputs to an estimation or productivity benchmarking process. However, not much work has been published to date on how to measure them.

In practice, requirements are initially typically addressed at the system level (Abran and Al-Sarayreh 2010a; Al-Sarayreh and Abran 2010b), either as high-level system functional user requirements (system-FUR) or as high-level system non-functional requirements (system-NFR). The latter must usually be detailed, allocated, and implemented in hardware, software as software FUR (software-FUR) or in a specific combination of hardware and software. To distinguish between these types of requirements, system-FUR describe the required functions in a system, while system-NFR describe how the required functions must behave in a system.

In the ECSS (European Cooperation on Space Standardization) standards for the aerospace industry (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009), ISO 9126 (ISO-9126 2004) and the IEEE 830 (IEEE-830 1998) standards, there are a number of

concepts provided to describe various types of candidate system-NFR at the system, software, and hardware levels. However, these standards vary in their views, terminology, and coverage of such system requirements.

Problem statement

In the system requirements phase, the focus is often on detailing and documenting the system functional requirements while their allocation to the software and hardware parts of the system being designed is being done in the system architecture phase. The NFR, in contrast, are often captured only generically at a fairly high level and they do not include the levels of details necessary for the system engineers to allocate them yet as specific functionalities to be handled either by the software or the hardware, or a combination thereof.

The European standards for the aerospace industry (ECSS series), include sixteen (16) types of NFR for embedded and real-time software as follows:

1. Reliability systems requirements;
2. Maintainability systems requirements;
3. Interfaces systems requirements;
4. Portability systems requirements;
5. Operations systems requirements;
6. Configuration systems requirements;
7. Data definitions and database systems requirements;
8. Adaptations and installations systems requirements;
9. Design and implementation constraints systems requirements;
10. Performance systems requirements;
11. Security and privacy systems requirements;
12. Safety systems requirements;
13. Resources systems requirements;
14. Human factor requirements;
15. Quality systems Requirements;
16. Other NFR requirements.

A number of concepts are provided in the ECSS and other standards to describe the various types of candidate NFR at the system, software, and hardware levels.

Prior to the research work reported in this thesis, there were no standard-based models for the identification and specification of software-FUR for implementing system-NFR based on the various views documented in international standards and in the literature. Consequently, it was challenging to specify and to measure these system-NFR related software-FURS, and to take them into account quantitatively for estimation purposes in software development projects.

This document reports on the research work carried out to develop fourteen standard-based models for system-NFR which could be allocated to software-FUR. The availability of these standard-based models aims to facilitate the early identification and specification of the system-NFR and their detailed allocation as specific functions to be handled by the specified allocation to hardware or software or a specific combination of the two. In the absence of such standard-based and detailed models, such NFR are typically handled in practice much later on the software development life cycle when, at system testing time, users and developers discover that NFR have been overlooked and additional effort must be expended to implement them.

The approach adopted in this research for the structure of these standard-based models is to use the generic model of software functional requirements proposed in the COSMIC (ISO-9761 2011) model, thereby allowing as well to measure the functional size of such system-NFR requirements allocated to software and, next, to take them into account for estimation purposes.

Thesis organization

This thesis contains nine chapters and four Annexes. The current introduction outlines the problem statement and the organization of the thesis.

Chapter 1 presents an overview of the non-functional requirements (NFR), as reported in the literature and in standards.

Chapter 2 presents the research project definition, including the research motivation, goal, objectives and users of the research results. Chapter 2 also presents the detailed methodology designed to tackle the research objectives, including the research phases and the research inputs.

Chapter 3 presents a survey of the system-NFR views, concepts, and terms in the ECSS, ISO and IEEE standards. It identifies which standards currently address aspects of the software-FUR derived from system-FUR and system-NFR. The outcome of Chapter 3 is the identification of the various elements that should be included in the design of standard-based model of software-FUR for each type of system-NFR.

Chapter 4 presents a standard-based model for the functions needed to address the system's reliability requirements. This chapter proposes the standard-based model of software-FUR for system reliability. This standard-based model can be considered as a kind of reference model for the identification of system reliability requirements and can be used for their allocation to software functions implementing such requirements.

Chapter 5 presents a standard-based model for specifying and measuring software requirements for the functions needed to address the system's maintainability requirements. This chapter proposes the standard-based model of software-FUR for system maintainability. This standard-based model can be considered as a kind of reference model for the identification of system maintainability requirements, and can be used for their allocation to software functions implementing such requirements.

Chapter 6 presents the system interface concepts dispersed in multiple standards and integrates them into a standard-based model of software-FUR for system interface-NFR. The availability of this standard-based model can facilitate the early identification and specification of the system interface-NFR and their detailed allocation as specific system

interface functions to be handled by hardware or software, or to a specific combination of the two.

Chapter 7 presents the other eleven types of system-NFR in the ECSS: specification and measurement models. The structures of the standard-based models are based on the generic model of software adopted by the COSMIC measurement standard: the necessary information for measuring their functional size is then readily available. Specifically, the standard-based models of system-NFR presented in this chapter are based on: the ECSS standards for the description of the NFR for system and the COSMIC measurement model of functional requirements.

Chapter 8 presents a case study using a standard-based model of software-FUR for system reliability-NFR. This chapter uses a valve control system (VCS) as a case study to illustrate the use of the standard-based model of software-FUR for system reliability-NFR. The selected case study aims at the identification and classification, then measurement, of the software-FUR for system reliability-NFR.

Chapter 9 presents the system requirements traceability matrix (RTM) in ECSS standards with the system life cycle for hardware and software. A modified RTM is proposed in this chapter by using the same ECSS traceability approach with some additional changes to tackle system-FUR and system-NFR. This chapter presents also a summary of the number of the traceability concepts and terms for the proposed fourteen standard-based models, as well as the detailed traceability to specific sections of the ECSS standards for the proposed reliability-NFR model.

The Conclusion chapter summarizes the results of this thesis, the contributions, and the expected impacts for the industry as well as suggestions for future work.

Annex I presents the NFR terms, concepts and vocabulary as defined in the literature.

Annex II (from Annex II-A to Annex IIK) presents the details of 11 standard-based models of software-FUR f derived from system-NFR.

Annex III presents the detailed traceability to the ECSS standards series of the standard-based models of software-FUR derived from system-NFR.

Annex IV (from Annex IV-A to Annex IV-G) presents the published works in international conferences for seven (7) standard-based models of software-FUR derived from system-NFR.

CHAPTER 1

LITERATURE REVIEW

1.1 Introduction

The non-functional requirements (NFR), are often only generically captured at a fairly high level and they do not yet include the degree of detail that is necessary for the system engineers to allocate such NFR as specific functionalities to be handled either by the software or the hardware, or a specific combination of both.

In the literature, there are many published studies tackling the NFR. Some of these studies present methods for the identification and classification of the NFR. as well as proposed frameworks and NFR ontologies.

In practice, requirements are initially typically addressed at the system level (Abran, Al-Sarayreh et al. 2010a), (Al-Sarayreh, Abran et al. 2010b) and (Karl 2003), either as high level system functional user requirements (system-FUR) or as high level system non functional requirements (system-NFR). The latter must usually be detailed, allocated, and implemented in either hardware or software, or both, as software-FUR for example – see Figure 1.1.

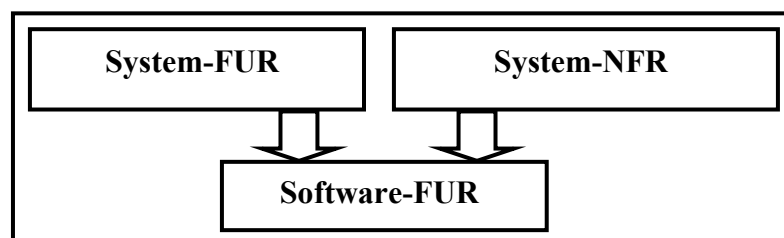


Figure 1.1 Mapping system-FUR and NFR to software-FUR

To distinguish between these types of requirements, system-FUR describes the required functions in a system, while system-NFR describes how the required functions must behave in a system. In the software requirements engineering step, system-NFR can then be detailed

and specified as software-FUR, to allow a software engineer to develop, test, and configure the final deliverables to system users.

The term "functional" refers to the set of functions the system (including the software) must offer, while the term "non functional" refers to the manner in which such functions perform. An FUR is typically phrased with a subject and a predicate (i.e., noun/verb), such as: "The system must print 5 reports". NFR, by contrast, are typically phrased with an adverb or modifying clause, such as: "The system will print 5 reports quickly", or "The system will print 5 reports with a high degree of reliability".

Currently, the European Cooperation on Space Standardization (ECSS) and the Institute of Electrical and Electronics Engineers (IEEE) propose two NFR lists with some primitive concepts and vocabularies; similarly, ISO proposes in the ISO 9126 standard, a quality model of software products.

This chapter presents a survey of the literature consulted for this research work and is organized as follows:

- Section 1.2 surveys the NFR in the academic literature;
- Section 1.3 surveys the NFR in international standards;
- Section 1.4 surveys the NFR ontologies.
- A summary is presented in section 1.5.

1.2 NFR in the academic literature

In the literature on systems/software engineering, there are a number of published works on NFR. Some of the early works on NFR, such as that of (Chung 1993), presents the initial attempts to capture knowledge in this domain. Chung's work was followed by that of (Mylopoulos, Chung *et al.* 1999) who suggested viewing all requirements as goals, each goal being an umbrella for related requirements, both functional and NFR.

Chung (Chung 1993) and Andrew (Andrew 2000) aimed to make NFR more quantitative in nature, while Andrew (Andrew 2000) found that there are often gaps between the stakeholder vision and requirements representation.

(Chung, Nixon *et al.* 2000) proposed a taxonomy for NFR, indicating that it is unrealistic to expect designers and developers to incorporate an entity that they cannot readily identify. While taxonomies aim to be inclusive of the entire set of entities in question, these authors suggested (Chung, Nixon *et al.* 2000) that a one-level or two-level taxonomy would suffice initially and that there are over 161 identifiable types of NFR.

(Moreira, Araujo *et al.* 2002), (Rosa, Cunha *et al.* 2002), (Park and Kang 2004), and (Glinz 2005) have proposed new methods for classifying NFR early in the software development process, while (Kaiya, Osada *et al.* 2004) have presented a method for identifying stakeholders and their NFR preferences by means of case diagrams of existing systems.

(Paech, Dutoit *et al.* 2002) recommended that functional requirements (FR), NFR, and architecture be tightly co-developed and addressed in a coherent and integrated manner, suggesting that NFR be decomposable into more refined NFR and additional FR, as well as architectural decisions.

(Cysneiros and Leite 2004) presented a process to elicit NFR, analyze their interdependencies, and trace those to functional conceptual models using UML by integrating the NFR into class, sequence, and collaboration diagrams. This process also shows how certain cases and scenarios can be adapted to deal with NFR.

More recently, (Mylopoulos 2006) promoted goal-oriented requirements engineering and suggests a specific solution involving the establishment of an agent-oriented software development method. Called the TROPOS project, this software method covers not only the requirements but also the design phases and addresses the design of high-variability software for systems such as home care and business process design.

(Galster and Bucherer 2008) have proposed a taxonomy for NFR in a service-oriented context. Their taxonomy implements three main categories of NFR: process requirements, NFR external requirements, and NFR service requirements. The taxonomy can be applied with individual services as well as with a service-based system as a whole. This taxonomy is considered as a starting point and checklist when handling NFR issues in service-oriented and particularly highly-distributed environments.

(Bharadwaj and Nair 2009) presented an approach for some of the NFR used in the FPA method by the International Function Point Users Group (IFPUG) to determine the degree of influence for each of them. The study shows that NFR affect the FP value while attempting to capture the actual applicable attributes of the fourteen GSCs for a given application. This is further complicated because the influence of NFR on the project size is also difficult to quantify. Furthermore, several different scale types are used in the various steps and the results of many of the steps and sub-steps of the measurement designs of the GSCs are based on inappropriate use of mathematical properties of corresponding scale types (Abran, 2010).

More recently, (Kassab, Daneva et al. 2009) proposed some solutions for building a NFR framework. For example, (Kassab, Daneva et al. 2009) suggested adopting a sequence of systematic activities with the aim of identifying, specifying, and separating FR from NFR, as well as a discussion on NFR prioritization and risk assessment. They also reported (Kassab, Ormandjieva et al. 2008) an initial solution using the COSMIC method for determining the functional size of NFR based on "soft goal" concepts, to deal with the problem of quantitatively assessing the proposed NFR framework early in a project.

Recently, (Casamayor, Godoy et al. 2010) proposed a method based on a semi-supervised learning techniques for automatic identification and classification of NFR. The method is based on a reduced number of categorized requirements by taking advantage of the knowledge provided by uncategorized ones, as well as certain properties of text. The learning method also exploits feedback from users to enhance classification performance.

Moreover, (Supakkul, Hill et al. 2010) presented a pattern-based approach composed of four kinds of NFR patterns for capturing and reusing knowledge of NFR patterns, problem patterns, alternatives patterns, and selection patterns. According to these authors, the NFR patterns may be visually represented and organized by rules specialization to create more specific patterns, composition to build larger patterns, and of instantiation to create new patterns using existing patterns as templates. This NFR pattern approach is based on the TJX incident, one of the largest credit card thefts in history, as a case study.

More recently, (Bendjenna, Charrel et al. 2010) proposed a process to identify the NFR to model them using a fuzzy cognitive map. According to these authors using a fuzzy cognitive map to model NFR allows moving from the conventional modelling to computer-based modelling.

Finally, (Yakkali and Subramanian 2010) proposed an approach based on minimum spanning trees and the NFR Framework for Control and Data Acquisition (CADA) systems used to monitor and control critical infrastructures ranging from computer networks to manufacturing.

1.3 NFR in international standards

The software industry has been working on the description of NFR, in particular through international standardization bodies, such as the European Cooperation on Space Standardization (ECSS), the Institute of Electrical and Electronics Engineers (IEEE), and the International Organization for Standardization (ISO).

1.3.1 European international standards (ECSS)

The European Cooperation for Space Standardization (ECSS) is an organization that works to improve standardization within the European space sector. The ECSS frequently publishes standards targeted to the contractors working for the European Space Agency (ESA) (ECSS-ESA 2005). The ECSS standards series includes a number of NFR at the system level.

More specifically, (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) for the aerospace industry includes sixteen (16) types of the NFR for embedded and real-time software – see Table 1.1. A number of NFR-related concepts are dispersed throughout the ECSS standards to describe, at varying levels of details, the various types of NFRs at the system, software, and hardware levels.

Table 1.1 List of the 16 NFR types in ECSS Standards

ID	Types of Non Functional Requirement
1	Reliability requirements
2	Maintainability requirements
3	Interface requirements
4	Portability requirements
5	Operations requirements
6	Software Configuration & Delivery Requirements
7	Data Definitions & Database Requirements
8	Adaptation & Installation Requirements
9	Design & Implementation Constraints requirements
10	Performance requirements
11	Security & Privacy requirements
12	Safety requirements
13	Resources requirements
14	Human Factors requirements
15	Quality requirements
16	Other Requirements

1.3.2 IEEE 830 standard

The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute (ANSI). IEEE 830 “The recommended practice for software requirements specifications standard” (IEEE-830 1998) was developed within the IEEE societies and the standards coordinating committees of the IEEE standards association (IEEE-SA) standards board.

More specifically, (IEEE-830 1998) identifies thirteen types of NFR to be included in a software requirements document - see Table 1.2. A number of NFR-related concepts are dispersed throughout the IEEE standards to describe, at varying levels of detail, the various types of candidate NFR at the system, software, and hardware levels.

Table 1.2 List of the NFR types in IEEE-Std 830 - 1998

ID	Type of Non Functional Requirements
1	Performance requirements
2	Interface requirements
3	Operational requirements
4	Resource requirements
5	Verification requirements
6	Reliability requirements
7	Quality requirements
8	Acceptance requirements
9	Documentation requirements
10	Security requirements
11	Portability requirements
12	Maintainability requirements
13	Safety requirements

1.3.3 ISO 9126 standard series

The ISO 9126 series (ISO-9126 2004) proposes a model for the evaluation of the quality, and associated metrics, of a software product. ISO 9126-1 presents the quality model of six (6) quality characteristics and several quality sub-characteristics for internal and external quality, which are further subdivided into sub-characteristics. These sub-characteristics are manifested externally when the software is used as a part of a computer system, and are a result of internal software attributes - see Table 1.3.

Table 1.3 List of quality characteristics in ISO 9126

ID	Quality Characteristics and Sub characteristics
1	Functionality: is defined as a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs. The sub characteristics related to this quality characteristic are: suitability, accuracy, interoperability, compliance, and security

Table 1.4 List of quality characteristics in ISO 9126 (Contd)

ID	Quality Characteristics and Sub characteristics
2	Reliability: is defined as a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. The quality sub characteristics related to these factors are: maturity, recoverability, compliance and fault tolerance
3	Efficiency: is defined as a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions. The sub characteristics related to this characteristic are: time behaviour, resource behaviour and compliance.
4	Usability: is defined as a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. The sub characteristics related to this characteristic are: learnability, understandability, compliance and operability
5	Maintainability: is defined as a set of attributes that bear on the effort needed to make specified modifications. The sub characteristics related to this characteristic are: stability, analyzability, changeability, compliance and testability
6	Portability: is defined as a set of attributes that bear on the ability of software to be transferred from one environment to another. The sub characteristics related to this characteristic are: installability, replaceability, conformance and adaptability

1.3.4 ISO 19759 (SWEBOK guide)

The Guide to the Software Engineering Body of Knowledge (ISO-19759 2004) (SWEBOK Guide), written under the auspices of the IEEE Computer Society's professional practices committee, was initiated in 1998 to develop an international consensus in pursuing the following objectives:

- To characterize the content of the software engineering discipline;
- To promote a consistent view of software engineering worldwide;
- To provide access to the software engineering body of knowledge;
- To clarify the place and set the boundary of software engineering with respect to other disciplines;
- To provide a foundation for curriculum development and individual certification material.

The SWEBOK Guide (ISO-19759 2004) presents the ‘Software Requirements’ knowledge area (KA) as the first KA in the software engineering life cycle process. According to the SWEBOK Guide, the software requirements KA is concerned with the elicitation, analysis, specification, and validation of software requirements. It is widely acknowledged within the software industry that software projects are critically vulnerable when these activities are performed poorly.

In the ‘Software Requirements Fundamentals’ in the SWEBOK Guide (ISO-19759 2004) The functional requirements describe the functions that the software is to execute whereas the non functional requirements (NFR) are the ones that act to constrain the solution. They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements, or one of many other types of software requirements.

1.3.5 ISO 19761 (COSMIC method)

It is specified in ISO 14143-1(ISO-14143-1 2007) that a functional size measurement (FSM) method must measure software-FUR. In addition, (ISO-19761 2011) – COSMIC proposes a generic model of software-FUR that clarifies the boundary between hardware and software. Figure 1.2 illustrates the generic flow of data from a functional perspective from hardware to software. From this generic model of software functional requirements – see Figure 1.2, we observe the following:

- Software is bounded by hardware. In the so-called “front-end” direction (i.e., the left-hand side in Figure 1.2), software used by a human user is bounded by I/O hardware, such as a mouse, a keyboard, a printer, or a display, or by engineered devices, such as sensors or relays. In the so-called “back-end” direction (i.e., the right-hand side of Figure 1.2), software is bounded by persistent storage hardware, like a hard disk, and RAM and ROM memory;
- The software functionality is embedded within the functional flows of data groups. Four distinct types of data movements can characterize such data flows. In the “front end”

- direction, two types of movements (Entry and Exit) allow the exchange of data with the users across a ‘boundary’. In the “back end” direction, two types of movements (Read and Write) allow the exchange of data with persistent storage hardware;
- Different abstractions are typically used for different measurement purposes. In real-time software, the users are typically the engineered devices that interact directly with the software; that is, the user is ‘I/O hardware’. For business application software, the abstraction commonly assumes that the users are humans who interact directly with the business application software across the boundary, in which case the I/O hardware is ignored.

The COSMIC FSM method (ISO-19761 2011) is aimed at measuring the size of software based on identifiable FUR. Once identified, those requirements are allocated to hardware and software from the unifying perspective of a system integrating these two “components”. Since COSMIC is aimed at sizing software and only the requirements allocated to the software are currently considered in its measurement procedure.

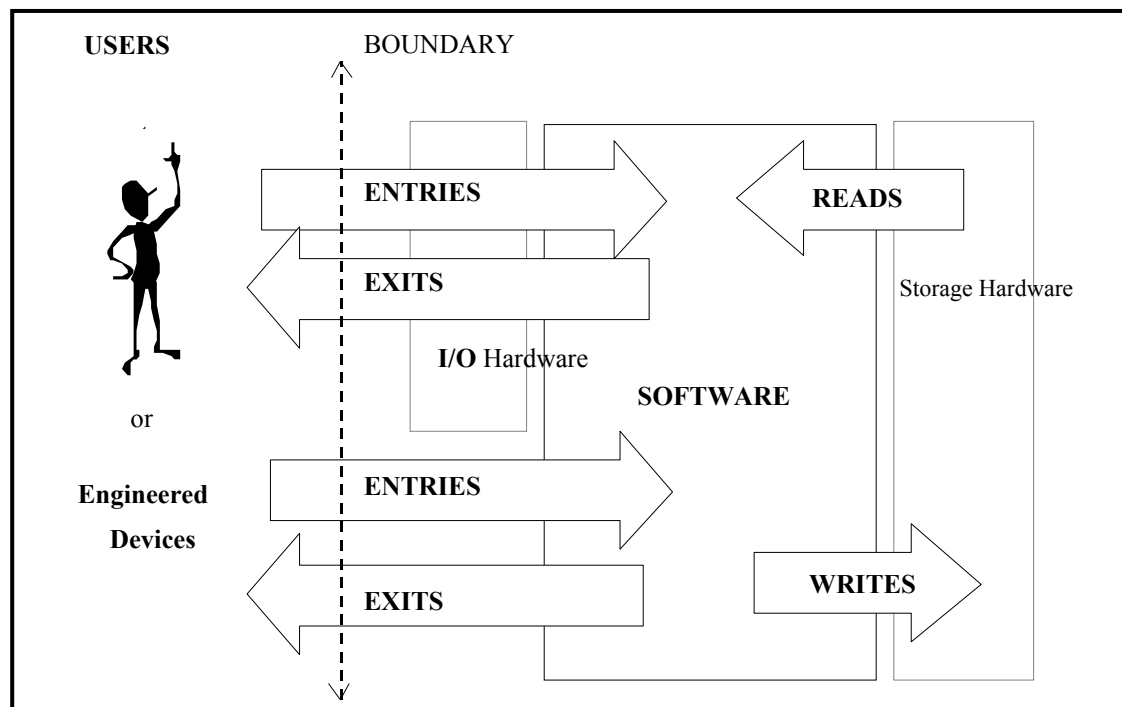


Figure 1.2 Generic flow of data groups for a functional perspective in COSMIC – ISO 19761

In this research work, the generic model of COSMIC (ISO 19761) will be used to measure the functional size of the functions of the system-NFR allocated to software-FUR, which functions are mentioned in other international standards such as ECSS, IEEE and Iso 9126. For example, the IFPUG (International Function Point User Group) sizing method has been published as an ISO Standard (ISO 20926 2003); this method attempts to capture and size the NFR through its set of the fourteen general system characteristics (GSCs) for a given application. This is further complicated in the IFPUG method structure since the influence of NFR on the project size is also difficult to quantify. Furthermore, several different scale types are used in the various steps of the IFPUG method and the results of many of the steps and sub-steps of the measurement designs of the GSCs are based on inappropriate use of mathematical properties of corresponding scale types (Abran, 2010).

1.3.6 Software Functional Size Measurement (FSM)

The functional size measurement (FSM) is used to measure software products from a user perspective. FSM must be independent of technical development and implementation decisions and it can be used to compare the productivity of different techniques and technologies (Abran 2010).

FSM has reached a high maturity level: for example, the basic concepts and definitions of FSM have been standardized by the International Organization for Standardization in (ISO-14143-1 2007) while five measurement methods have been adopted by ISO as International Standards, such as: COSMIC (ISO-19761 2011).

The COSMIC (ISO-19761 2011) standard is considered as a second generation of an FSM method. COSMIC method has been extensively tested and its use is increasing especially in the real-time and telecommunications; it is as well compatible with modern specification methods such as unified modeling languages (UML), and object-oriented (OO) techniques. COSMIC method defines the principles, rules, and a process for measuring the functional size of a piece of software. ‘Functional size’ is a measure of the ‘amount of functionality’ provided by the software.

1.3.7 COSMIC guideline for sizing service oriented architectural software

COSMIC-SOA is a supplementary guideline (COSMIC 2010) for the COSMIC standard published by the COSMIC Group in 2010. It is intended to be used by expert ‘measurers’ who have the task of measuring the functional size of software services according to the COSMIC method. In particular, the COSMIC method defines and standardizes particular concepts, such as layers, peer components, the unlimited size of a functional process, and that pieces of software can be functional users of each other; these concepts are perfectly suited for measuring SOA-based software requirements.

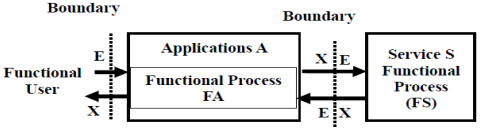
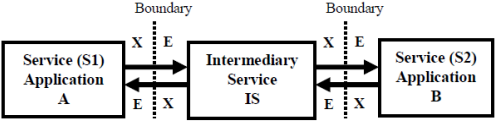
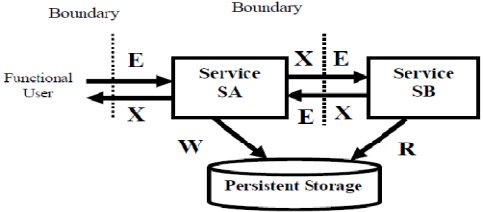
There are many definitions of a service-oriented architecture (SOA), such as: a flexible set of design principles used during systems development and integration (COSMIC 2010); a process including the definition of the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements (SoberIT: 2008).

SOA has been selected in this thesis as example of sets of more complex requirements. Of course, there could be other types of complex requirements from different viewpoints: for example, various types of architectures, and architecture design semantics in multi-tiered and distributed systems.

COSMIC Guideline (COSMIC 2010) aids measurers of services when there are SOA requirements by separating functions into distinct units, or services. These services communicate with each other by exchanging data in a well-defined, shared format, or by coordinating an activity between two or more services and aims to show how the COSMIC method can be applied to measure SOA software without needing to adapt the method in any way.

The COSMIC Guideline for SOA (COSMIC 2010) offers three types of data movements in Table 1.4.

Table 1.5 COSMIC SOA guideline - Tthree types of data movements (COSMIC 2010)

ID	COSMIC-SOA Data Movements Types	Generic Measurement Model
1	<p>COSMIC-SOA exchange messages:</p> <p>An application requiring commonly-used information from another application sends a request to the service of the application that can handle the request or the application may call upon its own services. Such calls are also called ‘messages’. Each message may consist of one or more data movements</p>	 <p>The diagram shows a 'Functional User' on the left. A vertical dashed line labeled 'Boundary' separates the user from a box containing 'Applications A' and 'Functional Process FA'. To the right of this box is another vertical dashed line labeled 'Boundary', followed by a box containing 'Service S Functional Process (FS)'. Arrows indicate data movements: an arrow labeled 'E' points from the user to the boundary, and an arrow labeled 'X' points from the boundary to the user. Similarly, an arrow labeled 'E' points from the 'Applications A' box to the second boundary, and an arrow labeled 'X' points from the second boundary to the 'Service S' box. Finally, an arrow labeled 'E' points from the 'Service S' box to the second boundary, and an arrow labeled 'X' points from the second boundary to the 'Applications A' box.</p>
2	<p>COSMIC-SOA intermediary services:</p> <p>When a functional process of an application service in application A requires data that are available via an application service in application B, the former application service calls upon a functional process of the intermediary service. This service functionality is also needed by other applications in the overall SOA framework, as it may itself be realized in the form of a utility service</p>	 <p>The diagram shows three boxes: 'Service (S1) Application A' on the left, 'Intermediary Service IS' in the middle, and 'Service (S2) Application B' on the right. Two vertical dashed lines labeled 'Boundary' separate the boxes. Between the first and second boundaries, arrows show 'E' from A to IS and 'X' from IS to A. Between the second and third boundaries, arrows show 'E' from B to IS and 'X' from IS to B.</p>
3	<p>COSMIC-SOA data exchanges:</p> <p>The data movements between components in the same layer, i.e., between peer components (where a component may be an application or a service). It shows <i>direct</i> and <i>indirect</i> exchanges of data between components. If components exchange data <i>directly</i>, then, for measurement purposes, the measurer will identify Exit and/or Entry data movements, as per the data movements between service A (SA) and service B (SB). An <i>indirect</i> exchange of data between components means that a service in one component writes data in a storage device, which is subsequently read by a service in another component. the measurer will identify a Write data movement in service SA and a Read in service SB</p>	 <p>The diagram shows three boxes: 'Service SA' on the left, 'Service SB' on the right, and 'Persistent Storage' (represented as a cylinder) at the bottom. Two vertical dashed lines labeled 'Boundary' separate the services from the user. On the left, an arrow labeled 'E' points from the user to the first boundary, and an arrow labeled 'X' points from the first boundary to the user. Between the first and second boundaries, arrows show 'E' from SA to SB and 'X' from SB to SA. Below the storage cylinder, an arrow labeled 'W' points from SA to the cylinder, and an arrow labeled 'R' points from the cylinder to SB.</p>

1.4 NFR ontologies

Ontology is a formal representation of a set of concepts within a domain and the relationships between these concepts. It is used to reason about the properties of that domain and may be used to define the domain (Liu 2010) .

An ontology provides a shared vocabulary that can be used to model a domain — that is, with the types of objects and-or concepts that exist, and their properties, relations, and the rules of the ontology meta model. The metadata can be filled in with concepts from an ontology to model a NFR.

An ontology deals with questions concerning what entities exist or can be said to exist and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences.

The use of NFR-ontologies is still as a research in progress in the software/system-NFR domain for the following reasons:

- The NFR-ontologies are built based on individual concepts and views, not on consensual views as agreed upon in international standards;
- The NFR domain is hard to model and the relationships between these requirements are often contradictory; because there is no abstract knowledge for NFR, it is hard to build a conceptualized view for the NFR-ontology;
- The NFR-ontology is difficult to enforce during software development projects;
- The NFR-ontology does not allow measuring these NFR with the available functional size measurement (FSM) methods such as COSMIC and FPA.

1.5 Summary

The term NFR has been used in academia and industry but there is still no common definition for this term and the NFR domain is hard to model and viewed: the NFR are defined, interpreted, and evaluated differently by authors and standards. The relationships between

these requirements are often contradictory. In practice, NFR have received less attention in the software engineering literature, including in software estimation models..

The European ECSS series of standards for the aerospace industry includes sixteen types of NFR for embedded and real-time software and the NFR-related concepts are dispersed throughout the ECSS standards to describe, at varying levels of detail, the various types of candidate NFRs at the system, software, and hardware levels.

The ECSS standards series uses a set of concepts and vocabularies to describe their 16 types of NFR. While conducting an inventory of the entire set of NFR-related concepts and terms described in the ECSS-E-40 and ECSS-Q-80 series and in ECSS-ESA as the integrated standard for ECSS-E and ECSS-Q, we observed that:

- The NFR elements list in ECSS is dispersed throughout various parts and there is therefore no integrated view of all types of NFR elements in the list.
- The NFR elements are described differently and at different levels of detail.
- There is no obvious guidance on how to measure the NFR in the ECSS standards series.

Additionally, in the ISO 9126 and IEEE 830, a number of concepts are provided to describe various types of NFR at the system and software levels in the testing and evaluation processes. However, these ISO and IEEE documents also vary in their views, terminology, and coverage of these requirements.

In addition, the SWEBOK Guide (ISO-19759 2004) includes in the ‘software requirements’ KA a description of the steps to move from system-NFR to software-NFR: requirements elicitation, requirements analysis, requirements specification, and requirements verification. These activities could be used to build a standard-based modeling from a high-level system-NFR to a detailed level of software-NFR such as Configuration Control and Data.

Furthermore, the basic concepts and definitions of FSM have been standardized by the International Organization for Standardization in (ISO-14143-1 2007) and five measurement

methods have been adopted by ISO as International Standards. The newest method in COSMIC (ISO-19761 2011). COSMIC method has been extensively tested and its use is increasing especially in the real-time and telecommunications. In particular, the COSMIC method is compatible with modern specification methods, such as UML, OO techniques and a COSMIC-SOA guideline that developed by the COSMIC group in 2010. This guideline aims to show how the COSMIC method can be applied to measure SOA software requirements.

In the work reported here, preference is given to the views, concepts, and vocabulary most widely used by the industry, as evidenced in its standardization infrastructure, rather than those in the academic literature. Similarly, for the structuring and description of models of FUR and for measurement purposes, the measurement views, concepts, and terminology from the standardization infrastructure have been adopted in this research thesis, rather than those in the literature.

CHAPTER 2

RESEARCH GOAL, OBJECTIVES, AND METHODOLOGY

2.1 Introduction

A research methodology is one of the keys to the success of a research project. It helps ensure that the research itself is valid and the methodology used is appropriate. (Ellis and Levy 2008) mentioned that “the problem is the axis around which the whole research effort revolves. The statement of the problem must first be expressed with the utmost precision; it should then be divided into more manageable sub-problems. Such an approach clarifies the goals and directions of the entire research effort”.

This chapter describes the research project definition including: the research motivation, the research goal, the research objectives, the users of the research results, key inputs to this research work and the research methodology.

2.2 Research motivation

The research project motivation is to contribute to the improvement of the estimation models of software development effort by including the system-NFR in the software estimation process through a quantitative view of such NFR.

2.3 Research goal

The goal for this research project is to help the project managers, organizations, and researchers make informed decisions during project planning and software development projects in the early identification, specification, and measurement of the system-NFR for embedded software. More specifically, this research project aims to contribute to better define, describe, and measure some of the inputs, which are the system-NFR required for *a priori* cost estimation.

2.4 Research objectives

The research objective is the early specification and measurement of software-FUR derived from system-NFR, using as a basis the systems and software engineering standards.

To achieve this research objective the following two specific research sub-objectives must be reached:

- Designs of standard-based models for the identification and specification of software-FUR for system-NFR.
- Measurement of the functional size of software-FUR for system-NFR using the COSMIC ISO 19761 standard.

The NFR measurements results should be available as early as possible in software projects and in particular be available for *a priori* estimating. The results of this research will be a set of standard-based generic requirements and measurement models for system-NFR for embedded software.

2.5 Users of research

The users of research in this research work are people who are working on the requirements and measurement of the software-FUR derived from system-NFR.

2.6 Research input

This research project for the specification and measurement of the software-FUR from the system-NFR using international standards has the following key inputs:

- ECSS European international standards, 2003-2010;
- ISO 9126, 2004;
- IEEE 830, 1998;

- ISO 19761 (COSMIC, 2011);
- COSMIC SOA guideline 2010;
- ISO 19759 (SWEBOK Guide, 2004).

2.7 Overview of the research methodology

This section presents an overview of the research methodology designed to pursue the research objective. This research methodology consists of five phases as seen in Figure 2.1.

Phase 1: NFR in the literature review

Phase 1 of the research methodology consists of surveying the literature on NFR, in both the academic literature and in the international standards on systems and software engineering – see chapter 1 and Annex I.

Phase 2: Identification of NFR concepts, terms and vocabularies in international standards

Phase 2 of the research methodology consists of surveying the concepts, vocabularies and terminologies from different standards for each of the 16 types of ECSS-NFR – see chapter 3.

Phase 3: Mapping and modeling of standard-based models of NFR types for software-FUR specifications and measurement

Phase 3 of the research methodology consists of the identification, specification and measurement of software-FUR derived from fourteen (14) types of system-NFR in the ECSS standards – see chapters 4, 5, 6 and 7 as well as Annex II.

Phase 4: Case study-Valve Control System (VCS)

Phase 4 of the research methodology consists of a case study to illustrate some of the proposed standard-based models of software-FUR derived from system-NFR – see chapter 8.

Phase 5: Traceability and Operationalization

Phase 5 of the research methodology consists of proposing a modified system requirement traceability matrix (RTM) within the system life cycle to tackle system-FUR and system-NFR in high and detailed levels, as well as the detailed traceability to specific sections and pages of the ECSS standards of the proposed reliability-NFR model.

2.8 Detailed research methodology Phase 1: NFR in the literature

It is noted from the literature survey of the NFR in chapter 1 that NFR are still hard to model and use in software projects and NFR are defined differently by different authors. This phase of the methodology consists of following steps:

- **Step 1.1: NFR in the academic literature**

This step presents a survey of the early and recent works on NFR in the academic literature – see chapter 1.

- **Step 1.2: NFR in international standards**

This step presents a survey of the NFR in standards such as ECSS, ISO, IEEE and ISO 19759 (SWEBOK Guide) as well as a standard measurement method such as ISO 19761 (COSMIC method) and its supplementary COSMIC-SOA guideline. This step is a key input for this research study – see chapter 1.

Phase 2: Identification of NFR concepts, terms and vocabularies in international standards

This phase 2 of the methodology consists of the following steps:

- **Step 2.1: ECSS views and concepts for 16 NFR types in the ECSS standard series**

This step identifies the NFR related views, concepts and terms in the ECSS standards.

- **Step 2.2: ISO 9126 views and concepts for some of the NFR types in the ECSS list**

This step identifies the NFR related views, concepts and terms in the ISO 9126 standards.

- **Step 2.3: IEEE views and concepts for some of the NFR types in the ECSS list**

This step identifies the NFR related views, concepts and terms in the IEEE standards.

- **Step 2.4: ISO 19759 (SWEBOK Guide) views and concepts for some of the NFR types in the ECSS list**

This step identifies the NFR related views, concepts and terms in the ISO 19759 for Configuration and design and implementation constraints requirements.

Phase 3: Mapping and modeling of standard-based models of NFR types for software-FUR specifications and measurement

This phase 3 of the methodology consists of following steps:

- **Step 3.1: Identification of functions to be specified for each type of system-NFR**

This step identifies the functions to be specified or corresponding functions to be measured for each type of system-NFR in the ECSS list.

- **Step 3.2: Identification of the system-NFR types allocated to software-FUR**

This step identifies the function types for each type of system-NFR in the ECSS list.

- **Step 3.3: Identification of the NFR relationships**

This step identifies the relationships between the specified function allocated to software-FUR of the system-NFR in the ECSS list by using the COSMIC functional modelling view.

- **Step 3.4: Model of function types relationships based on COSMIC and system views.**

This step identifies a standard-based models for each type of system-NFR allocated to software-FUR.

- **Step 3.5: A standard-based measurement model of software-FUR using COSMIC-SOA Guideline**

This step uses the COSMIC-SOA Guideline to identify the service oriented architecture requirements for the standard-based generic models of each type of the system-NFR allocated to software-FUR.

- **Step 3.6: Sizing a reference instantiation of the standard-based models of software-FUR**

This step identifies the functional measurement size for different specific instantiations for each type of system-NFR allocated to software-FUR.

Phase 4: A case study- Valve Control System (VCS)

This phase of the methodology includes the case study “Valve Control system”. The selected case study aims at the identification, specification and measurement of the standard-based model of software-FUR for system reliability-NFR using the following steps:

- **Step 4.1: Specification of the system reliability-NFR to be allocated to software.**
- **Step 4.2: Specification of the ECSS-based reliability allocated to software-FUR for the VCS components.**
- **Step 4.3: Measurements of the system-reliability-NFR for the VCS case study.**

Phase 5: Traceability and operations

This phase 5 of the methodology consists of the following steps:

- **Step 5.1: Traceability model in the ECSS standard series**
 These step presents the traceability model used in the ECSS standards series for the system-FUR.
- **Step 5.2: Improvement for a traceability model in ECSS standard series**
 This step identifies some improvements for extending the ECSS traceability model in the previous step to include system-FUR and NFR;
- **Step 5.3: The traceability of the concepts, terms and vocabularies for the proposed fourteen standard-based models**
 This step identifies the traceability of the functions identified in the various ECSS, IEEE and ISO standards for the proposed fourteen standard-based models.
- **Step 5.4: A detailed traceability for the standard-based model of software-FUR for system reliability-NFR**
 This step identifies the detailed traceability to specific sections and pages of the ECSS standards of the proposed reliability-NFR model.

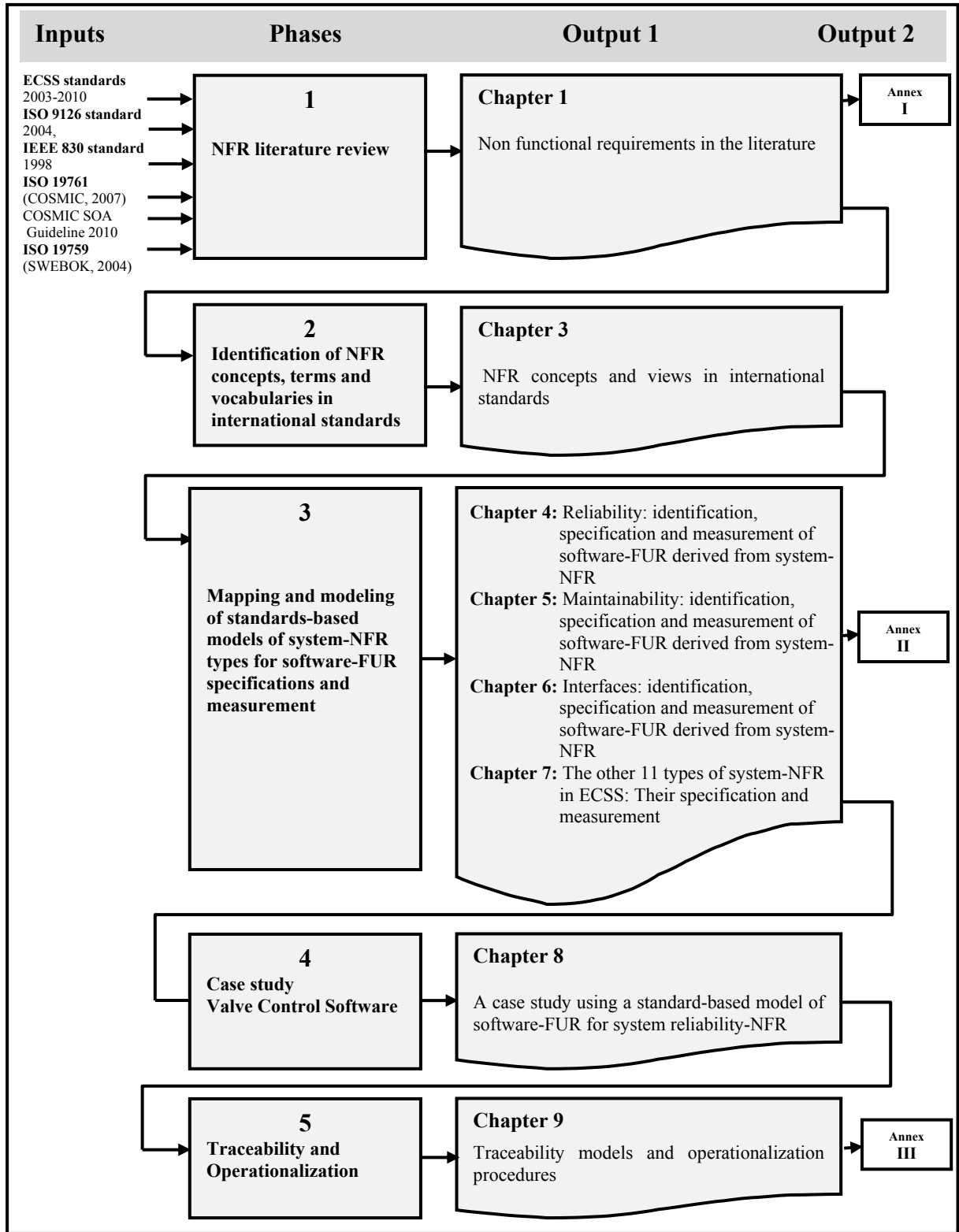


Figure 2.1 Research methodology – overview of phases

CHAPTER 3

IDENTIFICATION OF NFR CONCEPTS AND VIEWS IN STANDARDS

3.1 Introduction

The NFR are typically described at the system level and not at the software level. As yet, there is no consensus on how to describe and measure system-NFR. In current practice, they may be viewed, defined, interpreted, and evaluated differently by different people in the later project phases, particularly when they are stated vaguely and only briefly in the system requirements phase (Chung and do Prado Leite 2009).

In the ECSS (European Cooperation on Space Standardization) standards for the aerospace industry (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005), ISO 9126 (ISO-9126 2004) and IEEE 830 (IEEE-830 1998), a number of concepts are provided to describe various types of NFR at the system, software, and hardware levels. However, these standards vary in their views, terminology, and coverage of operations.

This chapter is organized as follows to describe the concepts and views of the 16 types of system-NFR in the ECSS series:

Section 3.2 Reliability systems requirements.

Section 3.3 Maintainability systems requirements.

Section 3.4 Interfaces systems requirements.

Section 3.5 Portability systems requirements.

Section 3.6 Operations systems requirements.

Section 3.7 Configuration systems requirements.

Section 3.8 Data definitions and database systems requirements.

Section 3.9 Adaptations and installations systems requirements.

Section 3.10 Design and implementation constraints systems requirements.

Section 3.11 Performance systems requirements.

Section 3.12 Security and privacy systems requirements.

Section 3.13 Safety systems requirements.

Section 3.14 Resources systems requirements.

Section 3.15 Human factor requirements.

A summary is presented in section 3.16

3.2 Reliability systems requirements

This section presents a survey of the reliability-related views, concepts, and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) , ISO 9126 (ISO-9126 2004), and IEEE-830 (IEEE-830 1998) standards. This section identifies which standards currently address aspects of the software-FUR that may be derived from system reliability FUR and NFR.

3.2.1 ECSS: views and concepts for reliability

Reliability, in the ECSS standards, shall be specified at the system level. The reliability requirements can be met by introducing adequate redundancy features. The ECSS standards consider reliability as the acceptable probability of system failure which is based on the equipment reliability and availability specifications.

According to the ECSS, reliability models shall be prepared to support predictions: FMEA (Failure Mode and Effects Analysis), FMECA (Failure Mode, Effects and Criticality Analysis) as well as reliability testing. Demonstration shall be performed according to the project reliability requirements in order to check the following:

1. Failure modes and effects;
2. Failure tolerance, failure detection and recovery;
3. Statistical failure data to support predictions and risk assessment;
4. Consolidated reliability assessments;

5. Capability of the hardware to operate with software or to be operated by a human being in accordance with the specifications;
6. Demonstrated reliability of critical items;
7. Justification of data bases used for theoretical demonstrations.

Table 3.1 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related reliability requirements. The ECSS standards specify that reliability requirements must be implemented in software, hardware, or a combination of the two.

3.2.2 IEEE: views and concepts for reliability

IEEE-830 (IEEE-830 1998) lists reliability as one of the thirteen (13) NFR types in their list. IEEE-830 (IEEE-830 1998) only defines the reliability requirements as the factors required to establish the required reliability of the software system at time of delivery; however, it does not provide guidance on how to describe and specify the reliability requirements and it does not provide guidance on how to measure them - see Table 3.1.

IEEE-1220 (IEEE-1220 2007) only defines the reliability requirement as the analysis of system effectiveness for each operational scenario, without mentioning how to describe and specify the reliability requirements.

3.2.3 ISO views and concepts for reliability

The key view on reliability in the ISO 9126 (ISO-9126 2004) series is from the perspective of the quality of the software product: reliability is presented as a quality characteristic, which is decomposed into quality sub-characteristics and then into proposed derived measures to quantify those quality sub-characteristics. The inventory of related concepts and vocabulary on software reliability, such as maturity, fault tolerance and recoverability, is presented in Table 3.1.

Furthermore, (ISO-24765 2008) for the systems and software engineering vocabulary defines the reliability as the probability that software will not cause the failure of a system for a specified time under specified conditions. (ISO-24765 2008) uses the following concepts with their definitions:

1. Function to identify error to input;
2. Function to identify error to output.

Table 3.1 Reliability views, concepts and terms in ECSS and ISO

ID	Standard Organization	Key view	Concepts and terms
1	ECSS	Acceptable probability of system failure	<ul style="list-style-type: none"> • Component failure • Redundancy feature • Data parameter • Reliability methods, operations and mechanism • Failure tolerance • FMEA and FMECA • Failure detection • Failure isolation • Failure recovery • Failure data
2	ISO	The capability of the software product to maintain a specified level of performance when used under specified conditions	<ul style="list-style-type: none"> • Maturity • Fault tolerance • Recoverability • Fault Density • Failure Resolution • Incorrect Operation • Availability • Breakdown Time • Recovery Time • Fault Removal • Failure Avoidance • Restart ability • Restorability

3.3 Maintainability systems requirements

This section presents a survey of the maintainability-related views, concepts, and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009), (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) standards. This section identifies which standards currently address aspects of the software-FUR which may be derived from the system maintainability FUR and NFR.

3.3.1 ECSS: views and concepts for maintainability

Maintainability in the ECSS standards is considered part of the integrated support requirements in system engineering, including related activities and procedures. Table 3.2 presents a list of the concepts and vocabulary used in the set of 13 ECSS standards to describe system-related maintainability requirements. For instance, the ECSS specifies that, for system maintainability, failure modes, effect, and criticality (FMECA) must be analyzed. It does not specify, however, whether such requirements must be implemented in software or hardware, or in a combination of the two.

3.3.2 IEEE: views and concepts for maintainability

IEEE-830 (IEEE-830 1998) standard lists maintainability as one of the NFR types on their list, but does not define it, nor does it provide guidance on how to describe and specify the maintainability requirements; neither, of course, does it provide guidance on how to measure any of these NFR. (IEEE-14764 2006) and (IEEE-982.1 2005) only define the maintainability requirement as the capability of the software product to be modified, without mentioning how to describe and specify the maintainability requirements.

3.3.3 ISO: views and concepts for maintainability

The key view on maintainability in the ISO 9126 (ISO-9126 2004) series is from the perspective of the quality of the software product. Maintainability is presented as a ‘quality characteristic’ of the software, which is decomposed into quality sub characteristics and then into proposed derived measures to quantify those quality sub characteristics. The inventory of related concepts and vocabulary on software maintainability, such as analyzability, changeability, etc., is presented in Table 3.2.

Table 3.2 Maintainability views concepts and terms in the ECSS and ISO standards

ID	Standard organization	Key view	Concepts and terms
1	ECSS	Part of the integrated logistical support requirements in system engineering, including activities and procedures	<ul style="list-style-type: none"> • Maintainability activities and procedures • Maintainability operations • Environment control and life support systems design (ECLSS) • FMECA: failure mode, effect, and criticality analysis • FMEA: failure mode and effect analysis • Mean time-to-repair and system downtime • Fault detection and isolation capability • System malfunction
2	ISO	<p>The maintainability quality characteristic denotes the capability of the software product to be modified.</p> <p>Modifications may include corrections, improvements, or adaptation of the software to changes in environment</p>	<ul style="list-style-type: none"> • Analyzability • Audit Trail Capability • Failure Analysis Capability • Status Monitoring Capability • Diagnostic Function Support • Changeability • Change Efficiency • Software Change Control Capability • Modifiability • Stability

Table 3.2 Maintainability views concepts and terms in the ECSS and ISO standards
(Continued)

ID	Standard organization	Key view	Concepts and terms
2	ISO	<p>The maintainability quality characteristic denotes the capability of the software product to be modified.</p> <p>Modifications may include corrections, improvements, or adaptation of the software to changes in environment</p>	<ul style="list-style-type: none"> • Modification Impact • Change Success Ratio • Testability • Availability of a built-in test function • Retest Efficiency • Test Restart Capability

3.4 Interface systems requirements

This section presents a survey of the interface-related views, concepts, and terms in the ECSS and IEEE-830 standards. This section identifies which standards currently address some aspects of the software-FUR derived from the interface system-FUR and system-NFR.

3.4.1 ECSS: views and concepts for interfaces

The ECSS standards ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) present software interfaces as a set of NFR for real time and embedded software. In particular, the ECSS-E-40 include the design of the external interface as part of the interface control document (ICD), while the design of the internal interface is included as part of the software design document (SDD). Also, (ECSS-ESA 2005) specifies that the detailed design of the software product interfaces should be defined during the interface design phase.

In ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), the majority of interfaces are software-to-software interfaces, and the ECSS requires

that they shall be defined in the requirements baseline (i.e., the requirements baseline must include the requirements applicable to the various elements of the system product tree).

Table 3.3 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related interface requirements. These standards specify that interface requirements must be implemented in software or hardware, or a combination of the two.

3.4.2 IEEE: view and concepts for interfaces

Software interface requirements are also presented in (IEEE-830 1998) as NFR, and the interface is defined through a detailed description of all inputs into, and outputs from, the software system. In particular, (IEEE-830 1998) mentions that the interface can be analyzed and understood through the user interfaces, the hardware interfaces, the software interfaces, and the communications interfaces. Note that, where (IEEE-830 1998) defines an interface as ‘inputs’ or outputs’, these are considered only as collections of data movements and not integrated into specific functional processes. (IEEE-830 1998) does not provide an analysis or explanation of the linkage between the set of interface concepts identified- see Table 3.3.

IEEE (IEEE-830 1998) identifies the system interface functionality of the software designed to accomplish the system requirements, and describes the interfaces that match the system for the following types of interfaces:

1. User interface: provides the logical characteristics for communication between the software product and its users. This includes the configuration characteristics (e.g. required screen formats, page or window layouts, content of any reports or menus, availability of programmable function keys, etc.) necessary to meet the software requirements;
2. Hardware interface: provides the logical characteristics for communication between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.) and addresses such matters as what devices are to be supported, how they are to be supported, and the related

- protocols. For example, full-screen support may be specified for the terminal, as opposed to line-by-line support;
3. Software interface: specifies other software products (e.g. a data management system, an operating system, a mathematical package, etc.) for the user, and provides the means of communication with other application systems (e.g. the linkage between an accounts receivable system and a general ledger system) required by the user;
 4. Communications interface: provides the user with network interconnection capability, such as with local network protocols, etc.

Table 3.3 Interface: views, concepts and terms in ECSS and IEEE

ID	Standards organization	Key view	Concepts and terms
1	ECSS	Interface requirements must be implemented in software or hardware, or a combination of the two	<ul style="list-style-type: none"> • User interface between the system and the product • Interface communications in various communication layers: <ul style="list-style-type: none"> ▪ External interfaces for telemetry, telecommands, ranging, and data: spacecraft-to-ground, spacecraft-to-spacecraft, ground-to-ground ▪ Internal interfaces between items of on-board equipment • Control software interfaces, including: <ul style="list-style-type: none"> ▪ Software interfaces for system applications ▪ Interface specifications through programming languages ▪ Interface specifications for each layer of socket programming, including data size control, data direction (unidirectional or bidirectional), data serialization, methods, and ports • Control Hardware interfaces: • Physical, thermal, and electrical interfaces
2	IEEE	Interface is defined through a detailed description of all inputs into, and outputs from, the software system	<ul style="list-style-type: none"> • User interfaces, • Hardware interfaces, • Software interfaces, and • Communications interfaces.

3.5 Portability systems requirements

This section presents a survey of the portability-related views, concepts, and terms used in international standards. It identifies which standards currently address some aspects of the software-FUR derived from the portability system-NFR. The elements of portability are dispersed in various system views throughout the ECSS standards, and are expressed as either:

- System portability functional user requirements (system portability-FUR);
- System portability non functional requirements (system portability-NFR).

3.5.1 ECSS: view and concepts for portability

The ECSS standards series ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), includes a number of portability requirements at the system level. Portability in the ECSS standards is considered as the capability of the system to be transferred from one environment to another. Table 3.4 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related portability requirements. For instance, the ECSS specifies minimum dependency on software and hardware (system portability) and independence of the operating system from hardware and software obsolescence. What it does not specify, however, is whether or not such requirements must be implemented in software or hardware, or a combination of the two.

3.5.2 IEEE: view and concepts for portability

The (IEEE-830 1998) lists the portability requirements as one of the NFR on their list. The IEEE describes portability by specifying the attributes of software that relate to the ease of porting the software to other host machines and/or operating systems, and provides some portability concepts – see Table 3.4.

3.5.3 ISO standards: views and concepts for portability

The key view on portability in the (ISO-9126 2004) is from the perspective of the quality of the software product: portability is presented as a ‘quality characteristic’ and is then decomposed into quality sub-characteristics and next into proposed derived measures to quantify those quality sub-characteristics. The inventory of related concepts and vocabulary on software portability, such as replaceability and co-existence, is presented in Table 3.4.

Portability in (ISO-24765 2008) is considered as a system or component that can be transferred from one hardware or software environment to another. Table 3.4 presents the concepts and vocabulary used in (ISO-24765 2008) to describe system-related portability requirements. While ISO 24765 states that portability in a system environment refers to a transfer between software and hardware, it does not specify whether portability requirements must be implemented in the software or the hardware, or in a combination of the two.

Portability in (ISO-2382-1 1993) is described as a program to be executed on various types of data processing systems. Table 3.4 presents a list of concepts and vocabulary used in (ISO-2382-1 1993) to describe system-related portability requirements. For instance, this standard refers to portability between a program and a sub part of the same program (sub program) when this program is executed using different data processing systems and system program calls (SPC) or remote procedural calls (RPC) between the program and sub program functions, independently of the language. It does not, however, specify whether such requirements must be implemented in the software or the hardware, or in a combination of the two.

Table 3.4 Portability views, concepts and terms in the standards

ID	Standard organization	Key views	Concepts and terms
1	ECSS	The capability of the system to be transferred from one environment to another	<ul style="list-style-type: none"> • Minimum system dependency • Independent from the operating system • Minimum hardware dependency • Obsolescence of hardware or software

Table 3.4 Portability views, concepts and terms in the standards (Continued)

ID	Standard organization	Key views	Concepts and terms
2	IEEE 830	Describe portability by specifying the attributes of software that relate to the ease of porting the software to other host machines and/or operating systems	<ul style="list-style-type: none"> • Percentage of components with host-dependent code • Percentage of code that is host-dependent • A proven portable language • A particular compiler or language subset • A particular operating system
3	ISO 9126	The capability of the software product to be transferred from one environment to another Environment may include the organizational, hardware, or software environment	<ul style="list-style-type: none"> • Sharing common resources • Independent software in a common environment • Continued use of data • Software running concurrently with other software • Replaceability • Co-existence
4	ISO 24765	A system or component can be transferred from one hardware or software environment to another	<ul style="list-style-type: none"> • Software environment • Hardware environment
5	ISO 2382-1	A program to be executed on various types of data processing systems	<ul style="list-style-type: none"> • Language independence • Data processing system • Isolating software system calls

3.6 Operations systems requirements

This section presents a survey of the operations-related views, concepts, and terms in the ECSS and IEEE-830 standards. It identifies which standards currently address aspects of the software-FUR derived from system operations FUR and NFR. The expected outcome is the identification of the various elements that should be included in the design of a standard-based framework for modeling software-FUR for system operations-NFR.

3.6.1 ECSS: views and concepts for operations

The ECSS standards series ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005) includes a number of operations requirements at the system level. Clearly, the ECSS focuses on the system-FUR for the early development phases, while the system-NFR are typically discussed within the context of later development phases, such as evaluation or testing.

The elements of operations are dispersed in various system views throughout various ECSS standards, and are expressed as either:

- System operations FUR;
- System operations NFR.

Operations in the ECSS standards include any specified operations mode and mode transition for the software, and, in the case of man-machine interaction, the intended use scenarios and diagrams may be used to show the intended operations and related transition modes. Moreover, operations engineering should cover all operations activities through all phases of the life cycle; i.e., preparation, validation, execution, and disposal.

Table 3.5 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related operations requirements. For instance, the ECSS specifies that, for system operations mode, an analysis of the operational functions (inter-operational function and operational function event) and of the system transitions mode (operational control interface and operational data interface) must be carried out.

The ECSS specifies that such requirements must be implemented in software or hardware, or a combination of the two.

3.6.2 IEEE: views and concepts for Operations

The (IEEE-830 1998) includes operations as one of the NFR types in their list of NFR, and considers the various modes of operation as part of the user interface. But it does not define what an operations requirement is, nor does it provide guidance on how to describe and specify the operations requirements. Of course, it does not provide guidance on how to measure any of these NFR either.

Table 3.5 Operations: view, concepts and terms in the ECSS standards

ID	Standard Organization	Key views	Concepts and terms
1	ECSS	Operational and transition modes	<ul style="list-style-type: none"> • Inter-operational function • Operational function event • Operational control interface • Operational data interface • System operations mode • System transitions mode • Operational scenario

3.7 Configuration systems requirements

This section presents a survey of the configuration-related views, concepts, and terms in the ECSS standards ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005) and in the SWEBOK Guide (ISO-19759 2004). It identifies which standards or views currently address some aspects of the software-FUR derived from system-NFR, specifically for the functional configuration requirements.

3.7.1 ECSS: views and concepts for configuration

Configuration in the ECSS standards is considered part of the “design and implementation engineering process”, which includes control activities and data flows for the operational

functions and data transfers of defined items. Table 3.6 presents a list of the concepts and vocabulary used in those standards to describe system-related configuration requirements.

For instance, ECSS standards specify that each item or element defined during the design phase can be configured. They also specify what configuration requirements shall be implemented in software.

3.7.2 ISO 19759 (SWEBOK Guide): views and concepts for configuration

The key view on configuration in the SWEBOK guide (ISO-19759 2004) is that of a software with minor system views for the functional and/or physical characteristics of hardware, firmware, or software, or a combination of these, as set forth in technical documentation and achieved in a product – see Table 3.6.

Configuration can also be thought of as a collection of specific versions of hardware, firmware, or software items combined according to specific procedures to serve a particular purpose. Configuration management (CM), then, is the discipline of identifying the configuration of a system at distinct times for the purpose of systematically controlling changes to that configuration.

The use of the functional configuration audit (FCA) and the physical configuration audit (PCA) can be considered as a prerequisite for the establishment of the product baseline. The purpose of the PCA is to ensure that the design and reference documentation are consistent with the product as built.

Table 3.6 Configuration: views, concepts and terms in ECSS and ISO 19759

ID	Standard organization	Key view	Concepts and terms
1	ECSS	Secure environment with controlled access linked to the required physical and functional characteristics	<ul style="list-style-type: none"> • Control activities of defined configuration items: <ul style="list-style-type: none"> – control flow

Table 3.6 Configuration: views, concepts and terms in ECSS and ISO 19759 (Continued)

ID	Standard organization	Key view	Concepts and terms
1	ECSS	Secure environment with controlled access linked to the required physical and functional characteristics of the system	<ul style="list-style-type: none"> – data flow • Each item or component defined during the design can be configured, such as: <ul style="list-style-type: none"> – modules, – processes and threads, – events and communication channels between a module and a sub software module • Control operational functions • Register data transfers
2	ISO 19759	Functional and/or physical characteristics of hardware, firmware, or software, or a combination of these	<ul style="list-style-type: none"> • Functional characteristics of hardware, firmware, and software • Systematic control of changes to configuration • Configuration control • Physical configuration audit (PCA)

3.8 Data definitions and database systems requirements

This section presents a survey of the data definitions and database -related views, concepts, and terms in the ECSS standards ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005). This section identifies which standards currently address aspects of the software-FUR derived from system data definition and database FUR and NFR

3.8.1 ECSS views and concepts for data definition and database

(ECSS-E-ST-70-31C 2008) includes a number of data definition and database requirements at the system level. Data definitions and database requirements are described in ECSS

standards series by data requirements and the corresponding mission data provided by a supplier to a customer. Formally, this data is part of the user manual for the corresponding element of the space system. Moreover, the ECSS requires these data definitions and database requirements to be defined in the requirements baseline (i.e., the requirements baseline must include the requirements applicable to the various elements of the system product tree).

More specifically, data definitions and database requirements are described through the data model requirements and the system model object. According to the ECSS standards, data model requirements (ECSS-E-ST-70-31C 2008) are composed of:

1. System data items which include:
 - System entity types (such as: event, parameter, system element, reporting data);
 - System value types (such as: simple value and record value);
 - System data types (simple type and complex type).

2. Product data schema which include:
 - Product configuration data;
 - Monitoring & data control.

A system model object refers to any object of the populated database that is uniquely identified by a name: e.g. a system element, a reporting data, an activity or an event. Moreover, a system model object is derived from data model requirements and data mission.

Table 3.7 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related data definition and database requirements. ECSS standards specify that data definitions and database requirements must be implemented in software, hardware, or a combination of the two.

Table 3.7 Data definitions and database views, concepts and terms in ECSS

ID	Standard organization	concepts and terms			
1	ECSS	Data model requirements	System Data items	System entity types	<ul style="list-style-type: none"> • Event • Parameter • System element • Reporting data • Activity
				System value types	<ul style="list-style-type: none"> • Simple value • Record value
				System data types	<ul style="list-style-type: none"> • Simple type • Complex type
			Product data schema	<ul style="list-style-type: none"> • Product configuration data • Monitoring & data control 	
		System model object	<ul style="list-style-type: none"> • Data model requirements • Data mission. 		

3.9 Adaptation and installation: systems requirements

This section presents a survey of the adaptation and installation-related views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) standards. This section identifies which standards currently address some aspects of the software-FUR derived from system-NFR, specifically for the adaptation and installation software-FUR.

3.9.1 ECSS: views and concepts for adaptation and installation

The ECSS-E-40 specifies that adaptation and installation requirements should be described or referenced; also (ECSS-ESA 2005), specifies that the supplier shall prepare the deliverable software product for its installation in the target platform or system environment as well as the resources and information to install shall be determined and available containing set-up activities.

While the (ECSS-E-ST-10C 2009) describes the adaptation and installation requirements as the adaptation data to specific installation for the system. ECSS-E-ST-10C 2009 identifies the adaptation data by making reference to all unique-to-site data contained in the released software as well as specifying all the instructions to build and install the software item, including:

1. Procedures to regenerate executable software from the delivered source code;
2. Procedures to install the software in the target environment;
3. Procedures to verify the correct execution of the installation;
4. Adaptation data, security issues relevant to the installation.

(ECSS-Q-ST-80C 2009) mentions that the adaptation and installation requirements should be described through approaches, methods, procedures, resources and organization to install, commission, and check the operation of the equipment in its fixed operational environment.

Moreover (ECSS-E-ST-10C 2009) describes the installation to be performed in accordance with the installation procedure and that the ground computer equipment and supporting services for implementing the final system shall be selected according to the project requirements regarding installation requirements conditions.

(ECSS-E-ST-10C 2009): the system adaptation as the resource reallocation between software, hardware and system environment. Table 3.8 illustrates the set of concepts and vocabulary used in the ECSS standards to describe adaptation and installation.

3.9.2 IEEE: views and concepts for adaptation and installation

The (IEEE-830 1998) lists adaptation and installation requirements as one of the NFR type. IEEE defines adaptation and installation as requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode.

In particular, (IEEE-830 1998) mentions that the site or mission-related features should be modified to adapt the software to a particular installation, but does not provide guidance on how to describe and specify the adaptation and installation requirements.

3.9.3 ISO: views and concepts for adaptation and installation

The key view on adaptation and installation in the (ISO-9126 2004) is from the perspective of the quality of the software product: adaptation and installation is presented as ‘sub quality characteristics’ of the portability quality characteristic. The inventory of related ISO concepts and vocabulary on software adaptation and installation is presented in Table 3.8.

Table 3.8 Adaptation and installation views, concepts and terms in the standards

ID	Standard organization	Key views	Concepts and terms
1	ECSS	Adaptation and installation requirements are described using approaches, methods, procedures, resources and check the operation of the equipment in its fixed operational environment.	<ul style="list-style-type: none"> • System Environments (adaptation) <ul style="list-style-type: none"> – Host-Target platform – Memory Resources – Storage resources – Transmission resources – I/O resources • Software and Data Environments (installation) <ul style="list-style-type: none"> – Registered Data Transfer – Control Data Transfer – Set Data Transfer with system resources
2	ISO	<p>Software product is adapted to different specified environments</p> <p>Software product installed in a specified environment</p>	<ul style="list-style-type: none"> • Adaptability of hardware, software and system environment. • Adaptability of software data structures. • Ease of software installation procedure • Ease of setup retry when the software is already installed

3.10 Design and implementation constraints (D&I) systems requirements

This section presents a survey of the design and implementation (D&I) constraints views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) standards and (ISO-19759 2004) (SWEBOK Guide) . This section identifies which standards currently address some aspects of the software-FUR derived from system requirements.

3.10.1 ECSS: views and concepts for D&I constraints

The elements of D&I constraints are dispersed in various system views throughout different ECSS standards and are expressed as either:

1. System D&I constraints functional user requirements (system D&I constraints-FUR);
2. System D&I constraints non-functional requirements (system D&I constraints-NFR).

The identification of D&I constraints in the ECSS standards is derived from an analysis of the requirements on the system and its functions. All system requirements are allocated to a set of D&I constraints. Moreover, hardware configuration D&I constraints, software configuration D&I constraints, and human operations D&I constraints shall be subsequently identified from these requirements. The supplier shall transform the requirements for the software D&I constraints into an architecture that describes its top-level structure and identifies the software components, ensuring that all the requirements for the software D&I constraints are allocated to its software components and later refined to facilitate detailed design.

The software architectural design shall describe the D&I constraints within:

1. The static architecture (i.e., decomposition into software elements such as packages and classes or modules);

2. The dynamic architecture, which involves active objects such as threads, tasks and processes;
3. The mapping between the static and the dynamic architecture, and the software behaviour.

The software D&I constraints requirements shall produce the physical model of the software components described during the software architectural design. For embedded software D&I constraints the following information should be included:

1. Type of D&I constraints participating to the real time behaviour, described by stating its logical and physical characteristics with D&I;
2. Scheduling types with D&I (e.g. single or multi-threads);
3. Scheduling model with D&I (e.g. pre-emptive or not, fixed or dynamic priority based);
4. Analytical model with its D&I (e.g. rate monotonic scheduling, deadline monotonic scheduling);
5. Tasks identification and D&I priorities;
6. Communication and synchronization with D&I;
7. Time management through D&I;
8. The dependencies of a component should be described by listing the D&I upon its use by other components.

The ECSS-ESA document covers the tailoring of the ECSS-E-40 requirements for the European Space Agency (ESA) software projects. In this document, the software design includes a program design, pseudo-code and flow charts. Software D&I may specify that the processing has to be performed using a particular algorithm and program parameters.

Table 3.9 presents a list of concepts and vocabulary used in ECSS to describe system related D&I constraints requirements and ECSS mentions that such requirements may be implemented in software.

3.10.2 ISO 19759 (SWEBOK Guide): views and concepts for D&I

According to the (ISO-19759 2004) SWEBOK Guide, “Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problem”. The (ISO-19759 2004) mentions explicitly D&I constraints as non-functional requirements in the “Software Requirements’ knowledge area (KA) and implicitly within the context of activities for design in the ‘Software Design’ KA.

Software design is defined in (ISO-19759 2004) as both “the process of defining the architecture, components, interfaces, and other characteristics of a system or component” and “the result of [that] process”. Furthermore software design in the software engineering life cycle is defined as activities in which software requirements are taken as inputs for analysis in the software design phase.

The architectural design is also described by (ISO-19759 2004) as the point at which the requirements process overlaps with software or systems design and illustrate how challenging it is to cleanly decouple the two tasks; software architecture is “a description of the subsystems and components of a software system and the relationships between them”. This means that ISO 19759 is describing the D&I constraints in the Software Requirements KA and these D&I constraints should be reflected on the software design in the Software Design KA.

Moreover, software design consists of two activities that fit between software requirements analysis and software construction (ISO-19759 2004):

- Software architectural design (sometimes called top level design): describing software’s top-level structure and organization and identifying the various components;
- Software detailed design: describing each component sufficiently to allow for its construction.

The (ISO-19759 2004) decomposes the software D&I constraints into processes, tasks, and threads and deals with related efficiency, atomicity, synchronization, and scheduling issues.

Table 3.9 presents a list of concepts and vocabulary used in the (ISO-19759 2004) to describe system related D&I constraints. The (ISO-19759 2004) SWEBOOK Guide specifies that such requirements be implemented in software design.

Table 3.9 D&I constraints views, concepts and terms in ECSS and ISO

ID	Standard organization	Key view	Concepts and terms
1	ECSS	Design and implementation (D&I) constraints applicable to various components of the system product	<ul style="list-style-type: none"> • Software architectural D&I constraints on modules, classes, packages • Software detailed D&I constraints on tasks and processes • Physical model of the software D&I constraints described during the software architectural design • The logical model of the D&I constraints described in software architectural design
2	ISO 19759	Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problem	<ul style="list-style-type: none"> • Software architectural D&I constraints on modules, classes, packages or top level structure • Software detailed D&I constraints on tasks and processes • Physical model of the software D&I constraints described during the software architectural design • The logical model of the software D&I constraints described within the software architectural design • Static and dynamic D&I constraints with system design

3.11 Performance systems requirements

This section presents a survey of the performance requirements views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) standards. The expected outcome is the identification of the various elements that should be included in the standard-based

framework for modelling software-FUR for system performance requirements.

3.11.1 ECSS: views and concepts for performance

The elements of performance requirements are dispersed in various system views throughout different ECSS standards and are expressed as either:

1. System performance functional user requirements (system performance -FUR);
2. System performance non-functional requirements (system performance -NFR)

The (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) present software performance as a system-NFR for embedded software – see Table 3.10.

According to (ECSS-E-ST-60-20C-Rev.1 2008) standard. Performance requirement is a specification that the output of the system does not deviate by more than a given amount from the target output.

According to (ECSS-E-ST-60-20C-Rev.1 2008) the performance can be measured by evaluating processing speed, response time, resource consumption and throughput.

In (ECSS-E-ST-60-20C-Rev.1 2008) and (ECSS-E-60A 2004) the performance requirements should assess that the controlled system performance is coherent with the control objectives generated by the requirement engineering process and the numerical requirements defined by the requirements analysis, furthermore, performance analysis should be conducted during all the phases of the control development process.

The response time (ECSS-E-ST-60-20C-Rev.1 2008) is defined as minimally acceptable of the rest time. A longer response time can cause users to think the system is down. You also need to specify *rest of time*; for example, the peak minute of a day, 1 percent of interactions. Response time degradations can be more costly or painful at a particular time of the day.

Response time is measured (ECSS-E-ST-60-20C-Rev.1 2008) from the time that the user performs the action until the user receives enough feedback from the computer to continue the task. It is the user's subjective wait time. It is not from entry to a subroutine until the first write statement.

Performance monitoring or performance observation (ECSS-E-ST-60-20C-Rev.1 2008) is often used in optimizing the use of software in a system. A performance monitor is generally regarded as a facility incorporated into a processor to monitor selected characteristics to assist in the debugging and analyzing of systems by determining a machine's state at a particular point in time. Often, the performance monitor (ECSS-E-ST-60-20C-Rev.1 2008) produces information relating to the utilization of a processor's instruction execution and storage control. For example, the performance monitor can be utilized to provide information regarding the amount of time that has passed between events in a processing system. The information produced usually guides system architects toward ways of enhancing performance of a given system or of developing improvements in the design of a new system.

The typical throughput refers to the number of event responses that have been completed over a given observation interval as in (ECSS-E-ST-60-20C-Rev.1 2008).

Performance can be measured using the following concepts and terms such as: response to reference signals (e.g. response time, settling time, and tracking error for command profiles), accuracy and stability errors in the presence of disturbances, measurement errors (e.g. attitude knowledge) and frequency domain requirements (e.g. bandwidth).

3.11.2 IEEE: views and concepts for performance

The (IEEE-830 1998) presents software performance requirements as a non-functional requirement; in addition, (IEEE-830 1998) defines the performance requirements as the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the number of terminals

to be supported, the number of simultaneous users to be supported and amount and type of information to be handled. Static numerical requirements are sometimes identified under a separate section entitled capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions. All of these requirements should be stated in measurable terms – see Table 3.10.

Table 3.10 Performance views, concepts and terms in ECSS

ID	Standard organization	Key view	Concepts and terms
1	ECSS	Performance requirement is a specification that the output of the system does not deviate by more than a given amount from the target output	<ul style="list-style-type: none"> • Response to reference signals <ul style="list-style-type: none"> – Response time, – Settling time, – Tracking error for command profiles • Throughput time. <ul style="list-style-type: none"> – Bandwidth – Workload • Resource consumption <ul style="list-style-type: none"> – Main memory time – Storage device time – processor execution time • Evaluation processing speed <ul style="list-style-type: none"> – Accuracy errors – Stability errors – System scalability
2	IEEE 830	Performance requirements as static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.	<ul style="list-style-type: none"> • Static numerical requirements <ul style="list-style-type: none"> • Capacity • Concurrency • Dynamic numerical requirements <ul style="list-style-type: none"> • Workload

3.12 Security systems requirements

This section presents a survey of the security requirements views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) and (ISO-9126 2004) standards. The expected outcome is the identification of the various elements that should be included in the standard-based model of software-FUR for system security requirements.

3.12.1 ECSS: views and concepts for security

The ECSS standards series present security as a system-NFR for real-time and embedded software – see Table 3.11 in these standards, the security requirements are described as specifications, including related factors, which might compromise sensitive information; and the ECSS requires that the system security shall be defined in the requirements baseline (i.e., the requirements base must include the requirements applicable to the various elements of the system product tree.

In the ECSS standards, the system security is described as:

1. Access control roles for person or group of persons and access control per system or entity;
2. Availability for redundant power or data and automatic restart;
3. System data integrity such as integrity with firewall, antivirus, external PKI (encryption and decryption of data) and integrity with different types of system backup (such as automatic, time interval, durability, data versioning and run-time backups).

1.12.2 IEEE: views and concepts for security

Security requirements are also presented in (IEEE-830 1998) as an NFR type : IEEE specifies the factors that protect the software from accidental or malicious access use,

modification, destruction, or disclosure. Specific requirements in this area could include the need to utilize certain cryptographically techniques; to keep specific log or history data sets; to assign certain functions to different modules; to restrict communications between some areas of the program and to check data integrity for critical variables – see Table 3.11.

1.12.3 ISO: views and concepts for security

The (ISO-9126 2004) lists the security as part of the software functionality to define the software product quality. In addition, (ISO-9126 2004) defines the security as the capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them – see Table 3.11.

Table 3.11 Security: views, concepts and terms in standards

ID	Standards organization	key views	Concepts and terms
1	ECSS	The key views of software security requirements in ECSS standards are described as specifications, including related factors, which might compromise sensitive information. Moreover, the ECSS standards require that the system security shall be defined in the requirements baseline which defines the requirements applicable to various elements of the system product tree	<ul style="list-style-type: none"> • Access control roles for the system, person and groups • Availability for redundant power or data and automatic restart man machined • System data integrity such as integrity with firewall, antivirus, external PKI
2	IEEE-830	The key views of software security requirements in the IEEE 830 standard are factors that protect the software from accidental or malicious access use, modification, destruction, or disclosure	<ul style="list-style-type: none"> • Cryptographic techniques; • Specific log or history data sets • Assign certain functions to different modules • Restrict communications between some areas of the program and • Check data integrity for critical variables

Table 3.11 Security: views, concepts and terms in standards (Continued)

ID	Standards organization	key views	Concepts and terms
3	ISO 9126	The key view of software security in ISO 9126 is described as a part of the software functionality to define the software product quality	<ul style="list-style-type: none"> • Access Auditability • Access Controllability • Data Corruption/Prevention • Data Encryption

3.13 Safety systems requirements

This section presents a survey of the safety-related views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005), (ISO-9126 2004) and (IEEE-830 1998) and (ISO-9126 2004) standards. This section identifies which standards currently address some aspects of the software-FUR derived from the safety system-NFR.

3.13.1 ECSS: views and concepts for safety

The ECSS present safety as an NFR for real-time and embedded software – see Figure 3.12 in these standards, the safety requirements are described as system states where an acceptable level of risk is not exceeded with respect to fatality, injury or occupational illness, damage to launcher hardware or launch site facilities, damage to an element of an interfacing manned systems, etc.

According to (ECSS-Q-ST-40C 2009), safety requirements shall be identified and traced from the system level into the design and then allocated to the lower levels; furthermore, the identified safety requirements shall be justified in the design and presented in an appropriate document.

The (ECSS-Q-ST-40C 2009) describes the mandatory aspects for safety requirements of a system safety programme to ensure that all safety risks associated with the design, development, production and operations of space product are adequately identified, assessed, minimized, controlled and finally accepted through the implementation of a safety assurance programme.

The (ECSS-Q-ST-40C 2009) safety policy is applied by implementing a system safety programme, supported by risk assessment, which can be summarized as follows:

1. Hazardous characteristics (system and environmental hazards) and functions with potentially hazardous failure effects are identified and progressively evaluated by iteratively performing systematic safety analyses;
2. The potential hazardous consequences associated with the system characteristics and functional failures are subjected to a hazard reduction sequence whereby:
 - Hazards are eliminated from the system design and operations;
 - Hazards are minimized;
 - Hazard controls are applied and verified.
3. The risks that remain after the application of a hazard elimination and reduction process are progressively assessed and subjected to risk assessment, in order to:
 - Show compliance with safety targets;
 - Support design trade-offs;
 - Identify and rank risk contributors;
 - Support apportionment of project resources for risk reduction;
 - Assess risk reduction progress;
 - Support the safety and project decision-making process (e.g. waiver approval, residual risk acceptance).
4. The adequacy of the hazard and risk control measures applied is formally verified in order to support safety validation and risk acceptance;
5. Approval obtained from the relevant authorities.

3.13.2 ISO: views and concepts for safety

The (ISO-9126 2004) includes safety as a quality sub-characteristic to assess the level of risk of harm to people, business, software, property or the environment in a specified context of use. It includes the health and safety of the both the user and those affected by use, as well as unintended physical or economic consequences – see Table 3.12.

3.13.3 IEEE: views and concepts for safety

The (IEEE-1220 2007) defines safety specifications as equipment/system design features, performance specifications, and training that reduce the potential for human or machine errors or failures that cause injury or death within the constraints of operational effectiveness, time, and cost throughout the equipment/system life cycle.

It describes also the safety plan as the approach and methods for conducting safety analysis and assessing the risk to operators, the system, the environment, or the public.

The (IEEE-1220 2007) describes software safety as falling into one or more of the following categories:

1. Software whose inadvertent response to stimuli, failure to respond when required, response out-of-sequence, or response in combination with other responses can result in an accident.
2. Software that is intended to mitigate the result of an accident;
3. Software that is intended to recover from the result of an accident.

The set of key views in standards on safety requirements, as well as the set of concepts, terminology and vocabulary to describe safety requirements are presented in Table 3.12, including the following standards.

Table 3. 12 Safety: views, concepts and terms in the standards

ID	Standards organization	Key views	Concepts and terms
1	ECSS	Safety requirements shall be identified and traced from the system level into the design and then allocated to the lower levels	<ul style="list-style-type: none"> • Safety control software hazards • Safety levels of software integration • Critical software catastrophic • Safety software functions • Safety failure mechanism and • Safety switching of redundant items • Safety audit software
2	IEEE 1220	Safety is specifications on equipment/system design features, performance, and training that reduce the potential for human or machine errors that cause injury or death	<ul style="list-style-type: none"> • Safety failures within the constraints of operational effectiveness, time, throughout the equipment/system life cycle • Safety approach and methods • Safety analysis and assessing the risk to operators, system, environment, or public
3	ISO 9126	Assessing the level of risk of harm to people, business, software, property or the environment in a specified context of use	<ul style="list-style-type: none"> • User health and safety • Safety of people affected by use of the system • Economic damage • Software damage
4	IEEE 830	Not clear	<ul style="list-style-type: none"> • Check data integrity for critical variables
5	IEEE 1228	<p>Safety is a freedom from software hazards.</p> <p>Safety program is a systematic approach to reducing software risks</p>	<ul style="list-style-type: none"> • Safety related software • Software safety hazard • Safety critical software • Levels of software integrity

3.14 Resources systems requirements

This section presents a survey of the resources-related views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) and (ECSS-ESA 2005),

(ISO-9126 2004) and (IEEE-830 1998) standards to identify the resources foundation in system-NFR see- Table 3.13.

3.14.1 ECSS: views and concepts for resources

The ECSS present resources as a system-NFR for real-time and embedded software: in these standards, the resources requirements are described as what the component needs from its environment to perform its function with computer resources (such as: CPU load and maximum memory size) to be considered by the supplier.

The ECSS-E-40 indicates the computer hardware resource requirements on the utilization (e.g. processor capacity and memory capacity) available for the software item (e.g. sizing and timing) and computer software resource requirements on the software items to be used by or incorporated into the system (or constituent software product) (e.g. a specific real time operating system).

The (ECSS-Q-ST-40C 2009) describes all the resource requirements related to the software and the hardware requirements (target hardware on which the software is specified to operate), as follows:

1. List of the requirements relevant to hardware environment in which the software is specified to operate;
2. List of the sizing and timing requirements applicable to the software item under specification;
3. Description of the computer software to be used with the software under specification or incorporated into the software item (e.g. operating system and software items to be reused);
4. Description of the real time constraints to respect (e.g. time management with respect to the handling of input data before its loss of validity).

The (ECSS-S-ST-00C 2008) describes hardware resources by the assignable, addressable bus paths that allow peripheral devices and system processors to communicate with each other.

Hardware resources typically include I/O port addresses, interrupt vectors, and blocks of bus-relative memory addresses. Resources are assigned to each device node in the device tree (assuming that the represented device needs resources and those resources are available).

3.14.2 IEEE: views and concepts for resources

The (IEEE-1220 2007) indicates to collect measurements, tracked and reported at pre-established control points during each stage of development, to enable the quality system and achievement of efficient use of resources – see Table 3.13.

3.14.3 ISO: views and concepts for resources

The (ISO-9126 2004) define the resources as the capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions. ISO 9126 identifies the resources as part of product efficiency and describes ways to measure the software recourses through:

1. I/O resource devices;
2. Memory resources;
3. Transmission recourses.

Table 3.13 Resources: views, concepts and terms in standards

ID	Standard organization	key views	Concepts and terms
1	ECSS	The resource requirements related to the software and the hardware requirements (target hardware on which the software is specified to operate)	<ul style="list-style-type: none"> • CPU load • Maximum memory size • Computer hardware resource requirements <ul style="list-style-type: none"> – Processor capacity for the software items

Table 3.13 Resources: views, concepts and terms in standards (Continued)

ID	Standard organization	key views	Concepts and terms
1	ECSS	The resource requirements related to the software and the hardware requirements (target hardware on which the software is specified to operate)	<ul style="list-style-type: none"> – Memory capacity for the software items • Computer software resource requirements <ul style="list-style-type: none"> – Specific real time operating system – Software elements. • I/O port addresses • Interrupt vectors • Blocks of bus-relative memory addresses • I/O Resource List • I/O Resource Descriptor
2	IEEE-1220	Collected measurements, tracked, and reported at pre-established control points during each stage of development to enable a quality system and achievement of efficient use of resources	<ul style="list-style-type: none"> • Not Clear
3	ISO 9126	Capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions	<ul style="list-style-type: none"> • I/O resource devices • Memory resources • Transmission resources

3.15 Human factors system requirements

This section presents a survey of the human factors requirements views, concepts and terms in the ECSS (ECSS-S-ST-00C 2008; ECSS-E-ST-10C 2009; ECSS-Q-ST-80C 2009) and (ECSS-E-40-Part-1B 2003; ECSS-Q-80B 2003; ECSS-E-40-Part-2B 2005) standards– see Table 3.14.

3.15.1 ECSS: views and concepts for human factors

The identification of human factors in the ECSS standards is derived from an analysis of the requirements on the system and its functions. ECSS standards include the human factors as one of 16 NFR for the embedded and real time software. Human factors engineering (ergonomics) specifications, including those related to manual operations, human equipment interactions, constraints on personnel, and areas requiring concentrated human attention, that are sensitive to human errors and training.

The (ECSS-E-ST-10-11C 2008) forms part of the system engineering branch of the Engineering area of the ECSS system. As such it is intended to assist in the consistent application of human factors engineering to space products by specifying normative provisions for methods, data and models to ensuring of the safety, performance and problem avoidance in space system and payload operations. Moreover, This standard belongs to the human factors discipline, as identified in (ECSS-E-ST-10-11C 2008) , and defines the human factors engineering and ergonomics requirements applicable to elements and processes.

According to (ECSS-E-ST-10-11C 2008) the application of human factors (that in the space domain includes ergonomics) to systems design enhances effectiveness and efficiency, improves human working conditions, and diminishes possible adverse effects of use on human health, safety and performance. Applying ergonomics to the design of systems involves taking account of human capabilities, skills, limitations and needs.

A space system design will consider human factors and especially the two following main aspects from the very beginning of the conceptual phase. Firstly, the human being will be correctly taken into account in the design of the hardware, software and operations products and, secondly, the corresponding organization and training will be addressed in parallel to the design of the hardware and software.

For instance, ECSS standards provide a set of requirements for a human centered design process applied to a space system compatible with the (ISO-13407 1999): Human centered design processes for interactive systems. The incorporation of the human centered design into the overall project structure shall be initiated during the feasibility phase to avoid risk of late and costly redesign or incorrect human integration.

Human factors considerations in (ECSS-E-ST-10-11C 2008) relevant to meeting system performance and having safety implications include:

1. Human performance (e.g., human capabilities and limitations, workload, function allocation, hardware and software design, decision aids, environmental constraints, and team versus individual performance);
2. Training (e.g., length of training, training effectiveness, retraining, training devices and facilities, and embedded training);
3. Staffing (e.g., staffing levels, team composition, and organizational structure);
4. Personnel selection (e.g., minimum skill levels, special skills, and experience levels);
5. Safety and health aspects (e.g., hazardous materials or conditions, system or equipment design, operational or procedural constraints, biomedical influences, protective equipment, and required warnings and alarms).

Table 3.14 Human factors views, concepts and terms in ECSS

Key view	Concepts and terms in ECSS
human factors relevant to meeting system performance and having safety implications	<ul style="list-style-type: none"> • Performance of the human factors (Cognitive ergonomics) <ul style="list-style-type: none"> ▪ Human capabilities and knowledge profiles and boundaries such as: <ul style="list-style-type: none"> – Workload – Function Allocation – Hardware and Software Design – Decision Aids – Team versus Individual Performance ▪ Training <ul style="list-style-type: none"> – Length of Training – Training Effectiveness – Retraining – Training Devices and Facilities

Table 3.14 Human factors views, concepts and terms in ECSS (Continued)

Key view	Concepts and terms in ECSS
	<ul style="list-style-type: none"> – Embedded Training ▪ Staffing <ul style="list-style-type: none"> – Staffing Levels – Team Composition – Organizational Structure ▪ Personnel Selection <ul style="list-style-type: none"> – Minimum Skill Levels – Special Skills – Experience Levels • Safety of the human factors (Environmental ergonomics) <ul style="list-style-type: none"> ▪ Mechanical Safety ▪ Electrical Safety ▪ Environmental Safety ▪ Operational Safety ▪ Psycho/physiological Safety • Human interface factors <ul style="list-style-type: none"> ▪ Visual, audio or tactile cues and information on interface characteristics and task performance ▪ Interface customization ▪ Identification of safety related controls

3.16 Discussion and observation

3.16.1 ECSS standards

While conducting the survey of all non functional concepts and terms described in the ECSS-E-40 and ECSS-Q-series and in ECSS-ESA as the integrated standard for ECSS-E and ECSS-Q, it was observed that:

1. The various system-NFR are described differently, and at different levels of detail within the standards contents;
2. The various system-NFR are dispersed throughout the various documents: there is therefore, no integrated view of all types of candidate of non functional requirements;
3. There is no obvious link for each type of system-NFR in ECSS-ESA as the integrated standard and between all other ECSS standards that describe these requirements within their contents or within their different ECSS standards contents;

4. It is also to be noted that ECSS does not propose a way to measure such requirements and, without measurement, it is challenging to take such an NFR as a quantitative input to an estimation process or in productivity benchmarking.

3.16.2 IEEE standards

While conducting the survey of all NFR concepts and terms described in the IEEE standards, it was observed that

1. IEEE standards do not provide guidance on how to describe and specify most of the NFR in their list or on the NFR list in ECSS;
2. IEEE standards do not provide guidance on how to measure any of these NFR.

3.16.3 ISO 9126 standards

While conducting the survey of all quality concepts and terms described in the ISO 9126 standards, it was observed that:

1. The key view in the ISO 9126 series is from the perspective of the quality of the software product;
2. ISO 9126 presents ‘quality characteristics’, which are decomposed into quality sub characteristics and then into proposed derived measures to quantify those quality sub characteristics;
3. A large number of measures are proposed in ISO 9126, but none addresses software-FUR;
4. ISO 9126 does not use of these concepts at the system level or looking at what functions must be performed at the software level (i.e., FUR allocation to software) to implement these system level NFR.

3.16.4 ISO 19759 (SWEBOK guide)

While conducting the survey of all NFR concepts and terms described in the ISO 19759 standards, it was observed that:

1. The key view in the ISO 19759 is from the perspective of the software level, and some parts at the system level;
2. The D&I constraints and Configuration requirements are described in ISO 19759 differently at the software and system levels, and at different levels of details.

3.17 Summary

This chapter has presented a survey of the system-NFR views, concepts, and terms in the ECSS, ISO, and IEEE standards. It has identified which standards currently address aspects of the software-FUR derived from system-FUR and NFR.

The outcome of this chapter is the identification of the various elements that should be included in the design of a standard-based framework for specifying software-FUR for system-NFR.

In the work reported here, preference has been given to the views, concepts, and vocabulary most widely used by the industry, as evidenced in its standardization infrastructure, rather than those in the academic literature. Similarly, for the structuring and description of models of FUR and for measurement purposes, the measurement views, concepts, and terminology from the standardization infrastructure are adopted, rather than those in the literature.

CHAPTER 4

RELIABILITY: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR

4.1 Introduction

Currently, there exists no standard-based model of software-FUR for the identification and specification of system reliability NFR based on the various views of reliability documented in international standards. Consequently, it is challenging to measure these reliability-related software-FUR and take them into account quantitatively for estimation purposes.

The European ECSS series of standards for the aerospace industry includes reliability requirements as one of sixteen types of NFR for embedded and real time software. As presented in chapter 3, a number of reliability related concepts are dispersed throughout the ECSS, ISO 9126, and IEEE 830 standards to describe at varying levels of details the various types of candidate reliability requirements at the system, software, and hardware levels.

This chapter organizes these dispersed reliability concepts into a standard-based model of software-FUR for system reliability NFR. The availability of detailed standard-based model of software-FUR for system reliability NFR can facilitate the early identification and specification of the system reliability-NFR and their detailed allocation as specific reliability functions to be handled by the specified allocation to hardware or software or in a specific combination of both.

The approach adopted in this research for the structure of the standard-based model of software-FUR for system reliability NFR is based on the generic model of software-FUR proposed in the COSMIC (ISO-19761 2011) model, thereby allowing the measurement of the functional size of such reliability requirements allocated to software and taking them into account for estimations purposes.

This chapter focuses on a single type of NFR, that is, system reliability requirements, and reports on the work carried out to define an integrated view within a standard-based model of software-FUR for system reliability NFR.

The reliability-related views, concepts, and terms in the ECSS, ISO, and IEEE standards identified in Chapter 3 should be included in the design of standard-based model of software-FUR for system reliability NFR. The elements of reliability are dispersed in various system views throughout various ECSS standards and are expressed as either – see Figure 4.1:

- System reliability functional user requirements (system reliability FUR);
- System reliability non-functional requirements (system reliability NFR).

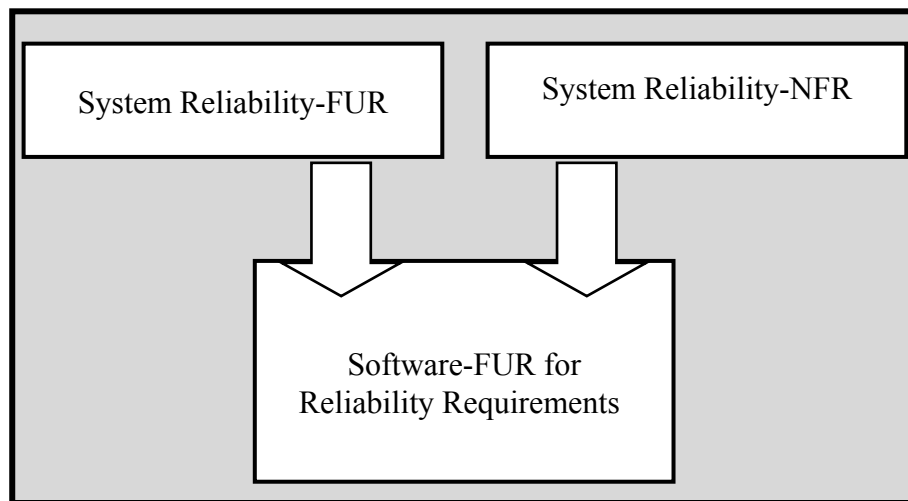


Figure 4.1 Mapping system requirements into software-FUR for reliability

The chapter is organized as follows. Section 4.2 presents a standard-based model of software-FUR for system reliability NFR. Section 4.3 presents a standard-based model of software-FUR for system reliability NFR using a service-oriented architecture (SOA). Section 4.4 presents the generic sizing of the standard-based model of software-FUR for system reliability NFR. Section 4.5 presents a measurement example. Finally, a summary is presented in section 4.6.

4.2 A standard-based model of software-FUR for system reliability NFR

The identified terminologies and concepts of reliability requirements in chapter 3 are mapped here into a proposed standard-based model of software-FUR for system reliability NFR.

4.2.1 Mapping reliability views and vocabulary from standards

Table 4.1 presents the system functions that are present either as system requirements in the ECSS standard or as reliability-related concepts in ISO 9126: each of these functions could be interpreted, and specified, as software-FUR.

Table 4.1 Reliability functions in ECSS, IEEE & ISO 9126

ID	Reliability Functions
1	Function to identify failure system tolerance
2	Function to identify fault recovery tolerance
3	Function to identify error data tolerance
4	Function to identify error to handle input
5	Function to identify error to produce output
6	Function to identify error to produce correct output
7	Function to identify fault prevention
8	Function to identify fault detection
9	Function to identify fault removal
10	Function to identify failure operation
11	Function to identify failure mechanism

Furthermore, various types of system-related reliability requirements can be derived from ISO 9126. Table 4.2 presents four (4) system reliability function types (left-hand side column) for system reliability requirements and corresponding software functions (middlecolumn) that may be specified to implement such reliability functions for the system reliability requirements.

Table 4.2 System reliability functions types and related software functions

ID	System reliability functions types	Software functions for reliability	System reliability requirements derived from ISO 9126
1	System reliability prediction (SRP)	Failure system tolerance function. Fault recovery tolerance function. Error data tolerance function.	System prediction tolerance
2	System reliability prediction failures (SRPF)	Failure operation function. Failure mechanism function.	System recoverability
3	System reliability prediction faults (SRPF1)	Fault prevention function. Fault detection function. Fault removal function.	System fault tolerance
4	System reliability prediction errors (SRPE)	Error to handle input function. Error to produce output function. Error to produce correct output function.	System maturity

4.2.2 Identification of the system reliability functional types allocated to software-FUR

This section identifies the four (4) function types and the relationships between these function types that may be allocated to software-FUR for system reliability.

System Reliability Prediction (SRP)

System reliability prediction (SRP) is used to predict the MTBF (mean time between failures) of items. The MTBF is determined by dividing the total cumulative operation hours for all fielded products by the number of system failures, error data and faults recovery occurrences. This is achieved by performing a prediction analysis method. The prediction analysis method can be used to define the quantitative parameters for components of a complete system. In this section according to ECSS standards, the system reliability prediction (SRP) allocated to software should be used a prediction algorithm that allows system architects to analyze the reliability of the system before it is built.

System modeling views for System Reliability Prediction (SRP)

Figure 4.2 illustrates a system modelling view of data movements for the system reliability prediction (SRP) (function type 1):

1. System reliability prediction (SRP): uses a prediction algorithm to exchange data movements between the failure system tolerance function (FSTF), the fault recovery tolerance function (FRTF) and the error data tolerance function (EDTF);
2. Failure system tolerance function (FSTF): exchanges data movements with other failures sub-system such as the failure operations and the mechanism functions in the system reliability prediction failure (SRPF) or function type 2;
3. Fault recovery tolerance function (FRTF): exchanges data movements with other faults sub-system such as the fault prevention and the detection and removal functions in the system reliability prediction fault (SRPF1) or function type 3;
4. Error data tolerance function (EDTF): exchanges data movements with other error sub-system such as the error to handle input and output functions in the system reliability prediction error (SRPE) or function type 4.

FSTF, FRTF and EDTF contact each other through intermediary services in order to deliver different types of data transfers (symbol \otimes in Figure 4.2).

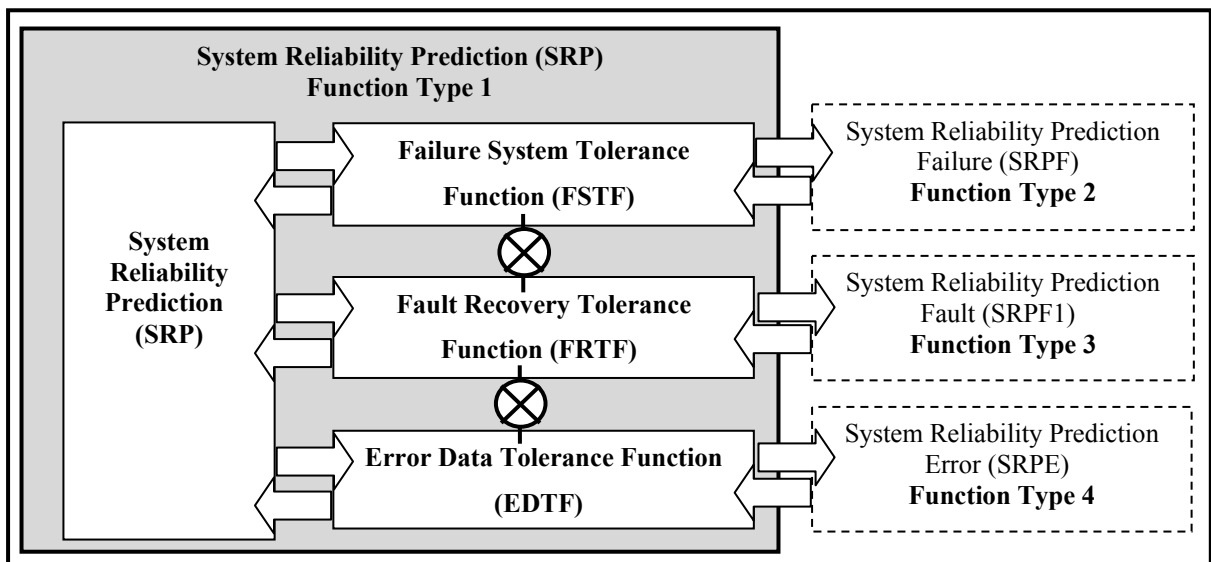


Figure 4.2 System Reliability Prediction (SRP): system modelling view

COSMIC modelling views for System Reliability Prediction (SRP)

Figure 4.3 illustrates a COSMIC modelling view of the data movements for the system reliability prediction (SRP) (function type 1) based on Figure 4.2:

1. SRP sends and receives a data group (i.e., Entry or Exit) to an FSTF, FRTF and EDTF;
2. FSTF, FRTF and EDTF send and receive data groups (i.e., Entry or Exit) to failures, faults and errors functions in function types 2, 3 and 4;
3. FSTF, FRTF and EDTF send and receive data groups (i.e., Entry and Exit) between each other using intermediary services (IS).

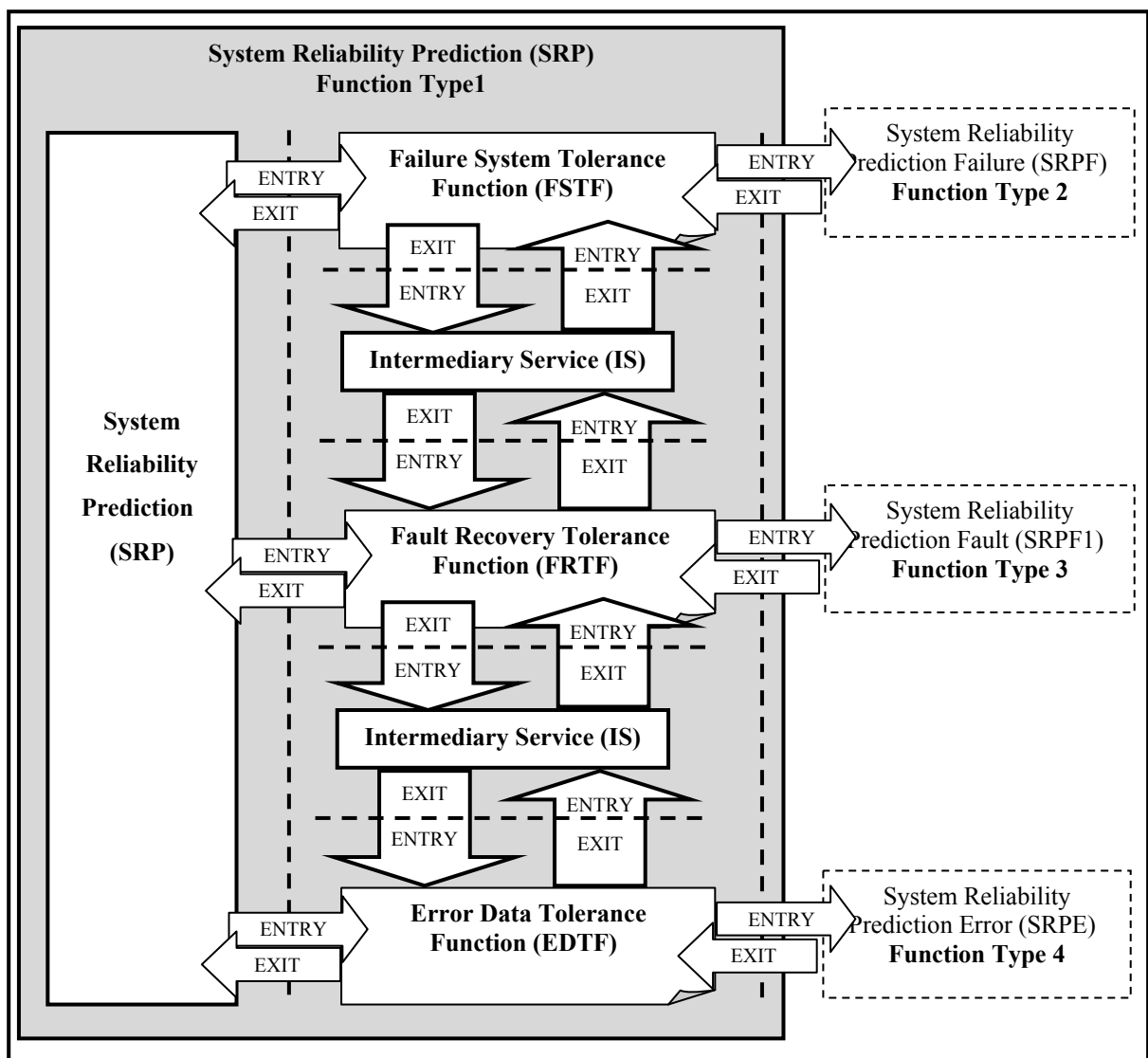


Figure 4.3 System Reliability Prediction (SRP): COSMIC modelling view.

System Reliability Prediction Failures (SRPF)

System reliability prediction failures (SRPF) usually depend on time, with the rate varying over the life cycle of the system. SRPF is divided into:

1. The Failure operation function (FOF) is defined as a particular way in which an equipment or machine failure can occur. The typical failure modes are: (1) premature operation, (2) failure to operate at the prescribed time, (3) failure to cease operation at the prescribed time, (4) failure during operation, and (5) degraded or excessive operational capability;
2. The Failure mechanism function (FMF) is defined by the means or methods by which a failure can be discovered by an operator under normal system operation or can be discovered by the maintenance crew by some diagnostic action.

System modelling views for System Reliability Prediction Failures (SRPF)

Figure 4.4 illustrates a system modelling view of data movements for the system reliability prediction failures (SRPF) (function type 2) which is divided into:

1. Failure operation function (FOF): exchange data movements with failure system tolerance function (FSTF) in function type 1, see- Figure 4.2;
2. Failure mechanism function (FMF): exchange data movements with failure system tolerance function (FSTF) in function type 1, see- Figure 4.2;

FOF and FMF contact each other through intermediary services in order to deliver different types of data transfers (symbol \otimes in Figure 4.4).

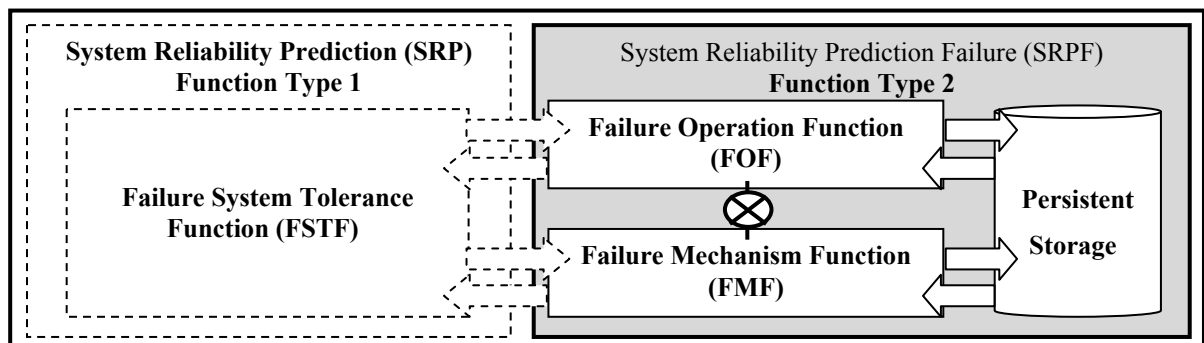


Figure 4.4 System Reliability Prediction Failures (SRPF): system modelling view

COSMIC modelling views for System Reliability Prediction Failures (SRPF)

Figure 4.5 illustrates a COSMIC modelling view of the data movements for the system reliability prediction failures (SRPF) (function type 2):

1. FOF and FMF read and write a data group (i.e., Read or Write) from a persistent storage;
2. FOF and FMF send and receive data groups (i.e., Entry or Exit) between each other using intermediary services.

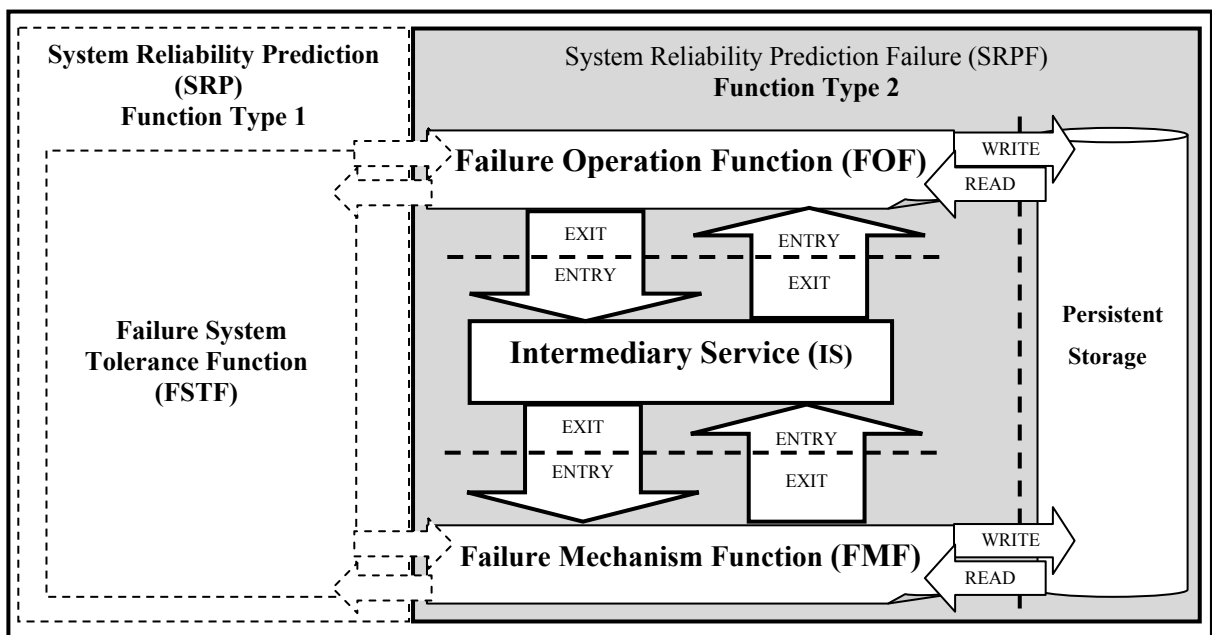


Figure 4.5 System Reliability Prediction Failures (SRPF): COSMIC modelling view

System reliability prediction Faults (SRPF1)

A system reliability prediction fault is defined by European standards as an abnormal condition or defect at the component, equipment, or sub-system level which may lead to a failure. A system reliability prediction fault is divided into fault prevention, detection and removal:

1. Fault Prevention deals with preventing faults being incorporated into a system. This can be accomplished by the use of development methodologies and good implementation techniques.

2. Fault detection and isolation is a subfield of control engineering which concerns itself with monitoring a system, identifying when a fault has occurred and pinpointing the type of a fault and its location.
3. Fault Removal can be sub-divided into two sub-categories: Removal during Development and Removal during Use. Removal during development requires verification so that faults can be detected and removed before a system is put into production. Once systems have been put into production a system is needed to record failures and remove them via a maintenance cycle.

System modelling views for System Reliability Prediction Faults (SRPF1)

Figure 4.6 illustrates a system modelling view of data movements for the system reliability prediction faults (SRPF1) (function type 3) which is divided into:

1. Fault prevention function (FPF): exchanges data groups with fault recovery tolerance function (FRTF) in a function type 1, see Figure 4.2;
2. Fault detection function (FDF): exchange data groups with fault recovery tolerance function (FRTF) in a function type 1, see Figure 4.2;
3. Fault removal function (FRF): exchange data groups fault recovery tolerance function (FRTF) in a function type 1, see Figure 4.2.

FPF, FDF and FRF contact each other through intermediary services in order to deliver different types of data transfers (symbol \otimes in Figure 4.6).

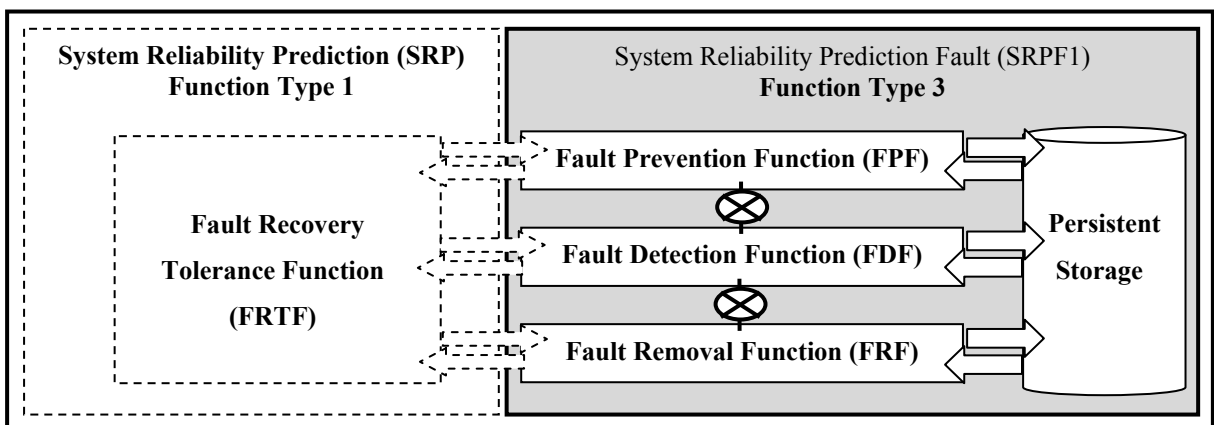


Figure 4.6 System Reliability Prediction Faults (SRPF1): system modelling view.

COSMIC modelling views for System Reliability Prediction Faults (SRPF1)

Figure 4.7 illustrates a COSMIC modeling view of the data movements for system reliability prediction faults (SRPF1) (function type 3):

1. FPF, FDF and FRF read and write a data group (i.e., Read or Write) from/to a persistent storage.
2. FPF, FDF and FRF send and receive data groups (i.e., Entry or Exit) between each other using intermediary services.

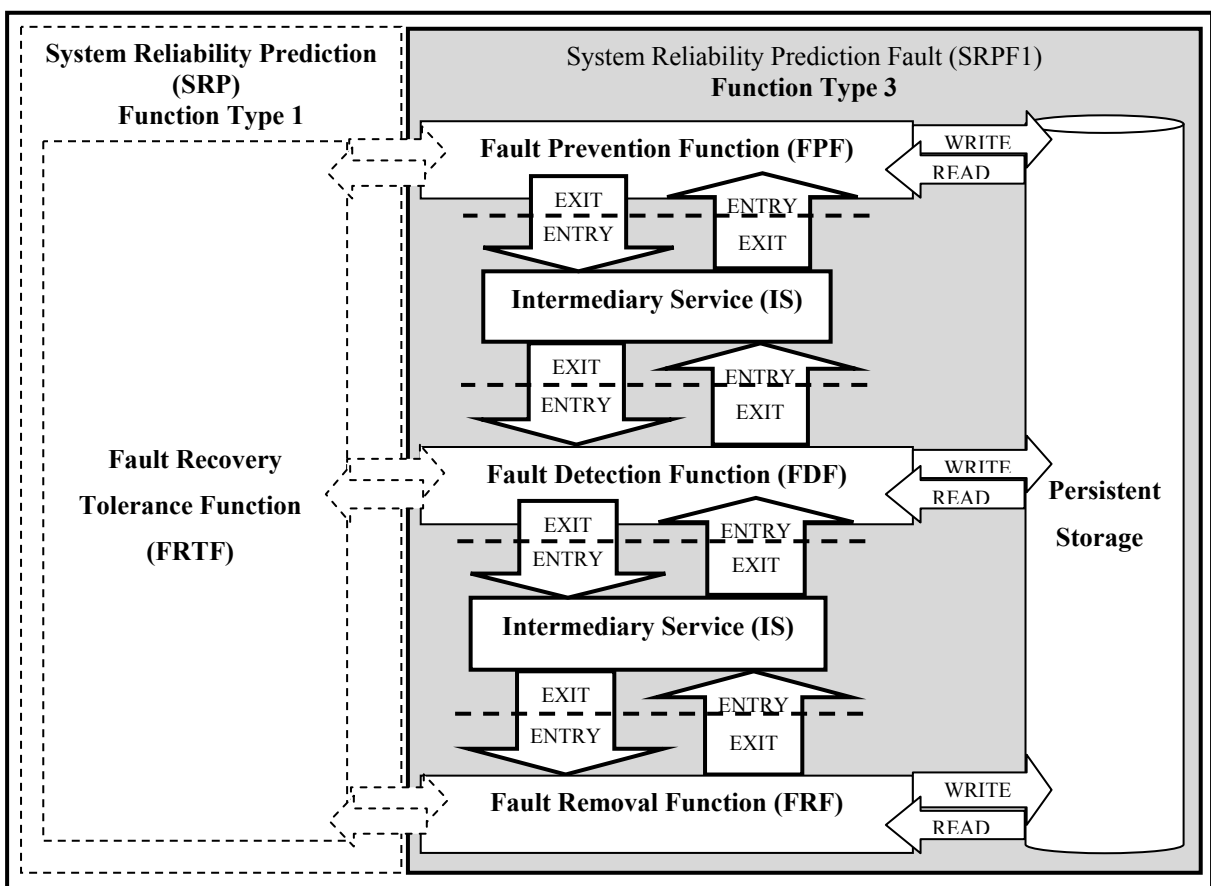


Figure 4.7 System Reliability Prediction Faults (SRPF1): COSMIC modelling view

System reliability prediction Errors (SRPE)

A system reliability prediction error is usually based on an algorithm model with a set of satisfaction conditions; in some cases this algorithm may give very bad parameter estimates

for systems not satisfying these conditions. This kind of problems frequently happens with reliability function errors due to the following reasons:

1. Function error to handle input in the reliability model;
2. Function error to produce output in the reliability model;
3. Function error to produce the correct output in the reliability model.

System modelling views for System Reliability Prediction Errors (SRPE)

Figure 4.8 illustrates a system modelling view of data movements for the system reliability prediction errors (SRPE) (function type 4) which can be divided into:

1. Error to handle input function (EHIF): exchanges data movement with error data tolerance function (EDTF) in a function type 1, see Figure 4.2;
2. Error to produce output function (EPOF): exchanges data movement with error data tolerance function (EDTF) in a function type 1, see Figure 4.2;
3. Error to produce correct output function (EPCOF): exchanges data movement with error data tolerance function (EDTF) in a function type 1, see Figure 4.2.

EHIF, EPOF and EPCOF contact each other through intermediary services in order to deliver different types of data transfers (symbol \otimes in Figure 4.8).

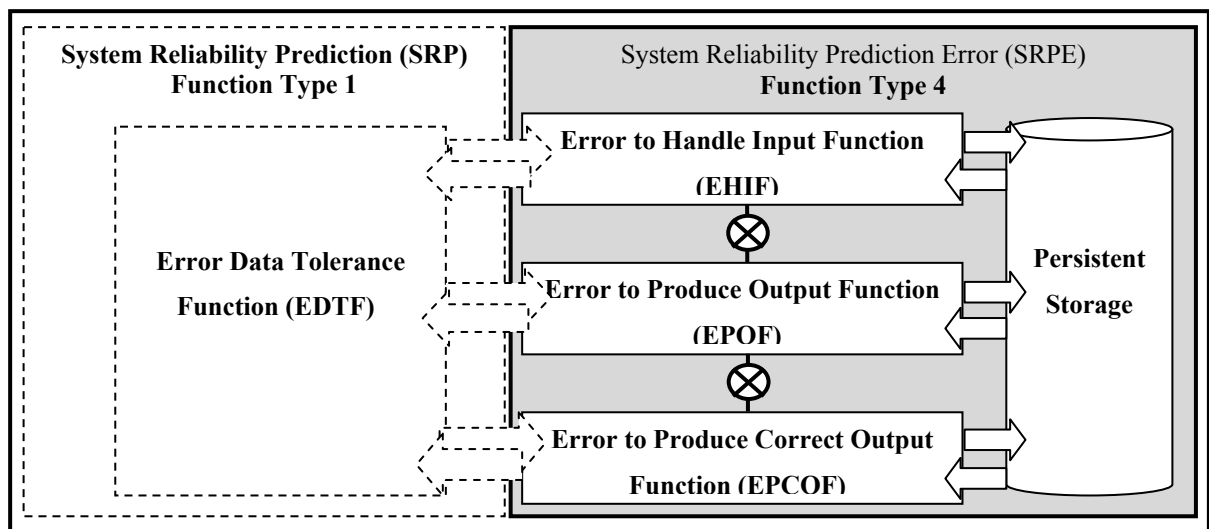


Figure 4.8 System Reliability Prediction Errors (SRPE): system modelling view

COSMIC modelling views for System Reliability Prediction Errors (SRPE).

Figure 4.9 illustrates a COSMIC modelling view of the data movements for system reliability prediction errors (SRPE) (function type 4):

1. EHIF, EPOF and EPCOF read and write a data group (i.e., Read or Write) from-to a persistent storage;
2. EHIF, EPOF and EPCOF send and receive data groups (i.e., Entry or Exit) between each other using intermediary services.

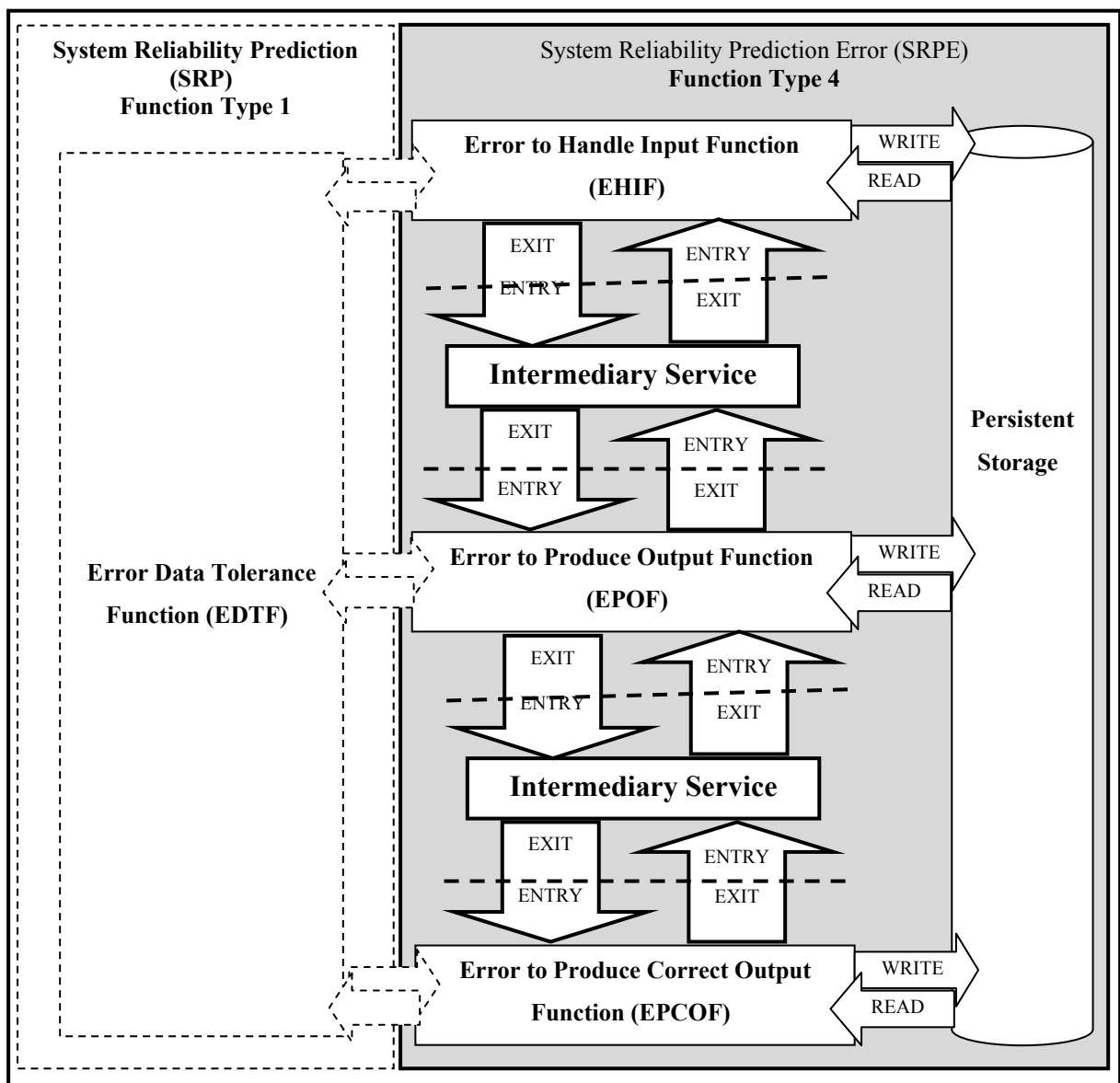


Figure 4.9 System Reliability Prediction Errors (SRPE): COSMIC modelling view

4.2.3 Model of the functions types relationships based on system views

Figure 4.10 presents an overview of the relationships between the function types for system reliability that may be allocated to software-FUR. More specifically, the system reliability requirements model is composed of 11 functions grouped into four function types. The data flows on the model are also divided into direct data flows and the intermediary data flows:

1. The SRP model (Function Type 1) can be used to specify the data flows between the three sub functions types and the data flows with the other functions on the system reliability model – see Figure 4.10;
2. The SRPF model (Function Type 2) can be used to specify the data flows between the two sub functions types and the data flows with other functions on the system reliability model. Function type 2 can be aligned with ISO 9126 on system recoverability– see Figure 4.10;
3. The SRPF1 model (Function Type 3) can be used to specify the data flows between the three sub functions and the data flows with other functions on the system reliability model. Function type 3 can be aligned with ISO 9126 on system fault tolerance – see Figure 4.10;
4. The SRPE model (Function Type 4) can be used to specify the data flows between the three sub functions and the data flows with other functions on the system reliability model. Function type 4 can be aligned with ISO 9126 on system maturity – see Figure 4.10.

4.2.4 Model of the functional types relationships based on COSMIC views

Figure 4.11 presents an overview of the relationships between the function types in the reliability software-FUR, using COSMIC for graphical representation. More specifically:

1. The SRP model can be used to specify and measure its functional size from the received/sent data groups from/to failure system tolerance function (FSTF), Fault recovery tolerance function (FRTF) and Error data tolerance function (EDTF) – see Figure 4.11;

2. The SRPF model can be used to specify and measure its functional size from the received/sent data groups from/to failure operation function (FOF) and failure mechanism function (FMF) – see Figure 4.11;
3. The SRPF1 model can be used to specify and measure its functional size from the received/sent data groups from/to fault prevention function (FPF), Fault detection function (FDF) and fault removal function (FRF) – see Figure 4.11;
4. The SRPE model can be used to specify and measure its functional size from the received/sent data groups from/to error to handle input function (EHIF), error to produce output function (EPOF) and error to produce correct output function (EPCOF) – see Figure 4.11.

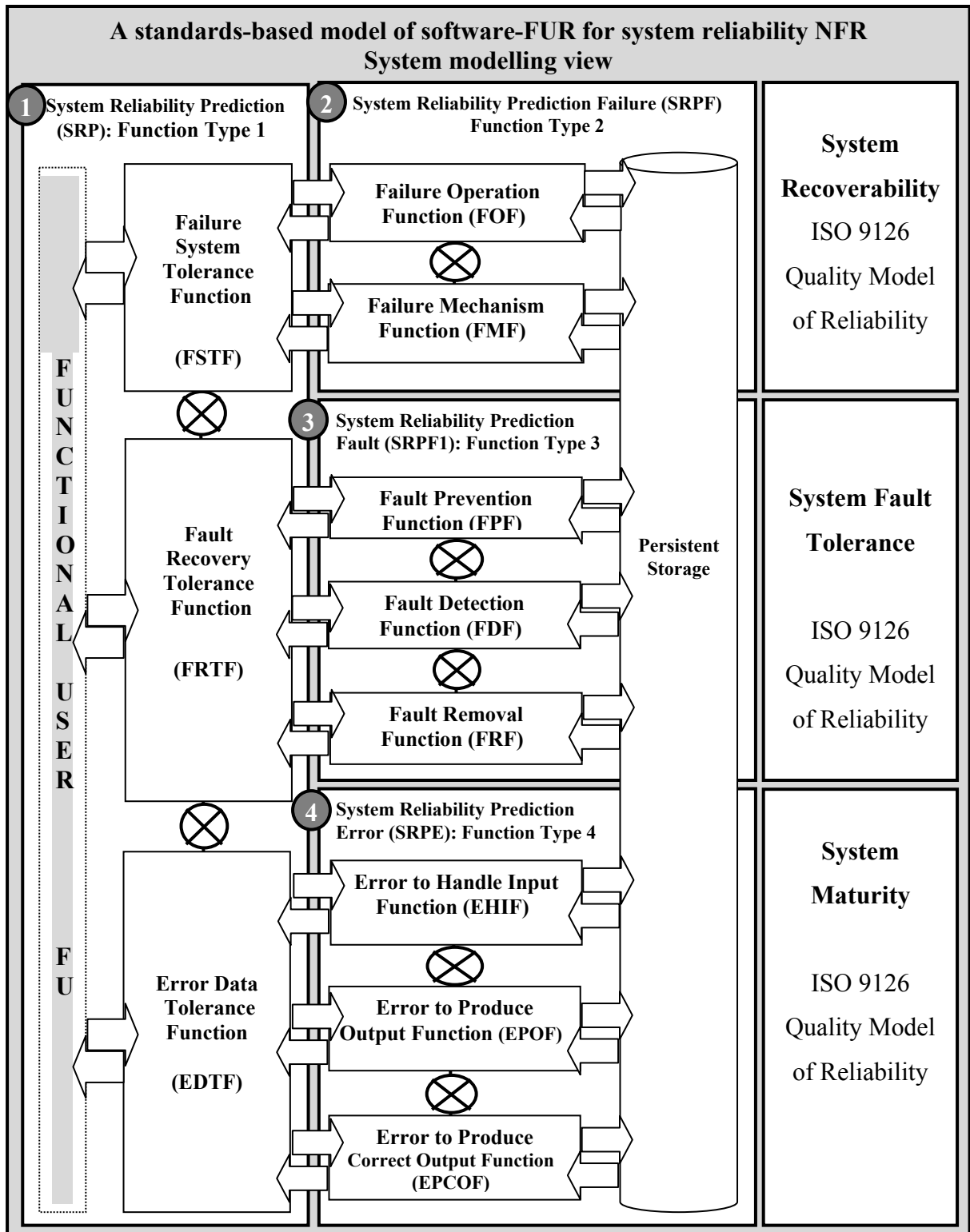


Figure 4.10 System modelling view for system reliability requirements

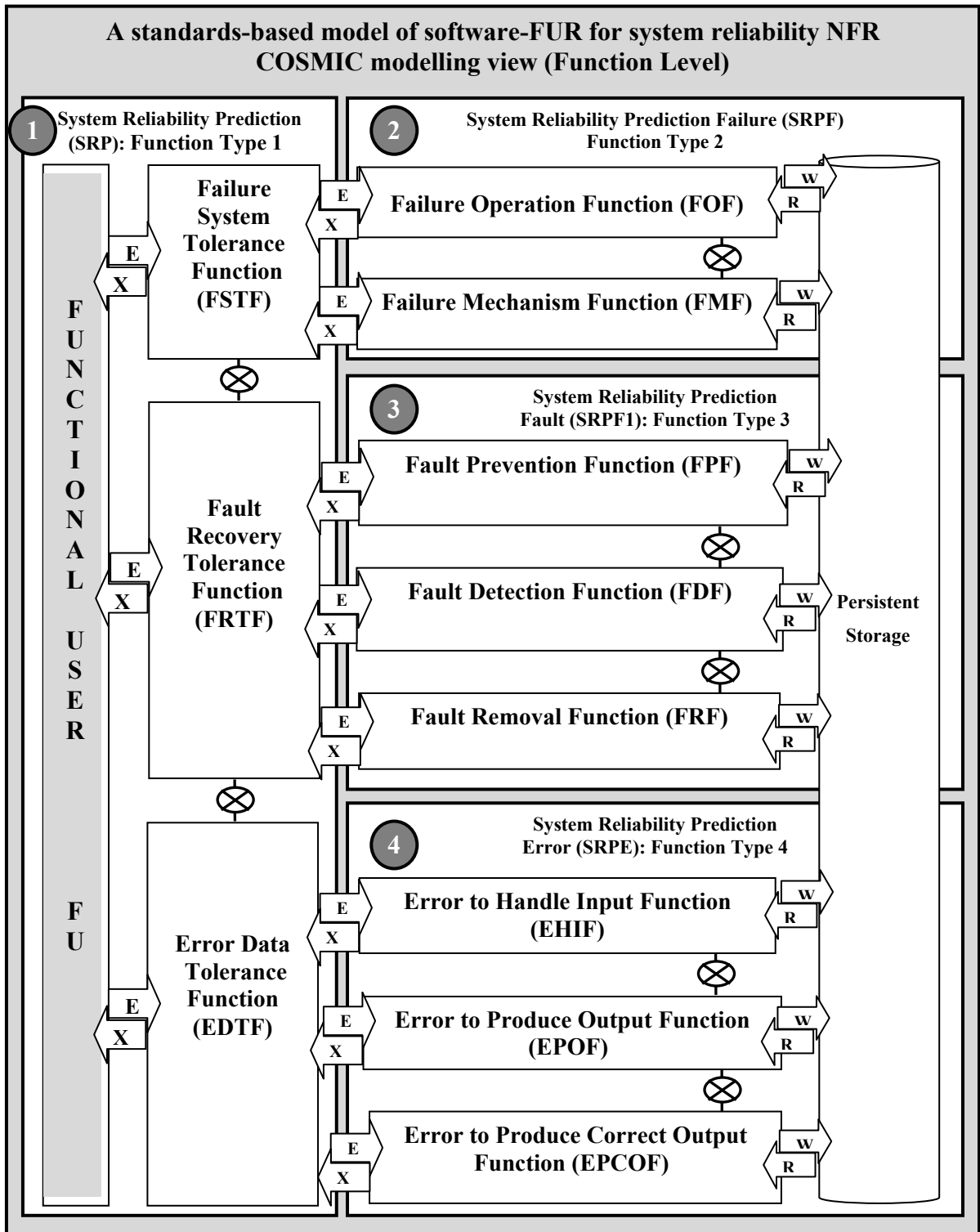


Figure 4.11 A standard-based model of software-FUR for system reliability NFR (Function Level)

4.3 A standard-based model of software-FUR for system reliability NFR using a COSMIC-SOA

Figure 4.11 illustrates the standard-based model of software-FUR for system reliability NFR. Figure 4.11 is considered a high-level model of requirements and describes the important concepts and relationships for system reliability requirements as defined in the ECSS international standards. In this section, the standard-based model of software-FUR for system reliability NFR using COSMIC-SOA is built in Figure 4.12 to elaborate on the model to show a more complete picture, which includes showing what is involved in instantiating the modeled entities in practice – for more details, see (COSMIC 2010). Figure 4.12 also describes the detailed measurement model which can be used to specify and measure the functionality at the service level.

The standard-based model of software-FUR for system reliability NFR using COSMIC-SOA in Figure 4.12 provides an integrated suite of services that can be used in multiple business domains to measure the functional size of software-FUR in an COSMIC SOA environment. In this model, the term “service” refers to a set of related software-FUR functions. The COSMIC-SOA guideline offers three types of data movements architecture in Table 1.4 - see chapter 1.

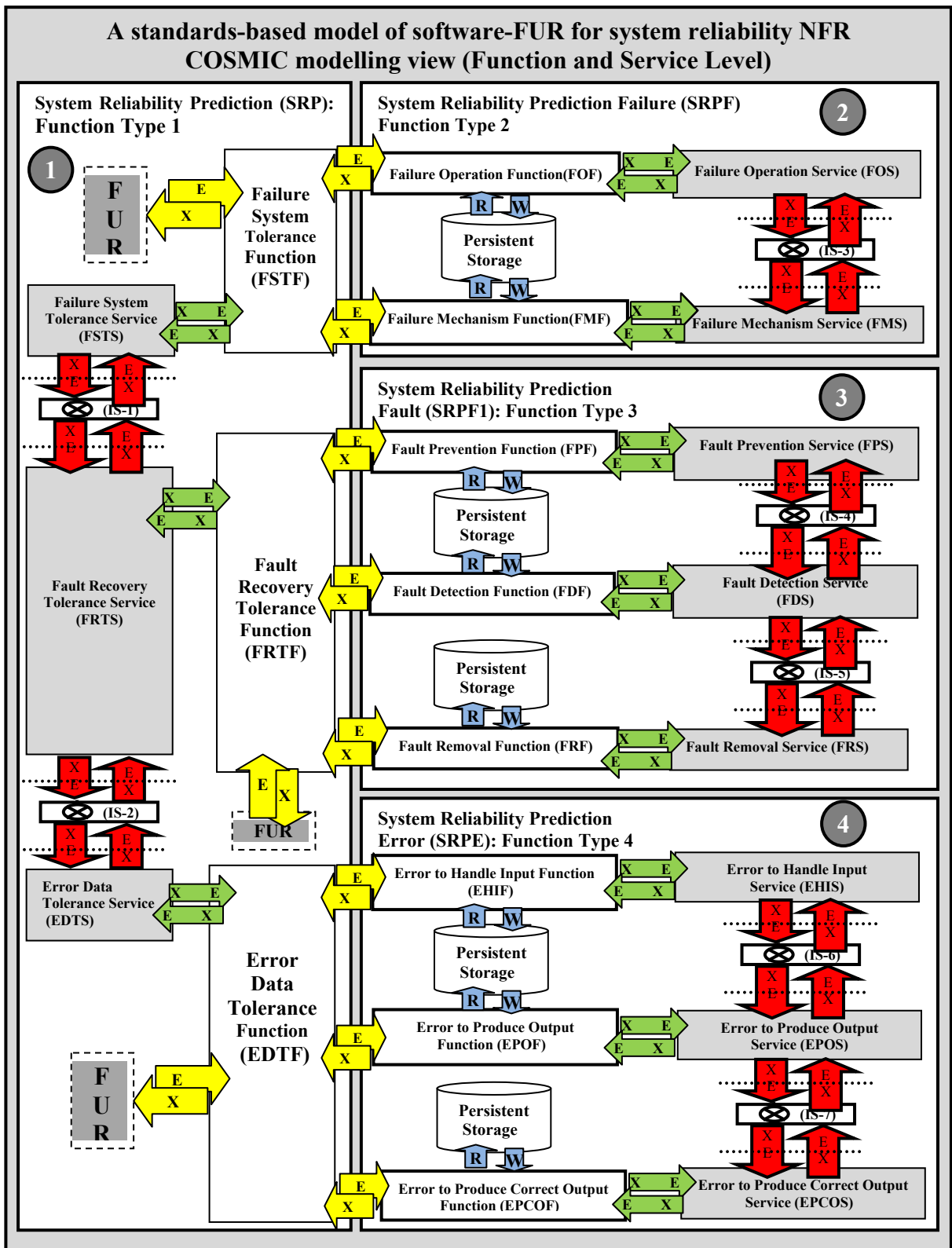


Figure 4.12 A standard-based model of software-FUR for system reliability NFR COSMIC modelling view (Function and Service Levels)

4.4 Sizing of the standard-based model of software-FUR for system reliability NFR

The specification of the standard-based model of software-FUR for system reliability NFR in any specific project is a specific instantiation of the proposed model described in Figure 4.12. When the software specification document is at the level of the movements of data groups, then these functional requirements can be directly measured using the COSMIC measurement rules. The measurement example presented next is illustrative of an instantiation of the standard-based model of software-FUR for system reliability NFR in an SOA context for a single data group for all the identified possible flows of data groups.

The measurement example in this section explains how to use the proposed standard-based model of software-FUR for system reliability NFR to size an hypothetical model composed of all of the kinds of software-FUR for system reliability-NFR.

4.4.1 Measurement of exchange messages for system reliability

There are eleven (11) functions types of system reliability, interacting with their own services, for the measurement of exchange messages in the standard-based model of software-FUR for system reliability NFR using COSMIC-SOA - see Figure 4.12. According to COSMIC-SOA guideline (COSMIC 2010), each functional process may interact with its own service by sending and receiving data movements (i.e., Entry and Exit).

Table 4.3 illustrates the measurement results for interactions between the system reliability functional processes with its own service processes i.e., the first line in Table 4.3 (Failure System Tolerance Function-FSTF) interacts with its own service process Failure System Tolerance Service-FSTS). For each interaction between each functional process with its own functional service process, the measurement result for this operation is equal to 4 CFP –see Table 4.4; the total measurement result is equal to 44 (see the green shaded arrows in Figure 4.12).

Table 4.3 Measurement of the exchange messages for the proposed model

Id. Of Functions	Types of Exchange Services for System Reliability		Quantity of Data Movements
	Application Functional Process	Service Functional process	
1	Failure System Tolerance Function (FSTF)	Failure System Tolerance Service (FSTS)	4
2	Fault Recovery Tolerance Function (FRTF)	Fault Recovery Tolerance Service (FRTS)	4
3	Error Data Tolerance Function (EDTF)	Error Data Tolerance Service (EDTS)	4
4	Failure Operation Function (FOF)	Failure Operation Service (FOS)	4
5	Failure Mechanism Function (FMF)	Failure Mechanism Service (FMS)	4
6	Fault Prevention Function (FPF)	Fault Prevention Service (FPS)	4
7	Fault Detection Function (FDF)	Fault Detection Service (FDS)	4
8	Fault Removal Function (FRF)	Fault Removal Service (FRS)	4
9	Error to Handle Input Function (EHIF)	Error to Handle Input Service (EHIS)	4
10	Error to Produce Output Function (EPOF)	Error to Produce Output Service (EPOS)	4
11	Error to Produce Correct Output Function (EPCOF)	Error to Produce Correct Output Service (EPCOS)	4
The Total of Data Movements			44 CFP

Table 4.4 Measurement example for the interactions between one application functional process and one service functional process

Application Functional Process	Service Functional process	Data Movement Description	Data Movement Type
Failure System Tolerance Function (FSTF)	Failure System Tolerance Service (FSTS)	FSTF sends a data group to FSTS	X
		FSTS receives a data group from FSTF	E
		FSTS sends a data group to FSTF	X
		FSTF receives a data group from FSTS	E
The Total of Data Movements			4 CFP

4.4.2 Measurement of intermediary services for system reliability

When a functional process service in Figure 4.12 requires data that is available via another functional process service, the former calls upon a functional process of the intermediary service. According to the standard-based model of software-FUR for system reliability NFR, the types of data movements for using the intermediary service must be an Entry and Exit.

Table 4.5 illustrates the measurement results for the intermediary services based on Figure 4.12 (see the red shaded arrows in Figure 4.12). This table presents an instantiation of a single data group for all possible flows of the data groups identified above, and listed as a data movement example for one intermediary service in Table 4.6. For this operation requirement the measurement results are equal to 8 CFP.

Table 4.5 Measurement of the intermediary services for the proposed model

Id. of Intermediary services	Types of Intermediary Services		Quantity of Data Movements
	Functional Service	Functional Service	
IS-1	Failure System Tolerance Service (FSTS)	Fault Recovery Tolerance Service (FRTS)	8
IS-2	Fault Recovery Tolerance Service (FRTS)	Error Data Tolerance Service (EDTS)	8
IS-3	Failure Operation Service (FOS)	Failure Mechanism Service (FMS)	8
IS-4	Fault Prevention Service (FPS)	Fault Detection Service (FDS)	8
IS-5	Fault Detection Service (FDS)	Fault Removal Service (FRS)	8
IS-6	Error to Handle Input Service (EHIS)	Error to Produce Output Service (EPOS)	8
IS-7	Error to Produce Output Service (EPOS)	Error to Produce Correct Output Service (EPCOS)	8
The Total of Data Movements			56 CFP

Table 4.6 COSMIC-SOA measurement example for the IS between functional Service

Intermediary Services		Data Movement Description	Data Movement Type
IS-1			
Failure System Tolerance Service (FSTS)	Fault Recovery Tolerance Service (FRTS)	FSTF sends a data group to IS-1	X
		IS-1 receives a data group from FSTF	E
		IS-1 sends a data group to FRTF	X
		FRTF receives a data group from IS-1	E
		FRTS sends a data group to IS-1	X
		IS-1 receives a data group from FRTF	E
		IS-1 sends a data group to FSTF	X
		FSTF receives a data group from IS-1	E
The Total of Data Movements			8 CFP

Note: IS-1 is the first intermediary service in Figure 4.12.

4.4.3 Measurement of the direct and indirect data movements for system reliability

This section is based on Figure 4.12 which illustrates the possible flows of data between components in the same layer, i.e., between peer components (where a component may be an application or a service). This section shows direct and indirect exchanges of data between components – one or both forms of which may be involved when services communicate. If components exchange data directly, the measurer will identify the Exit and/or Entry data movements, as per the data movements between service A and service B. An indirect exchange of data between components means that a service in one component writes data which are subsequently read by a service in another component. In this situation, the measurer will identify a Write data movement in the former component and a Read data movement in the other.

Specifically, Table 4.7 illustrates the measurement results for the exchange of data movements between the system reliability requirements model in a functional process or in service architecture layers – see Figure 4.12. This table presents an instantiation of this operation. The measurement results are equal to 38 CFP (see the yellow and blue shaded arrows in Figure 4.12).

Table 4.7 Measurements results for direct and indirect data groups

COSMIC-SOA Functions types	Data Movement Description	Data MovementType
Functional User (FU)	FU sends a data group to FSTF.	E
	FU sends a data group to FRTF.	E
	FU sends a data group to FDTF.	E
	FU receives a data group from FSTF	X
	FU receives a data group from FRTF	X
	FU receives a data group from FDTF	X
Failure System Tolerance Function (FSTF)	FSTF sends a data group to FOF	E
	FSTF sends a data group to FMF	E
	FSTF receives a data group from FOF	X
	FSTF receives a data group from FMF	X
Fault Recovery Tolerance Function (FRTF)	FRTF sends a data group to FPF	E
	FRTF sends a data group to FDF	E
	FRTF sends a data group to FRF	E
	FRTF receives a data group from FPF	X
	FRTF receives a data group from FDF	X
	FRTF receives a data group from FRF	X
Error Data Tolerance Function (EDTF)	EDTF sends a data group to EHIF	E
	EDTF sends a data group to EPOF	E
	EDTF sends a data group to EPCOF	E
	EDTF receives a data group from EHIF	X
	EDTF receives a data group from EPOF	X
	EDTF receives a data group from EPCOF	X
Failure Operation Function (FOF)	FOF reads and writes a data group from/to persistent storage.	R & W
Failure Mechanism Function(FMF)	FMF reads and writes a data group from/to persistent storage.	R & W
Fault Prevention Function (FPF)	FPF reads and writes a data group from/to persistent storage.	R & W
Fault Detection Function (FDF)	FDF reads and writes a data group from/to persistent storage.	R & W
Fault Removal Function (FRF)	FRF reads and writes a data group from/to persistent storage.	R & W
Error to Handle Input Function (EHIF)	EHIF reads and writes a data group from/to persistent storage.	R & W
Error to Produce Output Function (EPOF)	EPOF reads and writes a data group from/to persistent storage.	R & W
Error to Produce Correct Output Function (EPCOF)	EPCOF reads and writes a data group from/to persistent storage.	R & W
The Total functional size		38 CFP

4.5 Requirements and measurement examples

This section presents two specific measurement examples of the use of the standard-based model of software-FUR for system reliability NFR.

Example 1: Description of the requirements for a simple failure tolerance function (FTSF) to be allocated to software;

- Step 1: To implement the requirements for failure system tolerance function (FSTF) - see figures 4.2, 4.4 and 4.10, a software function must collect data of actual failure operations in the system (FOF) and data from the failure mechanisms (FMF) (a single or multiple failure mechanism requirement(s) must have been documented in the requirements at the system level, and allocated to the software – but is not described here for simplicity sake).
- Step 2: once the above FTSF requirements are detailed at a lower level of software requirements, the FTSF portion of the standard-based measurement model of the system reliability-NFR can be used for measuring the functional size of the functions allocated to Software-FUR.

Example 2: The set of functional requirements allocated to software for the system reliability requirements for a specific instantiation is the following (i.e. a subset of the full model in figure 4.2):

1. The functional user (FU) sends one data group to FSTF and another data group to FRTF;
2. The FSTF sends one data group to FOF and another data group to FMF.

Based on Figure 4.12 (arrows in yellow) and Table 4.7, the functional size measurement results are presented in Table 4.8 for the data movements identified by the measurer for this example. In this example, it is assumed for simplicity sake that there is a single data group involved in the requirements.

Table 4.8 Measurements results for direct and indirect data movements

COSMIC-SOA Functions	Data Movement Description	Data Movement Type
Functional User (FU)	FU sends a data group to FSTF.	E
	FU sends a data group to FRTF.	E
	FU receives a data group from FSTF	X
	FU receives a data group from FRTF	X
Failure System Tolerance Function (FSTF)	FSTF sends a data group to FOF	E
	FSTF sends a data group to FMF	E
	FSTF receives a data group from FOF	X
	FSTF receives a data group from FMF	X
Failure Operation Function (FOF)	FOF reads and writes a data group from/to persistent storage.	R & W
Failure Mechanism Function(FMF)	FMF reads and writes a data group from/to persistent storage.	R & W
The Total Functional Size		12 CFP

4.6 Summary

This chapter has introduced the standard-based model of software-FUR for system reliability NFR for specifying and measuring software requirements for the functions needed to address the system's reliability requirements.

The main contribution of this chapter is our proposed standard-based model of software-FUR for system reliability NFR. This model can be considered as a kind of reference model for the identification of system reliability requirements, and can be used for their allocation to software functions implementing such requirements. System requirements allocated to hardware have not been addressed in this chapter. Since the structure of the general model is based on the generic model of software adopted by the COSMIC measurement standard, the necessary information for measuring their functional size is readily available, and an example has been presented of a specific instantiation of this model. Specifically, the standard-based model of software-FUR for system reliability NFR presented in this chapter is based on:

- The ECSS standards for the description of the NFR for system reliability;
- The COSMIC measurement model of functional user requirements.

The proposed standard-based model of software-FUR for system reliability NFR is independent of the software type and the languages in which the software-FUR will be implemented. The proposed model provides:

- A specification model for each type, or all types, of reliability requirements: for example; the functional requirements to be allocated to software for the system reliability prediction.
- A specification measurement model for each type, or all types, of system reliability requirements allocated to software-FUR.

In the absence of such standard-based model of software-FUR for system reliability NFR, such NFR requirements are typically handled in practice much later on in the software development life cycle when, for example, at system testing time, users and developers find out that a number of reliability requirements have been overlooked and additional work has to be expanded to implement them.

CHAPTER 5

MAINTAINABILITY: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR

5.1 Introduction

Currently, there exists no standard-based model of software-FUR for system maintainability-NFR for the identification and specification of system maintainability requirements based on the various views documented in international standards and in the literature. Consequently, it is challenging to measure these maintainability-related software-FUR, and take them into account quantitatively for estimation purposes.

The ECSS includes maintainability requirements as one of sixteen (16) types of non functional requirement (NFR) for embedded and real time software. A number of maintainability related concepts are dispersed throughout the ECSS, ISO 9126 (ISO-9126 2004), and IEEE (IEEE-830 1998) standards to describe at varying levels of details the various types of candidate maintainability requirements at the system, software, and hardware levels.

This chapter organizes these dispersed maintainability concepts into a standard-based model of software-FUR for system maintainability-NFR. The availability and details of the model can facilitate the early identification and specification of the system maintainability-NFR and their detailed allocation as specific maintainability functions to be handled by the specified allocation to hardware or software or in a specific combination of both.

The approach adopted in this research for the structure of this model is based on the generic model of software-FUR proposed in the COSMIC – (ISO-19761 2011) model, thereby allowing the measurement of the functional size of such maintainability requirements allocated to software and taking them into account for estimations purposes.

This chapter focuses on a single type of NFR, that is, system maintainability requirements, and reports on the work carried out to define an integrated view of the standard-based model of software-FUR for system maintainability-NFR based on international standards, including the use of the generic COSMIC (ISO-19761 2011) model of software-FUR.

The maintainability related views, concepts and terms in the ECSS, IEEE, and ISO standards have been identified in chapter 3 and should be included in the design of the standard-based model of software-FUR for system maintainability-NFR. The elements of maintainability are dispersed in a number of system views throughout various ECSS standards, and are expressed as either – see Figure 5.1:

- System maintainability functional user requirements (system maintainability-FUR);
- System maintainability-NFR.

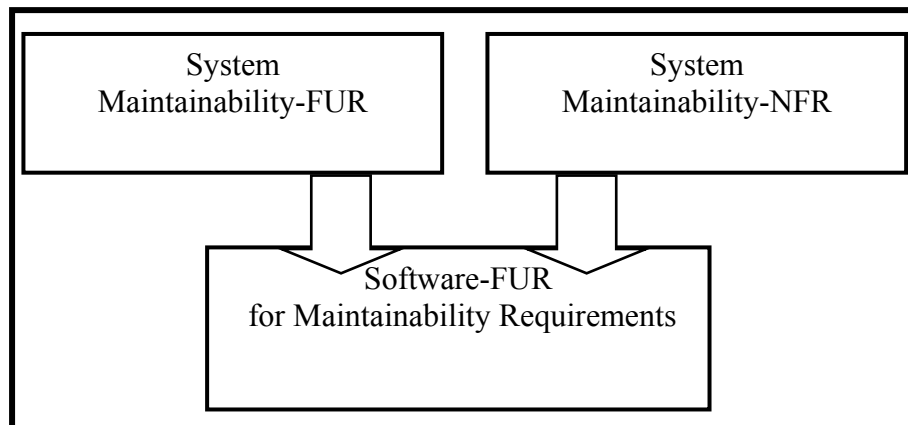


Figure 5.1 Mapping system-NFR to the maintainability FUR allocated to software

The chapter is organized as follows. Section 5.2 presents a standard-based model of software-FUR for system maintainability NFR. Section 5.3 presents a standard-based model of software-FUR for system maintainability NFR using a service-oriented architecture (SOA). Section 5.4 presents the sizing of a standard-based model of software-FUR for system maintainability NFR. Section 5.5 presents a measurement example. Finally, a summary is presented in section 5.6.

5.2 A standard-based model of software-FUR for system maintainability-NFR

The terminologies and concepts of maintainability identified in chapter 3 are mapped here into a proposed standard-based model of software-FUR for system maintainability NFR.

5.2.1 Mapping maintainability views, concepts, and terms from standards

During the mapping of maintainability requirements views and concepts from ISO 9126 and ECSS, it was observed that a high level of standard-based model of software maintainability requirements is defined by ISO, while a detailed (but disperse) view of system-maintainability requirements is provided by ECSS.

Table 5.1 presents the system maintainability requirements that are present either as system requirements in the ECSS standard or as maintainability-related concepts in ISO 9126. Each of these could be interpreted, and specified, at times as software-FUR.

Table 5.1 Maintainability requirements in ECSS & ISO 9126

ID	System Maintainability Requirements
1	Failure Data Operation
2	Failure Data Monitoring
3	Failure Data Control
4	System Failure Tasks
5	Failure Isolation
6	Failure Detection
7	Correct Data Faults
8	Correct System Defects
9	Fault Prevention of Data Control
10	Fault Prevention of System Functions
11	Fault Allocation Time

Furthermore, various types of system-related maintainability requirements can be derived from ISO 9126. Table 5.2 presents various procedures (middle column) associated with the system maintainability requirements and the corresponding software functions (right-hand

column) that may be specified to implement such procedures for the five types of system maintainability requirements.

Table 5.2 System maintainability requirements and related software functions

ID	System Maintainability in ISO 9126	System Maintainability Procedures	Software functions for maintainability
1	System Analyzability	System Maintainability Failure Procedure (SMFP)	<ul style="list-style-type: none"> • System Diagnostic Functions (SDF) • Failure Data Operation Function (FDOF) • Failure Data Monitoring Function (FDMF) • Failure Data Control Function (FDCF) • System Failure Tasks Function (SFTF)
2	System Analyzability & Changeability	System Registered Failures Procedure (SRFP)	<ul style="list-style-type: none"> • Failure Detection Function (FDF) • Failure Isolation Function (FIF)
3	System Changeability	System Malfunction Procedure (SMP)	<ul style="list-style-type: none"> • Correct Data Faults Function (CDFF) • Correct System Defects Function (CSDF)
4	System Stability	System Stability Procedure (SSP)	<ul style="list-style-type: none"> • Fault Prevention of Data Control Function (FPDCF) • Fault Prevention of System Function (FPSF)
5	System Testability	System Testability Procedure (STP)	<ul style="list-style-type: none"> • System Time Function (STF) • Fault Allocation Time Function (FATF)

5.2.2 Identification of system maintainability functions types allocated software-FUR

The section identifies the function types and the relationships between these function types allocated to software-FUR for system maintainability.

System Maintainability Failure Procedure (SMFP)

Figure 5.2 illustrates a system modeling view (i.e., a high-level view) of the data movements for the system maintainability failure procedure (SMFP) (Function Type 1) which is divided into:

1. The set of SDF constitutes a program or software written for the express purpose of examining the state of the hardware, or for locating problems with the hardware or operating system environment in/on which it is running. It sends data groups to the failure data operation, monitoring, and control functions (FDOF, FDMF, and FDCF) and to the system failure tasks function (SFTF);
2. The FDOF is the collection of failure activities required to operate the system diagnostic services and their execution. It reads about other services from stored information and writes their results on the system;
3. The FDMF keeps track of services in progress, and some information is provided from the FDOF results using intermediary services. It reads about other services from stored information and writes their results on the system;
4. The FDCF provides or assigns tasks, or brings about changes and verifies their service execution, to meet the deadlines and requirements. It reads about other services from stored information and writes their results on the system. The FDCF also uses intermediary services to connect the FDMF results, which provide some information;
5. The SFTF provides a complete description of a small unit of work. This description consists of two parts:
 - A data payload, which parameterizes the task;
 - Code, which implements the task.
 - SFTF reads about other services from stored information and writes their results on the system. In Figure 5.2, the intermediary services are represented by a cross in a small circle (\otimes).

The system maintainability failure procedure (SMFP) (Function Type 1) sends its results throughout the intermediary services to be used by the system stability procedure (SSP) (Function type 4) and system testability procedure (STP) (Function type 5).

The FDOF and FDMF in the system maintainability failure procedure (SMFP) (function type 1) send their results to FDF in the system registered failure procedure (SREP) (function type 2). The FDCF and SFTF in the system maintainability failure procedure (SMFP) (function type 1) send their results to FIF in the system registered failure procedure (SREP) (function type 2).

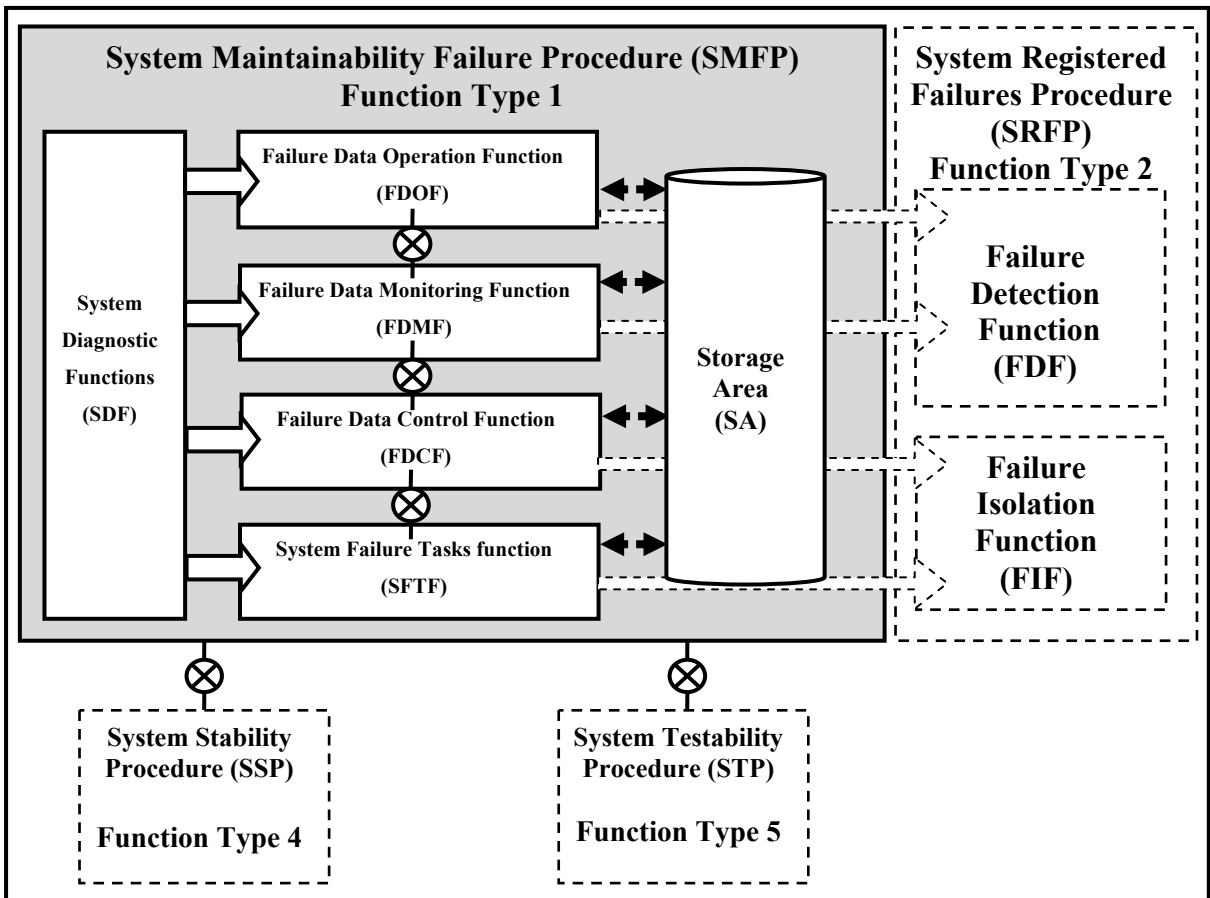


Figure 5.2 System Modeling View of a System Maintainability Failure Procedure (SMFP)

Figure 5.3 illustrates the COSMIC modeling scenario for the data movements for the System Maintainability Failure Procedure (SMFP).

The SDF sends a data group to the failure data operation, data monitoring, and data control functions and system failure tasks function (FDOF, FDMF, FDCF and SFTF):

1. The FDOF reads data groups about other services from stored information and writes their results as data movements on the system;
2. The FDMF reads data groups about other services from stored information and writes their results as data movements on the system;
3. The FDCF reads data groups about other services from stored information and writes their results as data movements on the system;
4. The SFTF reads data groups about other services from stored information and writes their results as data movements on the system.

The FDOF, FDMF, FDCF, and SFTF send and receive data groups to connect their functionality or service with one another by using intermediary services, which are represented by a cross inside a small circle \otimes – see Figure 5.3.

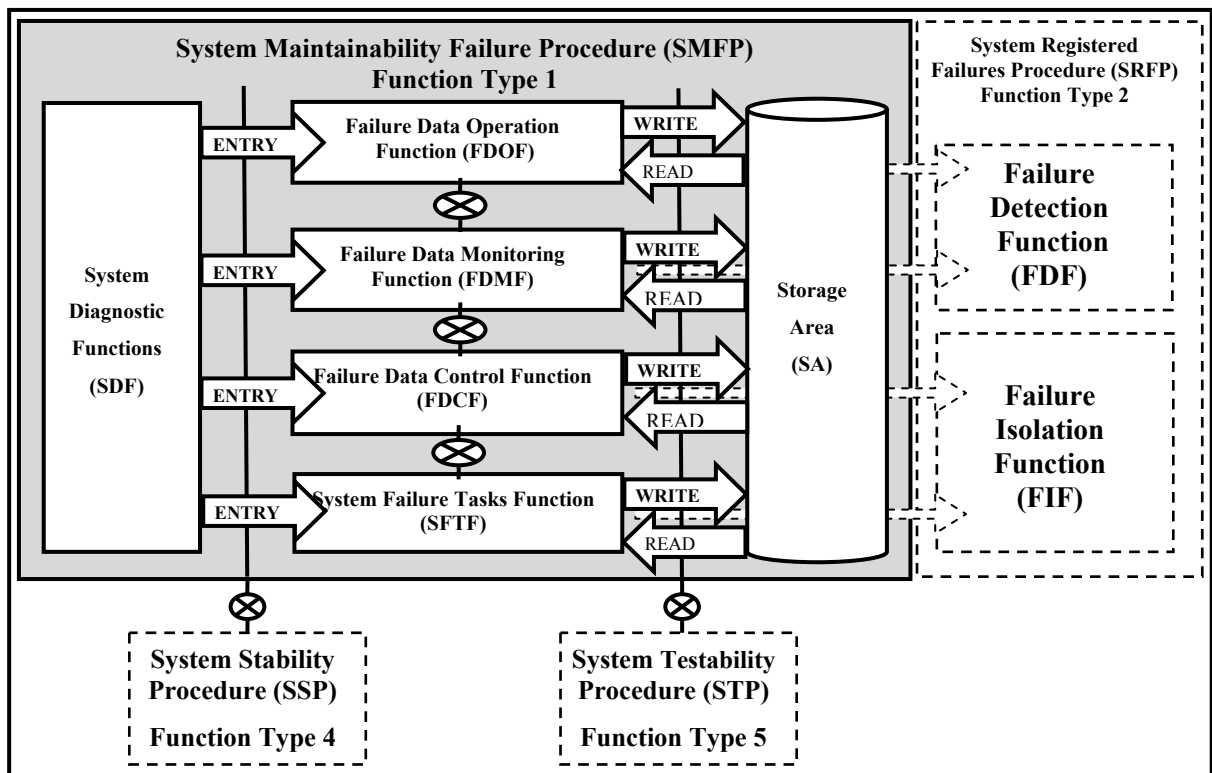


Figure 5.3 COSMIC Modeling View of a Maintainability Failures Procedure (SMFP) (i.e., with COSMIC data movements)

System Registered Failures Procedure (SRFP)

Figure 5.4 illustrates a system modeling view (i.e., a high-level view) of the data movements for the system registered failures procedure (SRFP) (Function Type 2):

1. The failure detection function (FDF) includes the ability of the system to detect and report a failure by saving the following results in the system:
 - The system correctly indicates a safe condition;
 - The system correctly indicates a malfunction requiring corrections;
 - The system erroneously indicates a safe condition in the event of a malfunction;
 - It provides information about data faults that could be occur.
2. The Failure Isolation Function (FIF) includes the ability of the system to identify the failure by saving the following results in the system:
 - System task operations cannot access data;
 - The modified data during a transaction that has not yet been completed;
 - The FIF provides information about system defects that could be occurred in the future.

The FDF and FIF contact each other through intermediary services to decide through different services which types of defects or faults can be appear in the system.

The FDF and FIF receive and send data movements from other function types in the maintainability model as follows:

1. The FDF receives its functionality based on the FDOF and FDMF results from System Maintainability Failure Procedure (SMFP) (Function Type 1) – see Figure 5.2;
2. The FIF receives its functionality based on the FDCF and SFTF results from System Maintainability Failure Procedure (SMFP) (Function Type 1) – see Figure 5.2;
3. The FDF sends its results to be used by the correct data faults function (CUFF) to system malfunction procedure (SMP) (Function type 3);
4. The FIF sends its results to be used by the correct system defects function (CSDF) to system malfunction procedure (SMP) (Function type 3).

5. The system registered failure procedure (SRFP) (Function Type 2) sends their results throughout the intermediary services to be used by the system stability procedure (SSP) (Function type 4) and system testability procedure (STP) (Function type 5);
6. In Figure 5.4, the intermediary services are represented by a cross in a small circle \otimes .

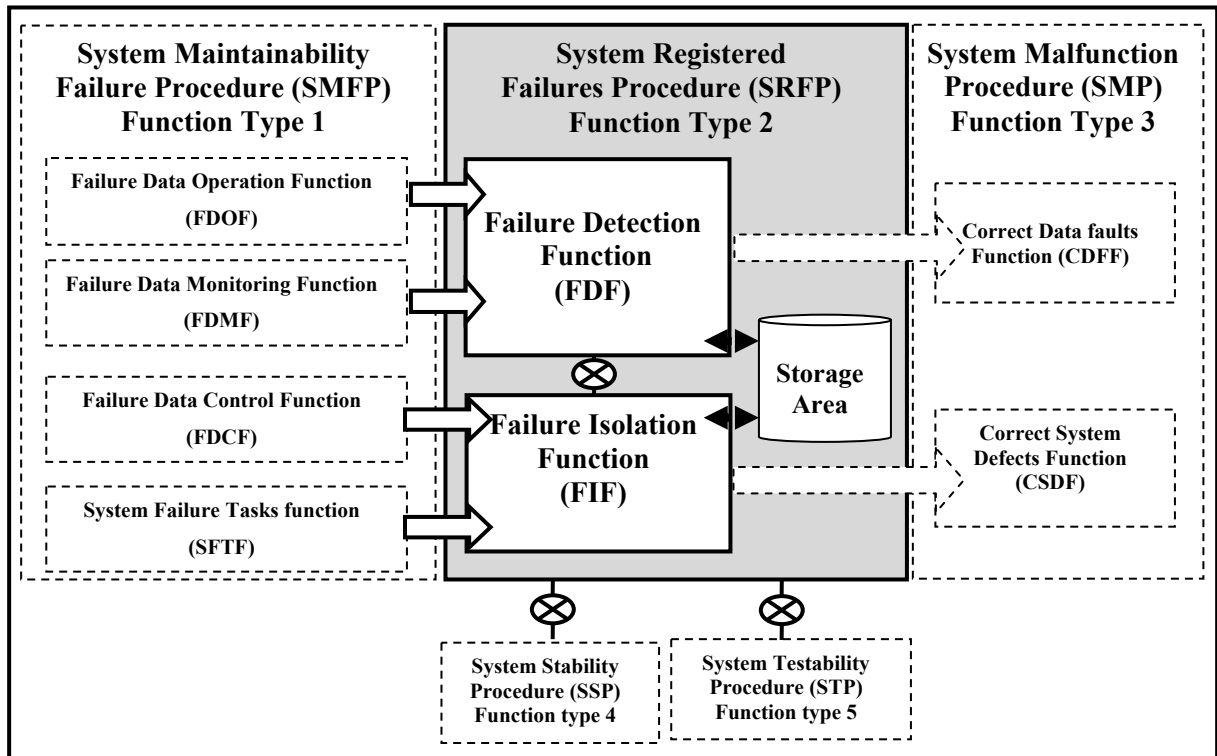


Figure 5.4 System Modeling Maintainability of the Registered Failures Procedure (SRFP)

Figure 5.5 illustrates a COSMIC modeling view of the data movements for the system registered failures procedure (SRFP) (Function Type 2):

1. The Failure Detection Function (FDF) receives a data group from the FDOF and FDMF.
2. The FDF reads and/or writes a data group to/from the storage area or system buffer.
3. The FIF receives a data group from the FDCF and SFIF.
4. The FIF reads and/or writes a data group to/from the storage area or system buffer.
5. The FDF and FIF contact each other by sending and receiving a data group using intermediary services.
6. In Figure 5.5, the intermediary services are represented by a cross in a small circle \otimes .

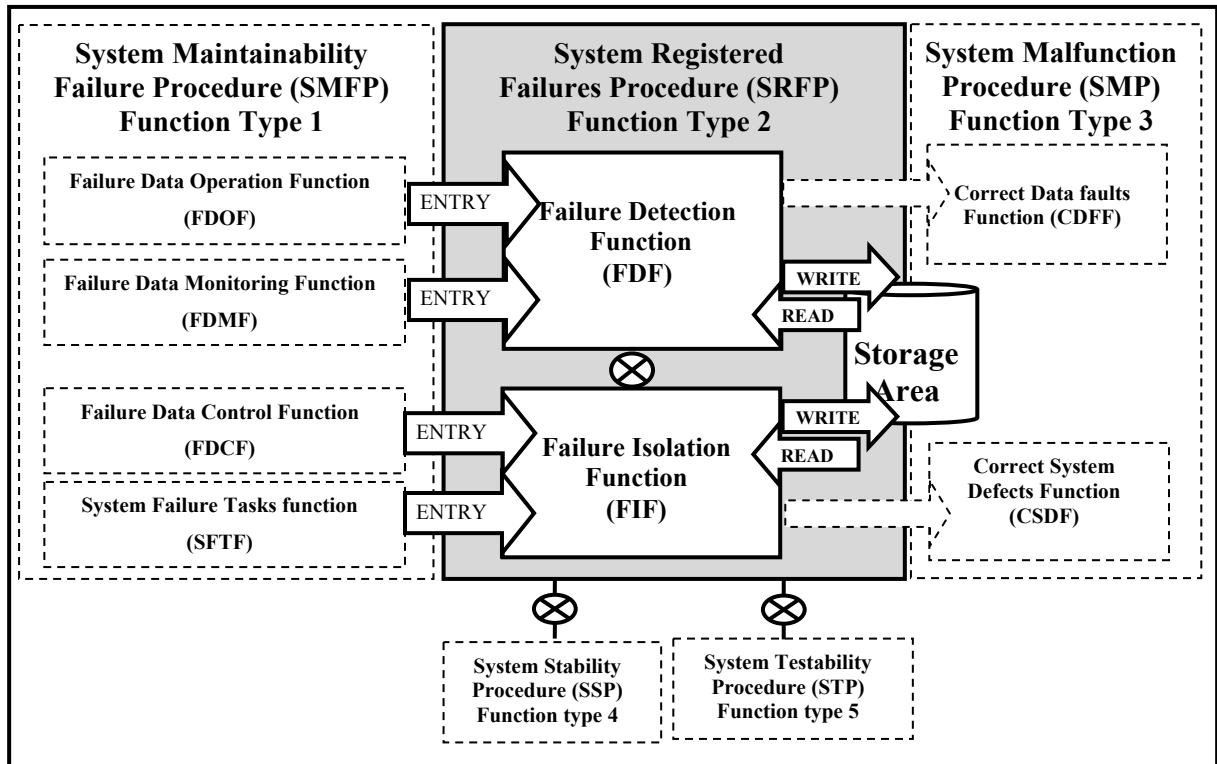


Figure 5.5 COSMIC Modeling View of the Maintainability Registered Failures Procedure (SRFP) (i.e., with COSMIC data movements)

System Malfunction Procedure (SMP)

Figure 5.6 illustrates a system modeling view (i.e., a high-level view) of the data movements for the maintainability system malfunction procedure (SMP) (Function Type 3):

1. The Correct Data Faults Function (CDF) is used when there is an abnormal condition at the component, equipment, or subsystem level, which may lead to failure in the functional unit or execution unit. The CDF provides information about asymmetric and symmetric data faults, a result which may be used by a next functionality in the maintainability systems requirements allocated to software;
2. The Correct System Defects Function (CSDF) is a functionality which is used when a reproducible or catastrophic malfunction occurs consistently under the same circumstances. It provides information about a failure of computer software to meet

requirements, a result which may be used by a next functionality in the maintainability systems requirements allocated to software.

The CDF and CSDF contact each other through intermediary services to decide through various services which type of defects or faults can be appear in the system.

The CDF and CSDF receive and send data movements from other function types in the maintainability model as follows:

1. The CDF receives its functionality based on the FDF and FIF results.
2. The CSDF receives its functionality based on the FIF results.
3. The CDF and CSDF send their results throughout the intermediary services to be used by the system stability procedure (SSP) (Function type 4) and system testability procedure (STP) (Function type 5).
4. In Figure 5.6, the intermediary services are represented by a cross in a small circle \otimes .

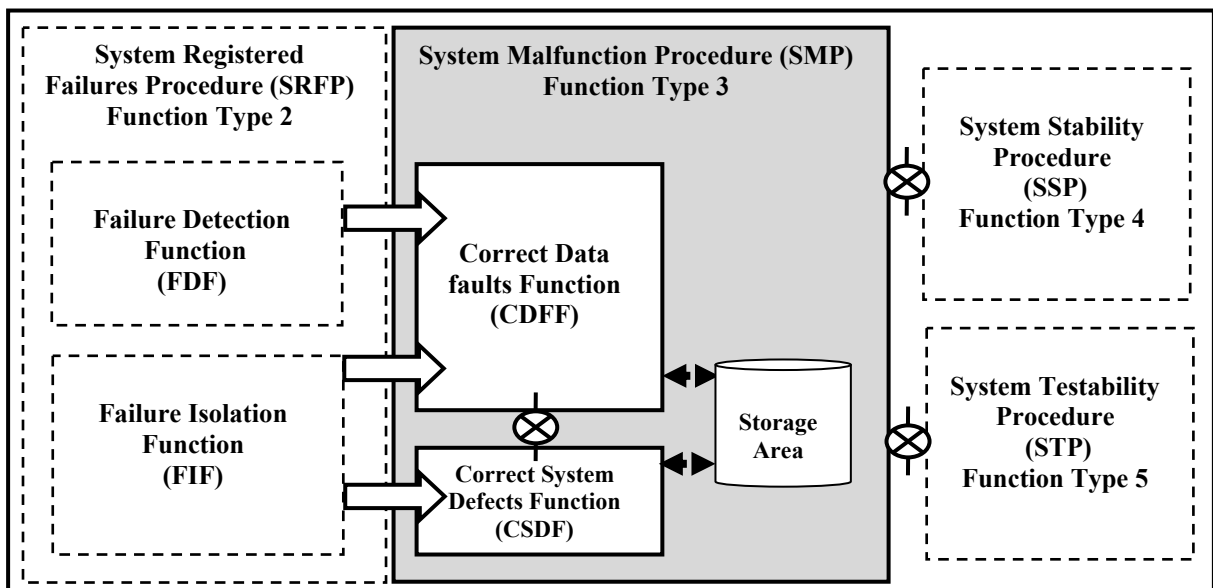


Figure 5.6 System Modeling View of a System Malfunction Procedure (SMP)

Figure 5.7, illustrates a COSMIC modeling view of the data movements for the maintainability system malfunction procedure (SMP) (Function Type 3):

1. The Correct Data Faults Function (CDF) receives data groups from the FDF and FIF;

2. The CDFF reads and/or writes a data group to/from a storage area or system buffer;
3. The Correct System Defect Function (CSDF) receives a data group from the FIF;
4. The CSDF reads and/or writes a data group to/from a storage area or system buffer;
5. The CDFF and CSDF contact each other by sending and receiving a data group using intermediary services;
6. The CDFF and CSDF in function type 3 send their results throughout the intermediary services to be used by the system stability procedure (SSP) (Function type 4) and system testability procedure (STP) (Function type 5);
7. In Figure 5.7, the intermediary services are represented by a cross in a small circle \otimes .

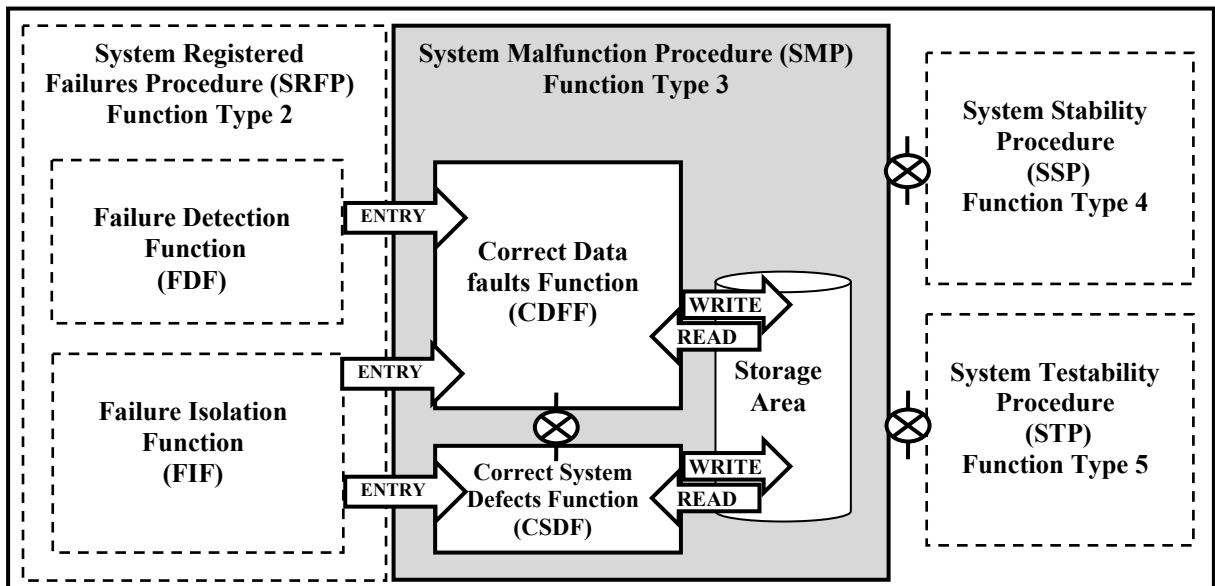


Figure 5.7 COSMIC Modeling View of a System Malfunction Procedure (SMP) (i.e., with COSMIC data movements)

System Stability Procedure (SSP)

Figure 5.8 illustrates a system modeling view (i.e., a high-level view) for the data movements for the system stability procedure (SSP) (Function Type 4):

1. The Fault Prevention of Data Control Function (FPDCF) is used when classifying the types of data faults being incorporated into a system. The FPDCF provides information about system data faults, a result which may be used by other functionalities, such as

SDF or system registered failures and system malfunctions in the maintainability systems allocated to software;

2. The Fault Prevention of System Function (FPSF) deals with preventing faults being incorporated into a system. The FPSF provides information about system faults, a result which may be used by other functionalities, such as SDF in the maintainability systems allocated to software.

The FPSF and FPDCF contact each other through intermediary services to provide the degree of system health. In Figure 5.8, the intermediary services are represented by a cross in a small (⊗).

The FPSF and FPDCF receive their functionality based on the results of the function types 1, 2, 3 and 5 using intermediary services.

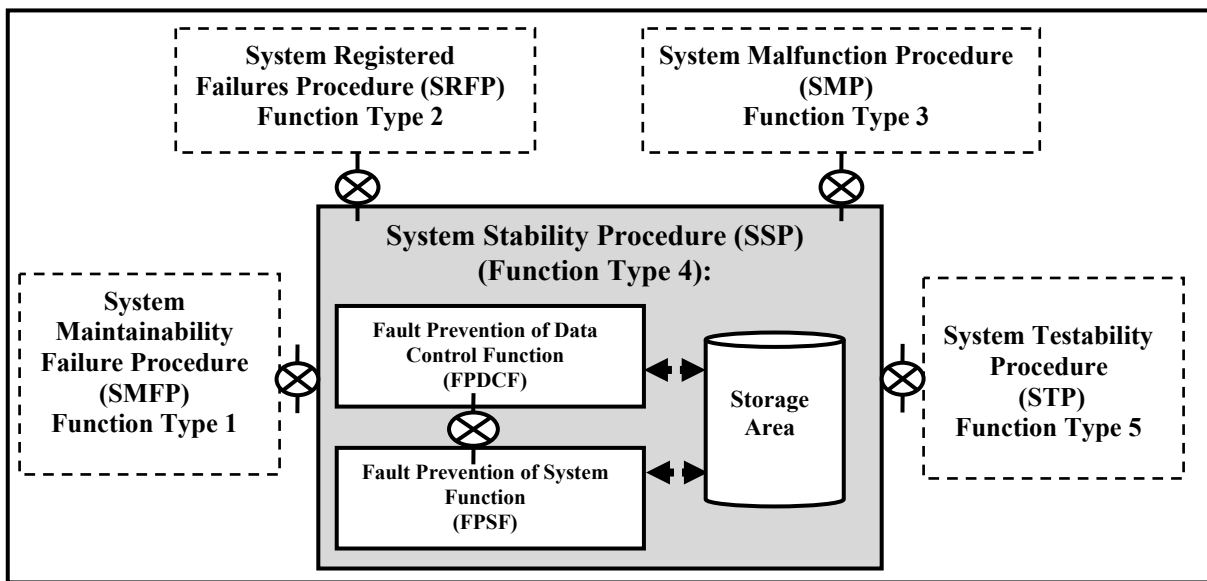


Figure 5.8 System Modeling View of System Stability Procedure (SSP)

Figure 5.9 illustrates a COSMIC modeling view of the data movements for the system stability procedure (SSP) (Function Type 4):

1. The FPDCF sending and receiving a data group using intermediary services from/to functions types 1, 2, 3 and 5. It reads and/or writes a data group to/from a storage area or system buffer;
2. The FPSF sending and receiving a data group using intermediary services from/to functions types 1, 2, 3 and 5. It reads and/or writes a data group to/from a storage area or system buffer.

The FPSF and FPDCF contact each other by sending and receiving a data group using intermediary services. In Figure 5.9, the intermediary services are represented by a cross in a small circle \otimes .

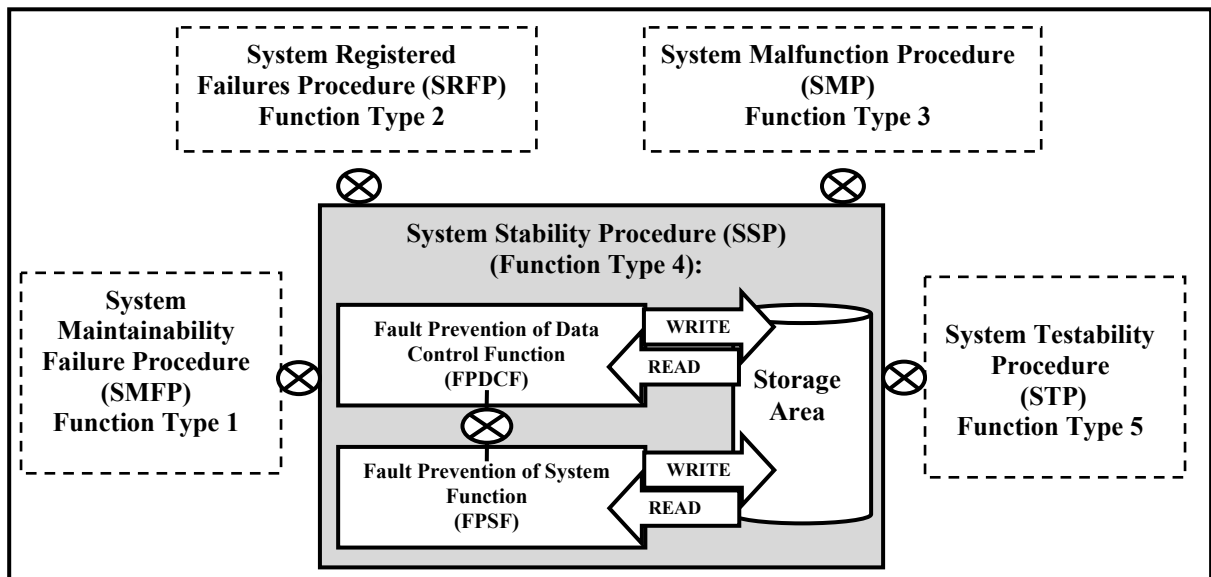



Figure 5.9 COSMIC Modeling View of a System Stability Procedure (SSP) (i.e., with COSMIC data movements)

System Testability Procedure (STP)

Figure 5.10, illustrates a system modeling view (i.e., a high-level view) of the data movements for the system testability procedure (STP) (Function Type 5):

1. The system time function (STF) is a system for describing points in time. It has two layers: the first encodes a point in time as a real number for each event in the system, and

- the second encodes that number as a sequence of bits or in another form. The STF provides time information about when the maintainability failure procedure occurred and the time of registering the failure for each event. This result may be used by other functionalities, such as the system stability function in the maintainability systems allocated to software;
2. The Fault Allocation Time Function (FATF) is used to provide both the execution time and the required memory for each event in the memory. It provides information about when the maintainability failure occurred: the time of registered failure for each event, and, when a system malfunctions, the faults or defects that occurred. Its result may be used by other functionalities, such as the system stability function in the maintainability systems allocated to software.

The STF and FATF contact each other through intermediary services. In Figure 5.10, the intermediary services are represented by a cross in a small circle .

The STF and FATF receive their functionality based on the results of function types 1, 2, and 3 in this model.

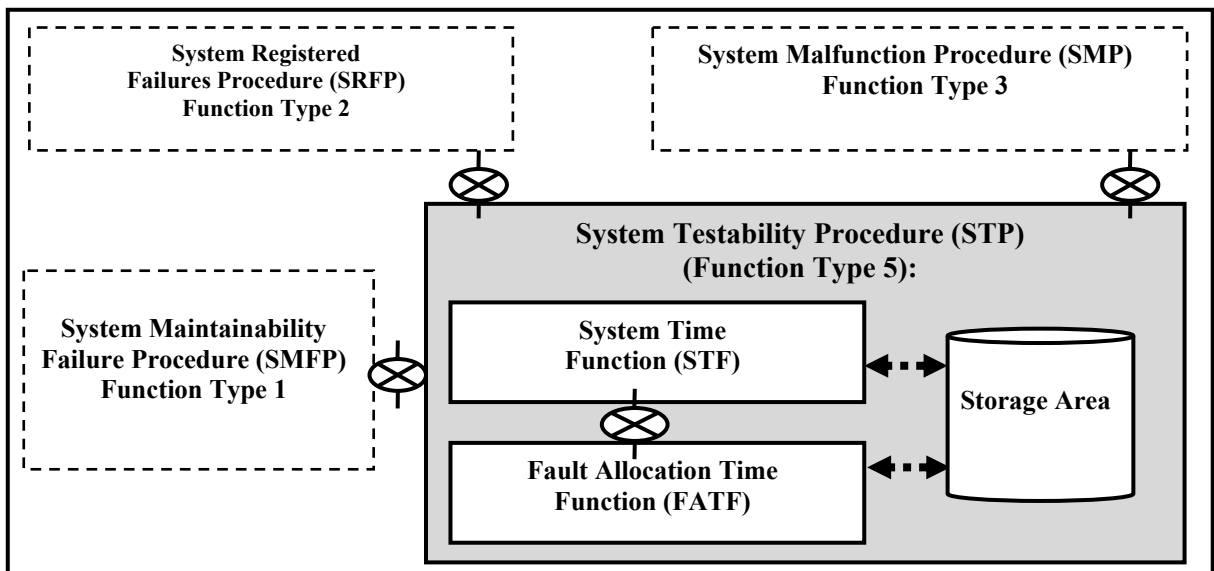


Figure 5.10 System Modeling View of the System Testability Procedure (ST)

Figure 5.11, illustrates a COSMIC modeling view of the data movements for the system testability procedure (STP) (Function Type 5):

1. The STF receives a data group from function types 1, 2 and 3 in the maintainability model using intermediary services. It reads and/or writes a data group to/from a storage area or system buffer;
2. An FATF receives a data group from function types 1, 2 and 3 in the maintainability model using intermediary services. It reads and/or writes a data group to/from a storage area or system buffer.

The STF and FATF contact each other by sending and receiving a data group using intermediary services. In Figure 5.11, the intermediary services are represented by a cross in a small circle \otimes .

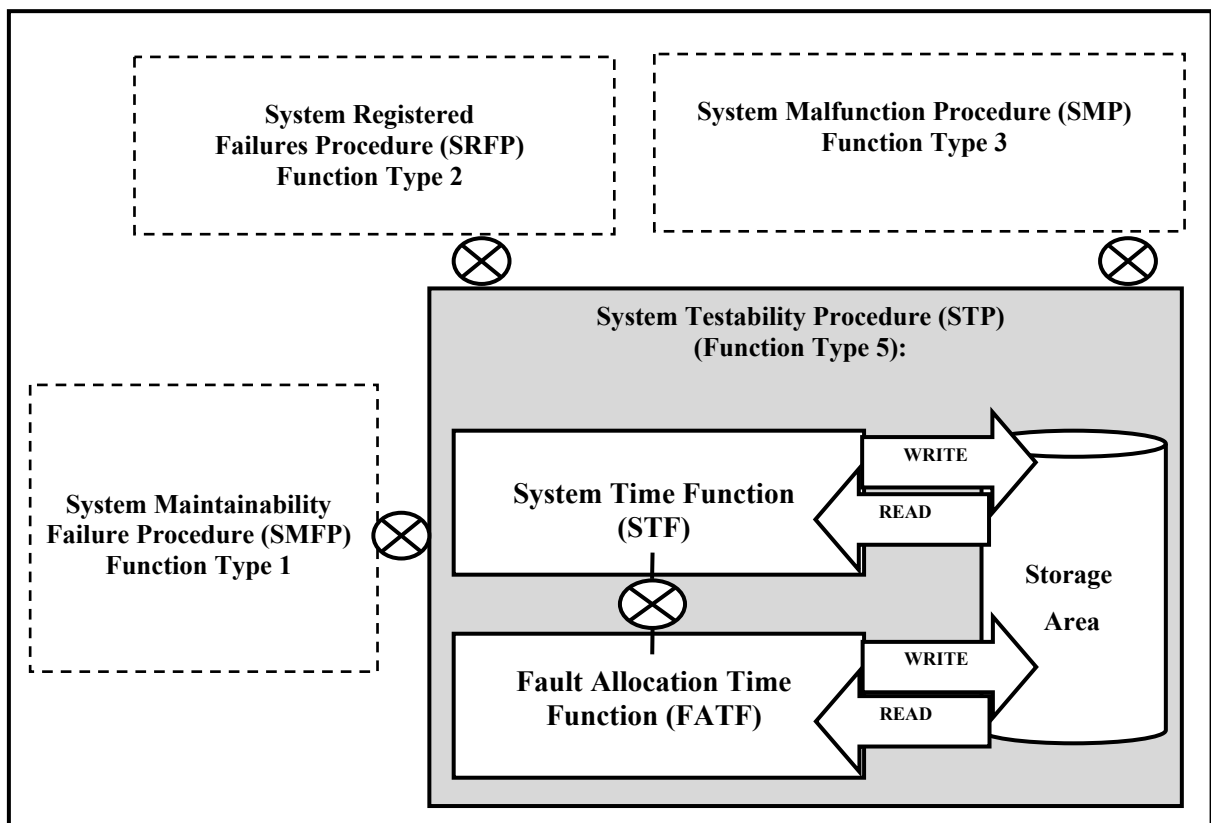


Figure 5.11 COSMIC Modeling View of a System Testability Procedure (ST) (i.e., with COSMIC data movements)

5.2.3 Model of the functions types relationships based on system views

Figure 5.12 presents an overview of the relationships between the function types for system maintainability that may be allocated to software-FUR. More specifically, the system maintainability requirements model is composed of 12 functions grouped into five function procedures types. The data flow on the model is also divided into direct data flows and the intermediary data flows:

1. The SMFP (Function Type 1) can be used to specify the data flows between its five sub functions and the data flows with the other functions on the system maintainability model – see Figure 5.12;
2. The SRFP model (Function Type 2) can be used to specify the data flows between its two sub functions and the data flows with the other functions on the system maintainability model – see Figure 5.12;
3. The SMP model (Function Type 3) can be used to specify the data flows between its two sub functions and the data flows with the other functions on the system maintainability model – see Figure 5.12;
4. The SSP (Function Type 4) can be used to specify the data flows between its two sub functions and the data flows with the other functions on the system maintainability model – see Figure 5.12;
5. The STP model (Function Type 5) can be used to specify the data flows between its two sub functions and the data flows with the other functions on the system maintainability model – see Figure 5.12.

Figure 5.13 presents an overview of the relationships between the function types for system maintainability that may be allocated to software-FUR, using COSMIC for graphical representation. More specifically:

1. The SMFP model can be used to specify and measure its functional size from the received/send data movements from/to SDF, FDOF, FDMF, FDCF and SFTF – see Figure 5.13;

2. The SRFP model can be used to specify and measure its functional size from the received/send data movements from/to FIF and FDF– see Figure 5.13;
3. The SMP model can be used to specify and measure its functional size from the received/send data movements from/to CDFP and CSDF – see Figure 5.13;
4. The SS model can be used to specify and measure its functional size from the received/send data movements from/to FPDCF and FPSF– see Figure 5.13;
5. The ST model can be used to specify and measure its functional size from the received/send data movements from/to STF and FATF – see Figure 5.13.

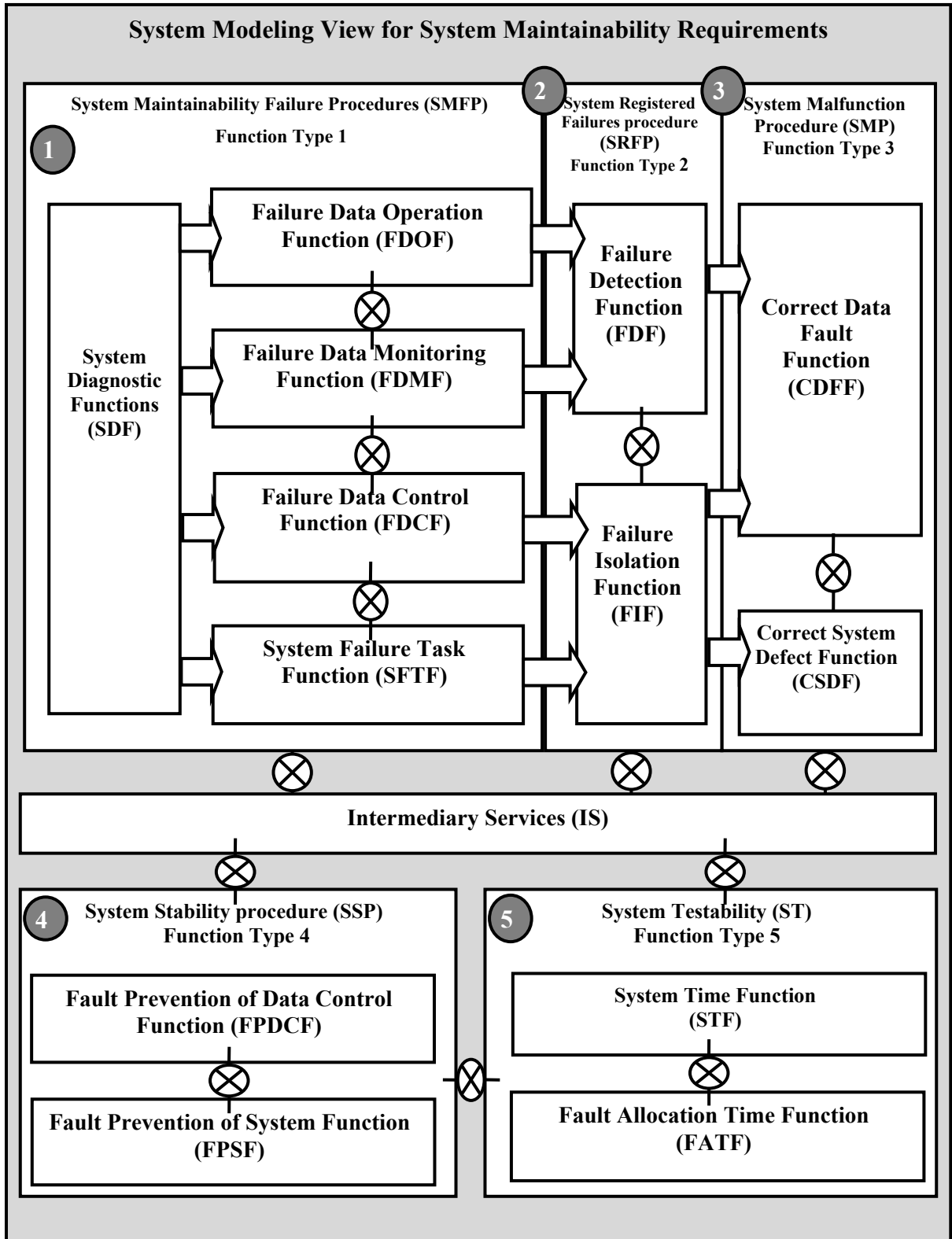


Figure 5.12 System Modeling View for System Maintainability Requirements

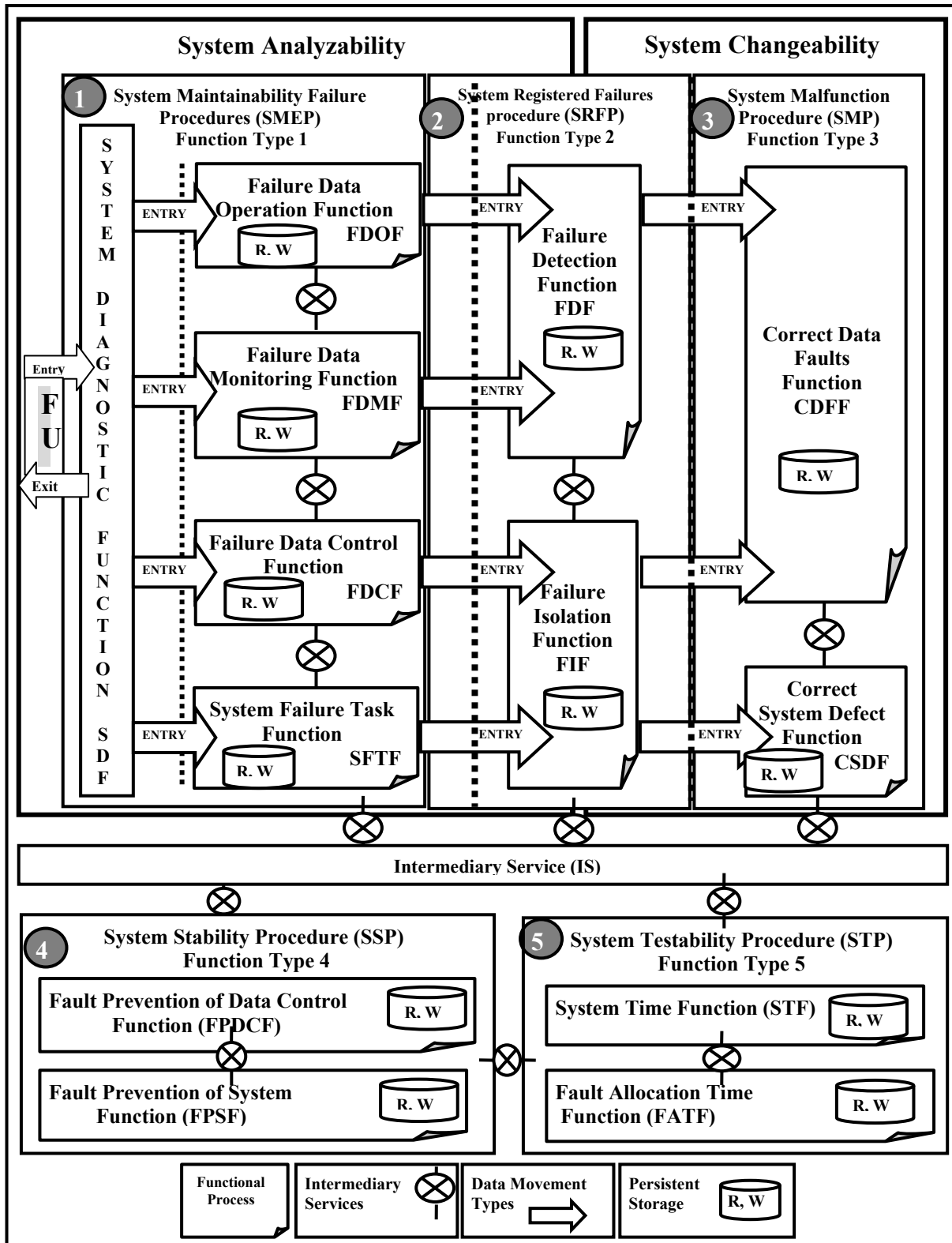


Figure 5.13 Standard-based model of software-FUR for system maintainability-NFR (Functional Level)

5.3 A standard-based model of software-FUR for system maintainability using SOA

A standard-based model of software-FUR for system maintainability-NFR is considered as a high-level model of requirements that helps explain, and position, the variety of maintainability-related functions described at the system level in the ECSS, IEEE, and ISO standards.

However, in practice, such a high-level model typically does not include detailed information documenting the required data groups necessary to unambiguously identify the specific corresponding data movements.

The standard-based model of software-FUR for system maintainability-NFR using SOA describes the detailed measurement model which can be used to specify and measure the functionality described in Figure 5.13.

5.3.1 Measurements of exchange messages for system maintainability

This section illustrates the standard-based model of software-FUR for system maintainability-NFR using SOA. This model is built based on Figure 5.13 and a role of the COSMIC-SOA explained in Table 1.4 in chapter 1.

System Maintainability Failure Procedure (SMFP)

Figure 5.14 describes the detailed measurements for the exchange of data messages between the application level and the services level for Function Type 1 (i.e., the System Maintainability Failure Procedure).

Table 5.3 contains the detailed measurement manual of the standard-based model of software-FUR for system maintainability-NFR for the SDF and their sub applications A, B, C, and D (i.e., the Maintainability Failure Procedure); in this case, they are triggered in the

requesting messages; the service functional process FS replies to FA and the sub applications with messages containing the requested data or an error message.

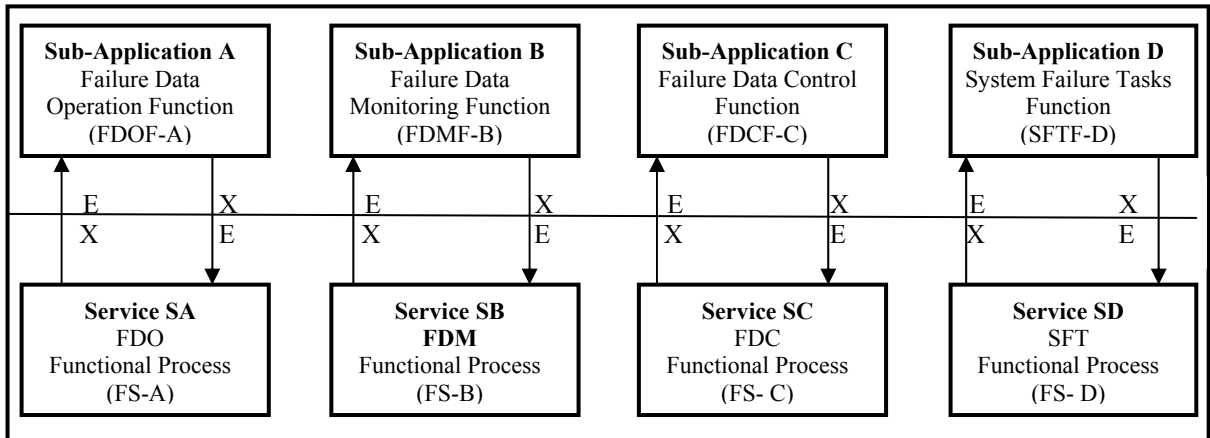


Figure 5.14 Exchange of data messages for sub applications (A, B, C, D) with their services for SMFP

Table 5.3 Measurement of the exchange messages of the application, sub application, and services for SMFP

ID	Functional Process Sub Application	CFP	Functional Process Services	CFP
Sub Application A	<ul style="list-style-type: none"> A functional process FDOF-A is triggered in the requesting messages from service functional process FS-A A functional process FDOF-A receives data from FS-A 	X	<ul style="list-style-type: none"> A service functional process FS-A receives a message from a functional process FDOF-A The service functional process FS-A replies to FDOF-A with a message containing the requested data or an error message 	E
		E		X
Sub Application B	<ul style="list-style-type: none"> A functional process FDMF-B is triggered in the requesting messages from service functional process FS-B A functional process FDMF-B receives data from FS-B 	X	<ul style="list-style-type: none"> A service functional process FS-B receives a message from a functional process FDMF-B The service functional process FS-B replies to FDMF-B with a message containing the requested data or an error message 	E
		E		X

Table 5.3 Measurement of the exchange messages of the application, sub application, and services for SMFP (Continued)

ID	Functional Process Sub Application	CF P	Functional Process Services	CFP
Sub Application C	<ul style="list-style-type: none"> • A functional process FDCF-C is triggered in the requesting messages from service functional process FS-C • A functional process FDCF-C receives data from FS-C 	X	<ul style="list-style-type: none"> • A service functional process FS-C receives a message from a functional process FDCF-C • The service functional process FS-C replies to FDCF-C with a message containing the requested data or an error message 	E
		E		X
Sub Application D	<ul style="list-style-type: none"> • A functional process SFTF-D is triggered in the requesting messages from service functional process FS-D • A functional process SFTF-D receives data from FS-D 	X	<ul style="list-style-type: none"> • A service functional process FS-D receives a message from a functional process SFTF-D • The service functional process FS-D replies to SFTF-D with a message containing the requested data or an error message 	E
		E		X
The total functional size = 16 CFP				

System Registered Failure Procedure (SRFP)

Figure 5.15 describes the detailed measurements for the exchange of data messages between application level and services level for SRFP. Table 5.4 contains the detailed measurement manual for the COSMIC-SOA model of system maintainability requirements for the SREP and their sub applications E, and F; in this case, they are triggered in the requesting messages; the service functional process FS replies to FA and the sub applications with messages containing the requested data or an error message.

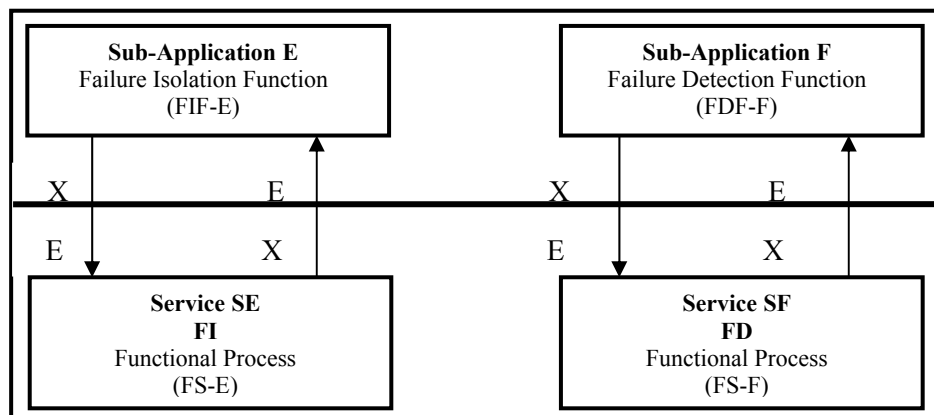


Figure 5.15 Interactions sub applications (E and F) with their services for SRFP

Table 5.4 Measurement of the exchange messages of the application, sub application, and services for SRFP

ID	Functional Process Sub Application	CFP	Functional Process Services	CFP
Sub Application E	<ul style="list-style-type: none"> • A functional process FIF-E is triggered in the requesting messages from service functional process FS-E • A functional process FIF-E receives data from FS-E 	X	<ul style="list-style-type: none"> • A service functional process FS-E receives a message from a functional process FIF-E • The service functional process FS-E replies to FIF-E with a message containing the requested data or an error message 	E
		E		X
Sub Application F	<ul style="list-style-type: none"> • A functional process FDF-F is triggered in the requesting messages from service functional process FS-F • A functional process FDF-F receives data from FS-F 	X	<ul style="list-style-type: none"> • A service functional process FS-F receives a message from a functional process FDF-F • The service functional process FS-F replies to FDF-F with a message containing the requested data or an error message 	E
		E		X
The total functional size = 8 CFP				

System Malfunction Procedure (SMP)

Figure 5.16 describes the detailed measurements for an exchange data message between the application level and services level for Function Type 3 (System Malfunction Procedure).

Table 5.5 contains the detailed measurement manual for the COSMIC-SOA model of system maintainability requirements for system malfunction procedure and their sub applications G, and H, in this case, are triggered in the requesting messages; the service functional process FS replies to FA and the sub applications with messages containing the requested data or an error message.

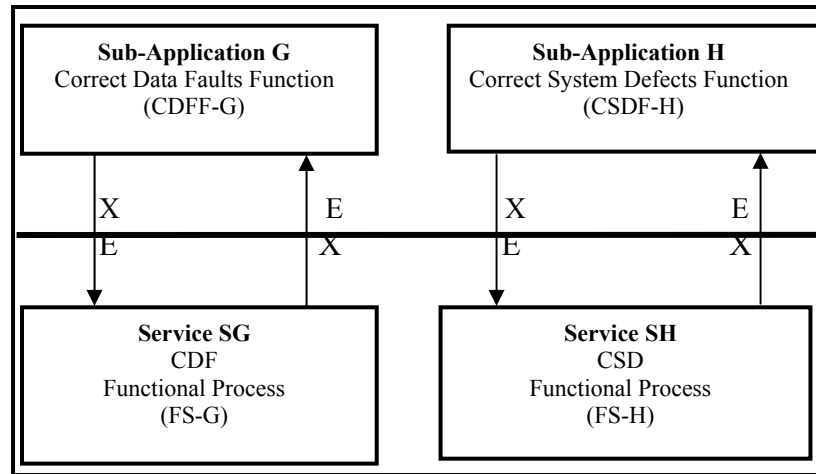


Figure 5.16 Interactions sub applications (G and H) with their services for SMP

Table 5.5 Measurement of the exchange messages of the application, sub application, and services for SMP

ID	Functional Process Sub Application	CFP	Functional Process Services	CF P
Sub Application G	<ul style="list-style-type: none"> A functional process CDFF-G is triggered in the requesting messages from service functional process FS-G A functional process CDFF-G receives data from FS-G 	X	<ul style="list-style-type: none"> A service functional process FS-G receives a message from a functional process CDFF-G The service functional process FS-G replies to CDFF-G with a message containing the requested data or an error message 	E
		E		X
Sub Application H	<ul style="list-style-type: none"> A functional process CSDF-H is triggered in the requesting messages from service functional process FS-H A functional process CSDF-H receives data from FS-H 	X	<ul style="list-style-type: none"> A service functional process FS-H receives a message from a functional process CSDF-H The service functional process FS-H replies to CSDF-H with a message containing the requested data or an error message 	E
		E		X
The total functional size = 8 CFP				

System Stability Procedure (SSP)

Figure 5.17 describes the detailed measurements for the exchange data messages between the level application and services level for Function Type 4 (System Stability Procedure).

Table 5.6 contains the detailed measurement manual for the COSMIC-SOA model of system maintainability requirements for system stability and their sub applications K, and L, in this case, are triggered in the requesting messages; the service functional process FS replies to FA and the sub applications with messages containing the requested data or an error message.

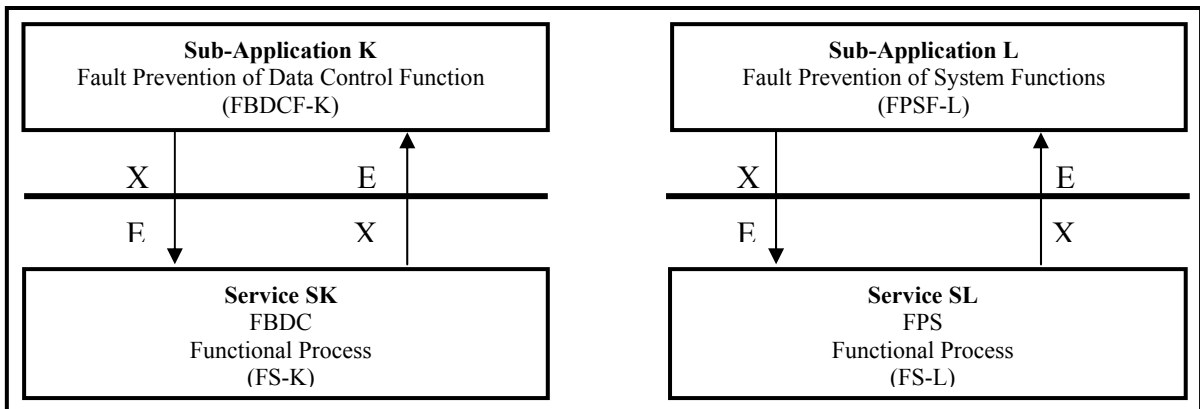


Figure 5.17 Interactions sub applications (K and L) with their services for SSP

Table 5.6 Measurement of the exchange messages of the application, sub application, and services for SSP

ID	Functional Process Sub Application	CFP	Functional Process Services	CFP
Sub Application K	<ul style="list-style-type: none"> A functional process FBDCF-K is triggered in the requesting messages from service functional process FS-K A functional process FBDCF-K receives data from FS-K 	X E	<ul style="list-style-type: none"> A service functional process FS-K receives a message from a functional process FBDCF-K The service functional process FS-K replies to FBDCF-K with a message containing the requested data or an error message 	E X

Table 5.6 Measurement of the exchange messages of the application, sub application, and services for SSP (Continued)

ID	Functional Process Sub Application	CFP	Functional Process Services	CFP
Sub Application L	<ul style="list-style-type: none"> A functional process FPSF-L is triggered in the requesting messages from service functional process FS-L 	X	<ul style="list-style-type: none"> A service functional process FS-L receives a message from a functional process FPSF-L 	E
	<ul style="list-style-type: none"> A functional process FPSF-L receives data from FS-L 	E	<ul style="list-style-type: none"> The service functional process FS-L replies to FPSF-L with a message containing the requested data or an error message 	X
The total functional size = 8 CFP				

System Testability Procedure (STP)

Figure 5.18 describes the detailed measurements for exchange data messages between the application level and services level for Function Type 5 (System Testability).

Table 5.7 contains the detailed measurement manual for the COSMIC-SOA model of system maintainability requirements for system testability procedure and their sub applications I, and J, in this case, are triggered in the requesting messages; the service functional process FS replies to FA and the sub applications with messages containing the requested data or an error message.

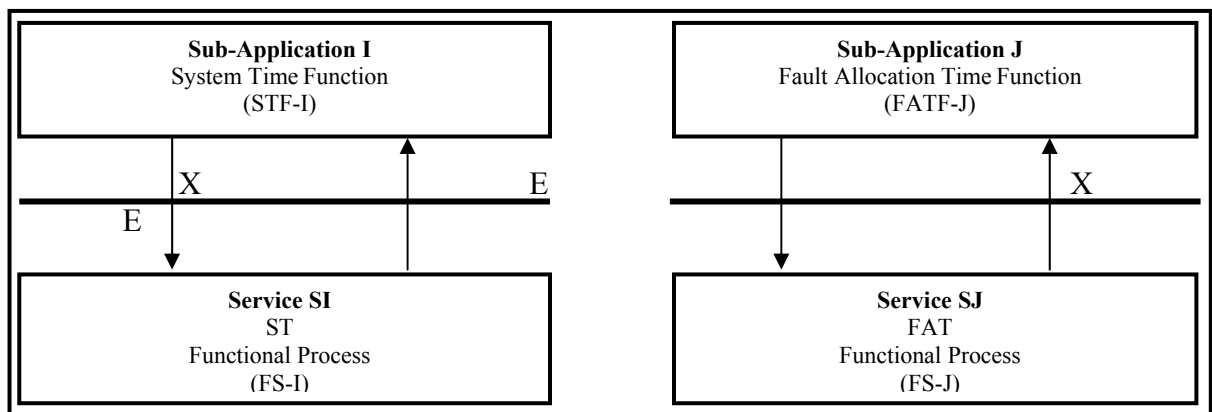


Figure 5.18 Interactions sub applications (I and J) with their services for STP

Table 5.7 Measurement of the exchange messages of the application, sub application, and services STP

ID	Functional Process Sub Application	CFP	Functional Process Services	CFP
Sub Application I	<ul style="list-style-type: none"> • A functional process STF-I is triggered in the requesting messages from service functional process FS-I • A functional process STF-I receives data from FS-I 	X E	<ul style="list-style-type: none"> • A service functional process FS-I receives a message from a functional process STF-I • The service functional process FS-I replies to STF-I with a message containing the requested data or an error message 	E X
Sub Application J	<ul style="list-style-type: none"> • A functional process FATF-J is triggered in the requesting messages from service functional process FS-J • A functional process FATF-J receives data from FS-J 	X E	<ul style="list-style-type: none"> • A service functional process FS-J receives a message from a functional process FATF-J • The service functional process FS-J replies to FATF-J with a message containing the requested data or an error message 	E X
The total functional size = 8 CFP				

5.3.2 Measurement of intermediary services for system maintainability

When a functional process of an application service in application A requires data that are available via an application service in application B, the former application service calls a functional process of the intermediary service, which may complete the following tasks as a separate utility service – for more details, see :

1. Control the handling of the request received from a service of application A;
2. Translate the ‘language’ of the message from application A into the ‘language’ of application B (and possibly applications C, D..., if services of other applications are involved) that must fulfill the request;
3. Call on a functional process of the application service of application B by means of the translated message;

4. Receive the reply message from the functional process of application B (and possibly reply messages from applications C, D ...);
5. Translate the results into a message in the language of application A;
6. Send the reply message to (the functional process of the service of) application A;
7. Manage exceptional situations;
8. Log data about the handling of services.

The COSMIC-SOA model for maintainability requirements describes the detailed measurement for the intermediary services. This section describes the second step of the COSMIC model for detailed measurement for intermediary services between the application and sub applications of the standard-based model of software-FUR for system maintainability-NFR to identify the software-FUR services.

System Maintainability Failure Procedure (SMFP)

Figure 5.19 describes the detailed measurements for the intermediary services between the application level and services level for Function Type 1 (SMFP) and Table 5.8 contains the detailed measurement between intermediary services (SA, SB, SC and SD).

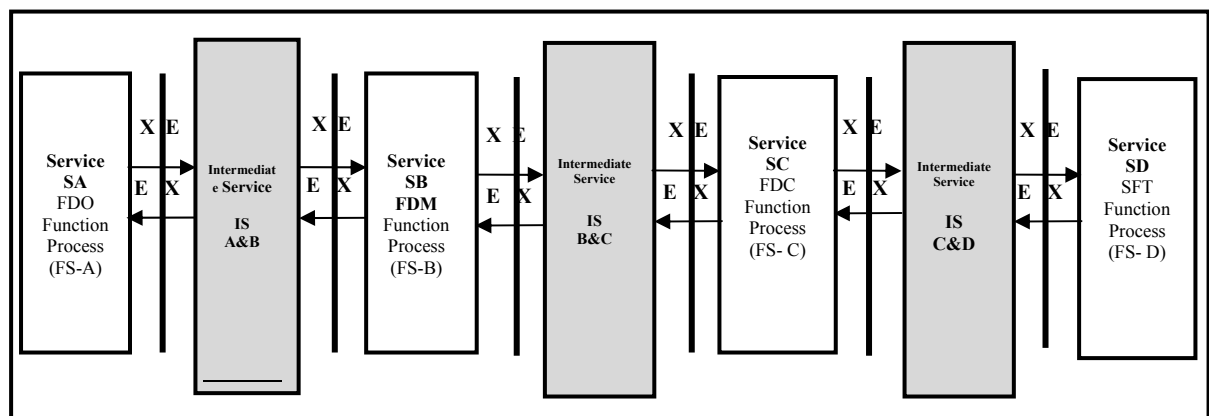


Figure 5.19 The intermediary services between sub application services (SA, SB, SC and SD) for SMFP

Table 5.8 Measurement of the intermediary services for SMFP

The intermediary services for Function Type 1		Functional Process Services	CFP
The intermediary services between FS-A and FS-B		<ul style="list-style-type: none"> • A service functional process FS-A sends one data group to intermediary service (IS) between service SA and service SB. • Intermediary service (IS) between service SA and service SB receives one data group from a service functional process FS-A. • Intermediary service (IS) between service SA and service SB sends one data group to a service functional process FS-B. • A service functional process FS-B receives one data group from intermediary service (IS) between service SA and service SB. • A service functional process FS-B sends one data group to intermediary service (IS) between service SA and service SB. • Intermediary service (IS) between service SA and service SB receives one data group from a service functional process FS-B. • Intermediary service (IS) between service SA and service SB sends one data group to a service functional process FS-A. • A service functional process FS-A receives one data group from intermediary service (IS) between service SA and service SB. 	X
Sub Application A	Sub Application B		E
			X
			E
			X
Sub Application B	Sub Application C		E
			X
			E
		X	
The intermediary services between FS-B and FS-C		<ul style="list-style-type: none"> • A service functional process FS-B sends one data group to intermediary service (IS) between service SB and service SC. • Intermediary service (IS) between service SB and service SC receives one data group from a service functional process FS-B. • Intermediary service (IS) between service SB and service SC sends one data group to a service functional process FS-C. • A service functional process FS-C receives one data group from intermediary service (IS) between service SB and service SC. • A service functional process FS-C sends one data group to intermediary service (IS) between service SB and service SC. • Intermediary service (IS) between service SB & service SC receives one data group from a service functional process FS-C. • Intermediary service (IS) between service SB and service SC sends one data group to a service functional process FS-B. • A service functional process FS-B receives one data group from intermediary service (IS) between service SB and service SC. 	X
Sub Application B	Sub Application C		E
			X
			E
			X
Sub Application C	Sub Application B		E
			X
			E
		X	

Table 5.8 Measurement of the intermediary services for SMFP (Continued)

The intermediary services for Function Type 1		Functional Process Services	CFP	
The intermediary services between FS-C and FS-D		<ul style="list-style-type: none"> • A service functional process FS-C sends one data group to intermediary service (IS) between service SA and service SD. • Intermediary service (IS) between service SC and service SD receives one data group from a service functional process FS-C. • Intermediary service (IS) between service SC and service SD sends one data group to a service functional process FS-D. • A service functional process FS-D receives one data group from intermediary service (IS) between service SC and service SD. • A service functional process FS-D sends one data group to intermediary service (IS) between service SC and service SD. • Intermediary service (IS) between service SC and service SD receives one data group movements from a service functional process FS-D. • Intermediary service (IS) between service SC and service SD sends one data group to a service functional process FS-C. • A service functional process FS-A receives one data group from intermediary service (IS) between service SC & service SD. 	X	
Sub Application C	Sub Application D			E
				X
				E
				X
				E
				X
				E
				X
The total functional size = 24 CFP				

System Registered Failure Procedure (SRFP)

Figure 5.20 describes the detailed measurements for intermediary services between the application level and the services level for Function Type 2 (system Registered Failures procedure) and Table 5.9 contains the detailed measurement between the intermediary services (SE and SF).

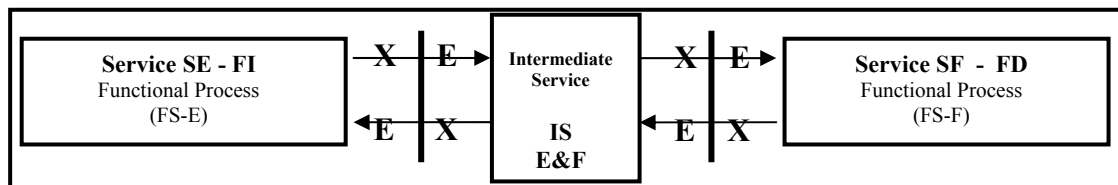


Figure 5.20 The intermediary services for sub application (SE and SF) for SRFP

Table 5.9 Measurement of the intermediary services for SRFP

The intermediary services for functional type 2		Functional Process Services	CFP
The intermediary services between FS-E and FS-F		<ul style="list-style-type: none"> • A service functional process FS-E sends one data group to intermediary service (IS) between service SE and service SF. • Intermediary service (IS) between service SE and service SF receives one data group from a service functional process FS-E. • Intermediary service (IS) between service SE and service SF sends one data group to a service functional process FS-F. • A service functional process FS-F receives one data group from intermediary service (IS) between service SE and service SF. • A service functional process FS-F sends one data group to intermediary service (IS) between service SE and service SF. • Intermediary service (IS) between service SE and service SF receives one data group from a service functional process FS-F. • Intermediary service (IS) between service SE and service SF sends one data group to a service functional process FS-E. • A service functional process FS-E receives one data group from intermediary service (IS) between service SE and service SF. 	X
Sub Application E	Sub Application F		E
			X
Sub Application E	Sub Application F		E
			X
			E
			E
The total functional size = 8 CFP			

System Malfunction Procedure (SMP)

Figure 5.21 describes the detailed measurements for intermediary services between the application level and services level for Function Type 3 (System Malfunction Procedure) and Table 5.10 contains the detailed measurement between the intermediary services (SG and SH).

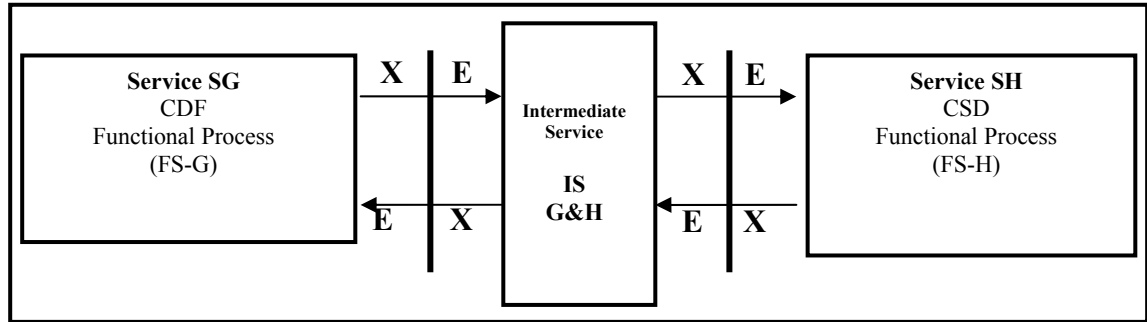


Figure 5.21 The intermediary services between sub application (SG and SH) for SMP

Table 5.10 Measurement of the intermediary services for SMP

The intermediary services for Function Type 3		Functional Process Services	CFP
The intermediary services between FS-G and FS-H		<ul style="list-style-type: none"> • A service functional process FS-G sends one data group to intermediary service (IS) between service SG and service SH. 	X
Sub Application G	Sub Application H	<ul style="list-style-type: none"> • Intermediary service (IS) between service SG and service SH receives one data group from a service functional process FS-G. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SG and service SH sends one data group to a service functional process FS-H. 	X
		<ul style="list-style-type: none"> • A service functional process FS-H receives one data group from intermediary service (IS) between service SG and service SH. 	E
		<ul style="list-style-type: none"> • A service functional process FS-H sends one data group to intermediary service (IS) between service SG and service SH. 	X
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SG and service SH receives one data group from a service functional process FS-H. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SG and service SH sends one data group to a service functional process FS-G. 	X
		<ul style="list-style-type: none"> • A service functional process FS-G receives one data group from intermediary service (IS) between service SG and service SH. 	E
		The total functional size = 8 CFP	

System Stability Procedure (SSP)

Figure 5.22 describes the detailed measurements for intermediary services between the application level and services level for Function Type 4 (System Stability) and Table 5.11 contains the detailed measurement of the intermediary services (SK and SL).

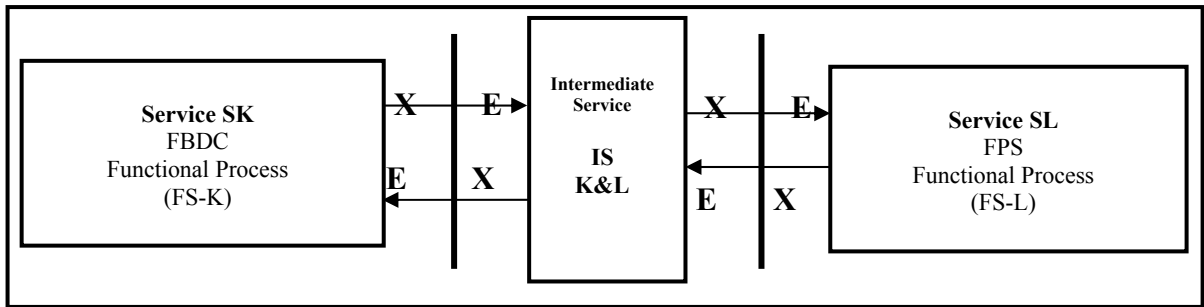


Figure 5.22 The intermediary services between sub application (SK and SL) for SSP

Table 5.11 Measurement of the intermediary services for SSP

The intermediary services for Function Type 5		Functional Process Services	CFP
The intermediary services between FS-K and FS-L		<ul style="list-style-type: none"> • A service functional process FS-K sends one data group to intermediary service (IS) between service SK and service SL. 	X
Sub Application K	Sub Application L	<ul style="list-style-type: none"> • Intermediary service (IS) between service SK and service SL receives one data group from a service functional process FS-K. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SK and service SL sends one data group to a service functional process FS-L. 	X
		<ul style="list-style-type: none"> • A service functional process FS-L receives one data group from intermediary service (IS) between service SK and service SL. 	E
		<ul style="list-style-type: none"> • A service functional process FS-L sends one data group to intermediary service (IS) between service SK and service SL. 	X
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SK and service SL receives one data group from a service functional process FS-L. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SK and service SL sends one data group to a service functional process FS-K. 	X
		<ul style="list-style-type: none"> • A service functional process FS-K receives one data group from intermediary service (IS) between service SK and service SL. 	E
		The total functional size = 8 CFP	

The intermediary services between SSP and STP

Figure 5.23 describes the detailed measurements for the intermediary services between the application levels and services level for Function Types 4 and 5 (System stability and System testability) and Table 5.12 contains the detailed measurement between the intermediary services (SI and SK) , (SI and SL) , (SJ and SK) and (SJ and SL) .

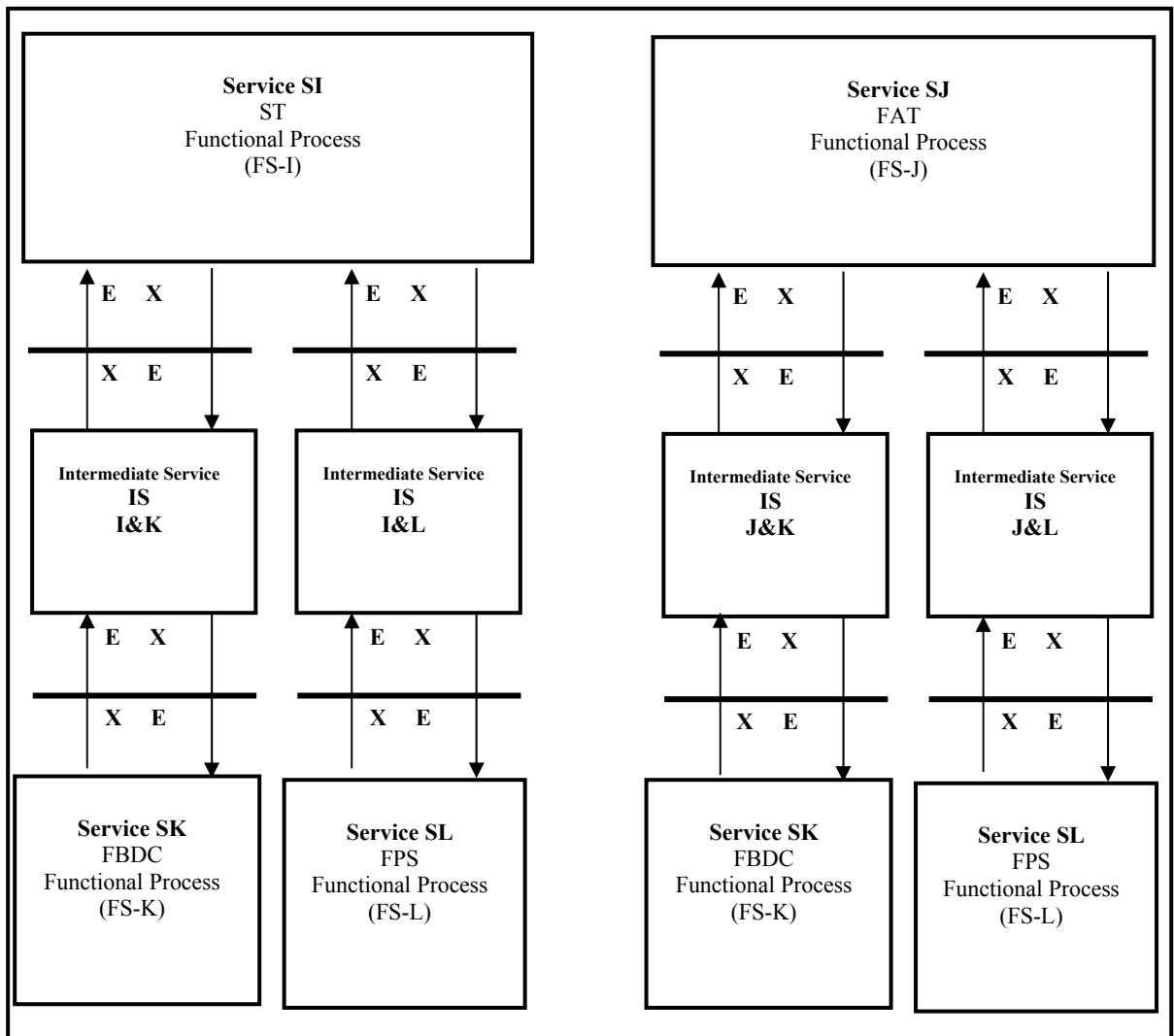


Figure 5.23 The intermediary services between sub application services (SI and SK), (SI and SL) and (SJ and SK), (SJ and SL) for (SSP and STP)

Table 5.12 Measurement of the intermediary services for SSP and STP

The intermediary services Function Types 4 & 5		Functional Process Services	CFP
The intermediary services between FS-I and FS-K		<ul style="list-style-type: none"> • A service functional process FS-I sends one data group to intermediary service (IS) between service SI and service SK. 	X
Sub Application I	Sub Application K	<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SK receives one data group from a service functional process FS-I. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SK sends one data group to a service functional process FS-K. 	X
		<ul style="list-style-type: none"> • A service functional process FS-K receives one data group from intermediary service (IS) between service SI and service SK. 	E
		<ul style="list-style-type: none"> • A service functional process FS-K sends one data group t to intermediary service (IS) between service SI and service SK. 	X
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SK receives one data group from a service functional process FS-K. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SK sends one data group to a service functional process FS-I. 	X
		<ul style="list-style-type: none"> • A service functional process FS-I receives one data group from intermediary service (IS) between service SI and service SK. 	E
		The intermediary services between FS-I and FS-L	
Sub Application I	Sub Application L	<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SL receives one data group from a service functional process FS-I. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SL sends one data group to a service functional process FS-L. 	X
		<ul style="list-style-type: none"> • A service functional process FS-L receives one data group from intermediary service (IS) between service SI and service SL. 	E
		<ul style="list-style-type: none"> • A service functional process FS-L sends one data group to (IS) between service SI and service SL. 	X
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SL receives one data group from a service functional process FS-L. 	E
		<ul style="list-style-type: none"> • Intermediary service (IS) between service SI and service SL sends one data group to a service functional process FS-I. 	X
		<ul style="list-style-type: none"> • A service functional process FS-I receives one data group from (IS) between service SI and service SL. 	E

Table 5.12 Measurement of the intermediary services for SSP and STP (Continued)

The intermediary services Function Types 4 & 5		Functional Process Services	CFP
The intermediary services between FS-J and FS-K		<ul style="list-style-type: none"> • A service functional process FS-J sends one data group to intermediary service (IS) between service SJ and service SK. • Intermediary service (IS) between service SJ and service SK receives one data group from a service functional process FS-J. 	X E
Sub Application J	Sub Application K	<ul style="list-style-type: none"> • Intermediary service (IS) between service SJ and service SK sends one data group to a service functional process FS-K. • A service functional process FS-K receives one data group from (IS) between service SJ and service SK. • A service functional process FS-K sends one data group to (IS) between service SJ and service SK. • Intermediary service (IS) between service SJ and service SK receives one data group from a service functional process FS-K. • Intermediary service (IS) between service SJ and service SK sends one data group to a service functional process FS-J. • A service functional process FS-J receives one data group from (IS) between service SJ and service SK. 	X E X E X E
The intermediary services between FS-J and FS-K		<ul style="list-style-type: none"> • A service functional process FS-J sends one data group to (IS) between service SJ and service SL. • Intermediary service (IS) between service SJ and service SL receives one data group from a service functional process FS-J. 	X E
Sub Application J	Sub Application L	<ul style="list-style-type: none"> • Intermediary service (IS) between service SJ and service SL sends one data group to a service functional process FS-L. • A service functional process FS-L receives one data group from intermediary service (IS) between service SJ and service SL. • A service functional process FS-L sends one data group to (IS) between service SJ and service SL. • Intermediary service (IS) between service SJ and service SL receives one data group from a service functional process FS-L. • Intermediary service (IS) between service SJ and service SL sends one data group to a service functional process FS-J. • A service functional process FS-J receives one data group from intermediary service (IS) between service SJ and service SL. 	X E X E X E
The total functional size = 32 CFP			

The intermediary services between SMFP, SRFP and SMP with STP

Figure 5.24 and Figure 5.25 describe the detailed measurements for the intermediary services between the application level and services level for Function Types 1, 2, 3, and 5 (and Table 5.13 contains the detailed measurement between the intermediary services for SK and SL with (SA, SB, SC, SD, SE, SF, SG and SH).

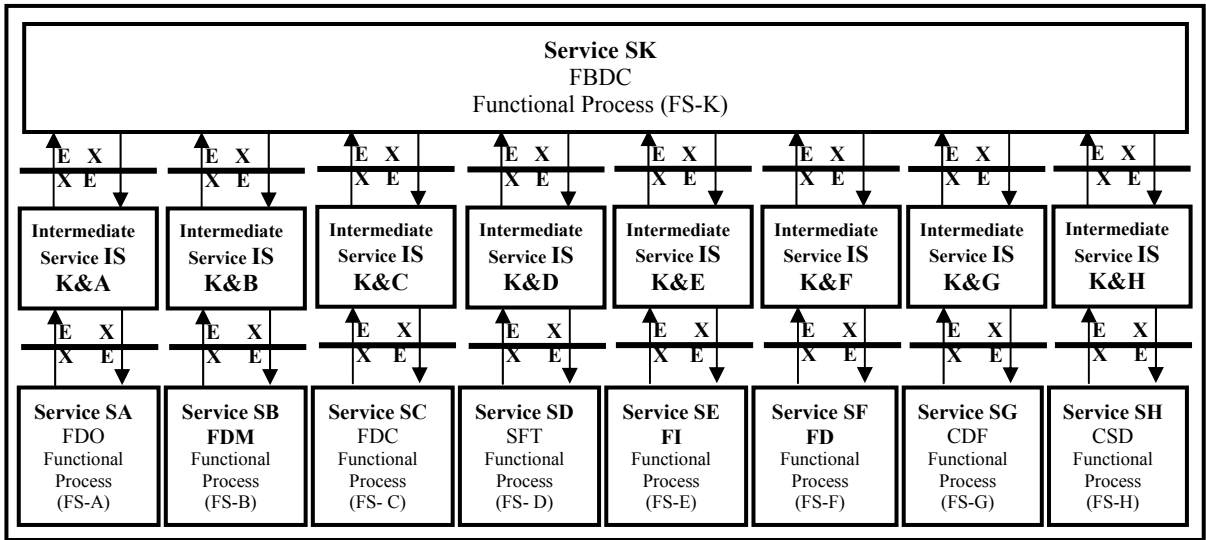


Figure 5.24 The intermediary services between sub application services SK with (SA, SB, SC, SD, SE, SF, SG and SH)

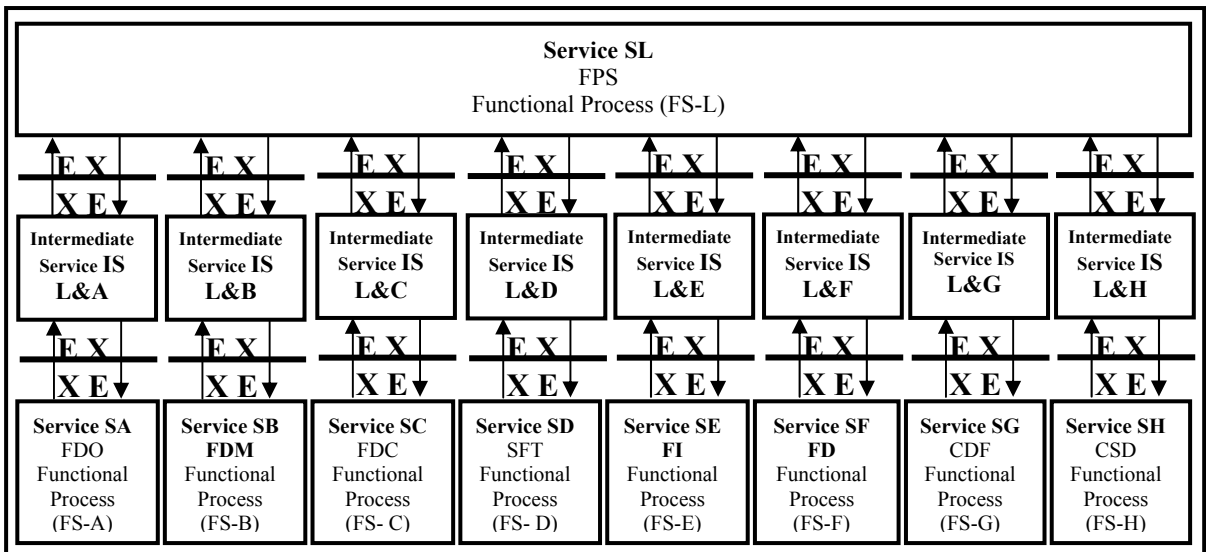


Figure 5.25 The intermediary services between sub application services SL with (SA, SB, SC, SD, SE, SF, SG and SH)

Table 5.13 Measurement of the intermediary services for SMFP, SRFP, SMP and STP

The intermediary services for Function Types 1, 2, 3, & 4		Functional Process Services	CFP
The intermediary services between FS-K and FS-A FS-K and FS-B FS-K and FS-C FS-K and FS-D FS-K and FS-E FS-K and FS-F FS-K and FS-G FS-K and FS-H		<ul style="list-style-type: none"> • A service functional process FS-K sends one data group to intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH • Intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH receives one data group from a service functional process FS-K. • Intermediary services (IS) between service SK and service SA, SB, SC, SD, SE, SF, SG, SH sends one data group to a service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H . • A service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H receives one data group from intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH. • A service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H sends one data group to intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH. • Intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH receives one data group from a service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H • Intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH sends one data group to a service functional process FS-K. • A service functional process FS-K receives one data group from intermediary services (IS) between service SK and services SA, SB, SC, SD, SE, SF, SG, SH. 	X
Sub Application K	Sub Application A B C D E F G H		E
The intermediary services between FS-L and FS-A FS-L and FS-B FS-L and FS-C FS-L and FS-D FS-L and FS-E FS-L and FS-F FS-L and FS-G FS-L and FS-H		<ul style="list-style-type: none"> • A service functional process FS-L sends one data group to intermediary services (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH • Intermediary services (IS) between service SL and service SA, SB, SC, SD, SE, SF, SG, SH receives one data group from a service functional process FS-L. • Intermediary services (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH sends one data group to a service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H . • A service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H receives one data group from intermediary service (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH. • A service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H sends one data group to intermediary services (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH. • Intermediary service (IS) between service SL and service SA, SB, SC, SD, SE, SF, SG, SH receives one data group from a service functional process FS-A, FS-B, FS-C, FS-D, FS-E, FS-F, FS-G, FS-H • Intermediary services (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH sends one data group to a service functional process FS-L. • A service functional process FS-L receives one data group from intermediary service (IS) between service SL and services SA, SB, SC, SD, SE, SF, SG, SH. 	X
Sub Application L	Sub Application A B C D E F G H		E
The total functional size = 128 CFP			

5.3.3 Measurements of data movements between Functional processes

Based on Figure 5.13, this section presents the possible flows of data movements between all functions in the system maintainability requirements.

System Maintainability Failure Procedure (SMFP)

Figure 5.26 describes the detailed measurements for direct data movements of services in the application level for Function Type1 (SMFP) and Table 5.14 contains the detailed measurement between the SDF and sub applications (A, B, C and D).

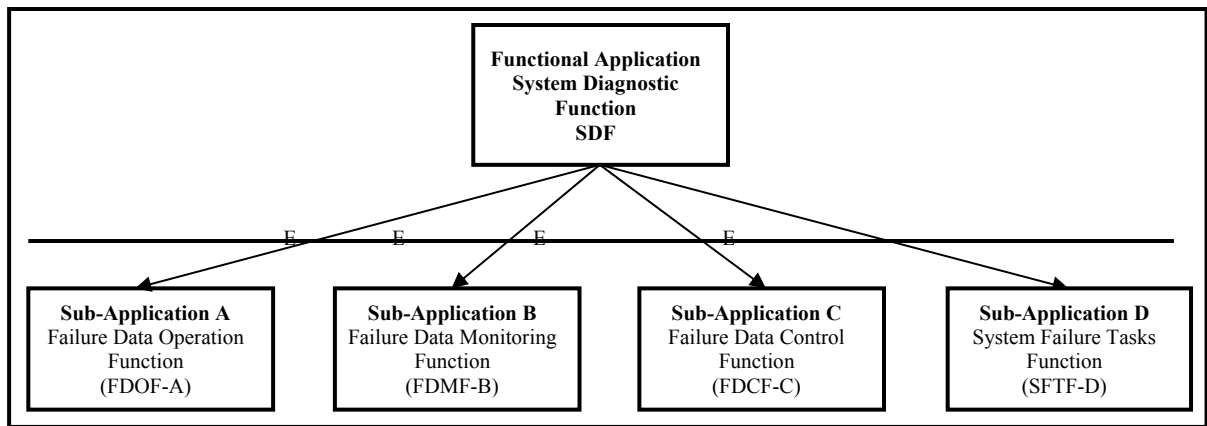


Figure 5.26 Direct data movements between the application (SDF) and sub applications (A, B, C, D) for SMFP

Table 5. 14 Measurement of the direct data movements for SMFP

ID	Functional Process Application (direct data movements)	CFP
Application with Sub Application A, B, C, and D	<ul style="list-style-type: none"> Failure Data Operation Function receives at least one data group from System Diagnostic Function. 	E*
	<ul style="list-style-type: none"> Failure Data Monitoring Function receives at least one data group from System Diagnostic Function. 	E*
	<ul style="list-style-type: none"> Failure Data Control Function receives at least one data group from System Diagnostic Function. 	E*
	<ul style="list-style-type: none"> System Failure Task Function receives at least one data group from System Diagnostic Function. 	E*
The total functional size = 4 CFP		

In the above table (*) means a variable numbers of data movements in this case.

System Registered Failure Procedure (SRFP)

Figure 5.27 describes the detailed measurements for direct data movements of services in the application level for Function Type 2 (SRFP) and Table 5.15 contains the detailed measurement for the direct data movements between sub applications (A, B, C and D) and sub applications (E and F).

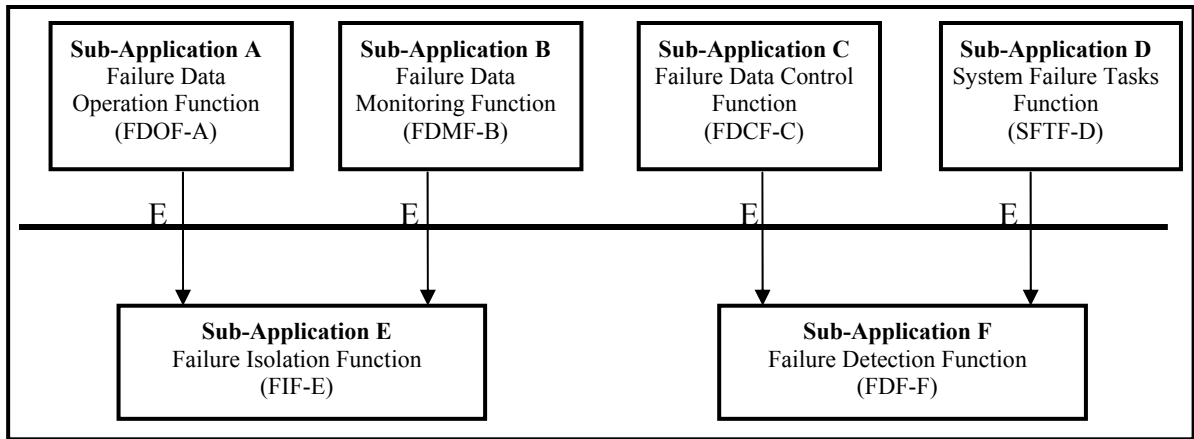


Figure 5.27 Direct data movements between sub applications (A, B, C, D) and sub applications (E and F) for SRFP

Table 5.15 COSMIC-SOA measurement of the direct data movements for SRFP

ID	Functional Process Application D (direct data movement)	CFP
Sub Applications A, B, C, and D with Sub Applications E and F	<ul style="list-style-type: none"> • FIF-E receives at least one data group from FDOF-A • FIF-E receives at least one data group from FDMF-B • FDF-F receives at least one data group from FDCF-C • FDF-F receives at least one data group from SFTF-D 	E* E* E* E*
The total functional size = 4 CFP		

System Malfunction Procedure (SMP)

Figure 5.28 describes the detailed measurements for direct data movements of services in the application level for SMP and Table 5.16 contains the detailed measurement for the direct data movements between sub applications (E and F) and sub applications (G and H).

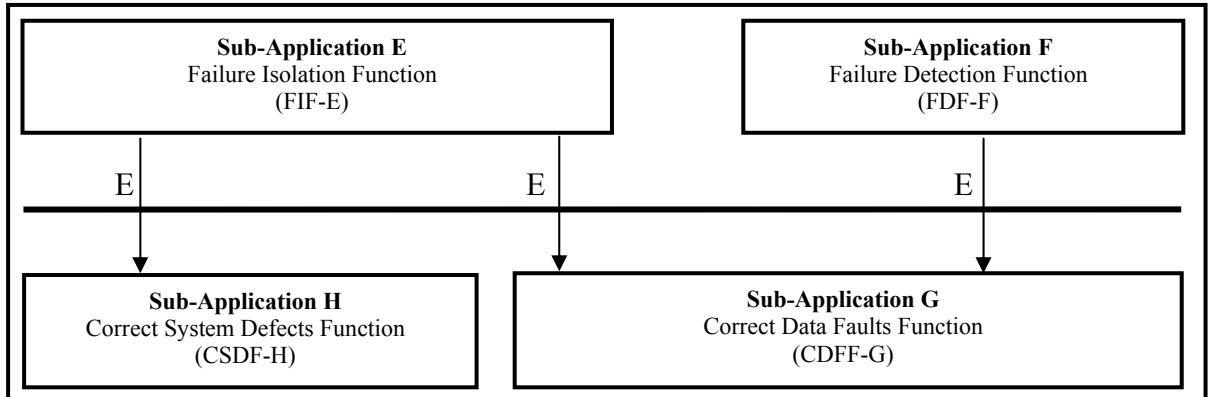


Figure 5.28 Direct data movements between sub applications (E and F) and sub applications (G and H) for SMP

Table 5.16 COSMIC-SOA measurement of the direct data movements for SMP

ID	Functional Process Application D (direct data movements)	CFP
Sub Applications E, F, G, and H with Sub	<ul style="list-style-type: none"> CSDF-H receives at least one data group from FIF-E CSDF-H receives at least one data group from FIF-E CDDF-G receives at least one data group from FDF-F 	E* E* E*
The total functional size 3 CFP		

5.3.4 Indirect data movements for all function types

Figure 5.29 describes the detailed measurements for the indirect data movements of services in the application level by using the same persistent storage for all the functional services and Table 5.17 contains the detailed measurement for the indirect data movements between the 12 sub applications.

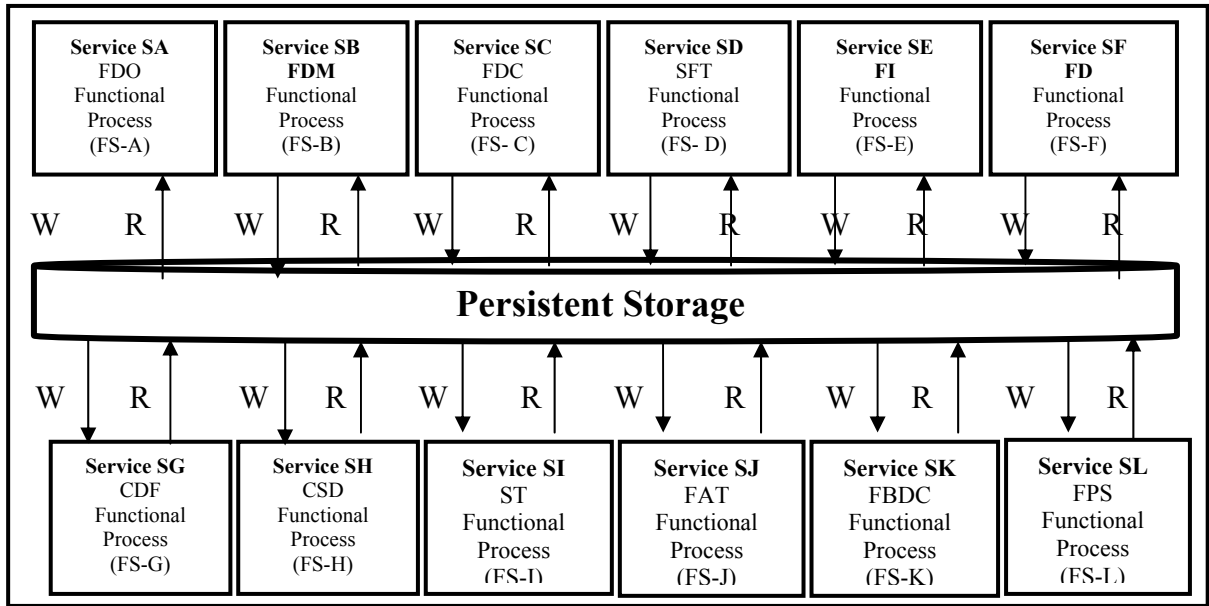


Figure 5.29 Indirect data movements between all sub applications in all functional types

Table 5.17 Measurement of the indirect data movements for the model

ID	Functional Process Application (Indirect Data Movements)	CFP
Sub Application A, B, C, D, E, F, G, H, I, J, K, and L	• Service SA writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SA reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SB writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SB reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SC writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SC reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SD writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SD reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SE writes a data group in the persistent storage to be used by other services in the maintainability model	W

Table 5.17 Measurement of the indirect data movements for the model (Continued)

ID	Functional Process Application (Indirect Data Movements)	CFP
Sub Application A, B, C, D, E, F, G, H, I, J, K, and L	• Service SE reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SF writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SF reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SG writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SG reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SH writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SH reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SI writes a data group in the persistent storage to be used by other service in the maintainability model	W
	• Service SI reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SJ writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SJ reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SK writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SK reads a data group from the persistent storage from another service in the maintainability model	R
	• Service SL writes a data group in the persistent storage to be used by other services in the maintainability model	W
	• Service SL reads a data group from the persistent storage from another service in the maintainability model	R
The total functional size = 24 CFP		

5.4 Sizing of the standard-based model for system maintainability-NFR

The specification of software-FUR for system maintainability in any specific project is a specific instantiation of the proposed standard-based model of software-FUR for system maintainability-NFR described in Figure 5.13. When the software specification document is at the level of the movements of data groups, then these functional requirements can be directly measured using the COSMIC measurement rules.

Table 5.18 presents the measurement results based on the detailed measurement manual using a specific instantiation of maintainability requirements, which would have one of each of the maintainability function types and relationships described in the previous sections and in Figure 5.13. For example, Table 5.18 and Table 5.19 illustrate the sizing of the standard-based model of software-FUR for system maintainability-NFR:

Table 5.18 Measurement of the maintainability model (Function Level)

Measurement of standard-based model of software-FUR for system maintainability-NFR			CFP
COSMIC-SOA Direct and Indirect Data Movements			
Direct Data Movements			
1	Function Type 1 (SMFP)	4	11
2	Function Type 2 (SRFP)	4	
3	Function Type 3 (SMP)	3	
Indirect Data Movements			
3	All Functions Types (SMFP, SRFP, SMP, SSP and STP)	24	24
Total Functional Size			35 CFP

Table 5.19 Measurement of the maintainability model (Service level)

Measurement of model of maintainability (Service level)			CFP
COSMIC-SOA exchange messages (services)			
A	1	Function Type 1 (SMFP)	16
	2	Function Type 2 (SRFP)	8
	3	Function Type 3 (SMP)	8
	4	Function Type 4 (SS)	8
	5	Function Type 5 (ST)	8
COSMIC-SOA intermediary services			
B	1	Function Type 1 (SMFP)	24
	2	Function Type 2 (SRFP)	8
	3	Function Type 3 (SMP)	8
	4	Function Type 4 (SS)	8
	5	Function Type 5 (ST)	8
	6	Function Type 1, Function Type 2, Function Type 3, with Function Type 5 (SMFP, SRFP and SMP with STP)	120
	7	Function Type 1, Function Type 2, Function Type 3, Function Type 5 with Function Type 4 (SMFP, SRFP, SMP and STP with SSP)	152
Total Size Functional = 376 CFP			

5.5 A measurement example

The specification of software-FUR for system maintainability requirements in any specific project is a specific instantiation of the proposed generic model described in Figure 5.13. When the software specifications document is at the level of the movement of data groups, then these functional requirements can be directly measured using the COSMIC measurement rules. This section presents a measurement example of the use of the COSMIC generic model of system maintainability requirements allocated to software.

The measurement example in this chapter explains how to use the proposed reference maintainability model to size a hypothetical framework with all of the kinds of software-FUR described in the framework.

Example: The functional requirements allocated to software for the system maintainability failures procedure (SMFP) for a specific instantiation are:

1. The SDF sends four data groups to the FDOF;
2. The SDF sends one data group to the FDMF;
3. The SDF sends one data group to the FDCF;
4. The SDF sends two data groups to the SFTF.

The next section presents the functional measurement sizing for the system Maintainability Failure Procedure using the standard-based model of software-FUR for system maintainability-NFR.

The Functional Measurement Solution

Based on Figure 5.13 of the standard-based model of software-FUR for system maintainability-NFR and the COSMIC-SOA guideline for specifying data movements, the functional size measurement method for the SMFP for this example is as follows:

5.5.1 Measurement of the exchange messages

1. The SDF in the application layer send four data groups to the FDOF, which means that four (4) functional processes will interact with four (4) functional services;
2. The SDF in the application layer sends a data group to the FDMF, which means that one (1) functional process will interact with one (1) functional service;
3. The SDF in the application layer sends a data group to the FDCF, which means that one (1) functional process will interact with one (1) functional service;
4. The SDF in the application layer sends two (2) data groups to the SFTF, which means that two (2) functional processes will interact with two (2) functional services;

Measurement Results

1. The number of functional services = 8;
2. Each functional process in the application layer will interact with each service. The data movements between each functional process and service = 4 CFP;
3. The functional size for the 8 services = $8 \times 4 = 32$ CFP.

5.5.2 Measurement of the intermediary services

1. The FDOF has four (4) functional services, which means that four (4) FDOF functional services need four intermediary services to contact the FDMF functional service;
2. The FDMF has one (1) functional service, which means one (1) FDMF functional service needs one (1) intermediary service to contact the FDCF functional service;
3. The FDCF has one (1) functional service, which means that one (1) FDCF functional service needs one (1) intermediary service to contact the SFTF functional service;
4. The SFTF has two (2) functional services, which means that two (2) SFTF functional services need two (2) intermediary services.

Measurement Results

1. Each intermediary service includes 8 data movements or 8 CFP;

2. In the example, 8 intermediary services are needed – see Figure 5.13;
3. The functional measurement size for the 8 intermediary services X 8 CFP for each = 64 CFP.

5.5.3 Measurement of data movements (Function Level)

Direct data movements:

1. The SDF will send 4 data groups directly to the FDOF, each functional data movement including one entry. The functional size = 4 CFP;
2. The SDF will send 1 data group directly to the FDMF, each functional data movement including one entry. The functional size = 1 CFP;
3. The SDF will send 1 data group directly to the FDCF, each functional data movement including one entry. The functional size = 1 CFP;
4. The SDF will send 2 data groups directly to the SFTF, each functional data movement including one entry. The functional size = $1 \times 2 = 2$ CFP;
5. FU sends 1 data group and receives another one (2 CFP);
6. The total functional measurement size (direct data movement case) = 10 CFP.

Indirect data movements:

1. Each of four services for the FDOF stores its results in a system buffer to be used by another functional service, and reads some data from the buffer to improve its work. The functional size = 4 services X 2 (Read and Write) = 8 CFP;
2. One service for the FDMF stores its results in a system buffer to be used by another functional service, and reads some data from the buffer to improve its work. The functional size = 1 service X 2 (Read and Write) = 2 CFP;
3. One service for the FDCF stores its results in a system buffer to be used by another functional service, and reads some data from the buffer to improve its work. The functional size = 1 service X 2 (Read and Write) = 2 CFP;

4. Each of two services for the SFTF stores its results in a system buffer to be used by another functional service, and reads some data from the buffer to improve its work. The functional size = 2 services X 2 (Read and Write) = 4 CFP;
5. The total functional size = 8 + 2 + 2 + 4 → 16 CFP.

5.6 Summary

Maintainability is typically described initially as NFR at the system level, and, subsequently, systems engineers must apportion these systems requirements very carefully, as either software or hardware requirements, to conform to the maintainability requirements of the system. Within the ECSS, ISO 9126, and IEEE standards, a number of views and concepts are provided to describe various types of maintainability requirements at the system, software, and hardware levels.

This chapter has collected and organized these concepts into a standard-based model of software-FUR for system maintainability-NFR. This model corresponds to a standard-based model for specifying software-FUR for system maintainability-NFR. This model is based on the generic model of software proposed in COSMIC-ISO 19761, which allows measurement of the functional size of the software maintainability requirements using this COSMIC international standard of measurement.

The proposed standard-based model of software-FUR for system maintainability-NFR is independent of the software type and the languages in which the software-FUR will be implemented. This standard-based model of software-FUR for system maintainability-NFR provides:

- A specification model for each type, or all types, of maintainability requirements. For example, the requirements to be allocated to software for the maintainability failure procedures for system analyzability, the registered failures and software/system malfunctions for system changeability, and for system/software stability and testability;

- A specification measurement model for each type, or all types, of maintainability requirements.

In the absence of such a standard-based model of software-FUR for system maintainability-NFR, such NFR requirements are typically handled in practice much later on in the software development life cycle when at system testing time, users and developers find out that a number of maintainability requirements have been overlooked and additional work has to be expanded to implement them.

CHAPTER 6

INTERFACES: IDENTIFICATION, SPECIFICATION AND MEASUREMENT OF SOFTWARE-FUR DERIVED FROM SYSTEM-NFR

6.1 Introduction

Currently, there exists no standard-based model of software-FUR for system interfaces NFR for the identification and specification of software-FUR for implementing system interfaces requirements (system-NFR) based on the various views documented in international standards and in the literature. Consequently, it is challenging to measure these interfaces-related software-FUR, and take them into account quantitatively for estimation purposes.

The ECSS includes interface requirements as one of sixteen (16) types of non functional requirement (NFR) for embedded and real time software. A number of concepts are provided in the ECSS and IEEE standards to describe the various types of candidate system interface requirements at the system, software, and hardware levels.

This chapter organizes these dispersed system interface concepts into a standard-based model of software-FUR for system interfaces NFR. The availability and the detailed model can facilitate the early identification and specification of the system interface-NFR and their detailed allocation as specific system interface functions to be handled by that allocation to hardware or software, or to a specific combination of the two.

The approach adopted to structuring this model is based on the generic model of software functional requirements proposed in the COSMIC (ISO-19761 2011) model, with which the functional size of the system interface requirements allocated to software can be measured, and to take them into account for estimation purposes.

This chapter focuses on a single type of NFR, that is, system interfaces requirements, and reports on the work carried out to define an integrated view for a standard-based model of

software-FUR for system interfaces NFR based on international standards, including the use of the generic COSMIC (ISO-19761 2011) model of software-FUR.

The interfaces-related views, concepts and terms in the ECSS and IEEE standards have been identified in chapter 3 and should be included in the design of a standard-based model of software-FUR for system interfaces NFR. The elements of interfaces are dispersed in various system views throughout a number of ECSS standards, and are expressed as either – see Figure 6.1:

- System interface functional user requirements (system interface-FUR);
- System interface non functional requirements (system interface-NFR).

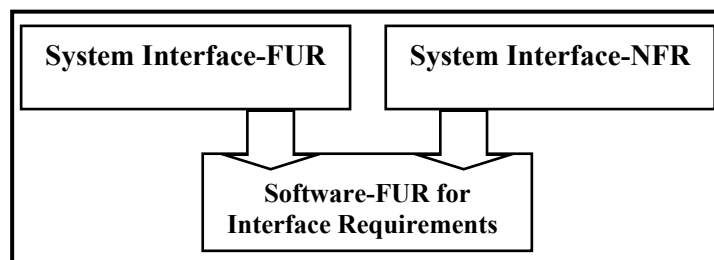


Figure 6.1 Mapping system requirements to software-FUR for an interface

The chapter is organized as follows. Section 5.2 presents a standard-based model of software-FUR for system interfaces NFR. Section 5.3 presents a standard-based model of software-FUR for system interfaces NFR using a service-oriented architecture (SOA). Section 5.4 presents the sizing of the standard-based model of software-FUR for system interfaces NFR. Section 5.5 presents a measurement example. Finally, a summary is presented in section 5.6.

6.2 A standard-based model of software-FUR for system interfaces NFR

The terminologies and concepts of interfaces identified in chapter 3 are mapped here into a proposed standard-based model of software-FUR for system interfaces NFR using the generic FUR model proposed in COSMIC. This COSMIC based model then becomes a

standard-based model of describing the software-FUR from system interfaces based on the ECSS standards.

6.2.1 Mapping system interface views and concepts and terms from standards

Table 6.1 presents the interface requirements that are present either as system requirements in the ECSS standards or as interface-related concepts in IEEE 830 (IEEE-830 1998), each of which could at times be interpreted, and specified, as software-FUR.

We observe that the general identification of interface requirements in all these standards is the same (e.g. user interface requirements, software and hardware interface requirements, and interface communication requirements), while the description of the detailed requirement views for general interface requirements differ from one standard to another.

Table 6.1 Interface requirements in ECSS and IEEE

ID	System interface requirements	System interface functionality	Description of system interface functionality
1	User interface	Logical characteristics of the interface(s) between the system software product and its users	What is needed to allow users (devices and humans) to interact with the system
2	Hardware interface	Hardware configuration	What is needed to ensure support for the hardware and the specific hardware configuration by the system (i.e., logical structure, physical address, and expected behavior)
3	Software interface	System applications Interface specifications through programming languages Interface specifications for each layer of interface socket programming	What is needed to allow communication with other software system components that are not part of the software to be designed (such as operating system, files, database management system, or other application software)

Table 6.1 Interface requirements in ECSS and IEEE (Continued)

ID	System interface requirements	System interface functionality	Description of system interface functionality
4	Communications interface	Communication layers and links	What is needed to allow communication between pieces of system software and software embodied in other systems or components

6.2.2 Interface functions to be specified

The system interface functions to be specified (and the corresponding entities to be measured) are divided into two types of system interface functions that may be allocated to software-FUR – see Table 6.2.

1. System interface components (SIC): components that permit high-level interaction between interface functions.
2. System interface specifications (SIS): specifications that describe the level of interaction required for interface component functions.

Table 6.2 System interface functions that may be allocated to software-FUR

ID	System interface function types	System interface functions
1	System interface components (SIC)	<ul style="list-style-type: none"> • User interface function (UIF) • Hardware interface function (HIF) • Software interface function (SIF) • Communication interface function (CIF)
2	System interface specifications (SIS)	<ul style="list-style-type: none"> • Interface specification function (ISF) • Interface specification Link function (ISLF)

6.2.3 Identification of the system interface function types allocated to software-FUR

In this section, the function types allocated to software-FUR for system interfaces, and the relationships between them are identified.

System Interface Components (SIC)

The system interface components (SIC) are considered to constitute a high-level control interface between the various external parts of the system. The standard-based identifications of system interface components (SIC) include user interfaces, hardware interfaces, communications interfaces, and high-level parts of a software interface.

Figure 6.2 illustrates a system modeling view of data movements for the System Interface Components (SIC):

1. User interface function (UIF): a user interface function is used to exchange data movements between HIF, SIF, and CIF;
2. Hardware interface function (HIF): it exchanges data movements with the user interfaces function (UIF), and with the other sub interface modules. It is used to configure the hardware items with the other parts of the system;
3. Software interface function (SIF): it exchanges data movements with the user interfaces function (UIF), and with the other sub interface modules. It is used to configure the software items, with the other parts of the system;
4. Communication interface function (CIF): it exchanges data movements with the user interfaces function (UIF), and with the sub interface modules. It is used to configure the hardware items or other data groups, with the other parts of the system.

HIF, SIF, and CIF use intermediary services to interact with one another to deliver different types of data interface (⊗ symbol in Figure 6.2).

HIF, SIF, and CIF send and receive data groups (i.e., Entry or Exit) from/to interface specification function (ISF) in system interface specification (SIS) (Function Type 2).

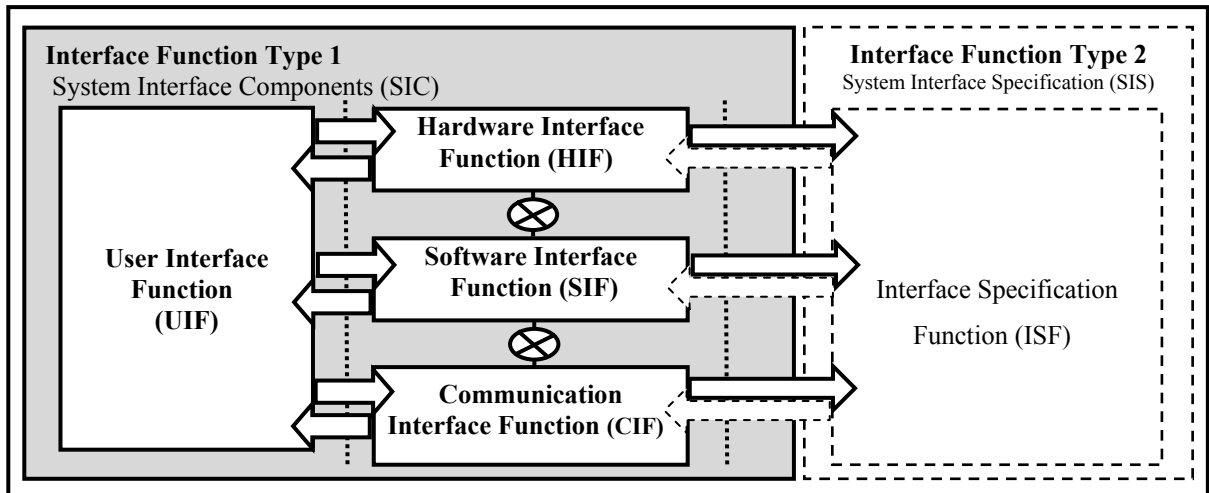


Figure 6.2 System interface components (SIC): a system modeling view

Figure 6.3 illustrates a COSMIC modeling view of the data movements for the system interface components (SIC) (Function Type 1):

1. UIF sends a data group (i.e., Entry) to an HIF, SIF, or CIF;
2. HIF, SIF, and CIF send and receive data groups (i.e., Entry or Exit) to interface specification function (ISF) in Function Type 2;
3. HIF, SIF, and CIF send and receive data groups (i.e., Entry or Exit) between them using intermediary services;
4. HIF, SIF, and CIF send data groups (i.e., Exit) to UIF.

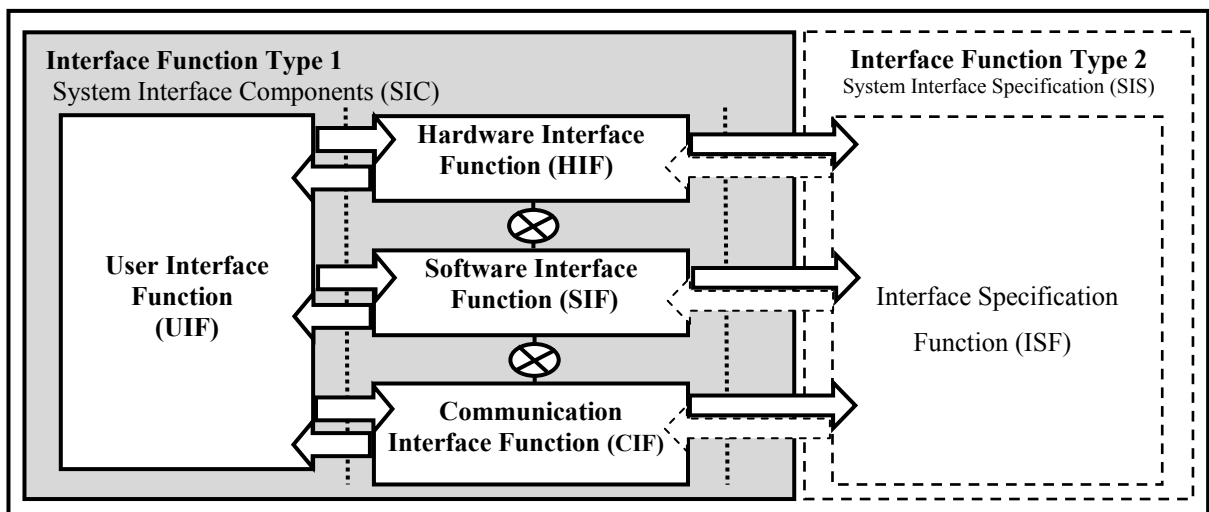


Figure 6.3 System interface components (SIC): COSMIC modeling view

System Interface Specifications (SIS)

According to the ECSS, most interfaces are software-to-software; therefore, system interface specifications may be considered to describe SIC at an internal or detailed level, e.g. programs running on the operating system or system device driver programs. Such interfaces are used to manage orders from HIF, SIF, and CIF to provide these components with detailed information, such as constants, data types, types of procedures, exception specifications, and method signatures, in order to build a network of interfaces inside the system.

Figure 6.4 illustrates a system modeling view of the data movements for the System Interface Specifications (SIS). These specifications can be divided into:

1. Interface specifications function (ISF): it exchanges data movements for the HIF, SIF, and CIF in system interface components (SIC) function type 1 and it exchanges data movements for the interface specification link function (ISLF) in function type 2;
2. Interface specification link function (ISLF): it exchanges data movements for interface specification function (ISF), and it reads and writes data from/to persistent storage.

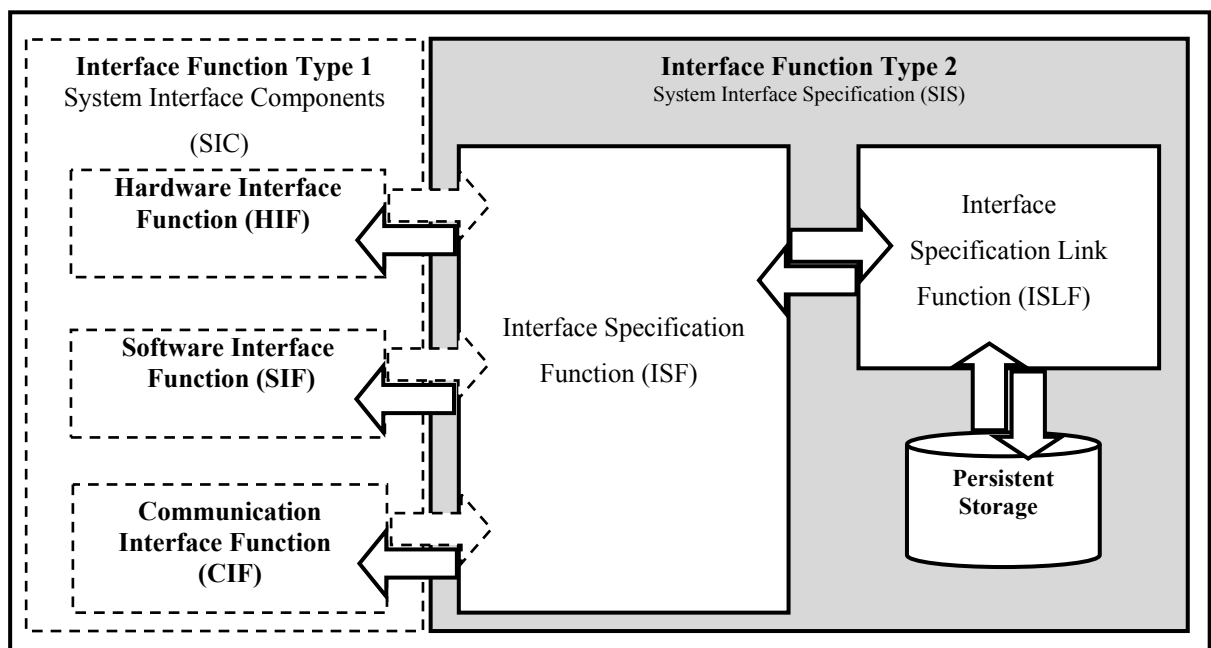


Figure 6.4 System interface specifications (SIS): a system modeling view

Figure 6.5 illustrates a COSMIC modeling view of the data movements for the system interface specifications (SIS):

1. ISF send a data group (i.e., Exit) to an HIF, SIF, or CIF in Function Type 1 (SIC);
2. ISF sends and receives data groups (i.e., Entry or Exit) with ISLF;
3. ISLF sends and receives data groups (i.e., Entry or Exit) with ISF;
4. ISLF reads and writes data groups to/from persistent storage.

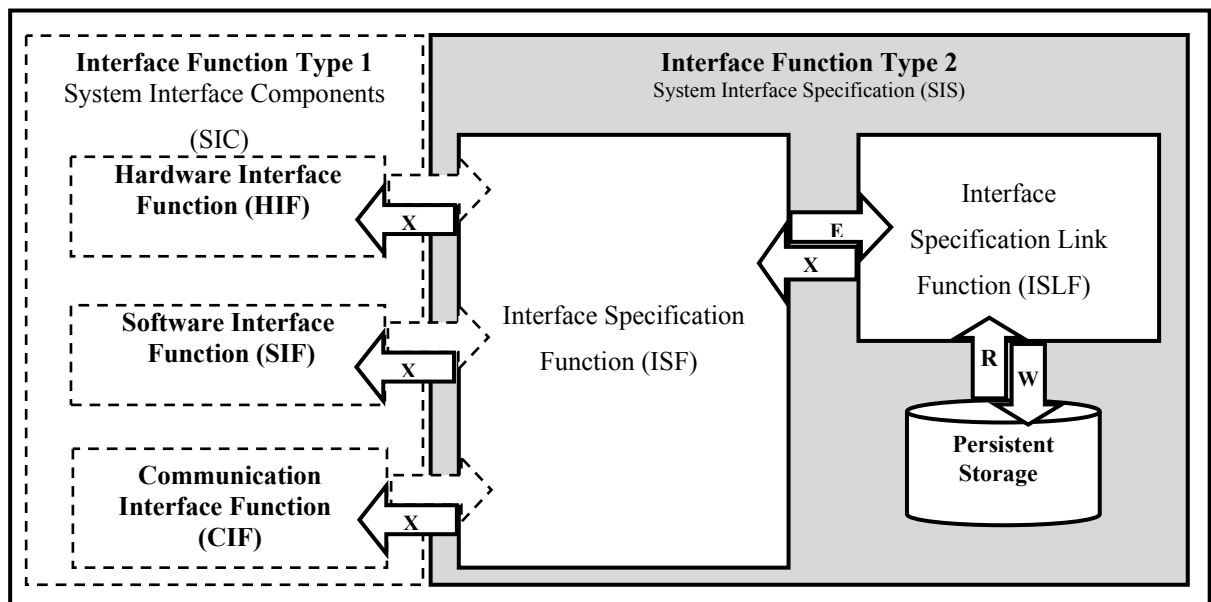


Figure 6.5 System interface specifications (SIS): COSMIC modeling view

Model of the functions types relationships based on system and COSMIC

Figure 6.6 presents an overview of the relationships between the function types for system interfaces that may be allocated to software-FUR. Specifically, the system interface requirements model is composed of six functions grouped into two Function Types. The data flow on the model is also divided into direct data flows and the intermediary services data flows:

1. The SIC model (Function Type 1) can be used to specify the data flows between four sub functions and the data flows with the other functions on the system interface model (see Figure 6.6);

2. The SIS model (Function Type 2) can be used to specify the data flows between the two sub functions and the data flows with the other functions on the system interface model (see Figure 6.6).

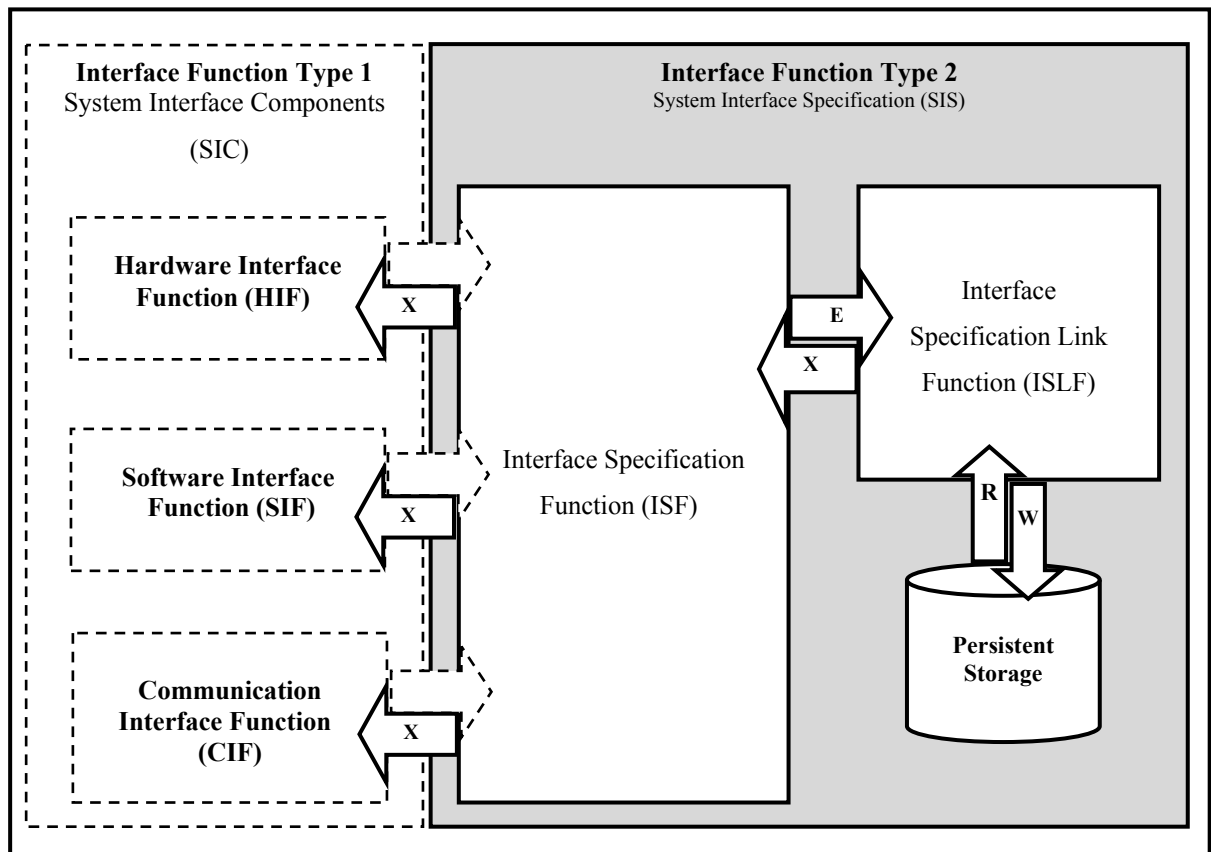


Figure 6.6 System modeling view for the system interface requirements

Figure 6.7 presents an overview of the relationships between the function types in the interface software-FUR, using COSMIC for graphical representation. Specifically:

1. The SIC model can be used to specify and measure the function size of the system user interface function (UIF) from the received/sent data groups from/to the HIF, SIF, and CIF – see Figure 6.7;
2. The SIS model can be used to specify and measure the functional size from the received/sent data groups from/to interface specifications function (ISP), and the interface specifications link function (ISLF) – see Figure 6.7.

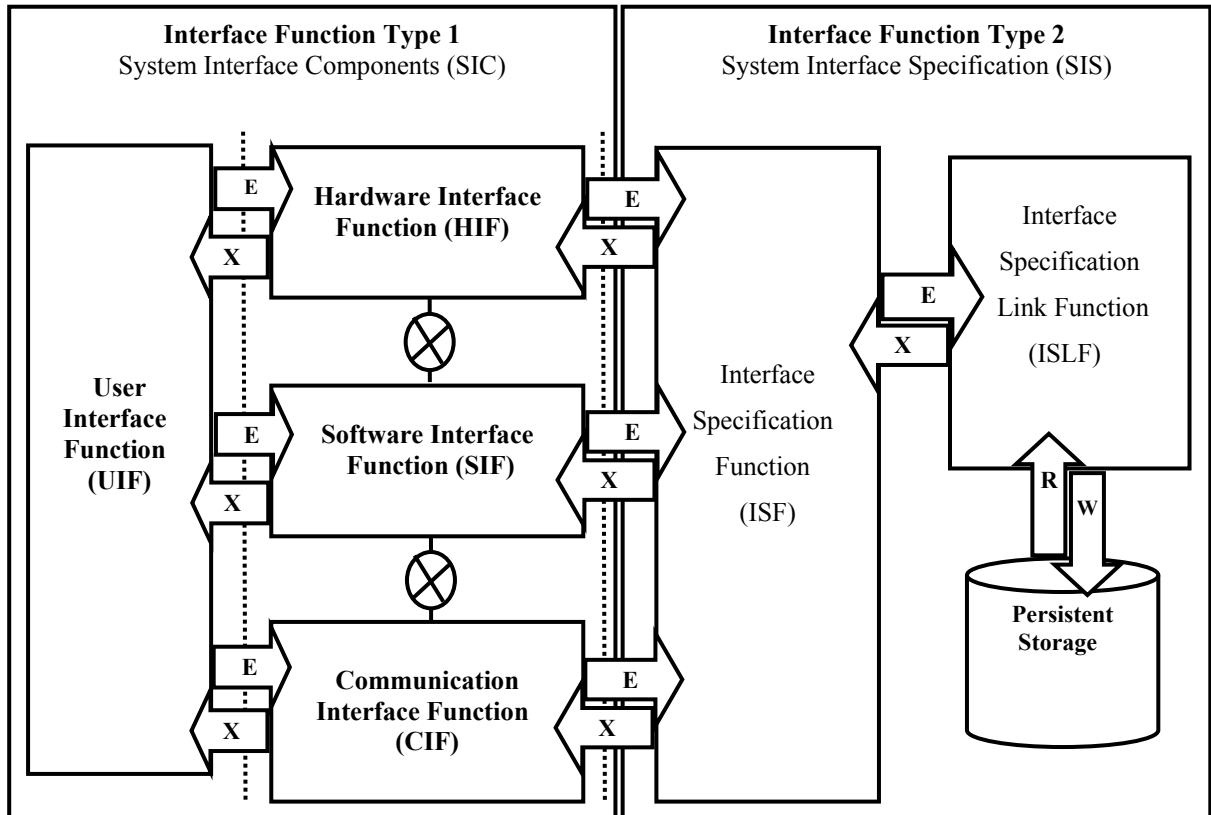


Figure 6.7 A standard-based model of software-FUR for system interfaces NFR (Function level)

6.3 A standard-based model of software-FUR for system interfaces NFR using SOA

In this chapter, Figure 6.7 illustrates the COSMIC standard-based model of software-FUR for system interfaces NFR. This model describes the important concepts and relationships for system interface requirements, as defined in the ECSS and IEEE standards. In this section, a standard-based model of software-FUR for system interfaces NFR using a service oriented architecture (SOA) is built to show a more complete picture, which includes showing what is involved in instantiating the modeled entities in practice – for more details, see .

Figure 6.8 illustrates a COSMIC standard-based model of software-FUR for system interfaces NFR using an SOA. This model is built based on Figure 6.7, and on the role of the COSMIC-SOA explained in (COSMIC 2010).

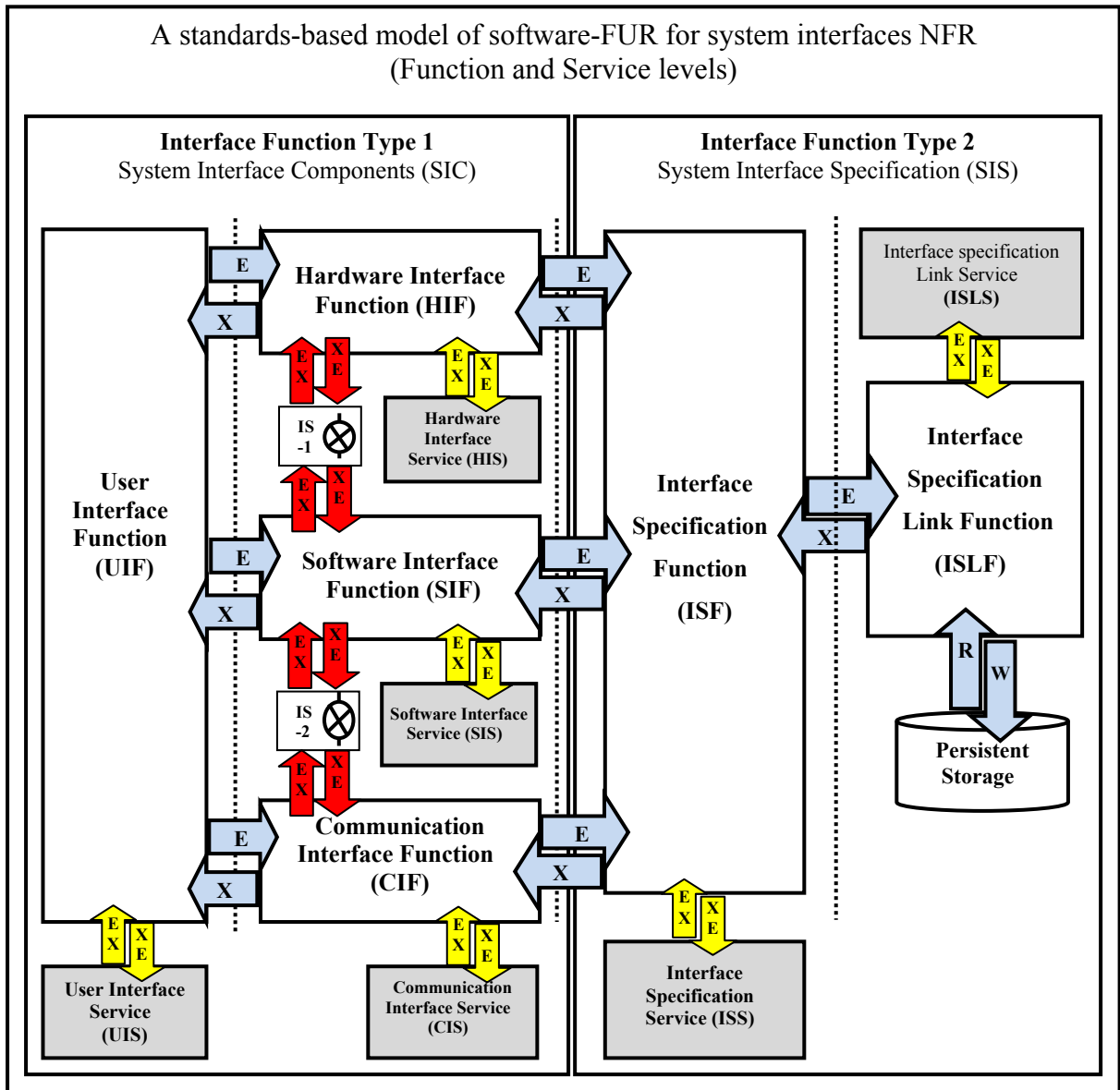


Figure 6.8 A standard-based model of software-FUR for system interfaces NFR (Function and Service levels)

6.4 Sizing of the standard-based model of software-FUR for system interfaces NFR

The specification of software-FUR for system interfaces in a project is a specific instantiation of the proposed standard-based model of software-FUR for system interfaces NFR described in Figure 6.8. When the software specification document is at the level of the movements of

data groups, then these functional requirements can be directly measured using the COSMIC measurement rules. The measurement example presented next illustrates a reference instantiation of the specification and measurement model of software-FUR for system interfaces in an SOA context for a single data group for all the possible flows of data groups identified.

The measurement example in this section explains how to use the proposed reference model of system interfaces to size a hypothetical model composed of all the kinds of software-FUR described in the framework.

6.4.1 Measurement of exchange messages for system interface

There are six functionality types of system interfaces, each interacting with its own services, for the measurement of exchange services for system interfaces using COSMIC-SOA – see Figure 6.8. According to COSMIC-SOA, each functional process may interact with its own service by sending and receiving data movements (i.e., Entry and Exit). Table 6.3 illustrates a measurement example for the interactions between a functional process and its own functional service process.

Table 6.3 COSMIC-SOA measurement example for the interactions between a functional process and its own functional service process

COSMIC-SOA Types		Data Movement Description	Data Movement Type
Functional Process	Functional Service		
Hardware Interface Function (HIF)	Hardware Interface Service (HIS)	HIF sends a data group to HIS	X
		HIS receives a data group from HIF	E
		HIS sends a data group to HIF	X
		HIF receives a data group from HIS	E
The total functional size			4 CFP

Table 6.4 illustrates the measurement results of the standard-based model of software-FUR for system interfaces NFR for interactions between a system interface functional process and

its own service processes, i.e., a hardware interface function (HIF) interacts with its own service process, the hardware interface service (HIS). The measurement result for this operation is equal to 4 CFP for each interaction between a functional process and its own functional service process. The total measurement result is equal to 24 CFP – see the yellow shaded arrows in Figure 6.8.

Table 6.4 Measurement for exchange messages

Function ID	Exchange Services for System Interfaces		No. of Data Movements
	Functional Process	Functional Service	
1	User Interface Function (UIF)	User Interface Service (UIS)	4
2	Hardware Interface Function (HIF)	Hardware Interface Service (HIS)	4
3	Software Interface Function (SIF)	Software Interface Service (SIS)	4
4	Communication Interface Function (CIF)	Communication Interface Service (CIS)	4
5	Interface specification function (ISF)	Interface specification Service (ISS)	4
6	Interface specification Link function (ISLF)	Interface specifications Link Service (ISLS)	4
The total functional size			24 CFP

6.4.2 Measurement of intermediary services for system interface

In this section, and based on Figure 6.8, when a functional process service requires data that are available via another functional process service, the former calls upon a functional process of the intermediary service. According to the COSMIC-SOA model of measurement for system interfaces, the types of data movements that can be used by the intermediary service are Entries and Exits – see Table 1.4 in chapter 1.

Table 6.5 illustrates a measurement example for the intermediary service between a functional process and its own functional service process.

Table 6.5 COSMIC-SOA measurement example for the intermediary service

COSMIC-SOA Intermediary Service		Data Movement Description	Data Movement Type
Functional Service	Functional Service		
Hardware Interface Service (HIS)	Software Interface Service (SIS)	HIS sends a data group to IS-1	X
		IS-1 receives a data group from HIS	E
		IS-1 sends a data group to SIS	X
		SIS receives a data group from IS-1	E
		SIS sends a data group to IS-1	X
		IS-1 receives a data group from SIS	E
		IS-1 sends a data group to HIS	X
		HIS receives a data group from IS-1	E
The total functional size			8 CFP

Note: IS-1 is the first intermediary service in Figure 6.8.

Table 6.6 illustrates the COSMIC-SOA measurement results for intermediary services – see the red shaded arrows in Figure 6.8. This table presents an instantiation of a single data group for all possible flows of the data groups identified above, and listed as a data movement example for one intermediary service in Table 6.5. For this interface requirement, the measurement results are equal to 8 CFP.

Table 6.6 COSMIC-SOA measurement for intermediary services

Intermediary Service ID	Intermediary Services for System Interfaces		No. of Data Movements
	Functional Process	Functional process	
1	Hardware Interface Service (HIS)	Software Interface Service (SIS)	8
2	Software Interface Service (SIS)	Communication Interface Service (CIS)	8
The total functional size			16 CFP

6.4.3 Measurement of the direct and indirect data movements for system interface

This section is based on Figure 6.8, which illustrates the possible flows of data between components in the same layer, i.e., between peer components (where a component may be an application or a service). This section shows direct and indirect exchanges of data between

components. If components exchange data directly, the measurer will identify the Exit and/or Entry data movements. An indirect exchange of data between components means that a service in one component writes data which are subsequently read by a service in another component.

Specifically, Table 6.7 illustrates the measurement results for standard-based model of software-FUR for system interfaces NFR for the exchange data movements at function level or in service architecture layers – see Figure 6.8. This table presents an instantiation of this operation. The measurement results are equal to 16 CFP – see the blue shaded arrows in Figure 6.8.

Table 6.7 Measurements of direct and indirect data groups for system interfaces

Interface Function	Data Movement Description	Data Movement Type
User Interface Function (UIF)	• UIF sends a data group to HIF	E
	• UIF sends a data group to SIF	E
	• UIF sends a data group to CIF	E
	• UIF receives a data group from HIF	X
	• UIF receives a data group from SIF	X
	• UIF receives a data group from CIF	X
Hardware Interface Function (HIF)	• HIF sends a data group to ISF	E
	• HIF receives a data group from ISF	X
Software Interface Function (SIF)	• SIF sends a data group to ISF	E
	• SIF receives a data group from ISF	X
Communication Interface Function (CIF)	• CIF sends a data group to ISF	E
	• CIF receives a data group from ISF	X
Interface specification function (ISF)	• ISF sends a data group to ISLF	E
	• ISF receives a data group from ISLF	X
Interface specification link function (ISLF)	• ISLF reads a data group from persistent storage (PS) • ISLF writes a data group to (PS)	R & W
The total functional size		16 CFP

6.5 A Measurement Example

The specification of software-FUR for system interface requirements in a particular project is a specific instantiation of the proposed model described in Figure 6.8. When the software

specifications document is at the level of the movements of data groups, then these functional requirements can be directly measured using the standard-based model of software-FUR for system interfaces NFR (Function and Service levels).

Example: The set of functional requirements allocated to software for the system interface requirements for a specific instantiation is the following:

1. UIF, HIF, and SIF call upon their own functional services to exchange their messages;
2. HIF uses an intermediary service with SIF;
3. UIF send one data group to HIF and another data group to SIF.

The Functional Measurement Solution

Based on Figure 6.8 for the standard-based model of software-FUR for system interfaces NFR using an SOA for specifying data movements, the measurement procedure to determine the functional size for interfaces on function types 1 and 2 for this example is as follows:

6.5.1 Measurement of exchange messages

The functional processes (UIF, HIF, and SIF) interacting with the functional services for UIS, HIS, and SIS in this example include the data movements shown in Figure 6.8 (arrows shaded in yellow). With the help of Table 6.4 the functional size measurement results are presented in Table 6.8.

Table 6.8 Measurement results for the interactions between three functional processes

Function ID	Exchange messages for System Interfaces		No. of Data Movements
	Functional Process	Service Process	
1	User Interface Function (UIF)	User Interface Service (UIS)	4
2	Hardware Interface Function (HIF)	Hardware Interface Service (HIS)	4
3	Software Interface Function (SIF)	Software Interface Service (SIS)	4
The Functional Size			12 CFP

6.5.2 Measurement of intermediary services

The functional process services (HIS and SIS) use intermediary services to interact with other functional process services in this example, including the data movements shown in Figure 6.8 (red arrows). With the help of Table 6.6, the functional size measurement results are presented in Table 6.9.

Table 6.9 Measurement results of intermediary services

Intermediary Services ID	Intermediary Services		No. of Data Movements
	Functional Process	Functional Process	
1	Hardware Interface Service (HIS)	Software Interface Service (SIS)	8
The Functional Size			8 CFP

6.5.3 Measurement of data movements

Based on Figure 6.8 (blue arrows) and Table 6.7, the functional size measurement results are presented in Table 6.10 for the data movements identified by the measurer for this example.

Table 6.10 Measurements of direct and indirect data movements for system interfaces.

Interface Functions	Data Movement Description	Data Movement
User Interface Function (UIF)	• UIF sends a data group to HIF	E
	• UIF sends a data group to SIF	E
	• UIF receives a data group from HIF	X
	• UIF receives a data group from SIF	X
Hardware Interface Function (HIF)	• HIF sends a data group to ISF	E
	• HIF receives a data group from ISF	X
Software Interface Function (SIF)	• SIF sends a data group to ISF	E
	• SIF receives a data group from ISF	X
Interface specification function (ISF)	• ISF sends a data group to ISLF	E
	• ISF receives a data group from ISLPF	X
Interface specifications Link function (ISLF)	• ISLF reads a data group from persistent storage (PS)	R & W
	• ISLF writes a data group to (PS)	
The functional size		12 CFP

6.6 Summary

The Interface requirements are typically described initially as non functional requirements at the system level, and system engineers must subsequently apportion these system requirements very carefully as either software or hardware requirements to conform to the interface requirements of the system. A number of views and concepts are provided in the ECSS and IEEE standards to describe various types of candidate interface requirements at the system, software, and hardware levels.

This chapter has introduced a standard-based model of software-FUR for system interfaces NFR (Function and Service levels) for specifying and measuring software requirements for the functions needed to address the system's interface requirements.

The main contribution of this chapter is our proposed standard-based model of software-FUR for system interfaces NFR. This model can be considered as a kind of reference model for the identification of system interface requirements, and can be used for their allocation to software functions implementing such requirements. The structure of the proposed model is based on the generic model of software adopted by the COSMIC measurement standard, the necessary information for measuring their functional size is readily available, and an example has been presented of a specific instantiation of this reference model.

Specifically, the standard-based model of software-FUR for system interfaces NFR presented in this chapter is based on:

- The ECSS and IEEE standards for the description of the NFR for system interfaces;
- The COSMIC measurement model of software-FUR.

The proposed standard-based model of software-FUR for system interfaces NFR is independent of the software type and the languages in which the software-FUR will be implemented. The proposed standard-based model of software-FUR for system interfaces NFR provides:

- A specification model for each type, or all types, of interface requirements: for example; the requirements to be allocated to software for the system interface components;
- A specification measurement model for each type, or all types, of interface requirements.

CHAPTER 7

THE OTHER ELEVEN TYPES OF SYSTEM-NFR IN ECSS: SPECIFICATION AND MEASUREMENT MODELS

7.1 Introduction

This chapter presents the other eleven (11) standard-based model of software-FUR for system -NFR that can be allocated to software FUR. The detailed modeling procedures for these standard-based models are presented in the Annex II.

This chapter is organized as follows:

Section 7.2 presents the portability system requirements.

Section 7.3 presents the operations system requirements.

Section 7.4 presents the configuration system requirements.

Section 7.5 presents the data definitions and database system requirements.

Section 7.6 presents the adaptation and installation system requirements.

Section 7.7 presents the design and implementation constraints system requirements.

Section 7.8 presents the performance system requirements.

Section 7.9 presents the security system requirements.

Section 7.10 presents the safety system requirements.

Section 7.11 presents the resources system requirements.

Section 7.12 presents the human factors requirements.

A summary is presented in section 7.13.

7.2 Portability system requirements

This section maps the portability terminologies found throughout the ECSS, IEEE, and ISO standards from chapter 3 into a proposed standard-based model of software-FUR for system portability-NFR, through the use of the generic model of FUR proposed in the COSMIC model presented in chapter 1.

7.2.1 Mapping views and concepts for portability from ECSS, ISO, and IEEE

Based on a synthesis of the various definitions, key views and concepts presented in chapter 3 the system portability requirements are listed in Table 7.1. It is important to note that Table 7.1 includes software, data, and hardware components which are interconnected. If the system can run on two or more kinds of devices, or with two or more kinds of operating systems that are easily or conveniently transported, then system portability is achieved. So we consider these components as environments for the software-FUR for the system portability-NFR – see also Table 7.2.

Table 7.1 Portability requirements in ECSS, ISO, and IEEE

ID	System portability requirements
1	• Isolating software system calls
2	• Independence of the operating system
3	• Independence of the middleware
4	• Independence of the programming language virtual machine
5	• Independence of browsers
6	• Client independence
7	• Server independence
8	• Storage independence
9	• Network independence
10	• Database independence
11	• Distributed data base management system (DDBMS)

In Table 7.2, portability requirements must be identified for each environment (from environment 1 to environment n), when required. In addition, the types of portability requirements should be identified for each environment and must be allocated to: software components, hardware components, and data components.

Table 7.2 Portability types, by environment

Environment 1	...	Environment n
<ul style="list-style-type: none"> – Software Components in Environment 1 <ul style="list-style-type: none"> ○ Independence of the operating system ○ Independence of the middleware ○ Independence of the programming language virtual machine ○ Independence of browsers – Hardware Components in Environment 1 <ul style="list-style-type: none"> ○ Independence of Client ○ Independence of Server ○ Independence of Storage ○ Independence of Network – Data Components in Environment 1 <ul style="list-style-type: none"> ○ Independence of Database ○ Distributed data base management system (DDBMS) 	...	<ul style="list-style-type: none"> – Software Components in Environment n <ul style="list-style-type: none"> ○ Independence of the operating system ○ Independence of the middleware ○ Independence of the programming language virtual machine ○ Independence of browsers – Hardware Components in Environment n <ul style="list-style-type: none"> ○ Independence of Client ○ Independence of Server ○ Independence of Storage ○ Independence of Network – Data Components in Environment n <ul style="list-style-type: none"> ○ Independence of Database ○ Distributed data base management system (DDBMS)

7.2.2 Software portability functions to be specified

The functions and corresponding entities to be specified and measured for software portability are listed in Table 7.3. Portability component functions and the corresponding entities for portability are represented by the environment of these components. Portability environment function and the corresponding entities are represented by the capability of the isolated software pieces in the environment to call each other.

Table 7.3 Portability functions that may be allocated to software

ID	Portability Type	Portability functions
1	Portability Components	<ul style="list-style-type: none"> • Independence of the operating system function • Independence of the middleware function • Independence of the programming language virtual machine function • Independence of the browser function • Client independence function • Server independence function • Storage independence function • Network independence function • Database independence function • Distributed data base management system (DDBMS) function
2	Portability Environment	<ul style="list-style-type: none"> • Isolating software system calls function

7.2.3 Identification of the function types in software portability

In this section, the portability function types are identified based on the findings of the portability functions as identified in the previous section. The system portability requirements allocated to software-FUR are divided into portability components and environments; each type in this division has its own functions. The proposed portability function types are illustrated in system and COSMIC modeling views, in order to propose a standard-based model of software-FUR for system portability-NFR based on the proposed system modeling view.

The proposed portability functions can be divided into four function types, three of them specified for portability components and the fourth for portability environments. Table 7.4 illustrates these portability function types, based on the identified portability functions.

Table 7.4 Function types for portability functions that may be allocated to software

ID	Function Types	Portability Functions
1	System Software Components	<ul style="list-style-type: none"> • Independence of the operating system function (IOSF) • Independence of the middleware function (IMF) • Independence of the programming language virtual machine function (IPLVMF) • Independence of the browser function (IBF)
2	System Data Components	<ul style="list-style-type: none"> • Independence of the database function (IDF) • Distributed data base management system function (DDBMSF)
3	System Hardware Components	<ul style="list-style-type: none"> • Independence of the client function (ICF) • Independence of the server function (ISF) • Independence of the storage function (ISTF) • Independence of the network function (INF)
4	Isolating System Calls	<ul style="list-style-type: none"> • Isolating software system calls function (ISSCF)

7.2.4 A standard-based model of software-FUR for system portability using SOA

Figure 7.1 illustrates a standard-based model of software-FUR for system portability-NFR using an SOA. This model is built based on the proposed portability functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

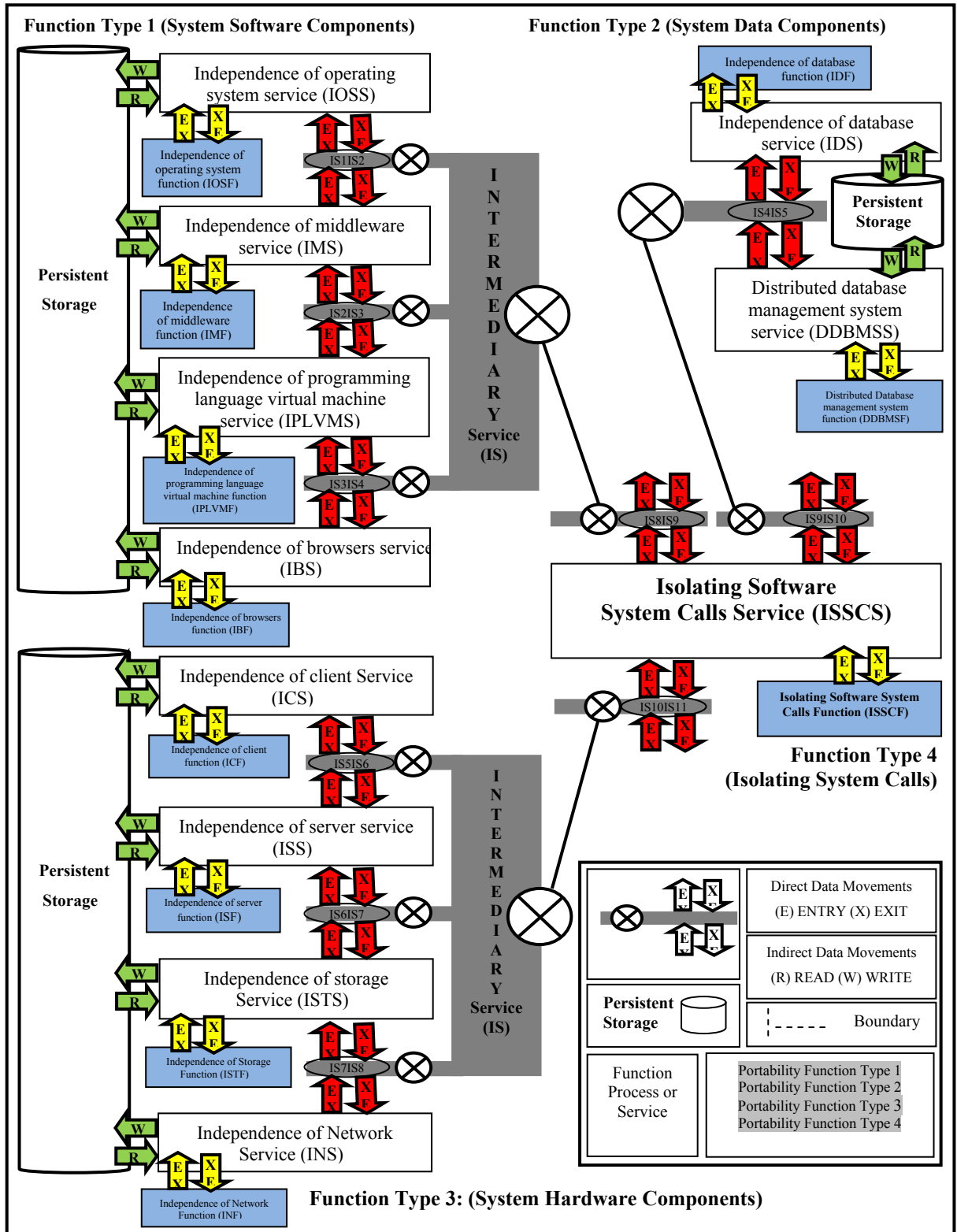


Figure 7.1 Standard-based model of software-FUR for system portability-NFR

7.3 Operations system requirements

This section assembles the terminologies and concepts of system operations dispersed throughout the ECSS standards. There are two types of system-related operations requirements that can be derived from the ECSS standards series: system operations mode, and system transitions mode.

7.3.1 Mapping system operations views and concepts from ECSS and IEEE standards

Table 7.5 presents the two types of system operations requirements, and related functions, which are included as system requirements in the ECSS and IEEE standards. These could, at times, be interpreted and specified, as software FUR:

1. System operations mode: this refers to the expected operations for the executed functions occurring in the system. The system operations mode consists of the inter-operational functions (IOPF) and the operational function events (OPFE);
2. System transitions mode: this refers to the expected data and control operations via the interface functionality that could occur in the system. The system transitions mode consists of operational data interface functions (OPDIF) and operational control interface functions (OPCIF).

Table 7.5 System operations FUR in the ECSS standards series

ID	Types of System Operations	Operations Functions to be Specified
1	System operations mode	<ul style="list-style-type: none"> • Inter-operational function (IOPF) • Operational function event (OPFE)
2	System transitions mode	<ul style="list-style-type: none"> • Operational data interface function (OPDIF) • Operational control interface function (OPCIF)

According to ECSS standards, the functions relationships across these two modes, as illustrated in Figure 7.2 are the following:

1. The inter-operational function (IOPF) in system operations mode, which are controlled by the operational control interface function (OPCIF) in system transitions mode. This relationship will be referred to the ‘System Operational Control’, or Function Type 1;
2. The operational function event (OPFE) in system operations mode, which sends and receives data movements from the operational data interface function (OPDIF) in system transitions mode. This relationship will be referred to the ‘System Operational Data’, or Function Type 2

For example, in embedded and real-time software:

1. A system scheduler sends distribution routines which form the operational control interface and the inter-operational functions;
2. The system device routines form the operational data interface and the operational function events.

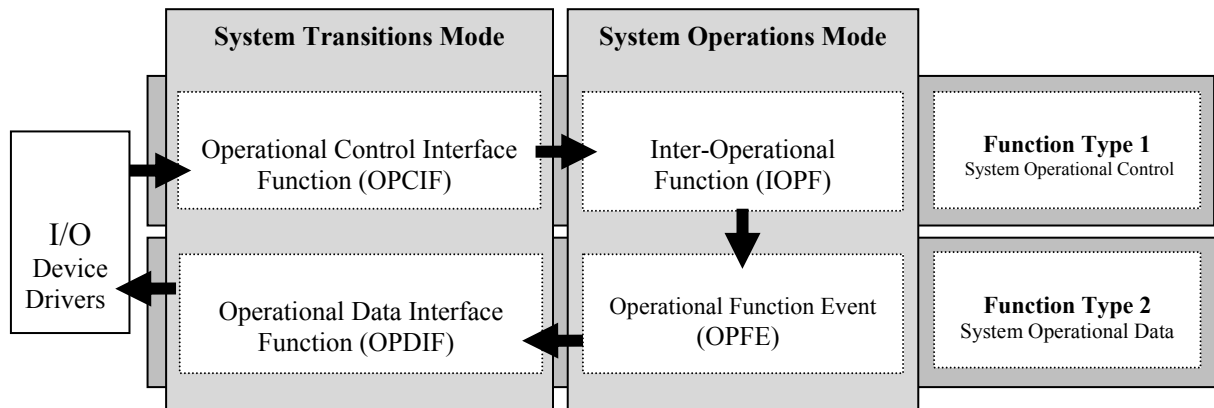


Figure 7.2 System operations functions and function types

In the next section, these terminologies are mapped into a proposed standard-based model of software-FUR for system operations-NFR, using the generic FUR model proposed in COSMIC – ISO 19761. This model is used for describing the software-FUR from system operations requirements based on the ECSS standards.

7.3.2 A standard-based model of software-FUR for system operation-NFR using SOA

Figure 7.3 illustrates a standard-based model of software-FUR for system operations-NFR using an SOA. This model is built based on the proposed system operations requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010)

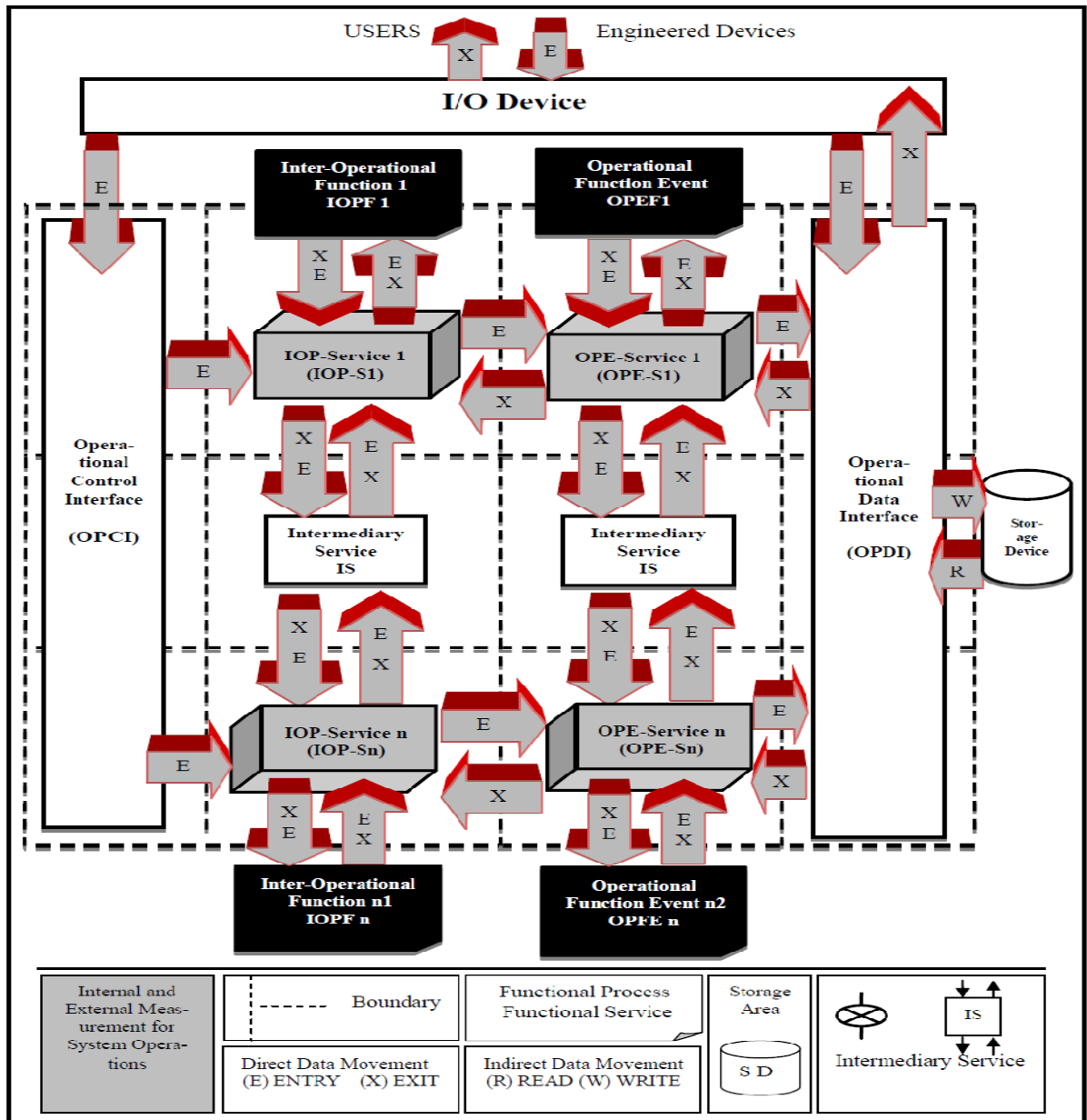


Figure 7.3 standard-based model of software-FUR for system operations-NFR Using an SOA

7.4 Configuration System requirements

This section assembles the terminologies and concepts associated with the configuration elements that are dispersed throughout the ECSS standards and the SWEBOK Guide (ISO-19759 2004). These terminologies are mapped to a standard-based model of software-FUR for system configuration NFR, through the use of the generic model of FUR proposed in the COSMIC model.

7.4.1 Mapping system configuration views and concepts from ECSS standards

From a synthesis of the previous configuration-related definitions, views, and concepts in the ECSS and the ISO 19759 standards, we can draw the following conclusions:

1. They all consider configuration as an important part of the design;
2. They all mention control configuration items or configuration elements, such as:
 - Control flow for operational functions;
 - The data flow register in each operational function.

The software-FUR for system configuration-NFR based on the previous mapping is presented in Table 7.6.

Table 7.6 Software-FUR for system configuration NFR

ID	Software-FUR for System Configuration NFR
1	Configuration control flow function
2	Configuration data flow function
3	Register data transfer function
4	Operational functions

Two types of configuration requirements must be identified:

1. Configuration control flows: the relationships between the operational functions for the configuration items or elements;

2. Configuration data flows: partition of an application into pieces that can be configured individually on configurable hardware or in software.

7.4.2 Configuration function types and functions to be specified

The configuration functions to be specified are divided into configuration data and control flows – see Table 7.7:

1. The configuration data flow specifies the register data that could come into the system view;
2. The configuration control flow specifies the expected operational functions in use in the system.

The ECSS view of system configuration NFR is that of a secure environment, including data flows and control flows. The ECSS view of software-FUR for system configuration NFR within a secure environment includes:

1. Register data transfer, containing a transfer history extraction unit, which extracts transfer history information from data subjected to data transfer each time the data transfer is performed, the extracted transfer history information being separate from the data subjected to data transfer in the secure environment for the system configuration NFR;
2. Operational functions, defining an area of responsibility within an operational function in a hierarchical structure in the secure environment for the system configuration NFR.

Table 7.7 Configuration functions that may be allocated to software

Candidate Function Types	Configuration Type	Configuration Functions
Function type 1	Configuration Data Flow	Register data transfer function
Function type 2	Configuration Control Flow	Operational functions

7.4.3 A standard-based model of software-FUR for system configuration-NFR using an SOA

Figure 7.4 illustrates a standard-based model of software-FUR for system configuration-NFR using an SOA. This model is built based on the system configuration requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

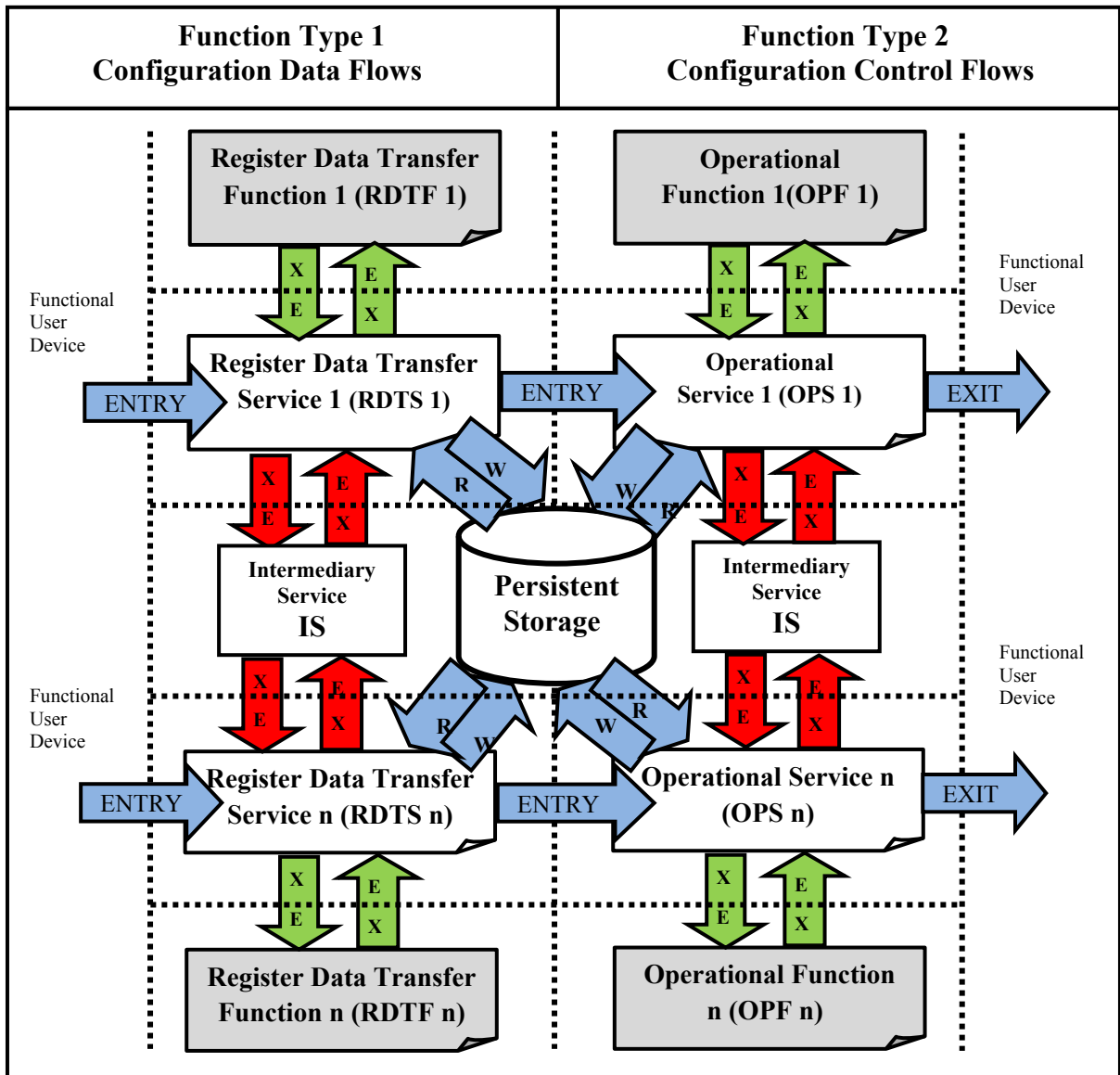


Figure 7.4 A standard-based model of software-FUR for system configuration-NFR using an SOA

7.5 Data definitions and database system requirements

This section assembles the terminologies and concepts of data definition and database dispersed throughout the ECSS standards. These terminologies are mapped into a proposed standard-based model of software-FUR for system data definition and database -NFR using the generic FUR model proposed in COSMIC. This model for describing the software-FUR from system data definition and database requirements is based on the ECSS standards.

7.5.1 Mapping data definition views and concepts from ECCS standards

Table 7.8 presents the functions to address system data definition and database requirements that are present as system requirements in the ECSS standard: each of these could be interpreted, and specified, at times as software-FUR.

Table 7.8 Functions to address system data definition and database requirements

ID	Functions to address system data definition and database requirements
1	Function to identify event
2	Function to identify parameter
3	Function to identify system element
4	Function to identify reporting data Function to identify activity
5	Function to identify simple value
6	Function to identify record value
7	Function to identify simple type
8	Function to identify complex type
9	Function to identify configuration data
10	Function to identify monitoring data
11	Function to identify control data

Various types of system-related data definition and database requirements can be derived from the following set of concepts:

1. System data items (SDI):
 - System entity types (SET);
 - System value types (SVT);

- System data types (SDT).
2. System product data schema (SPDS).

Table 7.9 presents various typical system data definition and database functions (middle column) for system data definition and database requirements and corresponding software functions (right-hand side column) that may be specified to implement such data definition and database functions for the system data definition and database requirements (and corresponding entities to be measured).

Table 7.9 System data definition requirements and related software functions

ID	Function types	System functions for DD and DB	Software functions for data definition and database requirements
1	Function Type 1 System data items (SDI)	System entity types (SET)	<ul style="list-style-type: none"> • Function to identify event (EF) • Function to identify parameter (PF) • Function to identify system element (SEF) • Function to identify reporting data (RDF) • Function to identify activity (AF)
		System value types (SVT)	<ul style="list-style-type: none"> • Function to identify simple value (SVF) • Function to identify record value (RVF)
		System data types (SDT)	<ul style="list-style-type: none"> • Function to identify simple type (STF) • Function to identify complex type (CTF)
2	Function Type 2 System product data schema (SPDS)		<ul style="list-style-type: none"> • Function to identify configuration data (SCDF) • Function to identify monitoring data (SMDF) • Function to identify control data (SCDF1)

7.5.2 A standard-based model of software-FUR for system data definition and database -NFR using an SOA

Figure 7.5 illustrates a standard-based model of software-FUR for system data definition and database-NFR using an SOA. This model is built based on the system data definitions and database requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

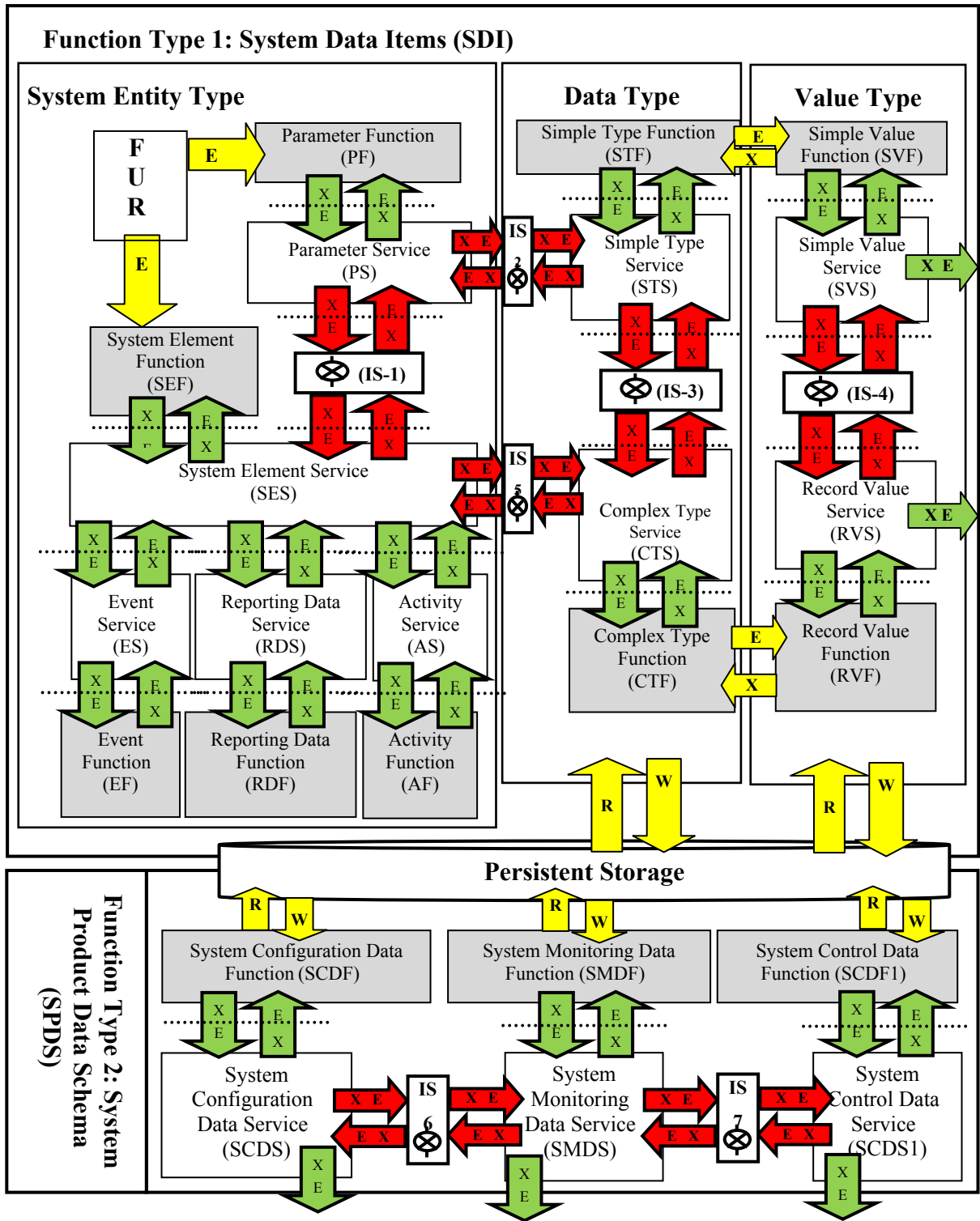


Figure 7.5 A standard-based model of software-FUR for system Data definition and database -NFR using an SOA

7.6 Adaptation and installation system requirements

This section assembles the dispersed terminologies and concepts of adaptation and installation dispersed throughout ECSS, IEEE and ISO standards into a proposed standard-based model of software-FUR for system adaptation and installation-NFR using an SOA through the use of the generic model of FUR proposed in the COSMIC model. This model can then be used for describing the software adaptation and installation requirements (i.e., from system-NFR into software-FUR) based on ECSS.

7.6.1 Mapping the adaptation and installation views and concepts from standards

Table 7.10 presents the system adaptation and installation requirements that are present either as system requirements in the ECSS standards or as adaptation and installation-related concepts in ISO 9126: each of these could be interpreted, and specified, at times as software FUR.

Table 7.10 Adaptation and installation in ECSS & ISO 9126

ID	System adaptation and installation requirements
1	Software Data Structure
2	Registered Data Transfer
3	Control Data Transfer
4	Set Data Transfer with System Resources
5	Operational Environment
6	Localizing I/O Resources
7	Host-Target Platform
8	Memory Resources
9	Storage Resources
10	Transmission Resources

Table 7.11 presents various typical procedures (left-hand side column) for system adaptation and installation requirements and corresponding software functions (right-hand side column) that may be specified to implement such procedures for the three types of system adaptation and installation requirements.

Table 7.11 System adaptation and installation requirements related software functions

ID	System adaptation and installation requirements	Software functions
1	System Software Environment	<ul style="list-style-type: none"> • Software Data Structure • Registered Data Transfer • Control Data Transfer • Set Data Transfer with System Resources
2	System Integrated Environment	<ul style="list-style-type: none"> • Operational Environment • Localizing I/O Resources
3	System Hardware Environment	<ul style="list-style-type: none"> • Host-Target Platform • Memory Resources • Storage Resources • Transmission Resources

7.6.2 Software adaptation and installation functions and function types to be specified

The adaptation and installation functions to be specified (and corresponding entities to be measured) are composed of ten functions that may be allocated to software adaptation and installation requirements; the specified functions are divided into three function types (and corresponding entities types) - see Table 7.12.

Table 7.12 System adaptation and installation functions and functions types

ID	Function types	Adaptation and installation functions
1	System Software Environment (SSE)	<ul style="list-style-type: none"> • Software Data Structure Function (SDSF) • Registered Data Transfer Function (RDTF) • Control Data Transfer Function (CDTF) • Set Data Transfer with System Resources Function (SDTF)
2	System Integrated Environment (SIE)	<ul style="list-style-type: none"> • Operational Environment Function (OPEF) • Localizing I/O Resources Function (IORF)
3	System Hardware Environment (SHE)	<ul style="list-style-type: none"> • Host-Target Platform Function (HTPF) • Memory Resources Function (MRF) • Storage Resources Function (SRF) • Transmission Resources Function (TRF)

7.6.3 A standard-based model of software-FUR for system adaptation and installation-NFR using an SOA

Figure 7.6 illustrates a standard-based model of software-FUR for system adaptation and installation-NFR using an SOA. This model is built based on the system adaptation and installation requirements, corresponding functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

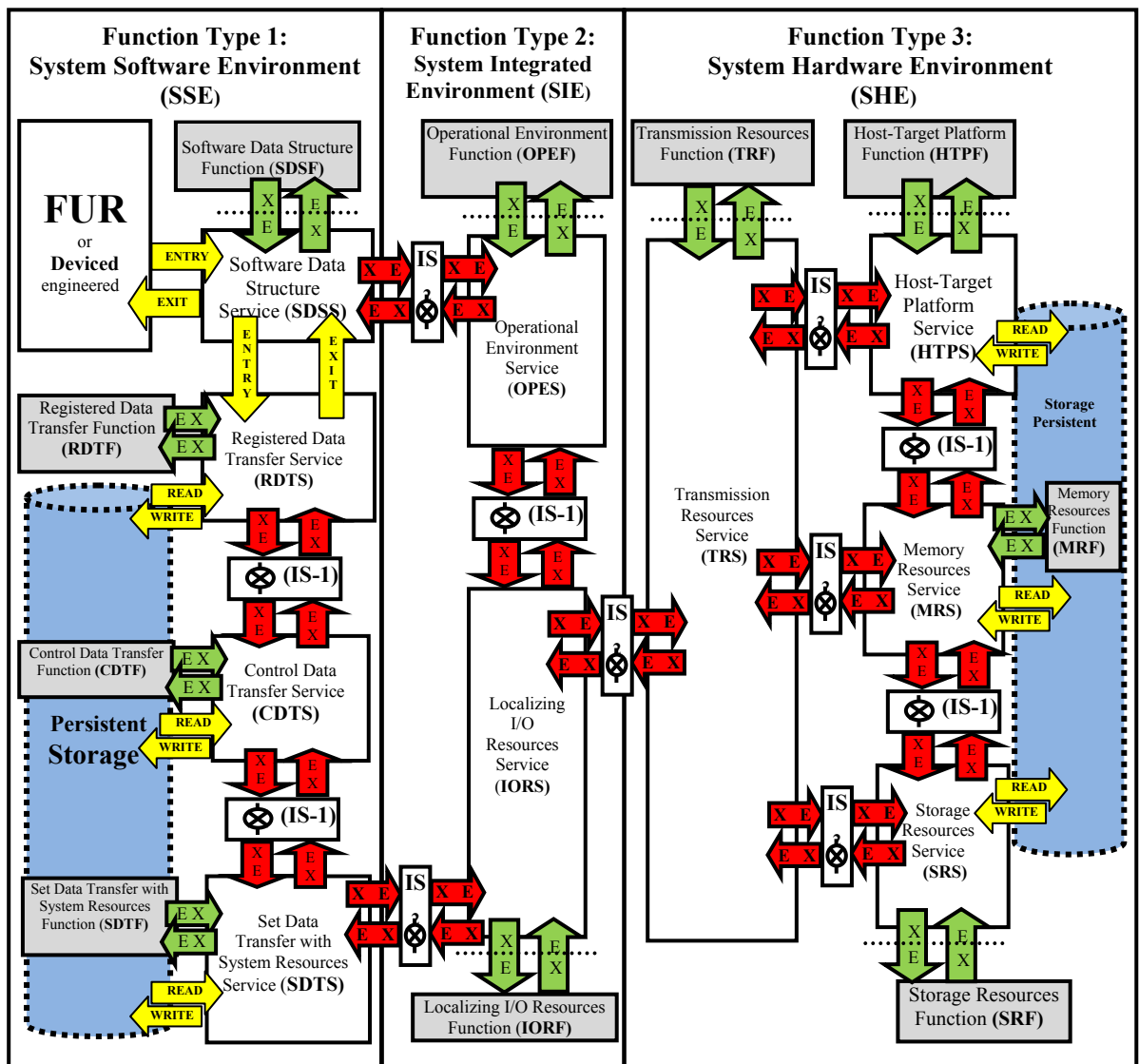


Figure 7.6 A standard-based model of software-FUR for system adaptation and installation-NFR using an SOA

7.7 Design and implementation (D&I) constraints system requirements

This section assembles the terminologies and concepts of D&I constraints dispersed in the ECSS standards into a proposed standard-based model of software-FUR for system of D&I constraints-NFR using an SOA through the use of the generic model of FUR proposed in the COSMIC model. This model can then be used for describing the software-FUR from system D&I constraints based on ECSS and ISO 19759.

7.7.1 D&I constraints requirements and functions to be specified

The types of system D&I constraints can be derived from the physical and logical models which include:

- The static design and its D&I constraints;
- The dynamic design and its D&I constraints;
- The mapping between both the static and the dynamic design and the D&I constraints views;
- The behaviour of the system design before and after implementation.

The functions to be specified (and corresponding entities to be measured) are divided into external and internal constraints functions - see Table 7.13. The internal D&I constraints refer to the expected logical D&I constraints that could appear from the system behaviour, while the external D&I constraints refer to the expected physical D&I constraints.

Table 7.13 Software D&I functions to be specified

D&I constraint	D&I constraint types	D&I components
Internal D&I constraints	Internal D&I constraints on module(s) and process(s)	<ul style="list-style-type: none"> • Module(s) • Process(s)
External D&I constraints	External D&I constraints on channels and event(s)	<ul style="list-style-type: none"> • Channel(s) • Event(s)

7.7.2 Model of function types relationships

Figure 7.7 presents an overview of the relationships between the function types in the D&I constraints software-FUR using the COSMIC model for graphical representation. More specifically:

1. The sub-model of internal D&I constraints function type 1 can be used to specify (and to measure the functional size of) the internal D&I constraints for the processes and the internal channels or events from the received/sent data movements from/to any other processes and internal channels in the same module – See Figure 7.7;
2. The sub-model of external D&I constraints on channels function type 2 can be used to specify (and to measure the functional size of) the external D&I constraints for the external channels from the received/sent data movement from/to any other processes in different modules – See Figure 7.7.

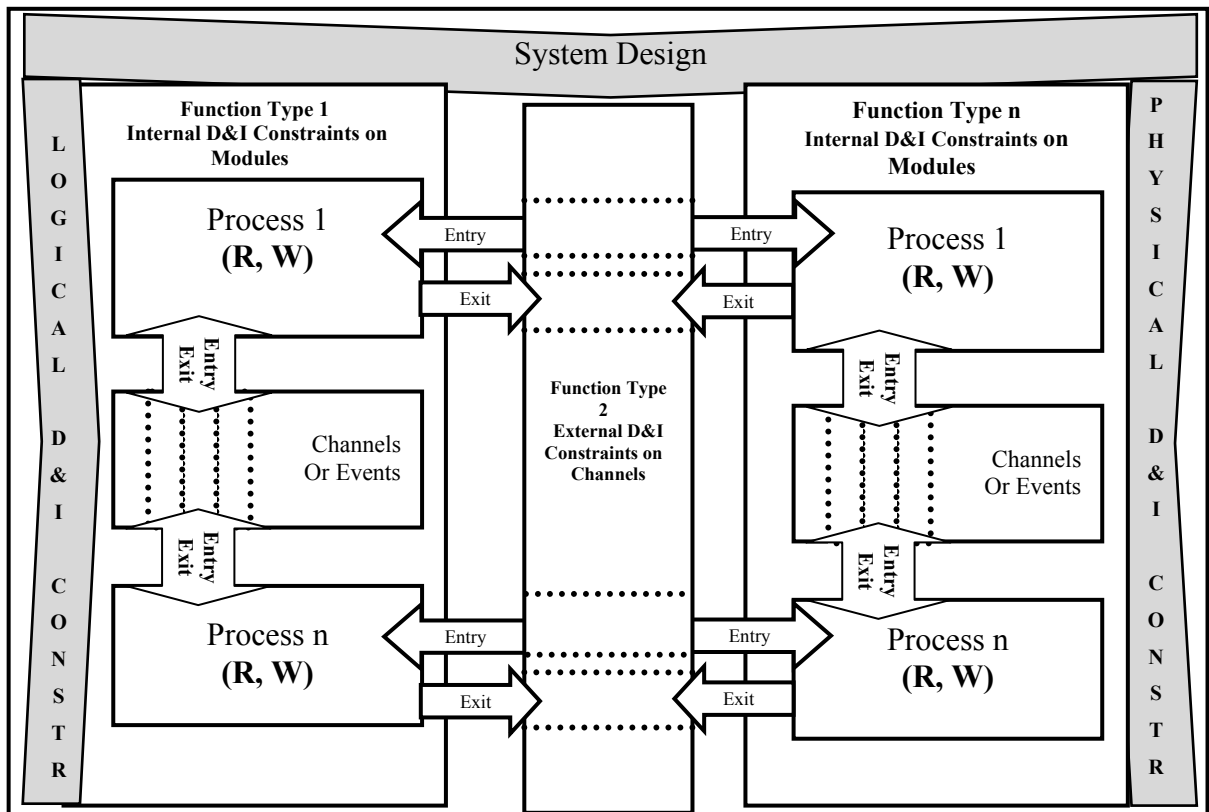


Figure 7.7 A standard-based model of software-FUR for system D&I constraints-NFR

7.7.3 A model of D&I constraints services

This model is referred here as a generic model of software-FUR for system D&I constraints:

1. The internal D&I constraints in modules (function type 1 in Figure 7.7): Each module may have many processes, each process may interact using an internal channel or event (for example, through an RPC or remote procedural call) for an internal connection; in this case the processes should be considered as a storage device for such kind of information before data marshalling between the other processes - see also Figure 7.8;
2. The external D&I constraints on channels (function type 2 in Figure 7.7): many modules may interact with each other through their own processes. In this case many processes in different modules may use external channels (for example: through an RMI or a remote method invocation) for external connection - see also Figure 7.8;
3. Process 1.1 starts sending to process 1.n in module 1 (for example process 1.1 represents function and process 1.n represents a sub-function in the same module);
4. Process n.1 should start sending to interact process n.n in a module 2 (for example process n.1 represent function n and process n.n represent sub-function in the same module).

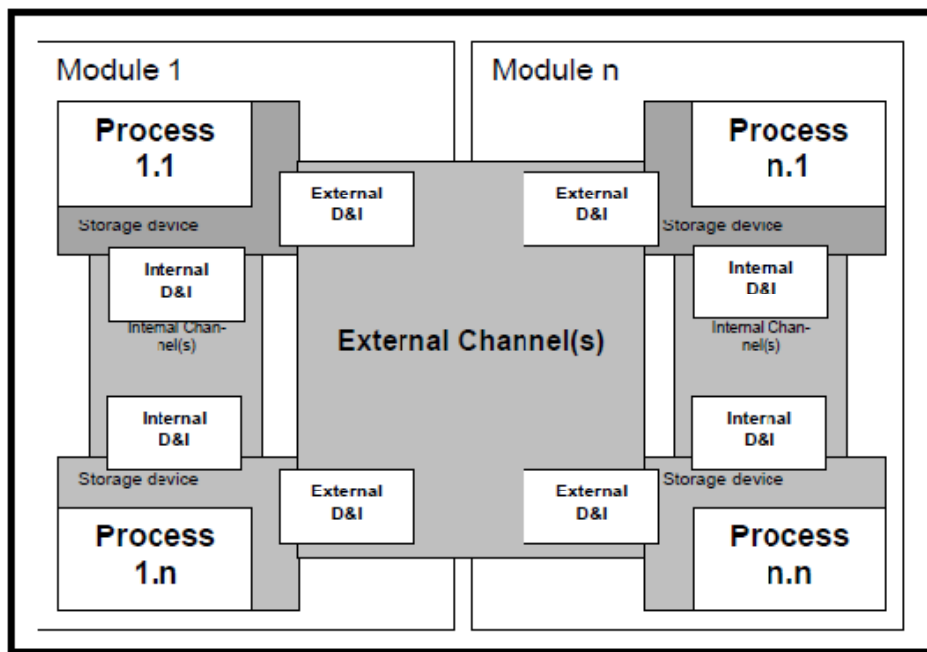


Figure 7.8 A model of D&I constraints requirements allocated to software

7.7.4 A model of D&I constraints of data movements in Software-FUR view

Figure 7.9 and Figure 7.10 show the possible flows of data movements between components; the exchange of data between components could be direct or indirect exchange of data movements to provide the functional user with services.

Figure 7.9 shows that each process (or component) in the figure could exchange the data directly to provide services to the functional user; in this case for the measurements uses, we identify Entry and/or Exit data movements.

Figure 7.10 shows indirect exchange of data between processes which means that a service in one process writes data which is subsequently read by another process. In this situation this identifies a write data movement in the next process and a read data movement by the latter.

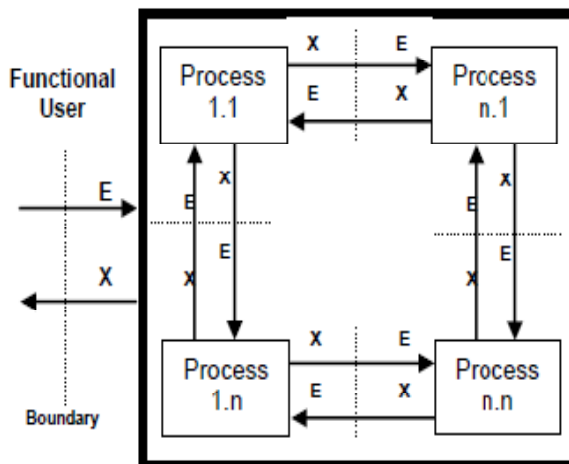


Figure 7.9 Direct Data Movements

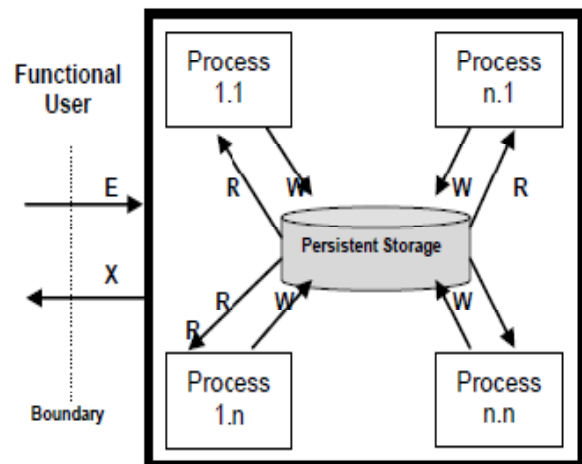


Figure 7.10 Indirect Data Movements

7.8 Performance system requirements

This section maps the performance terminologies found throughout the ECSS and IEEE standards from chapter 3 into a proposed standard-based model of software-FUR for system performance -NFR using an SOA through the use of the generic model of FUR proposed in the COSMIC model.

7.8.1 Mapping views and concepts for performance from ECSS and IEEE standards

Based on a synthesis of the various definitions, the key views and concepts presented in chapter 3 on software-FUR for system performance-NFR are presented in Table 7.14.

Table 7.14 Performance requirements in ECSS and IEEE

ID	System performance requirements
1	Static numerical requirements
2	Dynamic numerical requirements
3	Response to reference signals
4	Response time
5	Settling time
6	Tracking error for command profiles
7	Throughput time
8	Bandwidth
9	Workload
10	Resource consumption
11	Main memory time
12	Storage device time
13	Processor instruction execution
14	Evaluation processing speed
15	Accuracy errors
16	Stability errors
17	System scalability
17	Concurrency
18	Static numerical requirements

7.8.2 Software system performance functions to be specified

The functionality and corresponding entities to be specified (and measured) for system performance allocated to software are listed in Table 7.15.

Table 7.15 System performance functions that may be allocated to software

ID	System performance types	System performance functions
1	Static numerical requirements	<ul style="list-style-type: none"> • Resource consumption • Main memory time • Storage device time • Processor instruction execution • Evaluation processing speed • Accuracy errors • Stability errors • System scalability • Concurrency
2	Dynamic numerical requirements	<ul style="list-style-type: none"> • Response to reference signals • Response time • Settling time • Tracking error for command profiles • Throughput time • Bandwidth • Workload

7.8.3 Identification of the function types in the performance system requirements

In this section, the system performance function types are identified based on the findings of the performance functions, as discussed in the previous section. The system performance requirements allocated to software-FUR are divided into two types of requirements: static and dynamic numerical requirements. Each type in this division has its own functionality. The proposed performance function types are illustrated in system and COSMIC modeling views, in order to propose a standard-based model of software-FUR for system performance - NFR using an SOA- see Table 7.16.

Table 7.16 Function types for performance functions that may be allocated to software

System performance types	System performance function types	System performance functions
Static numerical requirements	Function type 1 Resource consumption (RC)	<ul style="list-style-type: none"> • Main memory time function (MMTF) • Storage device time function (SDTF) • Processor instruction execution function (PIEF)
	Function type 2 Evaluation processing speed (EPS)	<ul style="list-style-type: none"> • Accuracy errors function (AEF) • Stability errors function (SEF) • System scalability function (SSF) • Concurrency function (CF)
Dynamic numerical requirements	Function type 3 Response to reference signals (RRS)	<ul style="list-style-type: none"> • Response time function (RTF) • Settling time function (STF) • Tracking error for command profiles function (TECPF)
	Function type 2 Throughput time (TT)	<ul style="list-style-type: none"> • Bandwidth function (BF) • Workload function (WF)

7.8.4 A standard-based model of software-FUR for system performance-NFR using an SOA

Figure 7.11 illustrates a standard-based model of software-FUR for system performance - NFR using an SOA. This model is built based on the proposed performance functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

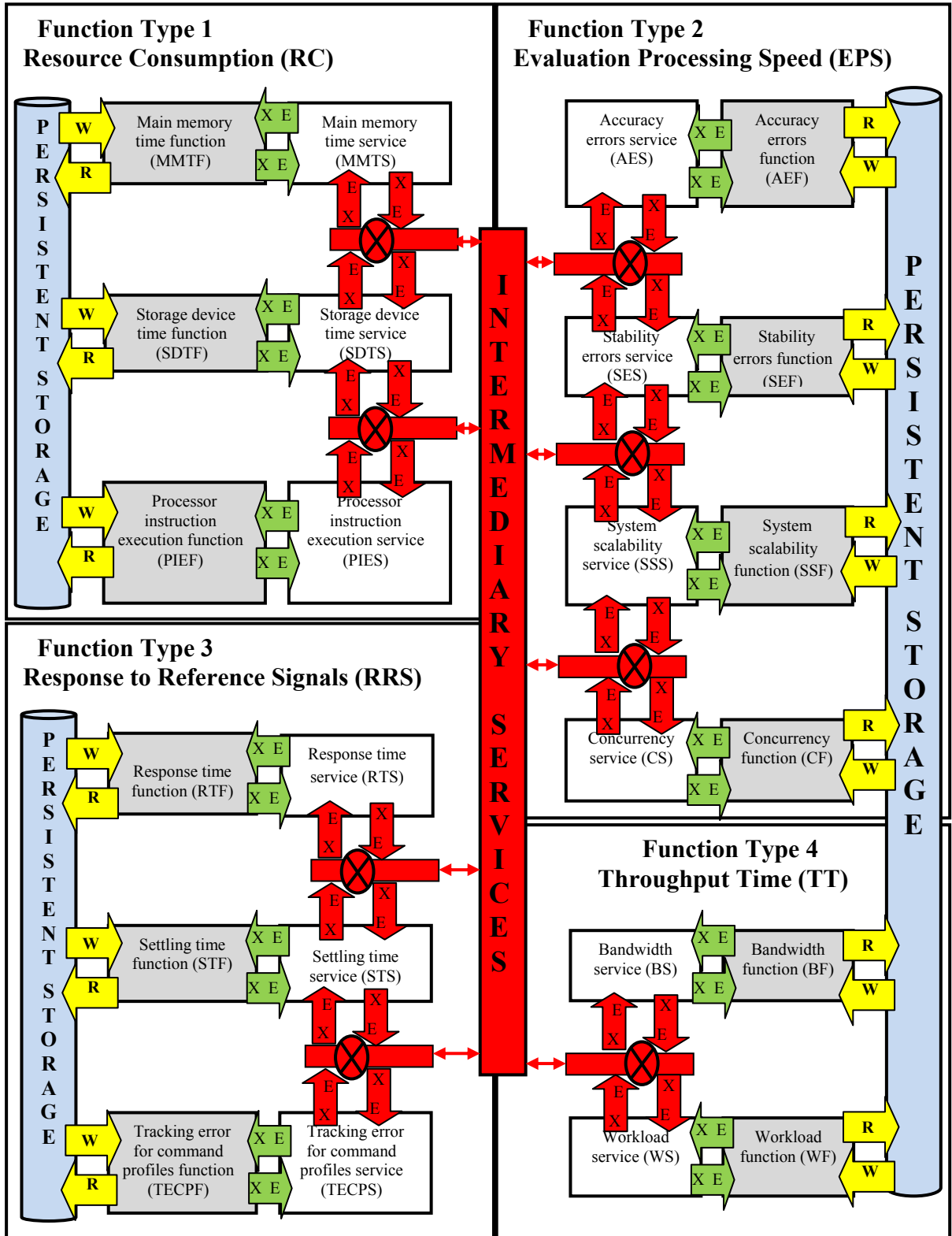


Figure 7.11 A standard-based model of software-FUR for system performance -NFR using an SOA

7.9 Security system requirements

This section maps the security terminologies found throughout the ECSS, IEEE, and ISO standards from chapter 3 into a standard-based model of software-FUR for system security-NFR using an SOA through the use of the generic model of FUR proposed in the COSMIC model.

7.9.1 Mapping views and concepts for security from ECSS, ISO, and IEEE standards

Based on a synthesis of the various definitions, the key views and concepts presented in chapter 3 on software-FUR for system security-NFR are presented in Table 7.17.

Table 7.17 Security requirements in ECSS, ISO, and IEEE

ID	System security requirements
1	Confidentiality
2	Availability
3	Integrity
4	Access control role
5	Security login
6	Authentication
7	Redundant power and network
8	Redundant data
9	Automatic restart
10	Firewall
11	Antivirus
12	External PKI
13	Backup type
14	Encryption and decryption

7.9.2 Software system security functions to be specified

The functionality and corresponding entities to be specified (and measured) for system security allocated to software are listed in Table 7.18.

Table 7.18 System security functions that may be allocated to software

ID	System security types	System security functions	Activity (examples)
1	Confidentiality	<ul style="list-style-type: none"> Access control role function 	<ul style="list-style-type: none"> Per person Per group
		<ul style="list-style-type: none"> Security login function 	<ul style="list-style-type: none"> User name & password Password change Smart card Single sign on Automatic login
		<ul style="list-style-type: none"> Authentication function 	<ul style="list-style-type: none"> Per person Per group Per entity Per system Smart card Biometrics
2	Availability	<ul style="list-style-type: none"> Redundant power and network function 	<ul style="list-style-type: none"> Available 24 H/ 7 Days
		<ul style="list-style-type: none"> Redundant data function 	
		<ul style="list-style-type: none"> Automatic restart function 	
3	Integrity	<ul style="list-style-type: none"> Firewall function 	<ul style="list-style-type: none"> Attack detection Hot and cold backup Encryption and decryption Algorithm
		<ul style="list-style-type: none"> Antivirus function 	
		<ul style="list-style-type: none"> External PKI function 	
		<ul style="list-style-type: none"> Backup type function 	
		<ul style="list-style-type: none"> Encryption and decryption function 	

7.9.3 Identification of the function types in the security

In this section, the system security function types are identified based on the findings on the security functions, as discussed in the previous section. The system security requirements allocated to software-FUR are divided into three types of requirements: confidentiality, availability and integrity. Each type in this division has its own functionality. The proposed security functional types are illustrated in system and COSMIC modeling views, in order to propose a standard-based model of software-FUR for system security-NFR using an SOA-see Table 7.19.

Table 7.19 Function types for security functions that may be allocated to software

ID	System security functional types	System security functions
1	Function type 1 System Confidentiality (SC)	<ul style="list-style-type: none"> • Access control role function (ACRF) • Security login function (SLF) • Authentication function (AF)
2	Function type 2 System Availability (SA)	<ul style="list-style-type: none"> • Redundant power and network function (RPNF) • Redundant data function (RDF) • Automatic restart function (ARF)
3	Function type 3 Security Integrity (SI)	<ul style="list-style-type: none"> • Firewall function (FF) • Antivirus function (AF) • External PKI function (EPKIF) • Backup type function (BTF) • Encryption and decryption function (EDF)

7.9.4 A standard-based model of software-FUR for system security-NFR using an SOA

Figure 7.12, illustrates a standard-based model of software-FUR for system security-NFR using an SOA. This model is built based on the security system requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

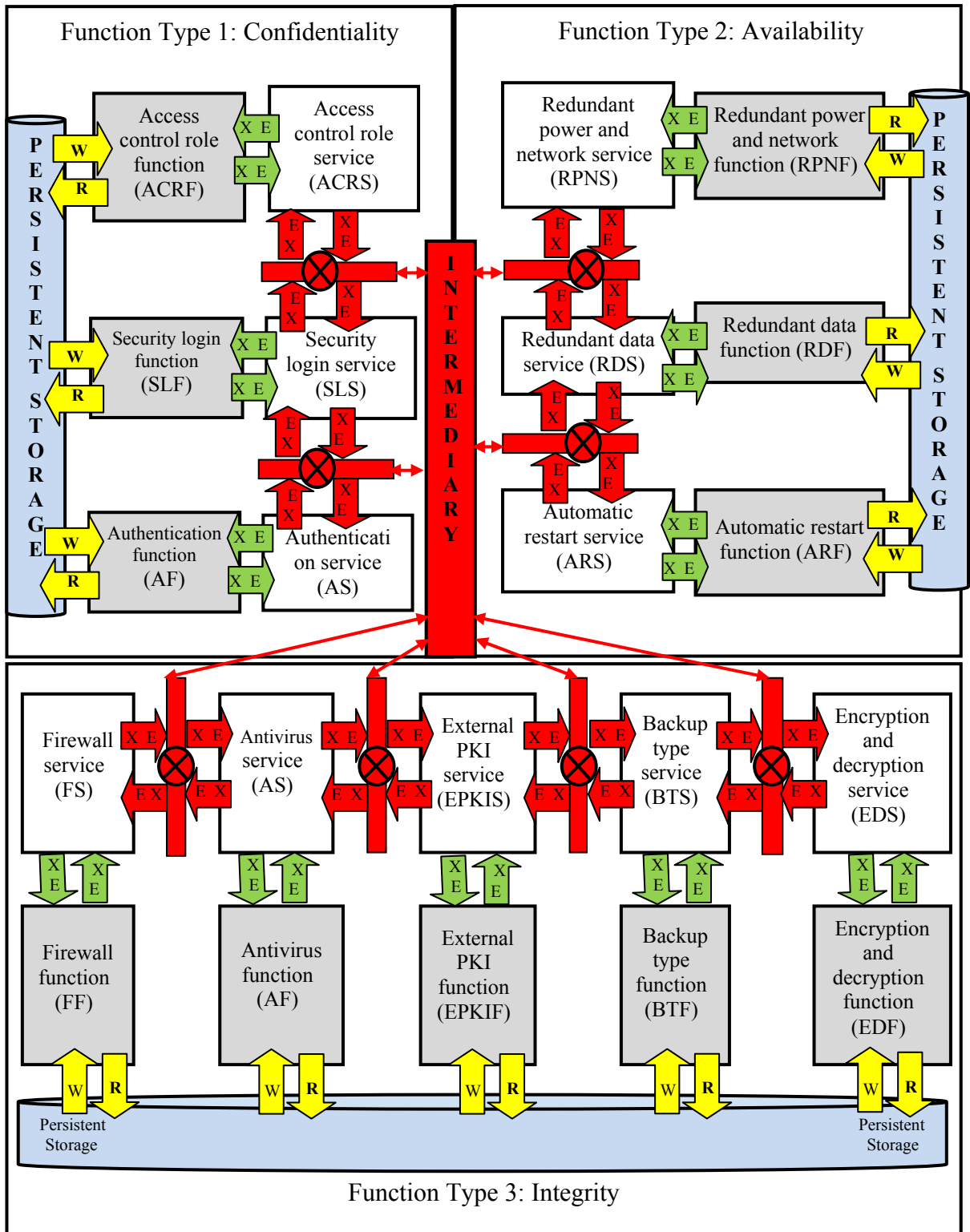


Figure 7.12 A standard-based model of software-FUR for system security-NFR Using an SOA

7.10 Safety system requirements

This section maps the safety terminologies found throughout the ECSS, IEEE, and ISO standards from chapter 3 into a proposed standard-based model of software-FUR for system safety-NFR using SOA through the use of the generic model of FUR proposed in the COSMIC model. This model can then become a framework for describing the safety requirements (i.e., from system-NFR to software-FUR) based on the ECSS standards.

7.10.1 Mapping views and concepts for safety from ECSS, ISO, and IEEE standards

Based on a synthesis of the various definitions, the key views and concepts presented in chapter 3 on software-FUR for system safety-NFR are presented in Table 7.20. It is important to note that Table 7.20 includes software, data, and hardware components which are interconnected.

Table 7.20 Safety requirements in ECSS, ISO, and IEEE

ID	System safety requirements
1	Software operation risk
2	Software design risk
3	Software configuration risk
4	System loss operation
5	System failure detection
6	System failure isolation
7	System safety audit
8	System redundancy status

7.10.2 Software system safety functions to be specified

The functions and corresponding entities to be specified and measured for system safety allocated to software are listed in Table 7.21.

Table 7.21 System safety functions that may be allocated to software

ID	System safety types	System safety functions
1	Control system hazards	<ul style="list-style-type: none"> • Software operation risk function • Software design risk function • Software configuration risk function • System loss operation function • System failure detection function • System failure isolation function
2	Critical system catastrophic	<ul style="list-style-type: none"> • System safety audit function • System redundancy status function

7.10.3 Identification of the function types in the software safety systems requirements

In this section, the system safety function types are identified based on the findings of the safety functions. The system safety requirements allocated to software-FUR are divided into system safety risk and mechanism and safety switching of redundant information. Each type in this division has its own functionality. The proposed safety function types are illustrated in system and COSMIC modeling views, in order to propose a standard-based model of software-FUR for system safety-NFR using an SOA.

The proposed safety functions can be divided into three function types, two of them specified for control system hazards and the third for critical system catastrophic. Table 7.22 illustrates these safety function types, based on the specified safety functions.

Table 7.22 Function types for safety functions that may be allocated to software

System safety types	System safety function types	System safety functions
Control system hazards	Function type 1 System safety risk (SSR)	<ul style="list-style-type: none"> • Software operation risk function (SROF) • Software design risk function (SDRF) • Software configuration risk function (SCRF)
	Function type 2 System safety mechanism (SSM)	<ul style="list-style-type: none"> • System loss operation function (SLOF) • System failure detection function (SFDF) • System failure isolation function (SFIF)
Critical system catastrophic	Function type 3 Safety switching of redundant information (SSRI)	<ul style="list-style-type: none"> • System safety audit function (SSAF) • System redundancy status function (SRSF)

7.10.4 A standard-based model of software-FUR for system safety-NFR using an SOA

Figure 7.13 illustrates a standard-based model of software-FUR for system safety-NFR using an SOA. This model is built based on the proposed safety functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010) .

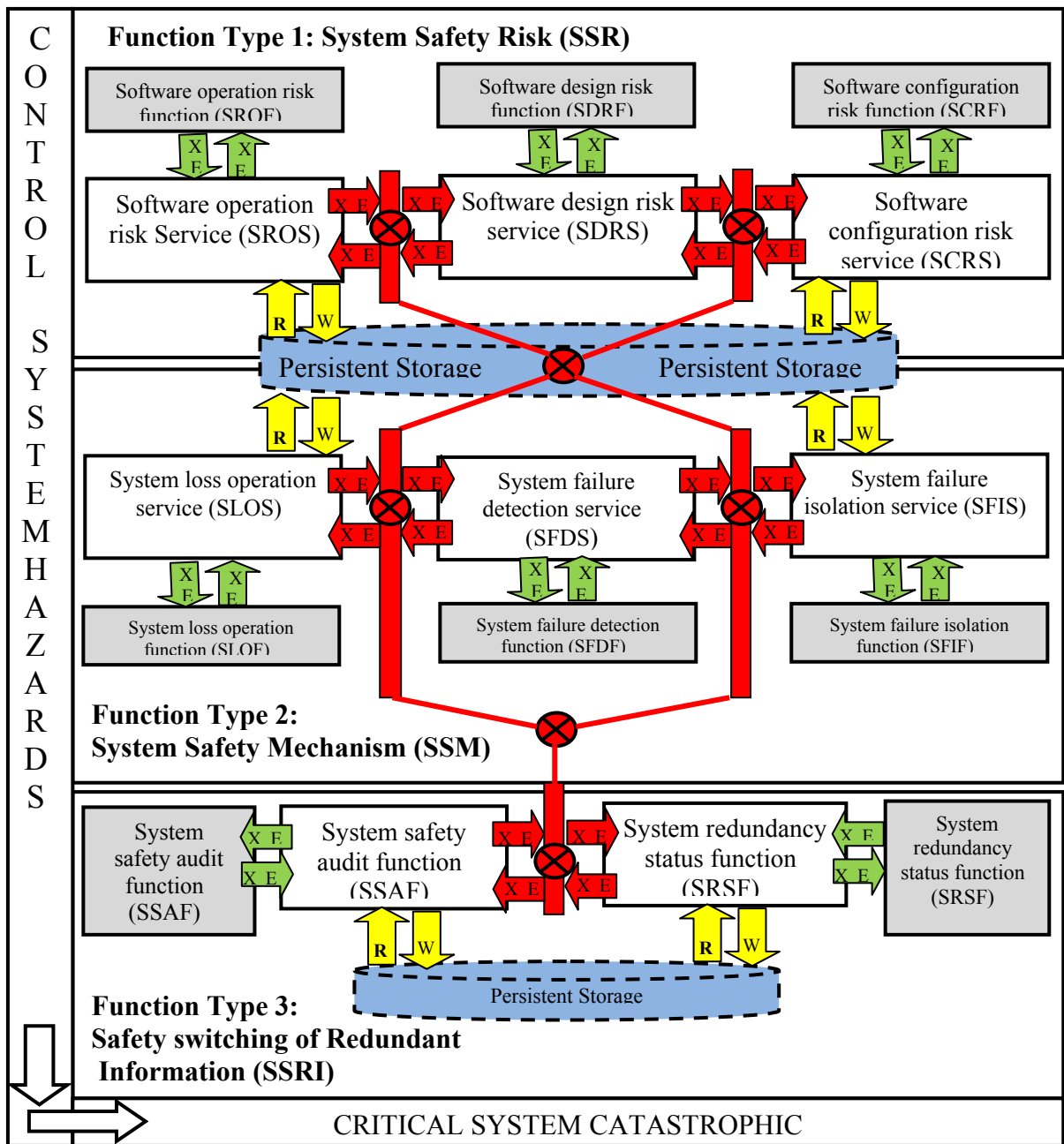


Figure 7.13 A standard-based model of software-FUR for system safety-NFR using an SOA

7.11 Resources system requirements

This section maps the resources terminologies found throughout the ECSS, IEEE, and ISO standards from chapter 3 into a standard-based model of software-FUR for system resources-NFR using an SOA, through the use of the generic model of FUR proposed in the COSMIC model. This model can then become a framework for describing the resources requirements (i.e., from system-NFR to software-FUR) based on the standards.

7.11.1 Mapping views and concepts for resources from ECSS, ISO, and IEEE standards

Based on a synthesis of the various definitions, the key views and concepts presented in chapter 3 on software-FUR for system resources-NFR are presented in Table 7.23.

Table 7.23 Resources requirements in ECSS, ISO, and IEEE

ID	System resources requirements
1	I/O recourse addresses
2	Hardware recourses
3	Software resources
4	I/O port addresses
5	I/O recourse list
6	I/O recourse addresses
7	I/O transmission addresses
8	Block of bus relative memory addresses
9	Processor capacity for software item
10	Memory capacity for software item
11	Storage device capacity for software item
12	Interrupt vectors
13	Software elements
14	Specific real time operating system

7.11.2 Software system resources functions to be specified

The functionality and corresponding entities to be specified and measured for system resources allocated to software are listed in Table 7.24.

Table 7.24 System resources functions that may be allocated to software

ID	System resources types	System resources functions
1	I/O recourse addresses	<ul style="list-style-type: none"> • I/O port addresses function • I/O recourse list function • I/O recourse addresses function • I/O transmission addresses function • Block of bus relative memory addresses function
2	Hardware recourses	<ul style="list-style-type: none"> • Processor capacity for software item function • Memory capacity for software item function • Storage device capacity for software item function • Interrupt vectors function
3	Software recourse	<ul style="list-style-type: none"> • Software elements function • Specific real time operating system function

7.11.3 Identification of the function types in the resources systems requirements

In this section, the system resources function types are identified based on the findings of the resources functions, as discussed in the previous section. The system resources requirements allocated to software-FUR are divided into three types of requirements: I/O recourse addresses, hardware addresses and software addresses. Each type in this division has its own functionality. The proposed resources function types are illustrated in system and COSMIC modeling views, in order to propose a standard-based model of software-FUR for system resources-NFR using an SOA- see Table 7.25.

Table 7.25 Function types for the resources functions that may be allocated to software

ID	System resources types	System resources function types	System resources functions
1	I/O resource addresses	Function type 1 System I/O resources (SIOR)	<ul style="list-style-type: none"> • I/O port addresses function (IOPAF) • I/O recourse list function (IORLF) • I/O recourse addresses function (IORAF) • I/O transmission addresses function (IOTAF) • Block of bus relative memory addresses function (BBRMAF)

Table 7.25 Function types for the resources functions that may be allocated to software
(Continued)

ID	System resources types	System resources function types	System resources functions
2	Hardware resources	Function type 2 Hardware resources (HR)	<ul style="list-style-type: none"> • Processor capacity for software item function (PCSIF) • Memory capacity for software item function (MCSIF) • Storage device capacity for software item function (SDCSIF) • Interrupt vectors function (IVF)
3	Software resource	Function type 3 Software resources (SR)	<ul style="list-style-type: none"> • Software elements function (SEF) • Specific real time operating system function (STOSF)

7.11.4 A standard-based model of software-FUR for system resources-NFR using an SOA

Figure 7.14 illustrates a standard-based model of software-FUR for system resources-NFR using an SOA. This model is built based on the resources requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010).

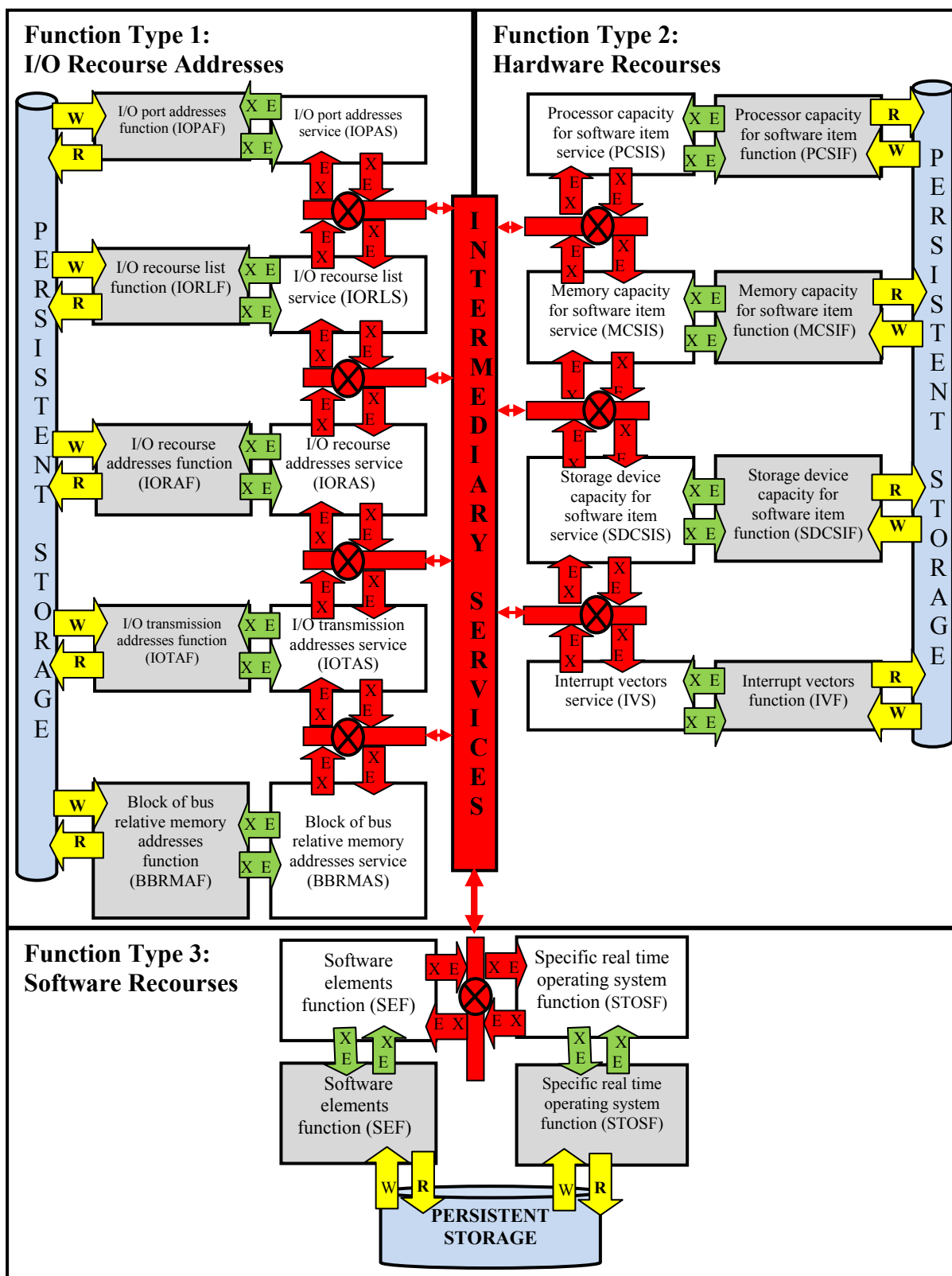


Figure 7.14 A standard-based model of software-FUR for system resources-NFR using SOA

7.12 Human factors system requirements

This section maps the human factors terminologies found throughout the ECSS standards from chapter 3 into a proposed standard-based model of software-FUR for human factors-NFR using an SOA, through the use of the generic model of FUR proposed in the COSMIC model.

7.12.1 Software system human factors functions to be specified

The functionality and corresponding entities to be specified (and measured) for human factors allocated to software are listed in Table 7.26.

Table 7. 26 Human factors functions that may be allocated to software

ID	Human factors types	System human factors functions
1	Cognitive ergonomics (performance of human factors)	<ul style="list-style-type: none"> • Human capabilities • Training • Staffing • Personal selection
2	Environmental of ergonomics (Safety of human factors)	<ul style="list-style-type: none"> • Mechanical safety • Electrical safety • Operational safety • Psychology and physiological safety • Environmental safety
3	Human interface factors	<ul style="list-style-type: none"> • Interface characteristics and task performance • Interface customization • Identification of safety related controls

7.12.2 Identification of the function types in the human factors

In this section, the system human factors function types are identified based on the findings of the human factors functions, as discussed in the previous section. Human factors requirements allocated to software-FUR are divided into three types of requirements: cognitive ergonomics, environmental of ergonomics and human factor interface

requirements. Each type in this division has its own functionality. The proposed human factors function types are illustrated in COSMIC modeling views, in order to propose a standard-based model of software-FUR for human factors-NFR using an SOA- see Table 7.27.

Table 7. 27 Function types for human factors functions that may be allocated to software

ID	System human factors types	System human factors function types	System human factors functions
1	Cognitive ergonomics (performance of human factors)	Function type 1 Cognitive ergonomics (CE)	<ul style="list-style-type: none"> • Human capabilities function (HCF) • Training function (TF) • Staffing function (SF) • Personal selection function (PSF)
2	Environmental ergonomics (Safety of human factors)	Function type 2 Environmental ergonomics (EE)	<ul style="list-style-type: none"> • Mechanical safety function (MSF) • Electrical safety function (ESF) • Operational safety function (OSF) • Psychology and physiological safety function (PSF) • Environmental safety function (ESF)
3	Human interface factors	Function type 3 Human interface factors (HIF)	<ul style="list-style-type: none"> • Interface characteristics and task performance function (ICTPF) • Interface customization function (ICF) • Identification of safety related controls function (ISRCF)

7.12.3 A standard-based model of software-FUR for human factors-NFR using an SOA

Figure 7.15 illustrates a standard-based model of software-FUR for human factors-NFR using an SOA. This model is built based on the human factors requirements, functions and function types and the role of the COSMIC-SOA explained in (COSMIC 2010) .

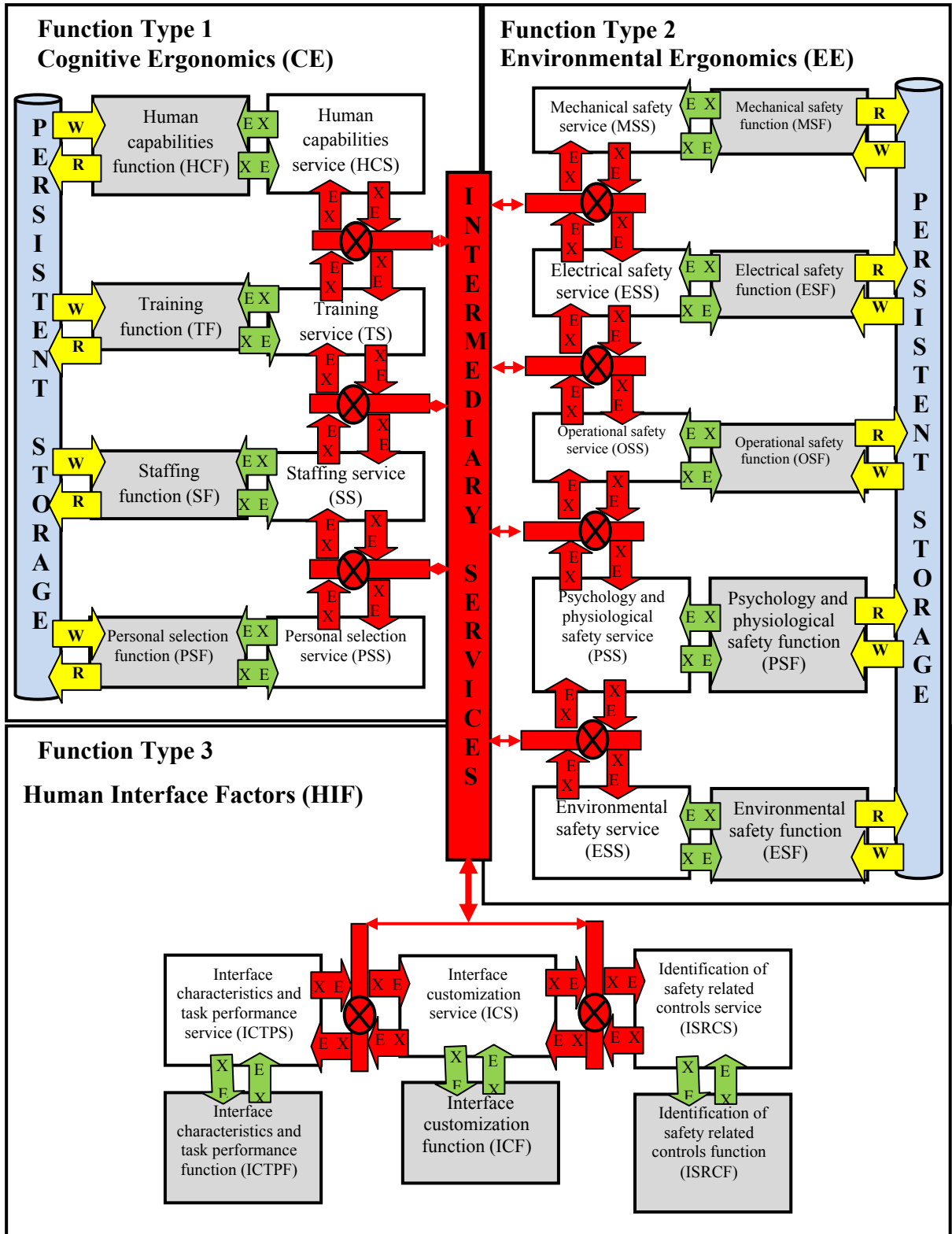


Figure 7.15 A standard-based model of software-FUR for human factors-NFR using SOA

7.13 Summary

This chapter has presented the standard-based models of eleven (11) types of system-NFR. The availability of these models can facilitate the early identification and specification of these system-NFR and their detailed allocation as specific functions to be handled by the specified allocation to hardware or software, or a specific combination of the two.

The main contribution of this chapter is our proposed eleven (11) standard-based models of software-FUR for the eleven (11) types of system-NFR. These models can be considered as a kind of reference models for the identification of these system-NFR, and can be used for their allocation to software functions implementing such requirements.

The structure of the standard-based models is based on the generic model of software adopted by the COSMIC measurement standard; the necessary information for measuring their functional size is readily available. More specifically, the standard-based models of system-NFR presented in this chapter are based on:

- The ECSS standards for the description of the NFR for system;
- The COSMIC measurement model of functional requirements.

CHAPTER 8

A CASE STUDY USING THE STANDARD-BASED MODEL OF SOFTWARE-FUR FOR SYSTEM RELIABILITY-NFR

8.1 Introduction

This chapter uses the Valve Control System (VCS) (COSMIC 2006) as a case study to illustrate the use of the standard-based model of software-FUR for system reliability-NFR. The selected case study aims at the identification and classification, then measurement, of the software-FUR for system reliability-NFR.

This chapter is organized as follows: Section 8.2 presents the description of the VCS case study. Section 8.3 presents the specification of the reliability requirements at the system level. Section 8.4 presents the allocation of these system reliability-FUR to software functions to be added to the VCS. Section 8.5 presents the specification of the ECSS-based reliability functions allocated to software-FUR for the VCS components. Section 8.6 presents the measurement of the system reliability-NFR for the VCS case study. A summary is presented in section 8.7.

8.2 The Valve Control System (VCS) Case Study

The VCS case study (COSMIC 2006) is a technology of variable valve timing used by automotive companies: the system varies the timing of the intake valves by using the hydraulic oil pressure to rotate the camshaft to provide optimal air flow in and out of the engine. The valve control system is a closed loop using camshaft sensors, crankshaft sensors, air flow meter, throttle position as well as oxygen sensors, and air fuel sensors to calculate the engine load.

Automotive companies develop real-time software with timing constraints to operate control valves that adjust the delivery of the hydraulic pressure to move the camshaft into the

position that will provide the engine with high timing reliability by using a multi-purposes logical clock for the operating cycle reference triggers.

The system functional requirements of the VCS case study are documented at a high-level in the ISO technical report: ISO/IEC TR 14143-4 (Version 2000). This ISO document provides various sets of reference user requirements (RUR), described in a textual formal.

A specific configuration of the VCS system functions allocated to hardware and software is documented as a case study and has been published by the COSMIC group, together with the measurement of the functional size of its software-FUR (COSMIC 2006). The VCS software requirements block diagram is reproduced in Figure 8.1: the software-FUR are specified and measured (with a software functional size of 12 CFP), while the system-NFR are neither specified nor measured.

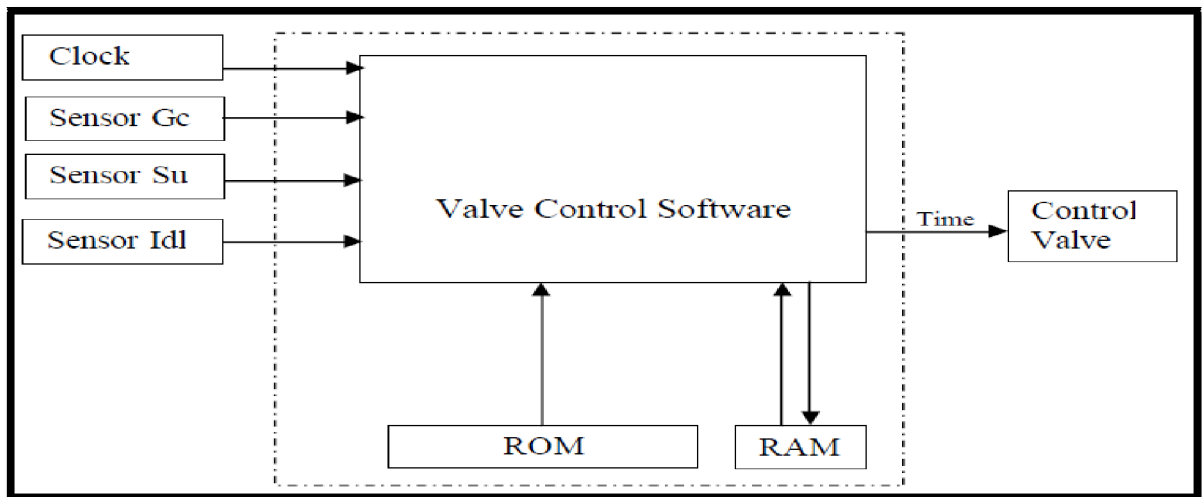


Figure 8.1 VCS blocks diagram with its hardware and software components (COSMIC 2006)

The use of standard-based model of software-FUR for system reliability-NFR can be illustrated with this VCS case study through the following steps:

1. Step 1: Specify some reliability functionality requirements at the (high) system level of the VCS components, using the proposed standard-based system reliability model as the reference for this type of specifications;

2. Step 2: Allocate these system reliability-FUR to software functions to be added to the VCS case study;
3. Step 3: Use the proposed ECSS-based reliability functions to specify, at the detailed level, these new software functions to be added to the VCS hardware and software components;
4. Step 4: Measurement of the software-FUR for the system reliability-NFR for this updated COSMIC VCS case study (COSMIC 2006).

8.3 Step 1: Addition of reliability requirements at the system level

In practice, stakeholders raise reliability requirements (R1 to R10) at the system level, such as those listed in the left-hand column of Table 8.1. To these reliability requirements correspond reliability functions (F1 to F11), which are described as such in the ECSS standards (right-hand column of Table 8.1).

Table 8.1 Alignment of system reliability requirements with the standard-based views of reliability-FUR

Stakeholder Reliability Requirements		Corresponding ECSS Standards Reliability Functions	
R1	Reliability MTBF requirements	F1	Failure system tolerance function
R2	Reliability data error requirements	F2	Error data tolerance function
R3	Reliability fault recovery requirements	F3	Fault recovery tolerance function
R4	Reliability failure operation requirements	F4	Failure operation function.
R5	Reliability failure mechanism requirements	F5	Failure mechanism function
R6	Reliability fault prevention requirements	F6	Fault prevention function
R7	Reliability fault detection and isolation requirements	F7	Fault detection function
R8	Reliability fault removal during the development requirements	F8	Fault removal function
R9	Reliability fault removal during use requirements	F8	Fault removal function.
R10	Reliability algorithm model with a set of satisfaction conditions or parameters requirements	F9 F10 F11	<ul style="list-style-type: none"> • Error to handle input function. • Error to produce output function. • Error to produce correct output function

8.4 Step 2: Allocate system reliability-FUR to software functions to be added to VCS

To meet these added system reliability-NFR requirements, all the corresponding reliability functions are allocated to new software functions to be added to the VCS components (both hardware and hardware). Table 8.2 presents the selected mapping to the VCS components of these added standard-based system reliability functions (Table 8.2), as follows;

1. Valve control software:

- The reliability requirements R1, R2, and R3 are allocated to the valve control software: therefore, R2 and R3 are mapped with F1, F2, and F3 respectively – see Figure 8.2 and Table 8.2. This means that the valve control software should be failure tolerant, have high fault recovery and minimum error tolerance.

2. Sensors

- The reliability requirements R4 and R5 are allocated to the sensors: therefore, R4 and R5 are mapped with F4 and F5 respectively – see Figure 8.2 and Table 8.2. This means that the different types of sensors should include a software component implementing failure operation and failure mechanism functions.

3. Control valve

- The reliability requirements R6, R7, R8, and R9 are allocated to the control valve. Therefore, R6, R7, R8, and R9 are mapped with F6, F7, and F8 respectively – see Figure 8.2 and Table 8.2. This means that the control valve used in the system should now have some software to implement the fault detection and removal functions for the signals received from the valve control software.

4. Clock

- The reliability requirement R10 is allocated to the clock. Therefore, R10 is mapped with F9, F10, and F11 respectively – see Figure 8.2 and Table 8.2. This means that the clock should now have added software functions to handle errors when it sends a reference time to other components in the system.

The allocation of the 10 new VCS reliability requirements (R1 to R10), and corresponding ECSS-based functions (F1 to F11) to the hardware and software components, is summarized in Table 8.2.

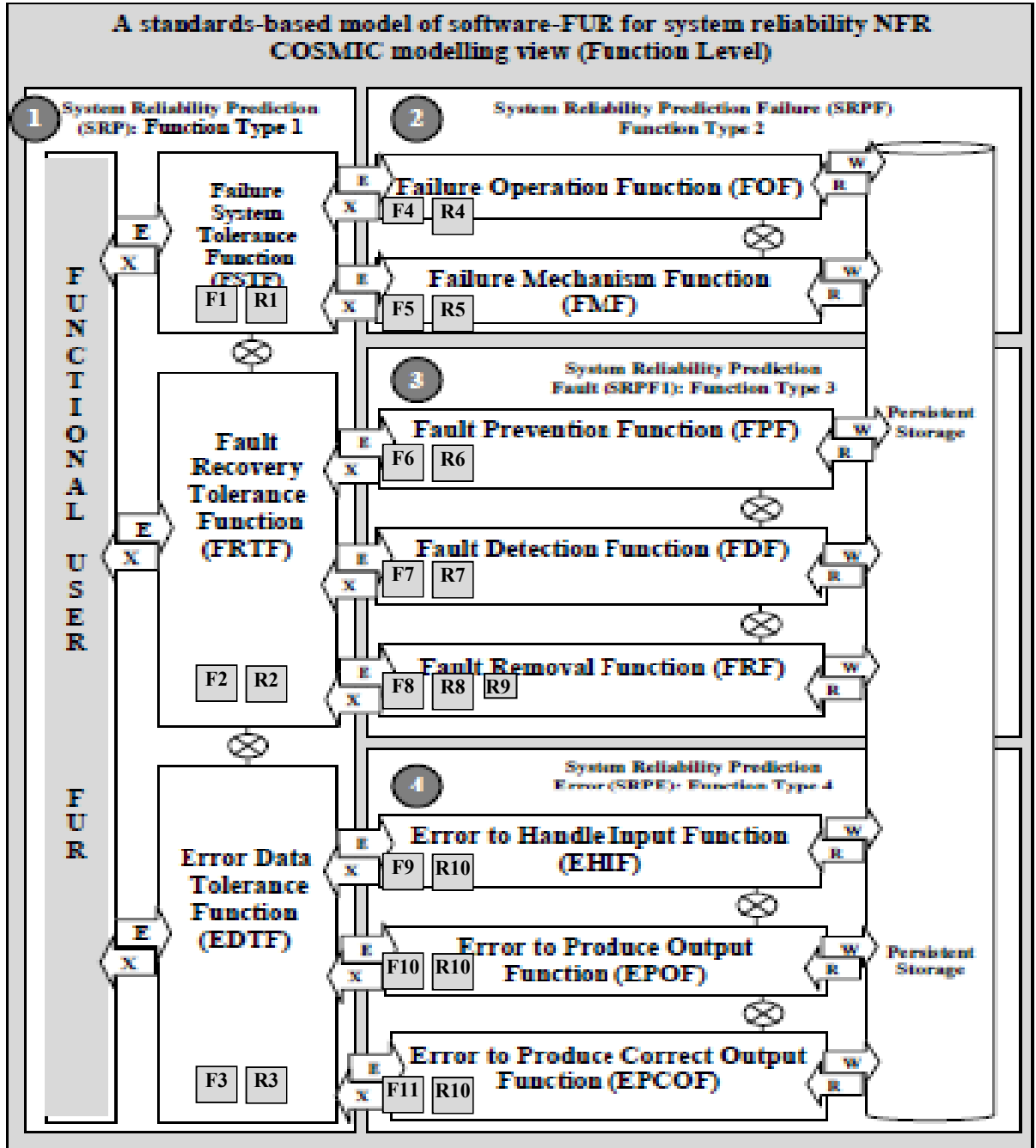


Figure 8.2 A standard-based model of FUR for system reliability-NFR (Function level)

Table 8.2 Allocation of Reliability-FUR to the VCS Components

ID	Standard-based of Reliability-FUR	F	R	VCS Components
1	Failure system tolerance function	F1	R1	Valve control software
2	Error data tolerance function	F2	R2	Valve control software
3	Fault recovery tolerance function	F3	R3	Valve control software
4	Failure operation function.	F4	R4	Sensors
5	Failure mechanism function	F5	R5	Sensors
6	Fault prevention function	F6	R6	Control Valve
7	Fault detection function	F7	R7	Control Valve
8	Fault removal function	F8	R8	Control Valve
9	Fault removal function	F8	R9	Control Valve
12	Error to handle input function	F9	R10	Clock
10	Error to produce output function	F10	R10	Clock
11	Error to produce correct output function	F11	R10	Clock

With the above set of additional reliability system-NFR, a number of software functions must be specified and added to the original VCS case study, as well as to the hardware components that did not initially have any software functions allocated to them (e.g., the clock device, the sensor devices, and the control valve, which was only receiving a signal from the ‘valve control software’).

8.5 Step 3: The requirements of the ECSS-based reliability functions allocated to software for the VCS components

For the purpose of this case study for this research work, the following more detailed requirements based on the proposed standard-based model of software-FUR for system reliability-NFR have been selected – see Figure 8.3:

- S1. All software components with their links to the components in the VCS should be defined based on system reliability prediction (SRP), i.e., F1, F2 and F3 (Derived Functions);
- S2. The sensors, control valve and clock components with their links allocated to software in the VCS should be defined based on system reliability prediction failures

(SRPF), system reliability prediction faults (SRPF1) and system reliability prediction Errors (SRPE), i.e., F4 to F11 (Base Functions).

8.6 Step 4: Measurement of the software-FUR for the system reliability-NFR

8.6.1 Measurement strategy phase

The measurement viewpoint in this case study is that of the software developer who is interested in quantifying the system reliability-NFR that have been added as new software functions that have to be developed. The measurement purpose is to measure the entire set of the functional user requirements (FUR) of the system reliability-NFR allocated to software for this case study using the COSMIC method (ISO 19761). The measurement scope is a subset of the system reliability-NFR requirements that is, only functions allocated to software and not those related to the hardware.

8.6.2 COSMIC mapping phase

When these reliability requirements are specified using the structure of the proposed standard-based model of software-FUR for the system reliability-NFR, it is already aligned with COSMIC model of functional user requirements and the necessary information for measuring their functional size is readily available.

To use Figure 8.3 to measure the functional size of the valve control software based on the proposed System-reliability-NFR, the operationalization sub-steps are:

- Reliability functions numbers are defined from: F1 to F11;
- Reliability function types for system reliability-NFR are defined from function types 1 to 4;
- Same persistent storage between system reliability functions.

8.6.3 COSMIC measurement phase

For the illustrative purpose of this case study, the following assumption is taken: there is a single data group for each reliability function specified (of course, for a reliability function specified in an industrial context, more than one data group may be needed). The total functional size according to the ISO 19761 for all the new reliability software functions added in this updated VCS is obtained with the addition of all data movements for each distinct reliability function– see Figure 8.2 and Table 8.3.

Table 8.3 The measurement details for the system reliability requirements allocated to software functions

ID	Standard-based Software-FUR for reliability functions	F	Data Movements identified				Size in CFP
			E	X	R	W	
1	Failure system tolerance function	F1	1	1	-	-	2
2	Error data tolerance function	F2	1	1	-	-	2
3	Fault recovery tolerance function	F3	1	1	-	-	2
4	Failure operation function.	F4	1	1	1	1	4
5	Failure mechanism function	F5	1	1	1	1	4
6	Fault prevention function	F6	1	1	1	1	4
8	Fault detection function	F7	1	1	1	1	4
9	Fault removal function	F8	2	2	2	2	8
10	Fault removal function	F8					
7	Error to handle input function	F9	1	1	1	1	4
11	Error to produce output function	F1 0	1	1	1	1	4
12	Error to produce correct output function	F1 1	1	1	1	1	4
Functional Size			12	12	9	9	42 CFP

The bottom line of Table 14 presents the measurement results for the system reliability functions allocated to the new software functions for the updated VCS case study: 42 CFP.

Observations:

- The software functional size for the initial VCS case study was equal to 12 CFP.

- The software functional size for the added software functions required to meet the system reliability requirements is equal to 42 CFP.
- Therefore, the total software functional size of this new version of the VCS case study (including the added reliability requirements for the specified hardware-software configuration) is equal to $12 \text{ CFP} + 42 \text{ CFP} = 54 \text{ CFP}$.

8.7 Summary

This chapter has presented a new version of the Valve Control System case study to illustrate the use of the proposed standard-based model of software-FUR for system-reliability requirements. This new version of this selected case study allows the identification and specification, as well as the measurement of the software functional size, of the system reliability-NFR allocated to new software functions.

This chapter has also presented the specification of the ECSS-based reliability allocated to software-FUR for the VCS components:

- S1. All software components with their links of defined components in the VCS were specified based on system reliability prediction (SRP), i.e., F1, F2 and F3;
- S2. The sensors, control valve and clock, components with their links allocated to software in the VCS were specified based on system reliability prediction failures (SRPF), system reliability prediction faults (SRPF1) and system reliability prediction Errors (SRPE), i.e., F4 to F11.

The system reliability-NFR for the VCS case study based on the specification of the ECSS-based reliability functions allocated to software-FUR for the VCS were specified and allocated to software and their size measured as follows:

- The functional size for S1 (software view) = 6 CFP of software functions added to the valve control software;
- The functional size for S2 (system view) = 36 CFP of software functions added to the other VCS hardware components.

CHAPTER 9

TRACEABILITY MODEL AND OPERATION PROCEDURES

9.1 Introduction

Requirements traceability links each single detailed requirement to its higher level of requirements inside the requirements set. This enables the derivation of a requirement tree which demonstrates the coherent flow-down of the requirements. For example: the ECSS standards series defines a requirement traceability matrix for the system engineering. Unfortunately, the ECSS matrix does not explicitly differentiate between system functional requirements and system-NFR.

This chapter presents a requirement traceability matrix that is considered as part of the design definition file as defined in ECSS-E-ST-10C Annex G (ECSS-E-ST-10C 2009): it includes the basic structure to perform the system functional requirements traceability and a modified traceability matrix for the system-NFR.

This chapter is organized as follows: Section 9.2 presents the ECSS requirement traceability matrix. Section 9.3 presents a proposed modified traceability matrix. Section 9.4 presents the traceability to ECSS standards of our proposed standard-based models for system-NFR. Section 9.5 presents the traceability to ECSS standards of the proposed standard-based system reliability-NFR. A summary is presented in section 9.6.

9.2 System Requirement Traceability Matrix (RTM) in ECSS standards

Currently, the ECSS standards series defines the requirement traceability matrix (RTM) as part of a system design definition file (DDF): the design definition file is a basic structure referring to all information relative to the functional and physical architectures of a system (i.e., information, necessary for its identification, manufacturing, utilization, support and removal from service).

The objective of the system design definition file (DDF) is to establish the technical definition of a system that complies with its technical requirements specification as defined in ECSS-E-ST-10-06 Annex A (ECSS-E-ST-10-06C 2009).

More specifically, the DDF is a collection of all the documentation that establishes the system such as: lower level technical specifications, design and interface description, drawings, electrical schematics, specified constraints (e.g. on materials, manufacturing, processes, and logistic) (ECSS-E-ST-10-06C 2009). The requirements traceability matrix (RTM) in the ECSS defines the relationships between the requirements of a system defined by a technical requirements specification and the apportioned requirements of the system's lower level elements.

The purpose of the RTM (ECSS-E-ST-10-06C 2009) and (ECSS-E-ST-10C 2009) is as follows:

- To state and derive requirements allocated to system components (forward trace);
- To determine the source of requirements (backward trace);
- To trace any information that satisfies the requirements;
- To ensure that all requirements are met and to locate affected system components when there is a requirements change.

The requirement traceability matrix (RTM) for software system in the ECSS standards (ECSS-E-ST-10C 2009) and (ECSS-E-ST-10-06C 2009) – see Figure 9.1 – includes the following steps:

- RTM uses a forward and backward traceability for system requirement sources:
 - High level system requirements (forward tracing);
 - Low level system requirements or detailed system requirements (backward tracing);
 - High level system requirements imposed management constraints: e.g. an applicable standard, an accepted lower level system constraint;
 - Each high level system requirement shall be linked to at least one requirement of a low level system requirement;

- When a low level system requirement is not linked to a high level system requirement, this requirement shall be justified and an evaluation of its existence or removal on the system shall be agreed between the customer and the supplier.
- RTM changes in the design inducing modifications of the system requirements;

RTM documented system requirements verification close-out in the ‘Verification Control Document’ (VCD) in conformance with ECSS-E-ST-10-02 Annex B (ECSS-E-ST-10-02C 2009).

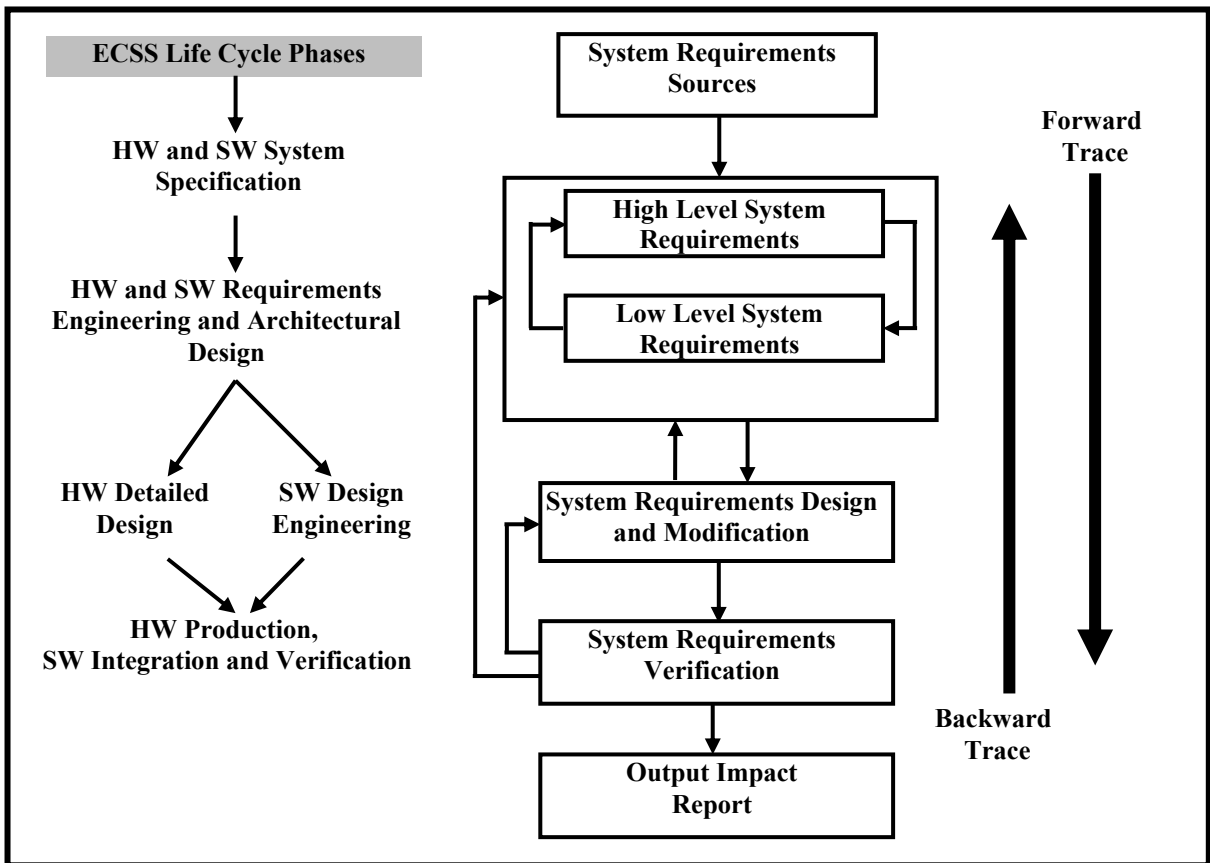


Figure 9.1 The requirement traceability matrix (RTM) in ECSS standards and cycle life

The requirements traceability matrix (RTM) using a forward trace approach for software systems in the ECSS standards (ECSS-E-ST-10-06C 2009), (ECSS-E-ST-10C 2009) and (ECSS-E-ST-10-02C 2009) – see Figure 9.1 – includes the following steps:

- Identify the system requirements sources, starting from current practices, projects' life cycle and phases;
- Derive a set of requirements covering the engineering, quality and management domains;
- Adapt the identified life cycle and phases to ECSS life cycles and map the identified engineering, quality and management requirements to the corresponding ECSS requirements (compliance/traceability);
- Integrate, with additional ECSS requirements, where necessary, the areas not adequately covered;
- The ECSS contains the compliant life cycle and requirements according to different projects characteristics, e.g. criticality, funding, technology, cost, organization, etc.

In addition, the requirement traceability matrix (RTM) using a backward trace approach for software systems in the ECSS standards (ECSS-E-ST-10-06C 2009), (ECSS-E-ST-10C 2009) and (ECSS-E-ST-10-02C 2009) – see Figure 9.1 – includes the following steps:

- Identify the ECSS hardware, software and system life cycle, phases and reviews;
- Examine all ECSS levels of the engineering, quality and management requirements and select the set of requirements possibly of interest for a selected project;
- The selected ECSS requirements initially tailored according to different projects characteristics, e.g. criticality, funding, technology, cost, organization, etc.

9.3 System functional & NFR traceability matrix

As mentioned in the previous section, the ECSS standards series have defined the requirements traceability matrix (RTM) as part of a system design definition file without direct links of the defined requirements (FUR) to their NFR.

This research study proposes a modified RTM as illustrated in Figure 9.2, using the same steps as in the original one in the ECSS standards. The proposed modifications are as follows:

- RTM has defined high and low levels of system requirements - see Figure 9.1, while in the modified RTM such high and low level requirements can be classified as system-FUR and system-NFR - see Figure 9.2. This part can be adapted with HW and SW Requirements Engineering and Architectural Design in phase 2 of the ECSS life cycle;
- RTM has defined system requirement design and modification - see Figure 9.1, while in the modified RTM such system requirements design can be classified as High and Low levels design for system-FUR and NFR in phase 3 of the ECSS life cycle;
- RTM has defined system requirement verification - see Figure 9.1, while in the modified RTM such system requirement verification can be specified as system-FUR and NFR verification in phase 4 of the ECSS life cycle.

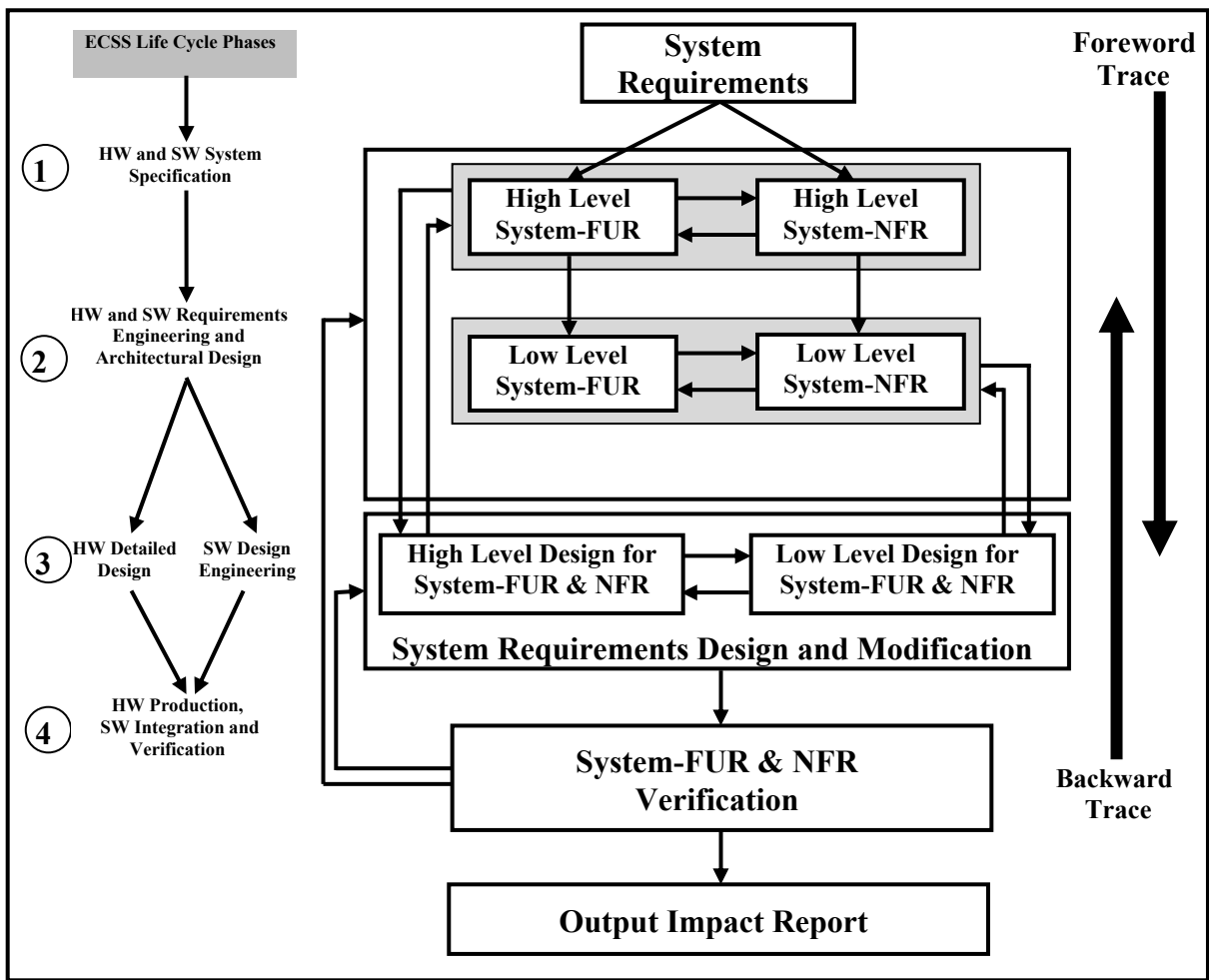


Figure 9.2 A modified requirement traceability matrix (M-RTM) in ECSS standards and cycle life

9.4 Traceability of standard-based models to system-NFR type

This section presents the traceability to the ECSS, ISO and IEEE standards for the concepts, terms and vocabularies included in the proposed standard-based models of software-FUR for the system-NFR listed in the ECSS standards. The summarized results are illustrated in Table 9.1. In summary:

- The number of the terms, concepts and vocabularies from the ECSS standards for the first fourteen (14) types of NFR requirements from the ECSS list = 153;
- The system quality requirements (requirements type fifteen (15) in the ECSS list) can be derived from the first fourteen (14) types of system requirements - see Annex III – (while the sixteenth type in the ECSS list ‘ other requirements’ is not defined);
- The number of the terms, concepts and vocabularies from the ISO standards = 65 (51 of them from ISO 9126; 9 from ISO 19759; and 4 from other ISO standards);
- The number of the terms, concepts and vocabularies from the IEEE standards = 27 (19 of them from IEEE-830 and 8 from other IEEE standards);
- The total number of terms, concepts and vocabularies from the ECSS, ISO and IEEE standards = 245;
- The total number of the system-NFR terms, concepts and vocabularies used to build the first fourteen models proposed in the research = 120 out of 245;
- The other terms, concepts and vocabularies) not used in the research = 125 out of 245 (Due to overlaps between the terms and concepts, or not belonging to system-NFR or out of this research scope).

Table 9.1 Traceability of the standard-based models

ID	NFR Type	Standards Identification (Terms, Concepts and Vocabularies)				Functions to Specified
		ECSS series	ISO	IEEE	Total	
1	System reliability requirements	10	13	-	23	11
2	System maintainability requirements	8	16	-	24	13
3	System interface requirements	10	-	4	14	6
4	System portability requirements	5	7 ISO 24765 2 ISO 2382-1 3	5	22	11
5	System operations requirements	7	-	-	7	2
6	System configuration requirements	9	ISO 19759 4	-	13	2
7	System data definitions and database requirements	20	-	-	20	12
8	System adaptation and installation requirements	10	4	-	14	10
9	System design and implementation requirements	4	ISO 19759 5	-	9	0
10	System performance requirements	15	-	5	20	12
11	System Security and privacy requirements	3	4	5	12	10
12	System safety requirements	7	4	IEEE 1220/1228 8	19	8
13	System resources requirements	13	3	-	16	11
14	Human factors requirements	32	-	-	32	12
15	System quality Requirements					
16	Other requirements					
Total # of (Terms, Concepts and Vocabularies)		153	65	27	245	120

9.5 Traceability matrix to ECSS of the standard-based model of software-FUR for system reliability-NFR

This section presents the traceability matrix to the ECSS standards for the system reliability-NFR – see Table 9.2 which lists the ECSS standards which discusses reliability aspects, together with their year of publication, their title as well as the related section number and page numbers. Such a traceability matrix will be of interest, and of use, to everybody interested in implementing in practice the standard-based models proposed in this research

work. It can be observed that this traceability matrix for the system reliability-NFR presents the details corresponding to line 1 of Table 9.1.

Table 9.2 Traceability to ECSS for the standard-based model of software-FUR for system reliability-NFR

ID	ECSS standards	Year	ECSS standards name	Section No.	Page No.
1	ECSS-E-40 part 1B	2003	Software-Part 1: Principles and requirements	-	33,77
2	ECSS-E-40 part 2B	2005	Software-Part 2: Document requirements definitions (DRDs)	-	28, 35
3	ECSS-Q-30-08A	2006	Components reliability data sources and their use	- 4.7.2 4.7.2 4.7.4 A.3 6.2.7.9	7,13 22 22 22 28 44
4	ECSS-Q-80B-10	2003		6.3.6.1 6.3.6.1 7.1.7 6.2.3	53 53 56 36
5	ECSS-E-HB-50A		Communication guideline	4.5.3.2. 4.7	50, 53 70
6	ECSS-E-ST-20C	2008	Electrical and electronics	6.3.4.2	59
7	ECSS-E-ST-33-01C		Mechanisms	4.8.2.9 4.2.2	41 17
8	ECSS-Q-ST-30-02C	2009	Failures mode, effects and critically analysis(FMEA/FMECA)	- 4.1 A.2	7-8 14-15 39-42
9	ECSS-Q-ST-30-09	2008	Availability analysis	3.2.12 A.4 A.2 A3 A-B	9-10 29 27 28 30
10	ECSS-Q-ST-30C	2009	Dependability	A-E A-F 6.4.2 7.1	36 38 20-24 26
11	ECSS-E-ST-70-26C	2008	Crimping of high-reliability electrical connection	4	12
12	ECSS-E-ST-70-08C	2009	Manual soldering of high-reliability electrical connections	A-A	101-104
13	ECSS-E-ST-70-30C	2008	Wire wrapping of high-reliability electrical connections	4	12
14	ECSS-Q-ST-10-09C	2008	Non conformance control systems	3.2.2	9
15	ECSS-M-ST-60C	2008	Cost and schedule management	-	74
16	ECSS-E-ST-50-14C	2008	Space craft discrete interfaces	4.2.4 4.2.5.2	18 21
17	ECSS-E-ST-50-04C	2008	Space data links-Telecommands protocols: synchronization and channel coding	7.5.4	85

Table 9.3 Traceability to ECSS for the standard-based model of software-FUR for system reliability-NFR (Continued)

ID	ECSS standards	Year	ECSS standards name	Section No.	Page No.
18	ECSS-E-ST-50-01C	2008	Space data links-Telemetry: synchronization and channel coding	6.1	21
19	ECSS-E-ST-33-11C		Explosive systems and device	4.2.2	17
20	ECSS-ST-35-10C	2009	Compatibility testing for liquid propulsion components, sub systems and systems	3.2.2 4.1.2	10 13
21	ECSS-ST-40C	2009	Software	5.4.2.1	46
22	ECSS-Q-ST-80C	2009	Software product assurance	5.2.7.2 6.3.2.4	27 52
23	ECSS-Q-ST-30-11C	2008	Derating-EEE components	5.1 5.2 5.3	13-16

9.6 Summary

This chapter has illustrated first the system requirement traceability matrix (RTM) in ECSS standards with the system life cycle for (HW and SW) using three distinct activities as defined in Figure 9.1:

1. Technical requirements which include (high and low level system requirements) adapted with phase 2 of the ECSS life cycle;
2. System requirements design and modifications adapted with phase 3 of the ECSS life cycle;
3. System requirements verifications adapted with phase 4 of the ECSS life cycle.

In the second part of this chapter and for the purpose of this research study, a modified RTM was proposed by using the same ECSS traceability approach by ECSS with some additional changes to tackle system-FUR and system-NFR in high and detailed levels in phase 2 of the ECSS life cycle, High and low (detailed) levels for design requirements in phase 3 as well as an extended verification part in phase 4 to tackle system-FUR and NFR as distinct requirements at high level design and as grouped at low level design.

The third part of this chapter has presented the summarized number of the traceability based concepts, terms and vocabularies for the proposed fourteen standard-based models, as well as the detailed traceability to specific sections and pages of the ECSS standards of the proposed reliability-NFR model. With this traceability to specific ECSS standards, and related sections and page numbers illustrated for the reliability-NFR, the users of the reliability model can find the details of each part of the proposed models in the ECSS standards, and they can use this traceability to implement the proposed models in practice and in conformity to the ECSS standards.

CONCLUSION

The research work presented in this thesis had one main research objective: Early specification and measurement of software-FUR derived from system-NFR, using as a basis the ECSS, ISO and IEEE systems and software engineering standards.

To achieve this objective, the following two specific research sub-objectives had reached:

- Designs of standard-based generic models for the identification and specification of software-FUR for system-NFR;
- Measurement of the functional size of software-FUR for system-NFR using the COSMIC ISO 19761 standard.

In this research study, this objective and the two sub-objectives were achieved by using three sets of international standards (ECSS, ISO and IEEE) and ISO 19761(COSMIC method) for the design of fourteen standard-based models of software-FUR for system-NFR: see Chapters 4, 5, and 6 for the details of these three models (system reliability, maintainability, and interface requirements), and Chapter 7 for the overviews of the 11 other models (system portability, operations, configuration, data definitions and databases, adaptation and installation, design and implementation constraints, performance, security, safety, recourses and human factors requirements).

Contributions of the Research

The research contributions of this PhD thesis are:

- The identification of the various concepts that should be included in the design of standard-based framework for modelling software-FUR for system-NFR based on ECSS, ISO and IEEE standards;
- The fourteen standard-based models for the identification, specification and measurement of software-FUR derived from system-NFR:
 1. Reliability systems requirements;

2. Maintainability systems requirements;
 3. Interfaces systems requirements;
 4. Portability systems requirements;
 5. Operations systems requirements;
 6. Configuration systems requirements;
 7. Data definitions and database systems requirements;
 8. Adaptations and installations systems requirements;
 9. Design and implementation constraints systems requirements;
 10. Performance systems requirements;
 11. Security and privacy systems requirements;
 12. Safety systems requirements;
 13. Resources systems requirements;
 14. Human factor requirements.
- A modified requirements traceability matrix (M-RTM) used in the ECSS for system-FUR by including the system-NFR.

A number of outcomes of this thesis have been published and-or submitted in the following conferences and journals.

- Published:
 1. Al-Sarayreh, Khalid T. and Abran, A., "A Generic Model for the Specification of Software Interface Requirements and Measurement of their Functional Size", 8th ACIS International Conference on Software Engineering Research, Management and Applications - SERA 2010, Montreal, May 24-26, 2010, IEEE-CS Press, Los Alamitos, pp. 217-222, (ISBN: 978-0-7695-4075-7), doi>10.1109/SERA.2010.35).
 2. Al-Sarayreh, Khalid T., Alain Abran and Juan J. Cuadrado-Gallego, "A Standard-based Model for the Specification and Measurement of Maintainability Requirements", 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), Redwood City, California, USA, July 2010, (ISBN 1-891706-26-8).

3. Abran, Alain, Al-Sarayreh, Khalid T. and Juan J. Cuadrado-Gallego, "Measurement Model of Software Requirements Derived from System Portability Requirements", 9th International Conference on Software Engineering Research and Practice (SERP 2010), Las Vegas, USA, July 2010, CSREA Press 2010, (ISBN 1-60132-167-8).
 4. Al-Sarayreh, Khalid T. and Alain Abran, "Measurement of Software Requirements Derived from System Reliability Requirements", 24th European Conference on Object-Oriented Programming (ECOOP 2010), ACM, Maribor, Slovenia, EU, 2010, (ISBN: 978-1-4503-0539-6), doi>[10.1145/1921705.1921706](https://doi.org/10.1145/1921705.1921706)).
 5. Alain Abran and Al-Sarayreh, Khalid T., "A Standard-based Model for the Specification of System Design and Implementation Constraints", 17th International Conference on European Systems and Software Process Improvements (EURO-SPI 2010), Grenoble Institute of Technology, Grenoble, France, Sept. 2010.
 6. Alain Abran and Al-Sarayreh, Khalid T., "Measurement of Software Requirements Derived from System Operations Requirements", 20th International Workshop on Software Measurement (IWSM 2010'), Stuttgart, Germany, Nov. 2010, (ISBN: 978-3-8322-9618-6).
 7. Al-Sarayreh, Khalid T. and Alain Abran, "Specification and Measurement of System Configuration Non Functional Requirements", 20th International Workshop on Software Measurement (IWSM 2010), Stuttgart, Germany, Nov. 2010, (ISBN: 978-3-8322-9618-6).
- Submitted
 1. Al-Sarayreh, Khalid T., Abran, A. and Cuadrado, J, "Measurement of Software Requirements Derived from System Maintainability Requirements", Submitted to Journal of Software Maintenance and Evolution: Research and Practice, John Wiley & Sons, Ltd. 2011.
 2. Al-Sarayreh, Khalid T. and Abran, A., "Early Identification, Specification and Measurement of System Non-Functional Interface Requirements", Submitted to International Journal of Metrology and Quality Engineering, 2011.

3. Al-Sarayreh, Khalid T. and Abran, A., "Early Identification, Specification and Measurement of Software Requirements Derived from System Reliability ", Submitted to Requirement Engineering Journal (RE), Springer, 2011.
4. Al-Sarayreh, Khalid T. and Abran, A., "Software Specification Framework for System Operations Requirements", Submitted to International Journal of Computer and Information Science (IJCIS), IEEE-INSPECT, 2010.
5. Abran, A., Al-Sarayreh, Khalid T. and Cuadrado, J, "Software Specification Framework of System Portability Requirements", Submitted to Journal of Software Maintenance and Evolution: Research and Practice, John Wiley & Sons, Ltd. 2011.

Expected impacts in the industry of the proposed standard-based models of software-FUR for system-NFR:

A) System engineers

The standard-based models of software-FUR for system-NFR proposed in this thesis can provide system engineers with:

- An integrated reference view of system-NFR that they can use to select the NFR necessary for a specific system to be developed (hardware-software);
- A methodology to specify these systems NFR: with the reference models, beginners may not require years of training before they are able to specify NFR at the levels of detail illustrated in the work reported in this thesis;
- An integrated model to be used as an input to make decisions on which of these detailed system-NFR will be allocated to: 1- hardware, 2- software, or 3- a combination of these for a specific context;
- Verification of system-NFR coverage and descriptions.

B) Software engineers

For software engineers, the proposed standard-based models of software-FUR for system-NFR can also provide them with reference models that they can use to verify whether or not the system engineers have provided them with this selection of system-NFR-derived

software-FUR, and at the necessary level of details. This means that the standard-based reference models can be used as a quality technique for the following:

- Elicitation of such requirements, in the software requirements phase, referred to as ‘both NFR and emergent properties’ in the SWEBOK Guide – ISO19759 (ISO-19759 2004);
- Achievement of this level of detailed inputs of software-FUR for system-NFR up front in the project life cycle (that is, at the software requirements phase, rather than much later, at the software testing phase, which is the common practice);
- The proposed standard-based models of software-FUR for system-NFR present a way to measure these software-FUR with COSMIC – ISO 19761, to take them into account in Function Points-based software estimation models, thereby avoiding late discovery of mandatory software-FUR that often lead to budget overruns and missed deadlines.

Future Work

- The measurement aspects presented in this thesis have been limited to the system requirements allocated to software. It will be interesting in future work to investigate whether or not this measurement approach can be extended to all such requirements at the system level: that is, to all hardware-software-manual requirements, and not only to software requirements;
- Document the other traceability matrices to standards for the other 13 models (such as table 9.3 for system reliability requirements in the chapter 9);
- Prepare a NFR specification Guidelines for each type of NFR, based on such traceability matrices;
- Suggest improvement to international standards on software requirements, such as IEEE 830, based on this research work;
- Suggest additions to the Requirements Knowledge Area of the SWEBOK Guide (ISO 19759), based on this research work;
- Suggest improvement to the ECSS standards based on this research work;
- Suggests improvements to the COSMIC group to document guidelines for the measurement of software-FUR derived from system-NFR

- Suggest improvement to the ISBSG data collection standards (www.isbsg.org) to capture information on NFR requirements

BIBLIOGRAPHY

- Abran, A. (2010). "Software Metrics and Software Metrology", IEEE Computer Society / Wiley, 10662 Los Vaqueros Circle, Los Alamitos,CA, P. 328.
- Abran, A. and K. T. Al-Sarayreh (2010a). "Standard-based Model for the Specification of System Design and Implementation Constraints", 17th International Conference on European Systems and Software Process Improvements (EURO-SPI 2010), Industry track, Grenoble Institute of Technology, Grenoble, France, Sept. 2010, P. 4.7-4.16 .
- Abran, A., K. T. Al-Sarayreh, et al. (2010b). "Measurement Model of Software Requirements Derived from System Portability Requirements", 9th International Conference on Software Engineering Research and Practice (SERP 2010), Las Vegas, USA, P. 553-559.
- Al-Sarayreh, K. T. and A. Abran (2010c), "A Generic Model for the Specification of Software Interface Requirements and Measurement of Their Functional Size", 8th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2010, Montreal, Canada, P. 217-223.
- Al-Sarayreh, K. T., A. Abran, et al. (2010d). "A Standard-based Model for the Specification and Measurement of Maintainability Requirements", 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), Redwood City, California, USA, P. 153-158.
- Andrew, J. R. (2000). "An Approach to Quantitative Non-Functional Requirements in Software Development", 34th Annual Government Electronics and Information Association Conference, P.13-20.
- Bendjenna, H., P. Charrel, et al. (2010). "Identifying and Modeling Non-Functional Concerns Relationships", Journal of Software Engineering & Applications 3(8), P. 820-826.
- Bharadwaj, K. and G. Nair (2009). "Mapping General System Characteristics to Non-Functional Requirements", IEEE International Advance Computing Conference, (IACC) Patiala, India, P. 1821-1825.
- Casamayor, A., D. Godoy, et al. (2010). "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach", Information and Software Technology 52(4), P. 436-445.

- Chung, L. (1993). "Representing and Using Non-functional Requirements for Information System Development: A Process Oriented Approach", Department of Computer Science, Canada, University of Toronto. Ph.D. Thesis, also Tech. Rpt. DKBS-TR-93-1.
- Chung, L. and J. do Prado Leite (2009). "On Non-Functional Requirements in Software Engineering. Conceptual Modeling: Foundations and Applications", A. Borgida, V. Chaudhri, P. Giorgini and E. Yu, Springer Berlin / Heidelberg. 5600, P. 363-379.
- Chung, L., B. Nixon, et al. (2000). "Nonfunctional Requirements in Software Engineering". Boston, Kluwer Academic Publishing, P. 439.
- COSMIC (2006). "Valve Control Software (VCS)", Software Engineering Research Laboratory, École de Technologie Supérieure - Université du Québec (Canada).
- COSMIC (2010). "The COSMIC Method v3.0.1, Guideline for Sizing SOA Software, v1.4", The Common Software Measurement International Consortium, MPC Review.
- Cysneiros, L. M. and J. C. S. d. P. Leite (2004). "Nonfunctional Requirements: From Elicitation to Conceptual Models", IEEE Trans. Softw. Eng. 30(5), P. 328-350.
- ECSS-E-40-Part-1B (2003). "Space Engineering: Software - Part 1 Principles and Requirements", European Cooperation for Space Standardization, The Netherlands.
- ECSS-E-40-Part-2B (2005). "Space Engineering: Software-part 2 Document Requirements Definitions", European Cooperation for Space Standardization, The Netherlands.
- ECSS-E-60A (2004). "Space engineering: Control engineering", Requirements & Standards Division, Noordwijk, The Netherlands.
- ECSS-E-ST-10-02C (2009). "Space engineering: Verification", Requirements & Standards Division, Noordwijk, The Netherlands.
- ECSS-E-ST-10-06C (2009). "Space engineering: Technical requirements specification", Requirements & Standards Division, Noordwijk, The Netherlands.
- ECSS-E-ST-10-11C (2008). "Space engineering: Human factors engineering", Requirements & Standards Division, Noordwijk, The Netherlands.

- ECSS-E-ST-10C (2009). "Space engineering: System engineering general requirements", Requirements & Standards Division Noordwijk, The Netherlands.
- ECSS-E-ST-60-20C-Rev.1 (2008). "Space engineering: Stars sensors terminology and performance specification", Requirements & Standards Division, Noordwijk, The Netherlands.
- ECSS-E-ST-70-31C (2008). "Space Engineering: Ground systems and operations - Monitoring and control data definition", Requirements & Standards Division Noordwijk, The Netherlands.
- ECSS-ESA (2005). "Tailoring of ECSS, Software Engineering Standards for Ground Segments, Part C: Document Templates", ESA Board of Standardization and Control (BSSC).
- ECSS-Q-80B (2003). "Space product assurance: Software product assurance", European Cooperation for Space Standardization, The Netherlands.
- ECSS-Q-ST-40C (2009). "Space product assurance: Safety", Requirements & Standards Division, Noordwijk, The Netherlands.
- ECSS-Q-ST-80C (2009). "Space Product Assurance: Software Product Assurance", Requirements & Standards Division Noordwijk, The Netherlands.
- ECSS-S-ST-00C (2008). "ECSS System: Description, Implementation and General Requirements", Requirements & Standards Division Noordwijk, The Netherlands.
- Ellis, T. and Y. Levy (2008). "Framework of Problem-Based Research: A Guide for Novice Researchers on the Development of a Research-Worthy Problem", Informing Science: the International Journal of an Emerging Transdiscipline Volume 11, P. 17-33.
- Galster, M. and E. Bucherer (2008). "A Taxonomy for Identifying and Specifying Non-Functional Requirements in Service-Oriented Development", IEEE Congress on Services Honolulu, HI, P. 126-132.
- Glinz, M. (2005). "Rethinking the Notion of Non-Functional Requirements", 3rd World Congress for Software Quality (3WCSQ 2005), Munich, Germany, P. 55-64.
- IEEE-830 (1998). "IEEE Recommended Practice for Software Requirements Specifications", Software Engineering Standards Committee, IEEE Computer Society.

- IEEE-982.1 (2005). "IEEE Standard Dictionary of Measures of the Software Aspects of Dependability", Software Engineering Standards Committee, IEEE Computer Society, New York, USA.
- IEEE-1220 (2007). "IEEE Standard for Application and Management of the Systems Engineering Proces", IEEE Computer Society, First edition.
- IEEE-14764 (2006). "Standard for Software Engineering-Software Life Cycle Processes-Maintenance", Software & Systems Engineering Standards Committee, IEEE Computer Society.
- ISO-2382-1 (1993). "Information technology - Vocabulary - Part 1: Fundamental terms", International Standards for Business, Government and Society.
- ISO-9126 (2004). "Software Engineering - Product Quality - Part 1: Quality Model 9126-1", International Organization for Standardization, Geneva (Switzerland).
- ISO-13407 (1999). "Human-centred design processes for interactive systems", International Organization for Standardization.
- ISO-14143-1 (2007). "Information technology-Software measurement - Functional size measurement Part 1: Definition of concepts", International Organization for Standardization, Geneva (Switzerland).
- ISO-19759 (2004). "Software Engineering Body of Knowledge (SWEBOK)", IEEE Computer Society.
- ISO-19761 (2011). "Software Engineering - COSMIC v 3.0 - A Functional Size Measurement Method", International Organization for Standardization, Geneva (Switzerland).
- ISO-24765 (2008). "Systems and software engineering vocabulary", British Standards Institution.
- ISO-15288 (2008). "Systems and software engineering – System life cycles processes", International Organization for Standardization/International Electrotechnical Commission, Geneva (Switzerland).
- ISO-12207 (2008). "Systems and software engineering – Software life cycles processes", International Organization for Standardization/International Electrotechnical Commission, Geneva (Switzerland).

- ISO-15979 (2002). "Open end blind rivets with break pull mandrel and protruding head-St/St", International Organization for Standardization/CEN-CENELEC International Regulation to Implement the ECSS, Geneva (Switzerland).
- Kaiya, H., A. Osada, et al. (2004). "Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems", IEEE International Conference on Requirements Engineering (RE04), P. 112-121.
- Karl, E. W. (2003). "Software Requirements", Second Edition, Microsoft Press, O'reilly, P. 544.
- Kassab, M., M. Daneva, et al. (2009). "Towards an Early Software Effort Estimation Based on Functional and Non-Functional Requirements", International Conference on Software Process and Product Measurement (MENSURA), Amsterdam, The Netherlands, Springer-Verlag Berlin, Heidelberg, P. 182-198.
- Kassab, M., O. Ormandjieva, et al. (2008). "Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP", Software Process and Product Measurement, J. C.-G. Juan, Ren, Braungarten, R. D. Reiner and A. Alain, Springer-Verlag: P. 168-182.
- Liu, C. (2010). "Ontology-Based Conflict Analysis Method in Non-functional Requirements", Computer and Information Science (ICIS), 2010 IEEE/ACIS, 9th International Conference on Information System, P. 491-496.
- Moreira, A., J. Araujo, et al. (2002). "Crosscutting Quality Attributes for Requirements Engineering", 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, SEKE: P. 167-174.
- Mylopoulos, J. (2006). "Goal-oriented Requirements Engineering", keynote talk at the 14th IEEE International Conference on Requirements Engineering (RE'06): Part II. Minneapolis, USA, IEEE Computer Society Press.
- Mylopoulos, J., L. Chung, et al. (1992). "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", IEEE Transactions on Software Engineering 18: P. 483-497.
- Mylopoulos, J., L. Chung, et al. (1999). "From Object-Oriented to Goal Requirements Analysis", Object-Oriented Modeling and Design C. ACM, ACM New York, NY, 42: P. 31-37.

- Paech, B., A. Dutoit, et al. (2002). "Functional requirements, non-functional requirements and architecture specification cannot be separated -- A position paper", International Workshop on Requirements Engineering: Foundations for Software Quality REFSQ, Essen, Germany, P. 37-49.
- Park, D. and S. Kang (2004). "Design Phase Analysis of Software Performance Using Aspect-Oriented Programming", 5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004, Lisbon, Portugal, Information and Communications University Munji-ro, Korea: P. 305-314.
- Rosa, N., P. Cunha, et al. (2002). "ProcessNFL: A language for Describing Non-Functional Properties", 35th Hawaii International Conference on System Sciences HICSS, USA, IEEE Press: P. 54-63.
- SoberIT: (2008). "Service-Oriented Architecture and Software Engineering", Seminar on Enterprise Information Systems by Software Business and Engineering Institute, Helsinki University of Technology, © 2008 Kari Hiekkanen.
- Supakkul, S., T. Hill, et al. (2010). "An NFR Pattern Approach to Dealing with NFRs", Requirements Engineering Conference (RE), 2010 18th IEEE International, P. 179-188.
- Yakkali, H. and N. Subramanian (2010). "Efficient Design of SCADA Systems Using Minimum Spanning Trees and the NFR Framework", 2nd South Eastern Symposium on System Theory. University of Texas at Tyler, USA: P. 346-351.