Measurement of Software Requirements Derived from System Reliability Requirements

Khalid T. Al- Sarayreh
Software Engineering Department
University of Quebec
1100 Notre-Dame west, Montréal
H3W 1T8, Canada
khalid.al-sarayreh.1@ens.etsmtl.ca

Alain Abran
Software Engineering Department
University of Quebec
1100 Notre-Dame west, Montréal
H3W 1T8, Canada
alain.abran@etsmtl.ca

Luca Santillo
Software Engineering Department
University of Quebec
1100 Notre-Dame west, Montréal
H3W 1T8, Canada
luca.santillo@gmail.com

ABSTRACT

Reliability is typically described initially as a non functional requirement at the system level. Systems engineers must subsequently apportion these system requirements very carefully as either software or hardware requirements to conform to the reliability requirements of the system. A number of concepts are provided in the ECSS, ISO 9126, and IEEE standards to describe the various types of candidate reliability requirements at the system, software, and hardware levels. This paper organizes these concepts into a generic standards-based reference model of the requirements at the software level for system reliability. The structure of this reference model is based on the generic model of software requirements proposed in the COSMIC – ISO 19761 model, thereby allowing the measurement of the functional size of such reliability requirements implemented through software.

Keywords—Reliability Requirements, Non functional requirements – NFR, Functional size, COSMIC – ISO 19761, ECSS International Standards, Reliability Measurement.

1. INTRODUCTION

Non-functional requirements (NFR) play a critical role in system development, including as selection criteria for choosing among alternative designs and ultimate implementations. NFR may also have a considerable impact on project effort, and should be taken into account for estimation purposes and when comparing project productivity.

Typically, these NFR are described at the system level and not at the software level, and there is no consensus yet on how to describe and measure these system NFR. In practice, NFR can be viewed, defined, interpreted, and evaluated differently by different people, particularly when they are stated vaguely and only briefly [1-3]. Therefore, it is challenging to take them into account in software estimation and software benchmarking: NFR have received less attention in the software engineering literature and are definitely less well understood than other cost factors [3]. Without measurement, it is challenging to take them as quantitative inputs into an estimation process and productivity benchmarking.

In practice, the requirements are initially typically addressed at the system level [4-10] either as high-level system functional user requirements (system FUR) or as high-level system non-functional requirements (system NFR).

ECOOP '2010 Maribor, Slovenia EU

The latter must usually be detailed, allocated and implemented in either hardware or software, or both, as software FUR, for instance – see Fig. 1.

For example, a system FUR will describe the required functions in a system, while a system NFR will describe 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) has to offer, while the term "non-functional" refers to the manner in which such functions are performed. FUR is typically phrased with subject or predicate constructions (i.e. noun/verb), such as: "The system must have a storage cluster (computer server and connection) for reliability purposes". NFR, by contrast, are typically phrased with adverbs or modifying clauses, such as: "The system must have high-availability on a storage cluster (computer servers and connection) for reliability purposes".

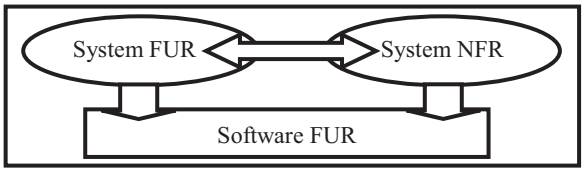

Fig. 1: Mapping system requirements into software-FUR

In the ECSS (European Cooperation on Space Standardization) standards for the aerospace industry [11-14], the ISO 9126 [15], ISO 24765 [16], IEEE 830 [17] and IEEE 1220 [18] standards, a number of concepts are provided to describe various types of candidate reliability requirements at the system, software, and hardware levels. However, these standards vary in their views, terminology, and coverage of reliability.

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

This paper focuses on a single type of NFR, that is, system reliability requirements, and reports on the work carried out to define an integrated view of software FUR for system reliability, on the basis of international standards including the use of the generic COSMIC – ISO 19761 [19] model of software FUR.

The paper is organized as follows. Section 2 presents the view of software FUR in ISO 19761. Section 3 identifies the standards describing reliability requirements. Section 4

presents a standards-based definition of a generic model of requirements for software to implement system reliability NFR. Section 5 presents the sizing of a reference instantiation of the generic model of reliability software FUR. Finally, a discussion is presented in section 6.

2. GENERIC VIEW OF SOFTWARE FUR IN ISO

ISO 14143-1 [20] specifies that a functional size measurement (FSM) method must measure the software functional user requirements (FUR). In addition, the COSMIC – ISO 19761 [19] model proposes a generic model of software FUR that clarifies the boundary between hardware and software. Fig. 2 illustrates the generic flow of data from hardware to software from a functional perspective. From this generic model of software FUR, depicted in Fig. 2, the following observations can be made:

- Software is bounded by hardware. In the so-called "front end" direction (i.e. center of Fig. 2), software used by a human 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 Fig. 2), software is bounded by persistent storage hardware like a hard disk, or RAM or ROM memory.
- Software functionality is embedded within the functional flows of data groups. Such data flows can be characterized by four distinct types of data movements. In the "front end" direction, two types of movements (ENTRIES and EXITS) allow the exchange of data with users across a boundary. In the "back end" direction, two types of movements (READS and WRITES) allow the exchange of data with the 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 users are considered the I/O hardware. For business application software, the abstraction commonly assumes that the user is one or more humans who interact directly with the business application software across the boundary; the I/O hardware is ignored.

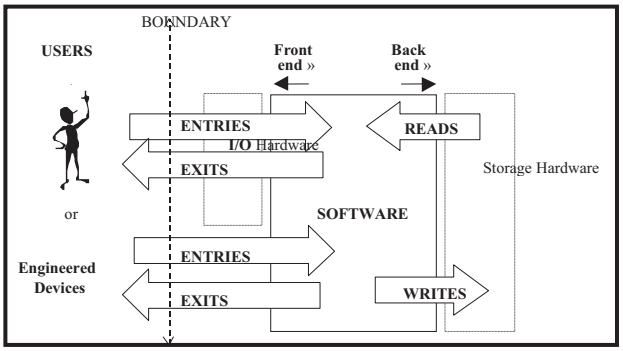

Fig. 2: Generic flow of data groups through software from a functional perspective in COSMIC – ISO 19761

As an FSM method, COSMIC 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, only those requirements allocated to the software are considered in its measurement procedure.

3. IDENTIFICATION OF STANDARDS DESCRIBING RELIABILITY

This section presents a survey of the reliability-related views, concepts, and terms in the ECSS, ISO, and IEEE standards. This section identifies which standards currently address aspects of the software FUR derived from system reliability FUR and NFR – see Fig. 3.

The expected outcome is the identification of the various elements that should be included in the design of a standards-based framework for modelling software FUR for system reliability. The elements of reliability are dispersed in various system views throughout various ECSS standards and are expressed as either:

- System reliability functional user requirements (system reliability FUR), or
- System reliability non-functional requirements (system reliability NFR)

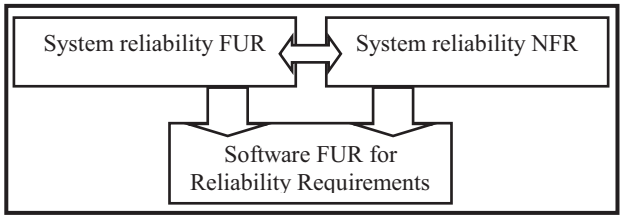

Fig. 3: Mapping system requirements into software FUR for reliability

3.1 ECSS views and concepts for reliability

Reliability in the ECSS standards shall be specified at the system level. These reliability requirements can be met by introducing adequate redundancy features. The ECSS standards consider reliability as 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:

- Failure modes and effects,
- Failure tolerance, failure detection and recovery,
- Statistical failure data to support predictions and risk assessment,
- Consolidated reliability assessments,
- Capability of the hardware to operate with software or to be operated by a human being in accordance with the specifications,
- Demonstrated reliability of critical items, and
- Justification of data bases used for theoretical demonstrations.

Table 1 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related reliability requirements. ECSS standards are specifying that reliability requirements must be implemented in software, hardware, or a combination of the two.

While conducting the survey of all the reliability concepts and terms described in the ECSS-E-40 and ECSS-Q series and in ECSS-ESA [21] as the integrated standard for ECSS-E and ECSS-Q, it was observed that:

- These various reliability elements are described differently, and at different levels of detail.

- The reliability elements are dispersed throughout the various documents: there is, therefore, no integrated view of all types of candidate reliability requirements.
- There is no obvious link between the reliability requirements in ECSS-ESA [21] as the integrated standard and all the other ECSS standards that describe reliability requirements.

Table 1: Reliability view and vocabulary in ECSS

Key views	Concepts and Vocabulary
Acceptable probability of system failure	• Component failure • Redundancy feature • Data parameter • Reliability methods, operations and mechanism • Failure tolerance • FMEA and FMECA • Failure detection • Failure isolation • Failure recovery • Failure data

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

3.2 IEEE views and concepts for reliability

IEEE-830 [17] lists reliability as one of the NFR type in their list. IEEE-830 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 any of these NFR either.

IEEE-1220 [18] 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.

3.3 ISO views and concepts for reliability

The key view on reliability in the ISO 9126 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 2.

A large number of measures are proposed in ISO 9126, but none addresses software-FUR, only the reliability NFR of the software itself. However, nothing precludes the 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.

Furthermore, ISO 24765 [16] 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 uses the following concepts with their definitions:

- Function to identify error to input.
- Function to identify error to output.

Table 2: Reliability view & vocabulary in ISO 9126

Key view	Concepts and Vocabulary
The capability of the software product to maintain a specified level of performance when used under specified conditions	• Maturity • Fault tolerance • Recoverability • Fault Density • Failure Resolution • Incorrect Operation • Availability • Breakdown Time • Recovery Time • Fault Removal • Failure Avoidance • Restart ability • Restorability

4 A STANDARDS-BASED DEFINITION OF A GENERIC MODEL OF SOFTWARE FUR FOR SYSTEM RELIABILITY REQUIREMENTS

This section identifies and assembles the terminologies and concepts of reliability dispersed throughout the ECSS, IEEE, and ISO standards. These terminologies are mapped next into a proposed model of software FUR for system reliability using the generic FUR model proposed in COSMIC– see Fig. 2,. This COSMIC-based generic model then becomes a framework for describing the software FUR from system reliability requirements based on the ECSS standards.

4.1 Mapping reliability views and vocabulary from standards

Table 3 presents the system reliability requirements that are present either as system requirements in the ECSS standard or as reliability-related concepts in ISO 9126: each of these could be interpreted, and specified, at times as software FUR.

Table 3: Reliability in ECSS, IEEE & ISO 9126

Functions to address system reliability requirements
• Function to identify error to handle input. • Function to identify error to produce output. • Function to identify error to produce correct output • Function to identify fault prevention • Function to identify fault detection • Function to identify fault removal • Function to identify failure operation. • Function to identify failure mechanism

4.2 Types of reliability requirements

Various types of system-related reliability requirements can be derived from the following set of concepts:

- System prediction tolerance
- System maturity
- System fault tolerance
- System recoverability

Table 4 presents various typical system reliability functions (middle column) for system reliability requirements and corresponding software functions (right-hand side column) that may be specified to implement such reliability functions for the system reliability requirements.

Table 4: System reliability requirements and related software functions

System Reliability Requirements derived from ISO 9126	System reliability functions	Software functions for reliability
System prediction tolerance	Reliability Models	• Failure system component tolerance • Fault recovery tolerance • Error data component tolerance
System maturity	Reliability Assessment #1	• Error to handle input. • Error to produce output. • Error to produce correct output
System fault tolerance	Reliability Assessment #2	• Fault prevention • Fault detection • Fault removal
System recoverability	Reliability Assessment #3	• Failure operation. • Failure mechanism

4.2.1 Reliability functions to be specified

The reliability functions to be specified (and corresponding entities to be measured) are divided into external and internal reliability functions that may be allocated to software – see Table 5:

- External reliability refers to the reliability prediction for faults, failures and errors that could occur in the system.
- Internal reliability refers to the reliability assessments for faults, failures and errors occurring in the system.

Table 5: Reliability functions that may be allocated to software

Types of reliability functions	Reliability Functions
External Reliability	• Error tolerance • Fault tolerance • Failure tolerance
Internal Reliability	• Error to handle input. • Error to produce output. • Error to produce correct output • Fault prevention • Fault detection • Fault Removal • Failure operation. • Failure mechanism

4.2.2 Relationships across reliability function types in software

This section identifies the function types and functional relationships in the software FUR for system reliability.

Function Type 1: Reliability models –Fig. 4

- Reliability prediction function sends at least one data group to error tolerance function or/and fault tolerance function or/and failure tolerance.
- Fault tolerance function sends/receives at least one data group to/from error tolerance function.
- Fault tolerance function sends/receives at least one data group to/from failure tolerance function.

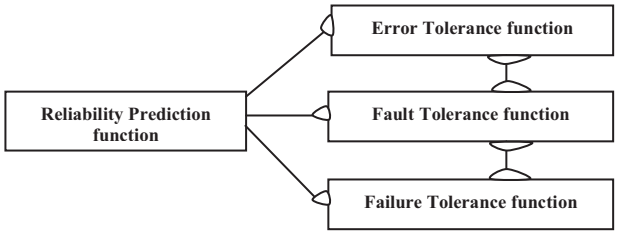

Fig. 4: Reliability models

Function Type 2: System Maturity

- Error to handle input, error to produce output and error to produce correct output functions send/receive at least one data group to/from each other's – Fig. 5.

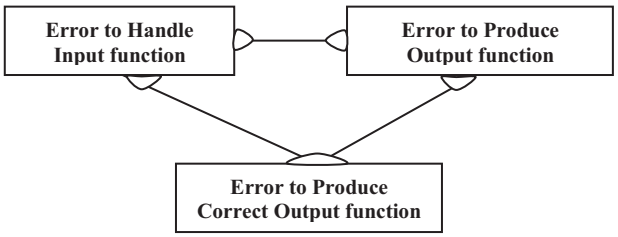

Fig. 5: System Maturity

Function Type 3: System Tolerance

- Fault prevention, fault detection and fault removal functions send/receive at least one data group to/from each other's – Fig. 6.

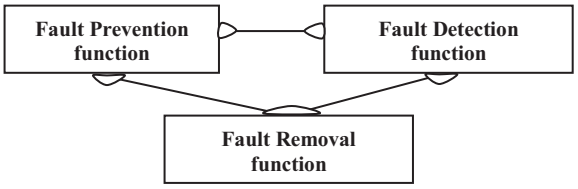

Fig. 6: System Tolerance

Function Type 4: System Recoverability

- Failure operation and failure mechanism functions send/receive at least one data group to/from each others – Fig. 7.

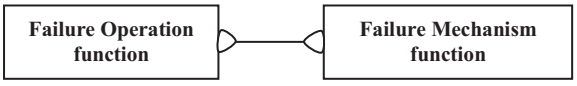

Fig.7: System Recoverability

4.2.3 Model of the functional relationships

Figure 8 presents an overview of the relationships between the function types in the reliability software FUR, using COSMIC for graphical representation. More specifically:

- The sub model of the Reliability Function Type 1 can be used to specify (and measure the functional size of) the external reliability functions for the reliability prediction models function type from the received/sent data movements from/to error, fault and failure tolerance functions – see Fig. 8:
- The sub model of the Reliability Function Type 2 can be used to specify (and measure the functional size of) the internal reliability for the system maturity function type from the received/sent data movements from/to the error to handle-produce input and output functions – see Fig. 8.
- The sub model of the Reliability Function Type 3 can be used to specify and measure the functional size of the internal reliability for the system fault tolerance function type from the received/sent data movements from/to fault prevention, detection and removal functions – see Fig. 8.

- The sub model of the Reliability Function Type 4 can be used to specify and measure the functional size of the internal reliability for system recoverability function type from the received/sent data movements from/to failure operation and failure mechanism functions – see Fig. 8.

This model is referred to here as a generic model of software FUR for system reliability.

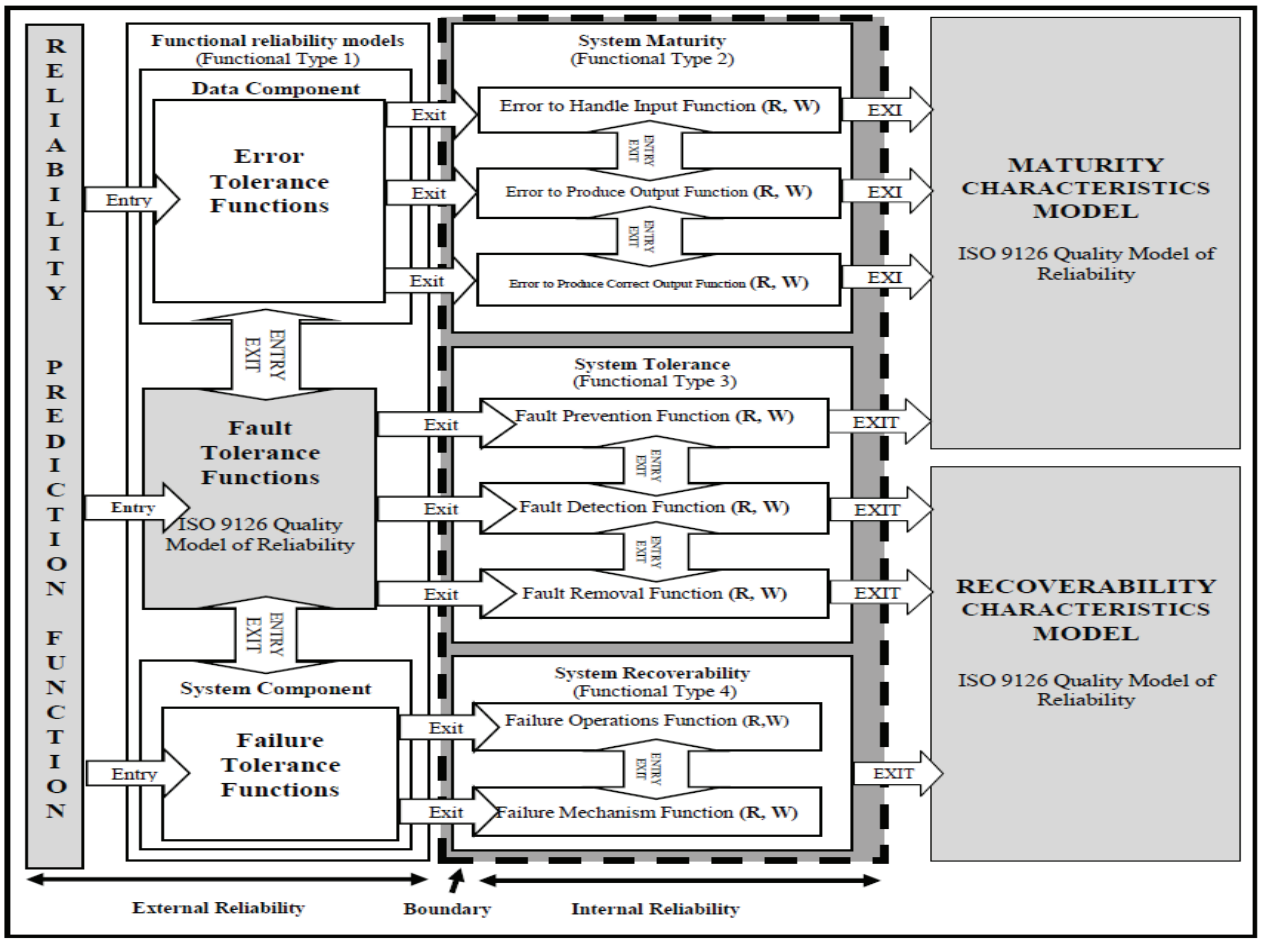

Fig. 8: COSMIC generic model for system reliability requirements allocated to software

5 Sizing a Reference Instantiation of the Generic Model of Software FUR for System Reliability

The specification of software FUR for system reliability in any specific project is a specific instantiation of the proposed generic model described in Fig. 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.

Table 6 presents the measurement results using a specific instantiation of reliability requirements which would have one of each of the reliability function types and relationships described in section 4 and Fig. 8. For example, for a reliability model (Function Type 1) – upper section of Table 6:

- The error tolerance function receives one data movement from the reliability prediction function **(1 Entry).**
- The error tolerance function sends one data movement to the error to handle input function, the error to produce output function and the error to produce correct output function in system maturity **(3 Exits).**
- The error tolerance function receives/sends one data movement from/to the fault tolerance function **(1 Entry + 1 Exit).**
- The fault tolerance function received one data movement from the reliability prediction function **(1 Entry).**
- The fault tolerance function sends one data movement to the fault prevention, fault detection and fault removal functions in the system tolerance functional type **(3 Exits).**
- The fault tolerance function receives/sends one data movement from/to the failure tolerance function **(1 Entry + 1 Exit).**
- The failure tolerance function received one data movement from the reliability prediction function **(1 Entry).**
- The failure tolerance function sends one data movement to the failure operation and failure mechanism functions in system recoverability **(2 Exits).**

The above requirements correspond to 15 COSMIC data movements, for a functional size of 15 COSMIC Function Points (i.e. 15 CFP).

The corresponding total functional size of this specific instantiation of Fig. 8 would therefore correspond to 45 data movements (of one data group each) – see table 6, which would then give a functional size of 45 CFP for this specific instantiation, using the COSMIC ISO 19761 measurement standard - see the bottom line of Table 6.

Table 6: Functional Size of a specific instantiation of the reference model of software FUR for system reliability

Functional Processes	Data Movement Description	Data Movement Type
Functional	– Error tolerance function receives one data movement from reliability prediction function	E
	– Error tolerance function sends one data movement to error handle input, error produce output and error produce correct output functions in system maturity.	X For each process
	– Error tolerance function sends/receives one data movement from/to fault tolerance function.	(E, X)
	– Fault tolerance function receives one data movement from reliability prediction function	E
	– Fault tolerance function sends one data movement to fault preventions, fault detection and fault removal functions in system reliability.	X For each process
	– Fault tolerance function sends/receives one data movement from/to failure tolerance function.	(E, X)
Reliability Models	– Failure tolerance function receives one data movement from reliability prediction function	E
	– Failure tolerance function sends one data movement to failure operation and failure mechanism functions in system recoverability.	X For each process
System Maturity	– Error to handle input, error to produce output and error to produce correct output function in system maturity read and write the data movements from error tolerance function.	R, W For each process
	– Error to handle input, error to produce output and error to produce correct output functions in system maturity send data movements to maturity characteristics model.	E For each process
	– Error to handle input, error to produce output and error to produce correct output functions in system maturity send/receive data movements with each others.	E, X For each process
System Tolerance	– Fault preventions, fault detection and fault removal functions in system tolerance read and write the data movements from fault tolerance function.	R, W For each process
	– Fault preventions function in system tolerance sends data movements to maturity characteristics model.	E
	– Fault detection and fault removal functions in system tolerance send data movements to recoverability characteristics model.	E For each process
	– Fault prevention function in system tolerance sends/receives data movements with each other	E, X For each process
System Recoverability	– Failure operation and failure mechanism functions in system recoverability read and write the data movements from failure tolerance function.	R, W For each process
	– Failure operation and failure mechanism functions in system recoverability send data movements to recoverability characteristics model.	E For each process
	– Failure operation and failure mechanism functions in system recoverability send/receive data movements with each others.	E, X For each process
The Total Functional Size using COSMIC		45 CFP

6 DISCUSSION

This paper has introduced a procedure for specifying and measuring software requirements for the internal and external reliability needed to address the system's reliability requirements.

The main contribution of this paper is our proposed Generic Model of software FUR for system reliability. This generic model can be considered as a kind of reference model for the identification of system reliability requirements and their allocation to software functions implementing such requirements. System requirements allocated to hardware have not been addressed in this paper. Since the structure of the generic 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 generic model.

Specifically, the generic model of reliability presented in this paper is based on:

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

The model is independent of the software type and the languages in which the software FUR will be implemented. The proposed generic reliability model (i.e. reference model) provides:

- A specification model for each type, or all types, of reliability requirements: for example, the requirements to be allocated to software for the reliability models system maturity, system tolerance and system recoverability.
- A specification measurement model for each type, or all types, of reliability requirements.

Future work includes documentation of the traceability of the elements of this generic model to the detailed elements of the ECSS standards, as well verification of this generic model to ensure full coverage of reliability requirements. Discussions with groups of experts will be necessary to ensure its usefulness across various communities and to develop a consensus on further refinements of such a generic model which could be proposed eventually as a candidate for standardization.

REFERENCES

[1] L. Chung and P. Leite, 2009, "On Non-Functional Requirements in Software Engineering", in "Conceptual Modeling: Foundation and Applications, Essays in Honor of John Mylopoulos", Springer.

[2] L. Chung, B.Nixon, E.Yu, J. Mylopoulos, 1999, "Non-Functional Requirements in Software Engineering" Springer, Heidelberg.

[3] J. Mylopoulos, L.Chung, B.Nixon, 1992, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", IEEE Transactions on Software Engineering, vol. 18, pp. 483-497.

[4] M. Shaw, 1989, "Larger Scale Systems Require Higher-Level Abstractions", Software Specification and Design, IEEE Computer Society, vol. 14, pp. 143-146.

[5] A. M. Davis, 1993, "Software requirements: objects, functions, and states", Prentice-Hall, Inc.

[6] I. Jacobson, G., Booth, J. Rumbaugl, 1999, "Excerpt from the Unified Software Development Process: The Unified Process", IEEE Software, vol. 16, pp. 96-102.

[7] K. Wiegers, 2003, "Software Requirements", 2nd edition, Microsoft Press.

[8] G. Roman, 1985, "A Taxonomy of Current Issues in Requirements Engineering", IEEE Computer, pp. 14-21.

[9] B. W. Boehm, 1978, "Characteristics of software quality", American Elsevier, North-Holland Pub. Co., Amsterdam, New York,.

[10] A. I. Antón, 1997, "Goal identification and refinement in the specification of software-based information systems", PhD Thesis, Georgia Institute of Technology.

[11] ECSS-E-40-Part-1B, 2003, "Space Engineering: Software - Part 1 Principles and Requirement", European Cooperation for Space Standardization,The Netherlands.

[12] ECSS-E-40-Part-2B, 2005, "Space Engineeing:Software- part 2 Document Requirements Definitions", European Cooperation for Space Standardization, The Netherlands.

[13] ECSS-Q-80B, 2003, "Space product assurance: Software product assurance", European Cooperation for Space Standardization, The Netherlands.

[14] ECSS-E-ST-10C, 2009, "Space engineering: System engineering general requirements," Requirements & Standards Division Noordwijk, The Netherlands.

[15] ISO/IEC-9126, 2004, " Software Engineering - Product Quality Model", International Organization for Standardization, Geneva (Switzerland).

[16] ISO-IEC-24765, 2008, "Systems and software engineering vocabulary", British Standards Institution.

[17] IEEE-Std-830, 1993, "IEEE Recommended Practice for Software Requirements Specifications", IEEE.

[18] IEEE-1220, 2007, "IEEE Standard for Application and Management of the Systems Engineering Process", IEEE Computer Society, First edition.

[19] ISO/IEC-19761, 2003, "ISO 19761: Software Engineering - COSMIC v 3.0 - A Functional Size Measurement Method", International Organization for Standardization, Geneva (Switzerland).

[20] ISO/IEC-14143-1, 1998, "ISO 14143-1: Information technology - Software measurement - Functional size measurement Part 1: Definition of concepts", International Organization for Standardization, Geneva (Switzerland).

[21] ECSS-ESA, 2005, "Tailoring of ECSS, Software Engineering Standards for Ground Segments, Part C: Document Templates", ESA Board of Standardization and Control (BSSC).