A Standards-based Model for the Specification of Portability Requirements

Khalid T. Al-Sarayreh[1], Alain Abran[2] and Juan J. Cuadrado-Gallego[3]
[1,2]Software Engineering Department, University of Quebec (ETS)
1100 Notre-Dame west, Montréal, Québec H3C 1K3, Canada
[3]Departamento de Ciencias de la Computación, Universidad de Alcalá
28805 Alcalá de Henares, Madrid, Spain

Abstract *- The European ECSS-E-40 standard for the aerospace industry includes portability as one of 16 types of non functional requirements. Portability requirements are typically described at the system level as non functional requirements, which may lead to specific portability-related functions to be implemented by software. According to the ECSS series of standards, portability is the capability of software to be transferred from one environment to another. ECSS standards and other international standards, such as ISO 9126 and IEEE-830, describe portability requirements using sets of concepts, terminologies, and views. This paper collects the concepts related to portability and organizes them into a generic model of functional requirements for software-FUR for system portability-NFR, which corresponds to a standards-based framework for modeling software-FUR for the portability requirements. This generic model may also be used to measure the functional size of software-FUR for system portability-NFR using the COSMIC–ISO 19761 measurement standard.*

Keywords: Portability requirements, Non functional requirements–NFR, Functional size, COSMIC–ISO 19761, ECSS International Standards, Software Portability Measurement, ISO 9126 and IEEE 830.

1 Introduction

Non functional requirements (NFR) play a critical role in system development, including as selection criteria for choosing among different 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, not at the software level, and there is no consensus yet on how to describe and measure them. In practice, they may be viewed, defined, interpreted, and evaluated differently by different people, particularly when they are stated briefly and vaguely [1-3] . Therefore it is a challenge to take NFR into account in software estimation and software benchmarking, and they are definitely less well understood than other cost factors [3]. Without measurement, it is not an easy matter to take them as quantitative inputs to an estimation process or to productivity benchmarking.

In practice, requirements are initially addressed at the system level [4-7] as either high level system functional user requirements (system-FUR) or as high level system non functional requirements (system-NFR). Normally, such high-level requirements must then be detailed and allocated to specifics-related functions, which may be implemented in both hardware and software, or both, as software functional user requirements (software-FUR). For example, system-FUR describe the functions required in a system, while system-NFR will describe how those functions must behave in the system [8-11]. In the software requirements engineering step, system-NFR may then be detailed and specified as software-FUR, to allow a software developer to develop, test, and configure the final deliverables to system users.

Functional requirements are the functions that the system (including the software) will offer, while non functional requirements detail the manner in which those functions are performed. FUR are described using subject or predicate constructions (i.e. noun/verb), such as: "The system can run on two or more kinds of devices or with two or more kinds of operating systems". NFR are described using adverbs or modifying clauses, such as: “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".

Within the ECSS European standard for the aerospace industry [12-16], ISO 9126 [17], IEEE-830 [18], ISO 24765 [19] and ISO 2382-1 [20], a number of concepts are provided to describe various types of candidate portability requirements at both the system, software, and hardware levels. However, these standards vary in their views, terminology, and portability coverage.

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

This paper focuses on a single type of NFR, that is, system portability requirements. It reports on the work carried out to define an integrated view of software-FUR for system portability-NFR on the basis of international standards, and on

the use of the generic COSMIC–ISO 19761 [21] model of software-FUR.

The paper is organized as follows. Section 2 presents the structured view of software-FUR as provided in ISO 19761. Section 3 identifies the standards that describe the software-FUR for system portability-NFR. Section 4 presents a standards-based definition of a generic model of requirements for software-FUR for system portability-NFR. Finally, a discussion is presented in section 5.

2 A generic view of software-FUR in ISO

ISO 14143-1 [22] specifies that a functional size measurement (FSM) method must measure software-FUR. In addition, COSMIC–ISO 19761 proposes a generic model of software-FUR that clarifies the boundary between hardware and software. Fig. 1 illustrates the generic flow of data from a functional perspective from hardware to software. From this figure, the following can be observed:

- Software is bounded by hardware. In the so-called "front-end" direction (i.e. center in Fig. 1), 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. right-hand side of Fig. 1), software is bounded by persistent storage hardware, like a hard disk and RAM and ROM memory.
- The functional flow of data groups can be characterized by four distinct types of 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.

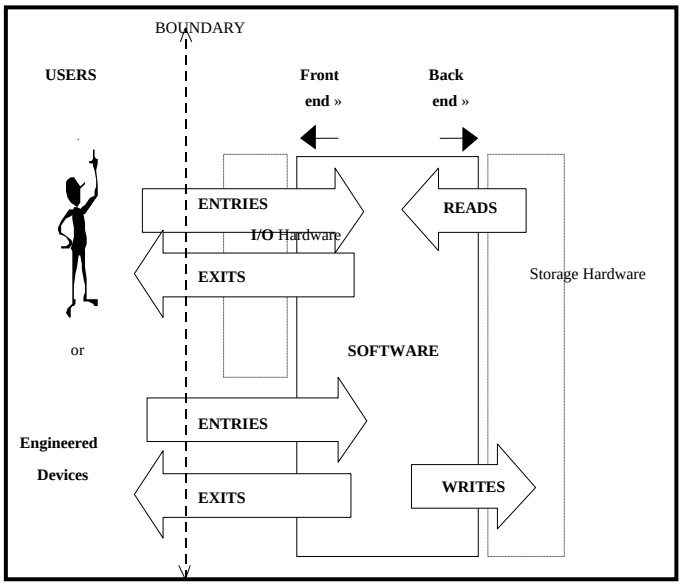

Fig. 1: Generic flow of data groups through software from a functional perspective in COSMIC – ISO 19761

- Different abstractions are typically used for different measurement purposes. In real-time software, the users are generally the engineered devices that interact directly with the software, that is, the users 'are' the I/O hardware. For business application software, the abstraction commonly assumes that the users are one or more humans who interact directly with the business application software across the boundary; the I/O hardware is ignored.

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-FUR, only those functional requirements allocated to the software are considered.

3 Identification of standards describing portability requirements

This section presents a survey of the portability-related views, concepts, and terms in international standards. It identifies which standards currently address some aspects of the software-FUR derived from system-NFR, specifically for the system portability-NFR – see Fig. 2. The expected outcome is the identification of the various elements that should be included in the design of a standards-based framework for modeling software-FUR for system portability-NFR.

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

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

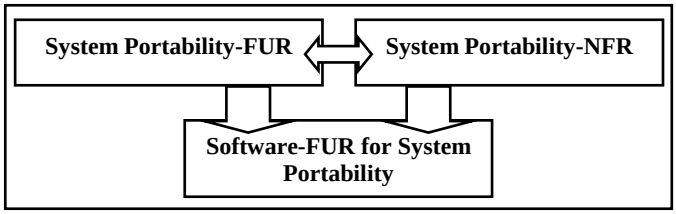

Fig. 2: Mapping system into software-FUR for portability

3.1 ECSS: views and concepts for portability

Portability in the ECSS standards is considered as the capability of the system to be transferred from one environment to another. Table 1 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. The ECSS does not specify, however, whether such requirements must be

implemented in software or hardware or in a combination of both.

Table 1: Portability view and vocabulary in ECSS

Key view	Concepts and vocabulary
The capability of system to be transferred from one environment to another	• Minimum System Dependency • Independence of Operating System • Minimum Hardware Dependency • Obsolescence of Hardware or Software • Technical Specification of Components

While conducting a survey of all the portability 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, we observed that:

- These various portability elements are described differently, and at different levels of detail;
- The portability elements are dispersed throughout the various documents: there is, therefore, no integrated view of all types of candidate portability requirements;
- There is no obvious link for the portability requirements between the ECSS-ESA standard as the integrated standard and all the other ECSS standards that describe portability requirements.

3.2 IEEE-830: views and concepts for portability

IEEE-830 [18] lists portability 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 a procedure for system portability – see Table 2. However, IEEE does not provide guidance on how to describe or specify portability requirements, nor, of course, does it provide guidance on how to measure any of these NFR.

Table 2: Portability view and vocabulary in IEEE

Key view	Concepts and vocabulary
Describe the portability by specifying the attributes of software that relate to the ease of porting the software to other host machines and/or operating systems	• 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.

3.3 ISO 9126: views and concepts for portability

The key view on portability in the ISO 9126 series is from the perspective of the quality of the software product: portability is presented as a 'quality characteristic' and is decomposed next into quality sub-characteristics and next into proposed derived measures to quantify such quality sub-characteristics. The inventory of related concepts and vocabulary on software maintainability is presented in Table 3, such as replaceability and co-existence.

Table 3: Portability view and vocabulary in ISO 9126

Key views	Concepts and vocabulary
• The capability of the software product to be transferred from one environment to another. • Environment may include organizational, hardware or software environment.	• Sharing common resources • Independence software in a common environment • Continued Use of Data • Function Inclusiveness • Software Concurrently with other Software • Replaceability • Co-existence

While a large number of measures have been proposed in ISO 9126, these measures do not address software-FUR, but rather the system portability-NFR of the software itself. However, nothing prevents using some of these concepts at the system level, or looking at what functions must be performed at the software level (i.e. FUR allocated to software) to implement these system-level NFR.

3.4 ISO 24765: views and concepts for portability

Portability in ISO 24765 [19] is considered as a system or component that can be transferred from one hardware or software environment to another. Table 4 presents a list of concepts and vocabulary used in ISO 24765 to describe system-related portability requirements. For instance, 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.

Moreover, ISO 24765 does not provide guidance on how to describe or specify portability requirements, nor, of course, does it provide guidance on how to measure any of these NFR.

Table 4: Portability view and vocabulary in ISO 24765

Key view	Concepts and vocabulary
A system or component can be transferred from one hardware or software environment to another	• Software Environment • Hardware Environment

3.5 ISO 2382-1: views and concepts for portability

Portability in ISO 2382-1 [20] is described as a program to be executed on various types of data processing systems. Table 5 presents a list of concepts and vocabulary used in ISO 2382-1 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 a combination of the two.

Moreover, ISO 2382-1 does not provide guidance on how to describe or specify the portability requirements, nor, of course, does it provide guidance on how to measure any of these NFR.

Table 5: Portability view and vocabulary in ISO 2382-1

Key view	Concepts and vocabulary
A program to be executed on various types of data processing systems	• Language independency • Data processing system • Isolating software system calls

4 A standard-based definition of a generic model of software-FUR for system portability requirements

This section maps the portability terminologies found throughout the ECSS, IEEE, and ISO standards into a proposed model of software-FUR for system portability-NFR through the use of the generic model of FUR proposed in the COSMIC model. This COSMIC-based generic model can then become a framework for describing the portability requirements (i.e. from system-NFR into software-FUR) based on the ECSS standards.

4.1 Mapping views and concepts

Based on a synthesis of the various definitions, the key views and concepts presented in the previous sections on software-FUR for system portability-NFR are presented in Table 6.

It is important to note that Table 6 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.

Table 6: Portability Requirements in ECSS, ISO & IEEE

System portability requirements
• Isolating software system calls • Independence of operating system • Independence of middleware • Independence of programming language virtual machine • Independence of browsers • Client independence • Server independence • Storage independence • Network independence • Database independence • Distributed data base management system (DDBMS)

4.2 Types of portability requirements

Portability requirements must be identified for each environment (from environment 1 to environment *n*), when required. Next, the types of portability requirements should be identified for each environment and must be allocated to: software components, hardware components, and data components – see Table 7.

Table 7: Portability types, by environment

Environment 1	…	Environment n
– **Software Components in Environment 1** o Independence operating system o Independence middleware o Independence of programming language virtual machine o Independence of browsers – **Hardware Components in Environment 1** o Client independence o Server independence o Storage independence o Network independence – **Data Components in Environment 1** o Database independence o Distributed data base management system (DDBMS)	…	– **Software Components in Environment n** o Independence operating system o Independence middleware o Independence programming language virtual machine o Independence of browsers – **Hardware Components in Environment n** o Client independence o Server independence o Storage independence o Network independence – **Data Components in Environment n** o Database independence o Distributed data base management system (DDBMS)

4.2.1 Software portability functions to be specified

The functionality and corresponding entities to be specified (and measured) for software portability are listed in Table 8. External functionality and its corresponding entities for portability are represented by the environment of these components. Internal functionality and its corresponding entities are represented by the software isolated pieces capabilities inside the environment to call each other.

Table 8: Portability functions that may be allocated to software

Portability Type	Portability Functions
External Portability	• Independence of operating system function • Independence of middleware function • Independence of programming language virtual machine function • Independence of browsers function • Client independence function • Server independence function • Storage independence function • Network independence function • Database independence function • Distributed data base management system (DDBMS) function
Internal Portability	• Isolating software system calls

4.2.2 Identification of the functional types in software portability

This section identifies the component types and functional relationships in the software portability.

- **Function Type 1: Software Components – Fig. 3**
  - A distributed browser function receives or sends at least one data group from/to a programming language virtual machine function.
  - A programming language virtual machine function receives or sends at least one data group from/to an independence operating system function.
  - An independence operating system function receives or sends at least one data group from/to independence middleware function.
  - Independence middleware function receives or sends at least one data group from/to an independence operating system function.

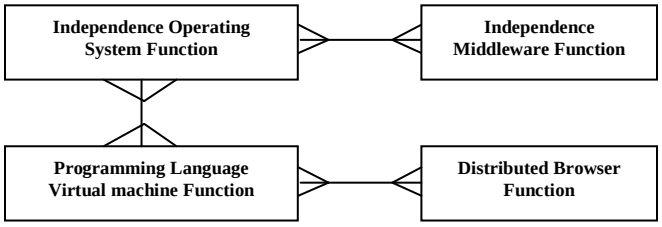

Fig. 3: Software components

- **Function Type 2: Data Components– Fig. 4**
  - A database independence function receives or sends at least one data group from/to a DDBMS function.
  - A DDBMS receives or sends at least one data group from/to a database independence function.

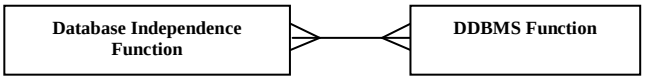

Fig. 4: Data components

- **Function Type 3: Hardware Components– Fig. 5**
  - A client independent function receives or sends at least one data group from/to an independence server function.
  - A server independent function receives or sends at least one data group from/to an independence network function.
  - A network independent function receives or sends at least one data group from/to independence storage function.
  - A storage independent function receives or sends a least one data group from/to an independence client function.

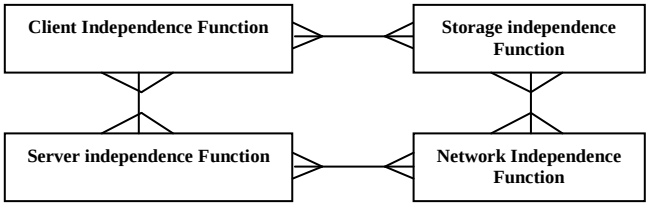

Fig. 5: Hardware components

- **Function Type 4: Isolating Software System Calls – Fig. 6**
  - Software components receive or send at least one data group from/to isolating software system calls function.
  - Software components receive or send at least one data group from/to data components.

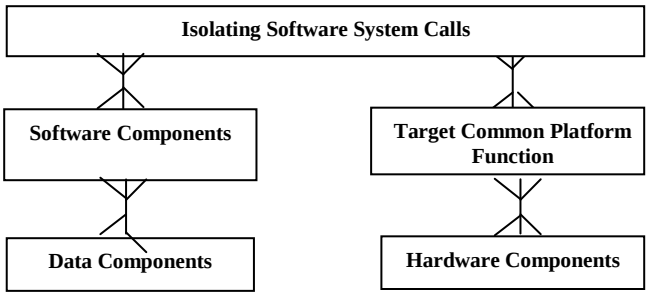

Fig. 6: Isolating Software System Calls

  - Isolating software system calls receives or send at least one data group from/to a target common platform.
  - A target common platform function receives or sends at least one data group from/to hardware components.

4.3 Functional relationships in the software portability

Figure 7 presents an overview of the relationships between the functional types in the software portability-FUR, using the COSMIC model for graphical representation. Specifically:

- The sub model of Functional Type 1 can be used to specify the external portability (and measure its functional size) for the software components from the received/sent data movements from/to an independence operating system, independence middleware, a programming language virtual machine, and distributed browser functions – see Fig. 7.
- The sub model of Functional Type 2 can be used to specify the external portability (and measure its functional size) for the data components from the received/sent data movements from/to a database independence and a DDBMS functions – see Fig. 7.
- The sub model of Functional Type 3 can be used to specify the external portability (and measure its functional size) for the hardware components from the received/sent data movements from/to a client, a server, storage, and network independence functions – see Fig. 7.
- The sub model of Functional Type 4 can be used to specify the internal portability (and measure its functional size) for the isolating software system calls from the received/sent data movements from/to a target common platform and software components – see Fig. 7.

We refer to this model as a generic model of software-FUR for system portability.

5 Discussion

This paper has introduced a procedure for specifying the requirements of the software-FUR for the system portability-NFR needed to address the system-NFR for portability. The main contribution of this paper is our proposed generic model of portability requirements. This generic model is considered as a kind of reference model for the measurement of the functional size of software-FUR for system portability-NFR. It is based on:

- ECSS, IEEE, and ISO standards for the description of software-FUR for system portability-NFR; and
- The COSMIC measurement model of requirements.

The model is independent of software type and the languages in which the portability requirements will be implemented.

The proposed generic portability model (i.e. reference model) provides:

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

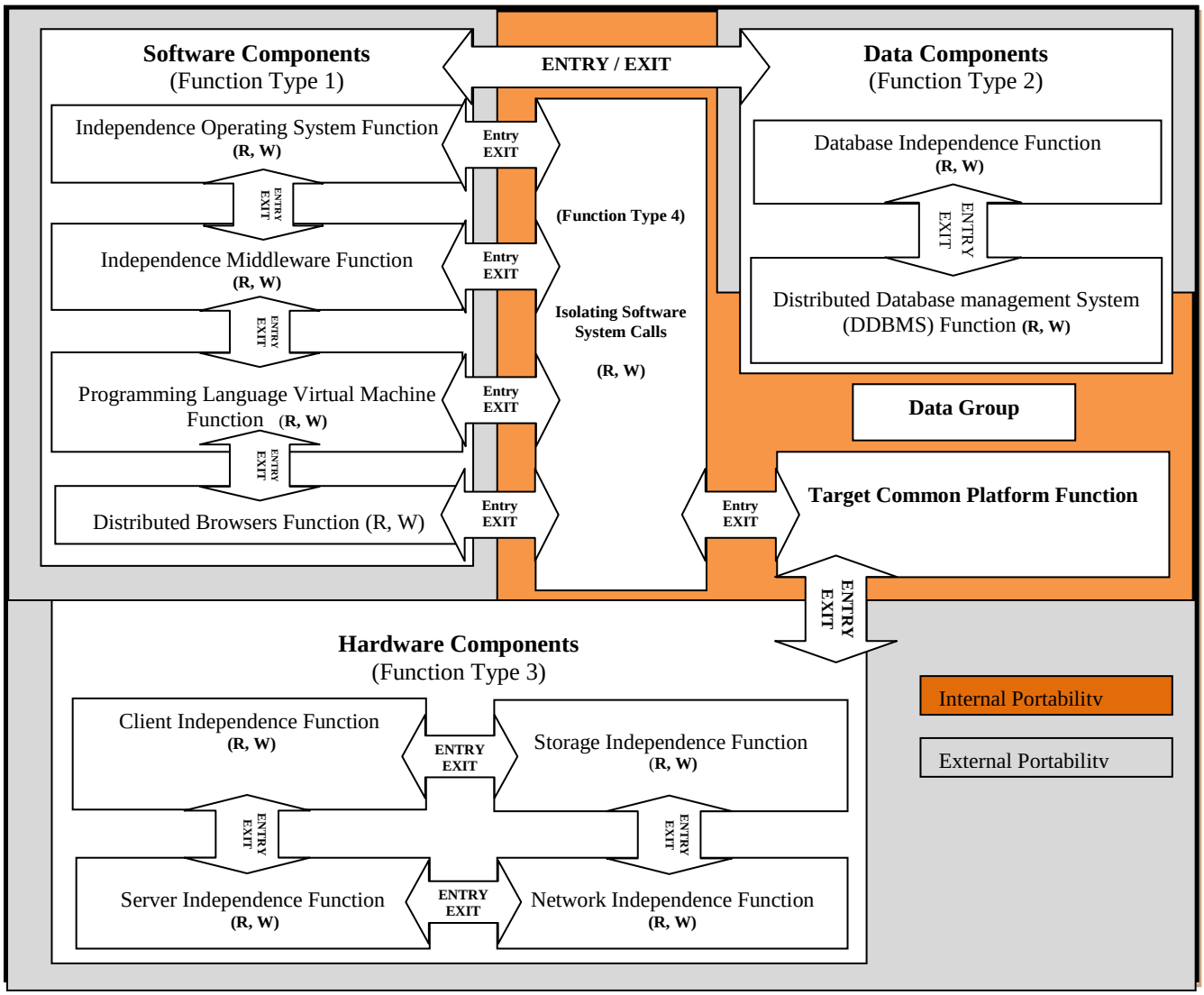

Fig. 7: COSMIC reference model of portability requirements allocated to software

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

It is important to note that the design measurement procedure for software-FUR for system portability-NFR has been developed to apply the COSMIC measurement method to the system portability requirements in order to obtain the functional size of the software-FUR for system portability-NFR in the early stages of the system development process.

Future work includes verification of this generic model to ensure full coverage of portability requirements, and verification with a group of experts to develop a consensual generic model which could be proposed as a candidate for standardization.

REFERENCES

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

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

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

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

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

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

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

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

[9] J. Mostow, "A Problem Solver for Making Advice Operational", National Conference on Artificial Intelligence (AAAI-83). AAAI, Menlo Park, Calif, pp. 279-283, 1983.

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

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

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

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

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

[15] ECSS-E-ST-40C, "Space engineering: Software Requirements & Standards Division", Noordwijk, The Netherlands, 2009.

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

[17] ISO/IEC-9126, "Software Engineering - Product Quality - Part 1: Quality Model", International Organization for Standardization, Geneva (Switzerland), 2004.

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

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

[20] ISO-2382-1, "Information technology - Vocabulary - Part 1: Fundamental terms", International Standards for Business, Government and Society, 1993.

[21] ISO-19761, "Software Engineering - COSMIC v 3.0 - A Functional Size Measurement Method", ISO, Geneva (Switzerland), 2003.

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