Specification and Measurement of System Configuration Non Functional Requirements

Khalid T. Al-Sarayreh, Alain Abran

Software Engineering Department, University of Quebec (ETS)
1100 Notre-Dame West Montréal, Québec H3W 1T8, Canada

khalid.al-sarayreh.1@ens.etsmtl.ca, alain.abran@etsmtl.ca

Abstract:

The European ECSS-E-40 series of standards for the aerospace industry includes configuration as one of 16 types of non functional requirements (NFR) for embedded and real-time software. Configuration requirements are typically described at the system level as non functional requirements, and a number of concepts and terms are provided in the ECSS series to describe various types of candidate configurations. This paper collects and organizes these configuration related descriptions into a generic model for the specification of software functional user requirements (software FUR) for system configuration NFR, and for measuring their functional size for estimation purposes using the COSMIC ISO 19761 standard.

Keywords

Configuration Requirements, Non functional requirements – NFR, Functional size, COSMIC – ISO 19761, ECSS International standards.

1 Introduction

Non functional requirements (NFR) play a critical role during system development, including as selection criteria for choosing among alternative designs and ultimate implementations. NFR may also considerably impact project effort, and should be taken into account for estimation purposes and when comparing project productivities. Typically, NFR are initially described at the system level, and there is no consensus yet on how to describe and measure them at the software level.

In practice, NFR may be viewed, defined, interpreted, and evaluated differently by different people, particularly when they are stated briefly and imprecisely [1-3]. Therefore, it is a challenge to take them into account in software estimation and software benchmarking: NFR have received less attention in the literature than other cost factors in software engineering, and are definitely less well understood [3]. Without measurement, it is not easy to take NFR as quantitative inputs to an estimation process and to productivity benchmarking.

In practice, requirements are typically initially addressed at the system level [4-7] as either high-level system functional user requirement system FUR or high-level system non functional requirements system NFR. Such high-level requirements must usually be detailed next and allocated to specific-related functions, which may be implemented in both hardware and software, or both, as software FUR [8-12], for instance – see Fig. 1.

For example, system FUR will describe the functions required in a system, while system NFR will describe how the required functions must behave in a system [13-15]. In the software requirements engineering step, such system NFR may be detailed next and specified as software FUR to allow a software engineer to develop, test, and configure the final deliverables to system users.

"Functional" refers to the set of functions the system is to offer, while "non functional" refers to the manner in which such functions are performed. FUR are typically phrased with subject or predicate constructions, or noun/verb, such as: "The system configuration has to register 5 personal computers to be connected with a shared printer" (i.e. the system configuration specifies the elements that define and/or prescribe the components of the system). NFR are typically phrased with adverbs or modifying clauses, such as: "The system configuration has to register 5 personal computers to be connected with a shared printer with high accessibility or controllability".

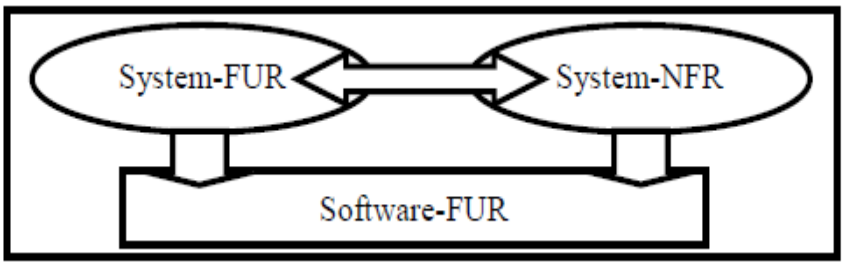

Fig. 1. Mapping system-FUR and -NFR into software-FUR

Currently, there is no generic model for the identification and specification of software FUR for system configuration NFR from the various views documented in international standards and in the literature. Consequently, it is a challenge to measure them and take them into account quantitatively for estimation purposes.
This paper reports on the work carried out to define an integrated view of software FUR for system configuration NFR on the basis of international standards, and on the use of a generic model of software FUR for system configuration NFR to measure their functional size using the COSMIC standard [16], independently of software development and implementation methodologies and technologies.

This paper is organized as follows. Section 2 presents the generic view of software FUR in ISO 19761. Section 3 identifies the standards describing a

configuration as either system NFR or as functional requirements for software and hardware. Section 4 presents a standards-based definition of a generic model of requirements for system configuration NFR. Section 5 presents a procedure for describing and measuring their functional size, and a discussion is presented in section 6.

2 A generic view of software-FUR in ISO

In the collection of ISO standards, it is specified in the ISO 14143-1 [17] that a functional size measurement method must measure the software functional user requirements (FUR). In addition, ISO 19761 – COSMIC [18] proposes a generic model of software-FUR that clarifies the boundary between hardware and software. Fig. 2 illustrates the generic flow of data from a functional perspective from hardware to software. From this generic model of software functional requirements in Fig. 2 the followings can be observed:

- Software is bounded by hardware. In the so-called "front-end" direction (i.e. left-hand side in Fig. 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. right-hand side of Fig. 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. 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 the 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 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, only those requirements allocated to the software are considered in its measurement procedure.

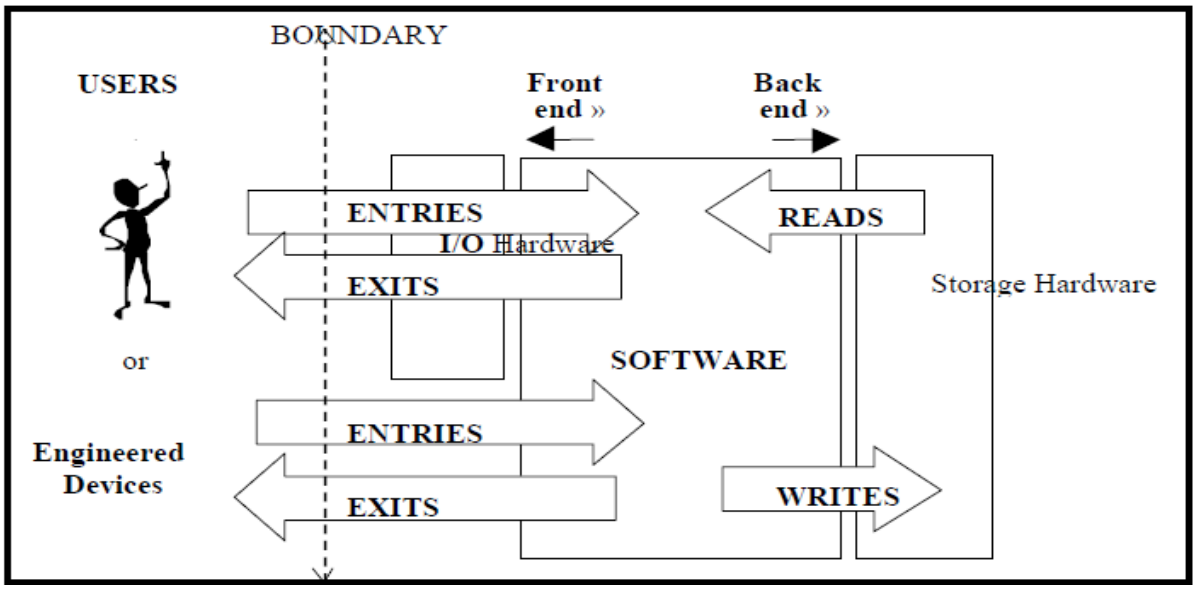

Fig. 2. Generic flow of data groups through software from a functional perspective in COSMIC – ISO 19761

3 Identification of standards for describing configuration requirements

This section presents a survey of the configuration-related views, concepts, and terms in the ECSS standards [19-22], Vincenti view of design [23] and in the SWEBOK Guide (ISO 19759) [18].

This section identifies which standards or views currently address some aspects of the software FUR derived from system NFR, specifically for the functional configuration requirements – 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 specifying software FUR for system configuration NFR.

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

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

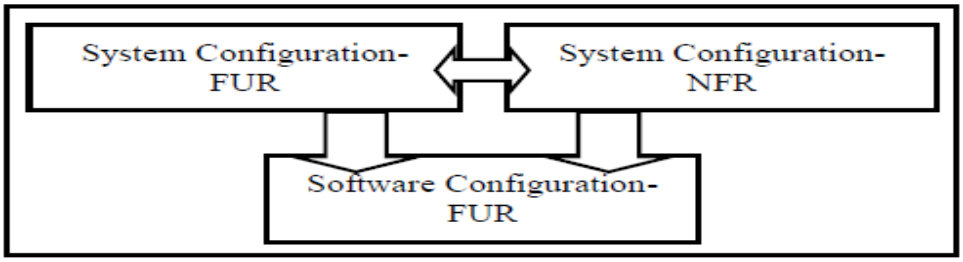

Fig. 3. Mapping system-requirements into software-FUR for Configuration requirements

3.1 Configuration requirements in ECSS standards

Configuration in the ECSS standards is considered part of the "design and implementation engineering process", including control activities and data flows for the operational functions and data transfers of defined items. Table 1 presents a list of concepts and the 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 can be configured**.** They also specify what configuration requirements shall be implemented in software.

Key views	Concepts and vocabulary
Secure environment with controlled access linked with required physical and functional characteristics of the system	• Control activities of defined configuration items. – Control flow – Data flow • Each item or component defined during the design can be configured such as: – Modules, – Processes and threads, – Events and communication channels between a modules and a sub software module • Control operational functions • Register data transfers

Table 1. Configuration requirements view and vocabulary in ECSS

While conducting a survey of all the configuration 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:

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

3.2 Configuration requirements in the SWEBOK Guide (ISO 19759)

The key view on configuration in the SWEBOK Guide (ISO 19759) is that of a system with 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. 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 points in time for the purpose of systematically controlling changes to the 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.

Key views	Concepts and vocabulary
Functional and/or physical characteristics of hardware, firmware, or software, or a combination of these	• Functional characteristics of hardware, firmware, and software. • Systematic control of changes to the configuration • Configuration control • Physical configuration audit (PCA)

Table 2. Configuration view and vocabulary in the SWEBOK

While conducting the survey of all the configuration concepts and terms described in the SWEBOK Guide, it was observed that:

- The configuration is described using both system and software views.
- The configuration elements are described at different levels of detail.
- There is no detailed measurement view in the configuration knowledge area.

3.3 Configuration requirements in Vincenti view of design

The key view on configuration as a non functional requirement in the Vincenti [23] view of design is from the engineering perspective of the quality of the system device. Configuration is presented as part of the fundamental design concepts in engineering. The inventory of related concepts and vocabulary on system NFR configuration is presented in Table 3.

For example, when Vincenti [23] defines the control volume analysis method as part of normal configuration, he is actually describing the operational functions of each item defined in the device; and each defined item should be configured. Therefore, Vincenti concludes that the normal configuration and operational functions are closely related, because both form a normal design. Similarly, for the propeller example in his book, Vincenti defines the propeller data as part of the configuration:

- The term ‘propeller’ refers to ‘a revolving shaft with spiral blades that cause a ship or an aircraft to move by the backward thrust of water or air’1: from the perspective of this research, this corresponds to registered data transfers between a set of operational functions in the system.

While conducting the survey of configuration concepts and terms described in the Vincenti view of design, it was observed that:

- Vincenti describes configuration at a highly abstract level as a philosophical view from past experience and the growth of engineering knowledge in the aeronautic field.
- This view could be applied to system and software configuration.

Key view	Concepts and vocabulary
The general shape and arrangement that, by general agreement, best embody the operational principle.	• Control volume analysis method • Propeller data

Table 3. Configuration view and vocabulary in Vincenti [23]

4 A standard based generic model of software-FUR for system configuration requirements

This section first identifies, and then assembles, the concepts and vocabularies associated with configuration elements dispersed throughout the ECSS standards, the SWEBOK Guide, and Vincenti’s view of design. These concepts and vocabularies are mapped into a proposed model of software FUR for system configuration NFR – see Fig. 3, 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 software FUR from system configuration NFR based on the ECSS standards.

Based on the synthesis of the previous configuration-related definitions, views and concepts in ECSS standards, the SWEBOK guide, and Vincenti, we can conclude that:

- They all consider configuration as an important part of the design.
- They all mention control configuration items or configuration elements such as:
  - Control flow for operational functions;
  - Data flow registered in each operational function.

1 From the World Lingo translator (online dictionary used by Google and Yahoo).

The set of software FUR for system configuration NFR based on the previous mapping is presented in Table 4.

No.	Software FUR for System Configuration NFR
1.	Configuration control flow function
2.	Configuration data flow function
3.	Register data transfer function
4.	Operational functions

Table 4. Software FUR for system configuration NFR

Two types of configuration requirements must be identified:

- Configuration control flows: the relationships between the operational functions for the configuration items or elements;
- Configuration data flows: partition of an application into pieces that can be configured individually on configurable hardware or in Software.

The entities and functional relationships of the software FUR for system configuration NFR can then be identified.

4.1 Configuration functions to be specified

The configuration functions to be specified are divided into external and internal configuration – see Table 5: The external configuration specifies the registered data that could come into the system view, while the internal configuration 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 and control flows. The ECSS view of software FUR for system configuration NFR within a secure environment includes:

- Registered data transfer, which contains 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;
- Operational functions, which define an area of responsibility within an operational function in a hierarical structure in the secure environment for the system configuration NFR.

In an embedded system, for example, the execution of an interrupt control flow is initiated by hardware. In the case of an Interrupt request (IRQ) signal, the CPU interrupts the current executing control flow and branches into an IRQ handler function. This function must not block, as doing so might freeze the system. If an IRQ handler needs to access some resource that is currently in use by a thread (or

some other IRQ handler), it cannot wait for the resource to be released. Therefore, every OS needs some mechanism to *delay* the execution of the interrupt code, or at least those parts of it accessing the resource, until the resource is available.

To solve the problem in this example, a secure environment for the system configuration NFR should not allow the blocking of IRQ handling function, because the interrupt should be registered before data transfer: more specifically, the problem in this example comes from the emergent properties between the functional and non functional requirements, which also reflect the problem on the NFR-configuration.

An IRQ line is a hardware line over which devices can send interrupt signals to the microprocessor. When, for example, a new device is added to a PC, these IRQ cases sometimes need to be reconfigured.

Configuration Types	Configuration Functions
External configuration function	• Registered data transfer function
Internal configuration function	• Operational functions

Table 5. Configuration functions that may be allocated to software

4.2 Identification of the function types in software FUR from system configuration NFR

Function Type 1: Configuration data flow

- Register data transfer 1 sends a data group to register data transfer 2.
- Register data transfer 2 receives a data group from register data transfer 1.
- Register data transfer 2 sends a data group to register data transfer n.
- Register data transfer n receives a data group from register data transfer 2.

Fig. 4: Configuration data flow

Function Type 2: Configuration control flow

- The operational function 1 sends and receives a data group to/from operational function 2.
- The operational function 2 sends and receives a data group to/from operational function n.

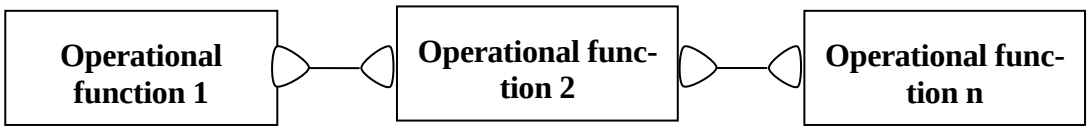

Fig. 5: Configuration control flow

Function Type 3: Configuration flow

- Register data transfer 1 to n sends a data group with at least one and possibly more operational functions;
- Operational function 1 to n receives a data group from register data transfer 1 to n;
- Control flow (function type 2) sends a data group to register data transfer 2 (function type 1) – see Figure 6, after having executed all operational functions for Register data transfer 1, to order Register data transfer 2 to accept any data group from Register data transfer 1.

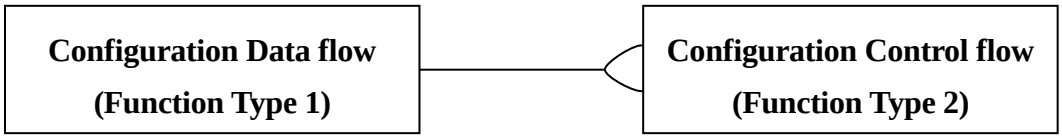

Fig. 6: Configuration flow.

4.3 Identification of the functional relationships in the software FUR for system configuration NFR

Fig. 7 presents a conceptual view of the relationships between embedded software and non functional configuration requirements. In classic embedded software, data flow and control flow are tightly coupled, and execute in lock-step [24]**.** In distributed embedded software and heterogeneous applications, the links between data flow and control flow are loosened [24]**.** A data flow configuration consists of a set of registers, data transfers, and control operational functions [25].

Fig. 7 also presents an overview of the relationships between the entity types in the software FUR for the configuration NFR, using the COSMIC model for graphical representation.

More specifically:

- The sub model of function type 1 can be used to specify the external configuration for the data flow (and to measure its functional size) throughout the registered data transfers – see the unshaded area in Figure 7.
- The sub model of function type 2 can be used to specify the internal configuration for the control flow (and to measure its functional size) throughout the operational functions – see the shaded area in Figure 7.

- The model of function type 3 can be used to specify the configuration flow for the configuration control and data flow (and to measure its functional size) – see Figure 7.

This model is referred to here as a COSMIC reference model of software FUR for system configuration NFR.

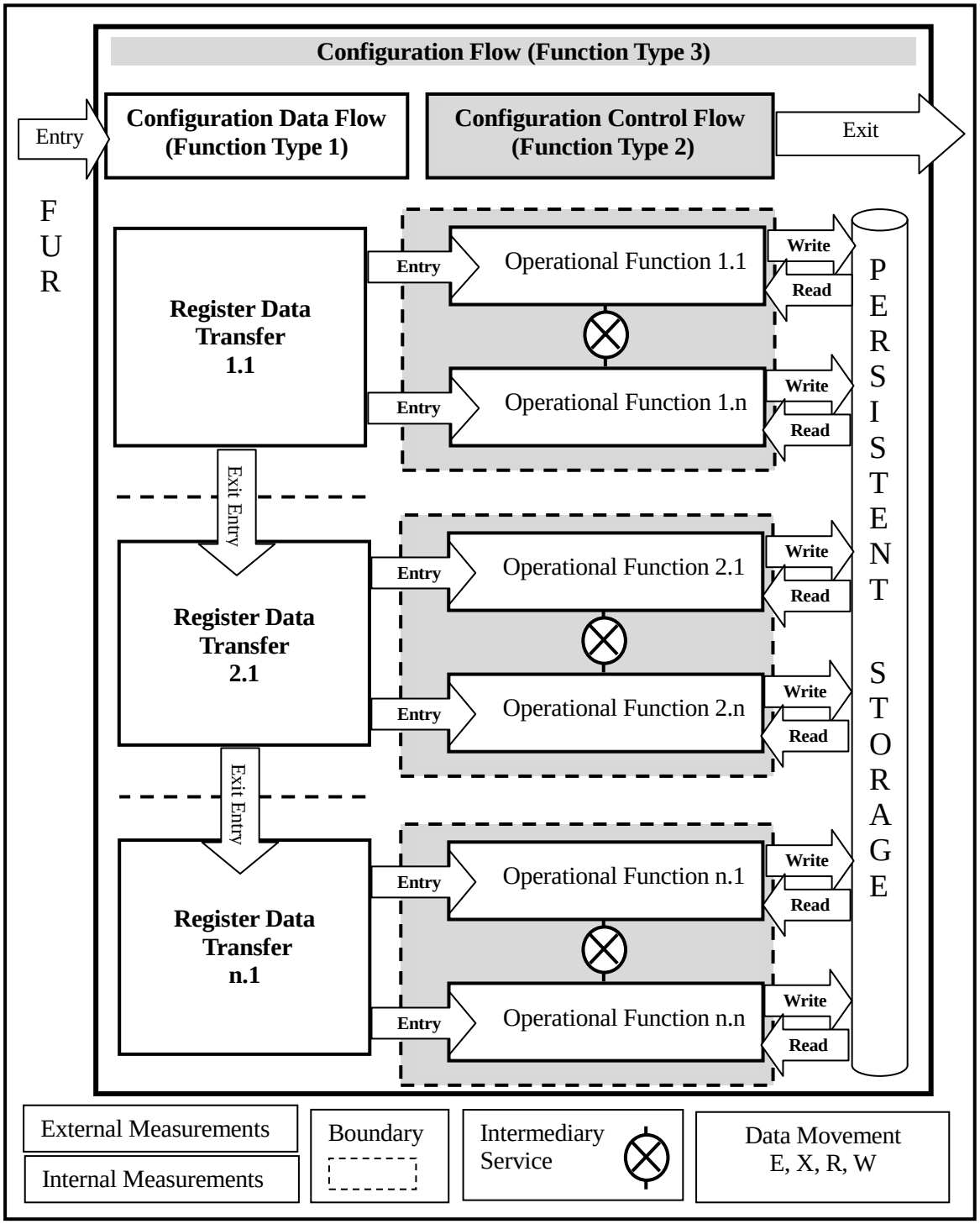

Fig. 7: COSMIC reference model of configuration requirements allocated to software

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

The specification of software FUR for system configuration NFR in any specific project is a specific instantiation of the proposed generic model, as described in Figure 7 (from left to right). When the specification document is at the level of the movements of data groups, then these configuration requirements cans be directly measured using the COSMIC measurement rules.
Table 6 presents the measurement results, using COSMIC, of a specific instantiation of configuration requirements which would have one of each of the functions and relationships described in section 4 and Figure 7. For example, for configuration data flow (function type 1): *Register data transfer* sends one data group to an operational function. This requirement corresponds to one COSMIC Entry data movement, for a functional size of 1 CFP. The corresponding total functional size of this specific generic instantiation would therefore consist of at least 15 data movements of one data group, for a total functional size of 15 CFP with the COSMIC ISO 19761 standard – see Table 6, bottom line.

Example of a functional size measurement of software FUR for system configuration NFR.

Functional Processes	**Data Movement Description**	**Data Movement Type**
Configuration data flow	• Each *Register data transfer* sends one data group to another *Register data transfer*. • Each another *Register data transfer* receives one data group from *Register data transfer* • Each *Register data transfer* entry one data group to each operational function.	**X** **E** **E**
Configuration control flow	• One operational function reads and writes one data group for each *Register data transfer*. • Each operational function sends and receives one data group to/from another operational function or (intermediary service).	**R, W** **4E, 4X**
Configuration flow	• FUR or Deviced Engineered sends one data group to the Configuration data flow • FUR or Deviced Engineered receives one data group from the Configuration Control flow.	**E** **X**
Total Cosmic Functional Size		**15 CFP**

Table 6. Example of a functional size measurement of software FUR for system configuration NFR

The intermediary used, when operational function 1 for example, requires data that is available via operational function 2, the former application service calls a functional process of the intermediary service – see Fig 8.

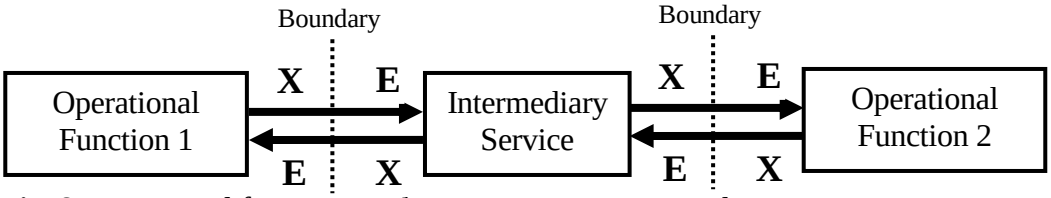

Fig. 8: operational functions and interconnecting intermediary service

6 Discussion

This paper has introduced a procedure for specifying and measuring the requirements of the software FUR for the internal and external system configuration needed to address the system NFR for configuration.

The software FUR for system Configuration NFR procedure is considered an integral part of design requirements. There is a close relationship between configuration NFR and software FUR for system-operational NFR

Each software FUR for system configuration NFR has at least one *Register data transfer* function and possibly more, and one or more control operational functions.

The main contribution of this paper is the proposed Generic Model of software FUR for system configuration NFR. This generic model is considered as a kind of reference model for the measurement of the functional size of system configuration NFR, and is based on:

- The ECSS standards, the SWEBOK Guide, and Vincenti's view of design for the description of the software FUR for the system configuration NFR.
- The COSMIC model of software functional requirements.

The model is independent of the software type and the languages in which software FUR for system configuration NFR will be implemented.

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

- A specification model for each type, or all types, of software FUR for configuration NFR: for example, the measurement for data flows (such as the Register data transfer function for configuration items) and control flows (such as operational functions on configuration items).
- A specification measurement model for each type, or all types, of configuration requirements as non functional requirements.

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

References

[1] L. Chung and P. Leite, "On Non-Functional Requirements in Software Engineering in Conceptual Modeling: Foundation and Applications, Essays in Honor of John Mylopoulos", 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] M. Shaw, "Larger Scale Systems Require Higher-Level Abstractions", Software Specification and Design, IEEE Computer Society, vol. 14, pp. 143-146, 1989.

[5] A. M. Davis, "Software requirements: objects, functions, and states", Prentice-Hall, Inc., 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] K. T. Al-Sarayreh and A. Abran, "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, 2010, pp. 217-222.

[9] K. T. Al-Sarayreh and A. Abran, "Measurement of Software Requirements Derived from System Reliability Requirements", 24th European Conference on Object-Oriented Programming (ECOOP 2010), Maribor, Slovenia, EU, 2010.

[10] K. T. Al-Sarayreh, A. Abran, J. Cuadrado, "A Standards-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, 2010.

[11] A. Abran, K. T. Al-Sarayreh, J. Cuadrado, "Measurement Model of Software Requirements Derived from System Portability Requirements", 9th International Conference on Software Engineering Research and Practice (SERP 2010), Las Vegas, USA, 2010.

[12] A. Abran and K. T. Al-Sarayreh, "Standards-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.

[13] G. Roman, " A Taxonomy of Current Issues in Requirements Engineering", IEEE Computer, pp. 14-21, 1985.
[14] B. W. Boehm, "Characteristics of software quality", Amsterdam, New York, North-Holland Pub. Co. , American Elsevie., 1978.
[15] A. I. Antón, "Goal identification and refinement in the specification of software-based information systems", PhD Thesis, Georgia Institute of Technology, 1997.
[16] ISO/IEC-19761, "Software Engineering - COSMIC v 3.0 - A Functional Size Measurement Method", International Organization for Standardization, Geneva (Switzerland), 2003.
[17] ISO/IEC-14143-1, " Information technology - Software measurement - Functional size measurement Part 1: Definition of concepts", International Organization for Standardization, Geneva (Switzerland), 1998
[18] ISO-19759, "Software Engineering Body of Knowledge (SWEBOK) ", IEEE Computer Society, 2004.
[19] ECSS-E-40-Part-1B, "Space Engineering: Software - Part 1 Principles and Requirements", European Cooperation for Space Standardaization,The Netherlands, 2003.
[20] ECSS-E-40-Part-2B, "Space Engineeing:Software- part 2 Document Requirements Definitions", European Cooperation for Space Standardaization, The Netherlands, 2005.
[21] ECSS-Q-80B, "Space product assurance: Software product assurance", European Cooperation for Space Standardaization, The Netherlands, 2003.
[22] ECSS-ESA, "Tailoring of ECSS, Software Engineering Standards for Ground Segments, Part C: Document Templates", ESA Board of Standardization and Control (BSSC), 2005.
[23] W. G. Vincenti, "What engineers know and how they know it", Baltimore, London, The Johns Hopkins University Press, 1990.
[24] P. Schaumont, K.Sakiyama, A.Hodjat, "Embedded Software Integration For Coarse-Grain Reconfigurable Systems", International Parallel And Distributed Processing Symposium ,IEEE, 2004.
[25] D. W. Lewis, "Fundamentals of Embedded Software: Where C and Assembly Meet", Prentice Hall Professional Technical, 2002.