A standard-based reference framework for system operations requirements

Khalid T. Al-Sarayreh and Ibrahim Al-Oqily*

Departments of Software Engineering
and Computer Science and Applications,
Hashemite University, Zarqa 13115, Jordan
E-mail: khalidt@hu.edu.jo
E-mail: izaloqily@hu.edu.jo
*Corresponding author

Kenza Meridji

Department of Software Engineering,
Petra University, Amman 11196, Jordan
E-mail: kmeridji@uop.edu.jo

Abstract: European series of standards for the aerospace industry, includes system operations requirements as one of 16 types of NFRs for embedded and real-time systems. A number of concepts are provided in the ECSS and Institute of Electrical and Electronics Engineers (IEEE) standards to describe the various types of candidate operations requirements at the system, software and hardware levels. In the absence of such a generic and detailed model, these NFRs are typically handled much later on in the software development life cycle, when, at system testing time, users and developers may discover that a number of system operations requirements have been overlooked and additional work is required to implement them. This paper organises these dispersed operations concepts into a standards-based reference framework of system operations requirements.

Keywords: operations requirements; NFRs; non-functional requirements; functional size; COSMIC – ISO 19761; ECSS and IEEE Standards; operations measurement.

Reference to this paper should be made as follows: Al-Sarayreh, K.T., Al-Oqily, I. and Meridji, K. (2013) 'A standard-based reference framework for system operations requirements', *Int. J. Computer Applications in Technology*, Vol. 47, No. 4, pp.351–363.

Biographical notes: Khalid T. Al-Sarayreh is an Assistant Professor of Software Engineering at Hashemite University in Jordan. He has a PhD in Software Engineering from École de Technologie Supérieure (ÉTS), University of Québec in Canada. He also has a Doctoral degree in Computer Information Systems, MSc in Computer Engineering (Embedded Systems) and BS in Computer Science from Jordanian Universities. From 2002 to 2005, he was at the KADDB (King Abdullah II Design and Development Bureau). His research interests include software real time, software quality, non-functional requirements for embedded systems and HCI.

Ibrahim Al-Oqily received his BSc from the Department of Computer Science, Mu'tah University, Jordan, in 1993, MSc from the Department of Computer Science, Jordan University, Jordan, in 2003, and PhD from the Department of Computer Science, the School of Information Technology and Engineering, University of Ottawa, Canada. He is currently working as a Vice Dean at Prince Al-Hussein bin Abdullah II Faculty for Information Technology in the Hashemite University. His current research interests include overlay networks management, autonomic management and policy-based network management. He is an active member of the IEEE since 2006.

Kenza Meridji is an Assistant Professor at Petra University Amman – Jordan. She is pursuing research on the fundamental principles of software engineering and non-functional requirements. She holds a PhD from the École de Technologie Supérieure, Université du Québec, Montreal, Canada, and a Master's degree in Software Engineering from Concordia University, Canada.

This paper is a revised and expanded version of a paper entitled 'Measurement of software requirements derived from system operations requirements' presented at *20th International Workshop on Software Measurement & International Conference on Software Measurement, IWSM/ MetriKon/Mensura*, Stuttgart, Germany, pp.101–114.

1 Introduction

During the system requirements gathering phase, the focus is often on the functional requirements (FRs) of the system, whereas Non-Functional Requirements (NFRs) are often captured by system analysts at a very global level, with a lack of attention to detail: detailing these NFRs is typically left to be handled (i.e., defined at the necessary level of detail) much later by system designers in the architecture and design phases (Al-Sarayreh et al., 2010).

The NFRs play a critical role in system development. They may have a considerable impact on project effort, and should be taken into account for estimation purposes and in comparing project productivity. In the system analysis phase, the NFRs are typically described at the system level (Abran et al., 2013; Al-Sarayreh et al., 2012a, 2012b; Meridji et al., 2013; Al-Khasawneh et al., 2013) and not at the software level. As yet, there is no consensus on how to describe and measure system NFR. In current practice, they may be viewed, defined, interpreted and evaluated differently by different people in the later project phases, particularly when they are stated vaguely and only briefly in the system requirements phase (Abran et al., 2013; Al-Sarayreh et al., 2012a, 2012b; Meridji et al., 2013). It is challenging, therefore, to take them into account in software estimation and software productivity benchmarking, particularly as they have received less attention in the software engineering literature and are definitely less well understood than other cost factors (Chung and Leite, 2009). Of course, measurement is essential if NFRs are to be taken as quantitative inputs to an estimation or productivity benchmarking process.

In practice, requirements are initially typically addressed at the system level (Abran et al., 2013; Al-Sarayreh et al., 2012a, 2012b; Meridji et al., 2013), either as high-level system Functional User Requirements (system FURs) or as high-level system NFRs. The latter must usually be detailed, allocated and implemented in either hardware or software, or both, as software FUR.

To distinguish between these types of requirements, system FUR describes the required functions in a system, whereas system NFR describes how the required functions must behave in a system (Abran and Al-Sarayreh, 2010a, 2010b). 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, whereas the term ‘non-functional’ refers to the manner in which such functions perform. A FUR is typically phrased with a subject and a predicate (i.e., noun/verb), such as: “The system must print 5 reports”. NFR, by contrast, is typically phrased with an adverb or modifying clause, such as: “The system will print 5 reports quickly”, or “The system will print 5 reports with a high degree of reliability”.

In the ECSS standards for the aerospace industry (ECSS-E-40-Part-1B, 2003; ECSS-E-40-Part-2B, 2005; ECSS-Q-80B, 2003; ECSS-E-ST-10C, 2009; ECSS-E-ST-32 C-Rev.1, 2008; ECSS-E-ST-33-01, 2009; ECSS-E-ST-70, 2008; ECSS-Q-ST-20, 2008; ECSS-Q-ST-80, 2009; ECSS-S-ST-00, 2008) and the IEEE 830 standard (IEEE-Std-830, 1998), a number of concepts are provided to describe various types of candidate operations requirements at the system, software and hardware levels. However, these standards vary in their views, terminology and coverage of operations.

This paper proposes a generic model for the identification and specification of software FUR for implementing system operations requirements (system NFR) based on the various views documented in international standards and in the literature (Abran et al., 2013; Al-Sarayreh et al., 2012a, 2012b; Meridji et al., 2013).

This paper focuses on a single type of NFR, i.e., system operations requirements, and reports on the work carried out to define an integrated view of software FUR for system operations NFR based on international standards, including the use of the generic Common Software Measurement International Consortium (COSMIC) – ISO 19761 (ISO/IEC-19761, 2011) model of software FUR.

The paper is organised as follows. Section 2 presents the related work. Section 3 presents the standards-based framework of system operations requirements: identification phase. Section 4 presents the standards-based framework of system operations requirements: specification phase. Section 5 presents a standards-based framework of system operations requirements: measurement phase. Section 6 presents a sizing of a reference instantiation of the generic model of operations software FUR (a measurement example). Finally, a discussion and conclusion are presented in Section 7.

2 Related work

In the literature, there are some early works on NFR in system/software engineering (Abran et al., 2013; Al-Sarayreh et al., 2012a, 2012b; Meridji et al., 2013). For instance, Chung and Leite (2009) presented one of the initial attempts to capture knowledge in this domain. His work was followed by that of Mylopoulos (2006), who suggested viewing all requirements as goals, each goal being an umbrella for related functional and NFRs. Chung and Leite (2009) and Andrew (2000) aimed to make NFR more quantitative in nature, while Andrew (2000) found that there are often gaps between the stakeholder vision and requirements representation. Chung et al. (2000) proposed taxonomy for NFR indicating that it is unrealistic to expect designers and developers to incorporate an entity that they cannot readily identify. While taxonomies aim to be inclusive of the entire set of entities in question, these authors suggested in Chung et al. (2000) that a one- or two-level taxonomy would suffice initially, and that there are over 161 identifiable types of NFR.

Paech et al. (2002) recommended that functional requirements (FR), NFR and architecture be tightly

co-developed and addressed in a coherent and integrated manner, suggesting that NFR be decomposable into more refined NFR and additional FR, as well as architectural decisions. Moreira et al. (2002), Rosa et al. (2002), Park and Kang (2004) and Glinz (2005) have proposed new methods for classifying NFR early in the software development process, while Kaiya et al. (2004) have presented a method to identify stakeholders and their NFR preferences by using use-case diagrams of the existing systems.

More recently, Mylopoulos (2006) promoted Goal-Oriented Requirements Engineering, and suggested a specific solution involving the establishment of an Agent-Oriented Software Development Method, called the Tropos project, which covers not only the requirements, but also the design phases, and addresses the design of high-variability software for applications such as home care software and business process design.

More recently still, Fernández-Sanz and Misra (2012) worked to apply a simple evaluation experience called Teamwork Benefits Awareness (TBA) to group the students of computing degrees with experience as junior IT professionals based on international software projects. In contrast, Tiako (2011) proposed a new approach of process modelling for its remote performance. The main advantage of this approach is to maintain collaboration among autonomous Process-centred Software Engineering Environments (PSEEs) during process performance. Jindan et al. (2012) presented a feature-based platform-specific modelling approach for coarse-grained components. The component is described from the perspectives of function and implementation by establishing mapping between the feature models.

In parallel with the work of researchers, the software industry has been working on the description of NFR, in particular through international standardisation bodies, such as the ECSS, the IEEE and the ISO. In the ECSS standards for the aerospace industry (ECSS-E-40-Part-1B, 2003; ECSS-E-40-Part-2B, 2005; ECSS-Q-80B, 2003; ECSS-E-ST-10C, 2009; ECSS-E-ST-32 C-Rev.1, 2008; ECSS-E-ST-33-01, 2009; ECSS-E-ST-70, 2008; ECSS-Q-ST-20, 2008; ECSS-Q-ST-80, 2009; ECSS-S-ST-00, 2008), a system operations requirement is identified as one of the 16 types of NFR. In addition, in the IEEE 830 (IEEE-Std-830, 1998) standards, a number of implicit concepts are provided to describe various types of candidate system operations requirements at the system and software levels in the testing and evaluation processes.

However, these standards vary in their views, terminology and coverage of operations. Currently, there exists no generic model for the identification and specification of software FUR for implementing system operations requirements (system NFR) based on the various views documented in these international standards and in the literature (Abran et al., 2010). Consequently, it is challenging to measure the system-operations-related software FUR, and take them into account quantitatively for estimating software projects.

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

This paper focuses on a single type of NFR, i.e., system operations requirements, and reports on the work carried out to define an integrated view of software FUR for system operations NFR on the basis of international standards, including the use of the generic COSMIC – ISO 19761 (ISO/IEC-19761, 2011) model of software FUR and COSMIC-SOA (2010) as the template for the description of measurable functional requirements.

3 A standards-based framework of system operations requirements (identification phase)

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

3.1 Operations requirements in ECSS standards

The ECSS is an organisation that works to improve standardisation within the European space sector. The ECSS frequently publishes standards targeting the contractors working for the European Space Agency (ESA). The ECSS standards series (ECSS-E-40-Part-1B, 2003; ECSS-E-40-Part-2B, 2005; ECSS-Q-80B, 2003; ECSS-E-ST-10C, 2009; ECSS-E-ST-32 C-Rev.1, 2008; ECSS-E-ST-33-01, 2009; ECSS-E-ST-70, 2008; ECSS-Q-ST-20, 2008; ECSS-Q-ST-80, 2009; ECSS-S-ST-00, 2008) includes a number of operations requirements at the system level. Clearly, the ECSS focuses on the system FUR for the early development phases, whereas the system NFRs are typically discussed within the context of later development phases, such as evaluation or testing. The elements of operations are dispersed in various system views throughout various ECSS standards, and are expressed as either system operations FUR, or system operations NFR.

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

An analysis of the overall operations requirements in the ECSS standard series has revealed that these standards list

the system and software operations FUR early in the development process. Later on, the system operations NFR are used in the evaluation and testing phases. This paper refines these system operations NFR to take them into account much earlier in the development cycle: in a feasibility study and at the software requirements phase.

Table 1 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related operations requirements. For instance, the ECSS specifies that, for system operations mode, an analysis of the operational functions (Inter-Operational Function (IOPF) and Operational Function Event (OPFE)) and of the system transitions mode (Operational Control Interface (OPCI) and Operational Data Interface (OPDI)) must be carried out. The ECSS specifies that such requirements must be implemented in software or hardware, or a combination of the two.

Table 1 ECSS operations requirements view and concepts

Key views	*Concepts and vocabulary*
Operational and transition modes	Inter-operational function
	Operational function event
	Operational control interface
	System operations mode
	Operational data interface
	System transitions mode
	Operational scenario

A survey of all the operations concepts and terms described in the ECSS-E-40 and ECSS-Q series and in the ECSS-ESA standard (i.e., the integrated standard for ECSS-E and ECSS-Q) reveals that:

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

We can also note that the ECSS does not propose a way to measure such operations requirements when allocated to software, and, without measurement, it is challenging to take such NFR either as quantitative input to an estimation process or in productivity benchmarking.

3.2 Operations requirements in IEEE standards

IEEE-830 (IEEE-Std-830, 1998) includes operations as one of the NFR types in their list of NFR, and considers the various modes of operation as part of the user interface. But it neither defines what an operations requirement is, nor provides guidance on how to describe and specify the operations requirements. Of course, it does not provide guidance on how to measure any of these NFRs either.

3.3 Operations requirements views into standards

This section assembles the terminologies and concepts of system operations dispersed throughout the ECSS and IEEE standards. There are two types of system-related operations requirements that can be derived from the ECSS standards series: system operations mode and system transitions mode.

Table 2 presents the two types of system operations requirements and related functions, which are included as system requirements in the ECSS and IEEE standards. These could at times be interpreted and specified as software FUR:

- *System operations mode*: This refers to the expected operations for the executed functions occurring in the system. The system operations mode consists of the IOPFs and the OPFEs.
- *System transitions mode*: This refers to the expected data and control operations via the interface functionality that could occur in the system. The system transitions mode consists of Operational Data Interface Functions (OPDIFs) and Operational Control Interface Functions (OPCIFs).

Table 2 System operations FUR in the ECSS standards series

Types of system operations	*Operations functions to be specified*
System operations mode	Inter-operational function (IOPF)
	Operational function event (OPFE)
System transitions mode	Operational data interface function (OPDIF)
	Operational control interface function (OPCIF)

According to ECSS standards (ECSS-E-40-Part-1B, 2003; ECSS-E-40-Part-2B, 2005; ECSS-Q-80B, 2003; ECSS-E-ST-10C, 2009; ECSS-E-ST-32 C-Rev.1, 2008; ECSS-E-ST-33-01, 2009; ECSS-E-ST-70, 2008; ECSS-Q-ST-20, 2008; ECSS-Q-ST-80, 2009; ECSS-S-ST-00, 2008), the functional relationships across these two modes, as illustrated in Figure 1, are the following:

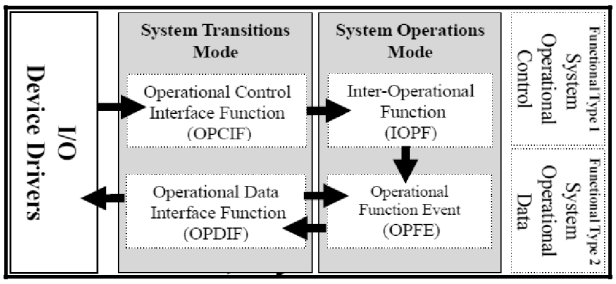
- The IOPFs (in operations mode), which are controlled by the OPCIF (in transitions mode). This relationship will be referred to in this paper as the 'System Operational Control', or Functional Type 1.
- The OPFEs (in operations mode), which send and receive data movements from the OPDIF (in transitions mode). This relationship will be referred in this paper as the 'System Operational Data', or Functional Type 2.

For example, in embedded and real-time software (ECSS-E-40-Part-1B, 2003; ECSS-E-40-Part-2B, 2005; ECSS-Q-80B, 2003; ECSS-E-ST-10C, 2009; ECSS-E-ST-32 C-

Rev.1, 2008; ECSS-E-ST-33-01, 2009; ECSS-E-ST-70, 2008; ECSS-Q-ST-20, 2008; ECSS-Q-ST-80, 2009; ECSS-S-ST-00, 2008):

- a system scheduler sends distribution routines, which form the OPCI and the IOPFs
- the system device routines form the OPDI and the OPFEs.

Figure 1 System operations functions and functional types in the ECSS standards series

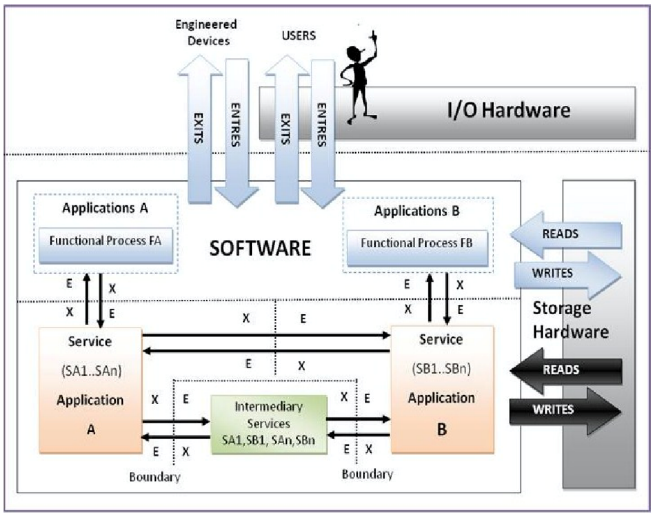


In the next section, these terminologies are mapped into a proposed model of operations software FUR, using the generic FUR model proposed in COSMIC – ISO 19761. This COSMIC-based generic model will be used as a framework for describing the software FUR from system operations requirements based on the ECSS standards.

4 A standards-based framework of system operations requirements (specification phase)

This section identifies and assembles the terminologies and concepts of reliability dispersed throughout the ECSS and IEEE standards. Next, these terminologies are mapped into a proposed model of software-FUR for system operations using the generic FUR model proposed in COSMIC – see Figure 2.

Figure 2 Generic flow of data groups through software from a functional perspective in COSMIC – ISO 19761 (2011) and COSMIC-SOA (2010) (see online version for colours)



4.1 A generic ISO view of software FUR and the Service Oriented Architecture (SOA)

ISO 14143-1 (ISO/IEC-14143-1, 2008) specifies the collection of ISO standards that a Functional Size Measurement (FSM) method must measure software FUR. In addition, ISO 19761 – COSMIC (ISO/IEC-19761, 2011) and COSMIC-SOA (2010) – proposes a generic model of software FUR that clarifies the boundary between hardware, software and a service architectural level for ISO 19761 (COSMIC-SOA, 2010). Figure 2 illustrates the generic flow of data from a functional perspective from hardware to software. From this generic model of software functional requirements in Figure 2, we observe the following:

- Software is bounded by hardware. In the so-called 'front-end' direction (i.e., the left-hand side in Figure 2), software used by a human user is bounded by I/O hardware, such as a mouse, a keyboard, a printer, or a display, or by engineered devices, such as sensors or relays. In the so-called 'back-end' direction (i.e., the upper-side of Figure 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 characterised 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; i.e., 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; i.e., the 'I/O hardware' is ignored.
- The software functionality in ISO 19761 can be measured at "a service architectural level by providing functional users with a set of services by exchanging messages in the application layer and a service between two peer pieces of software FUR. For instance, if an application requiring commonly used information from another application sends a request to the service of the application that can handle the request, or the application may call upon its own services. Such calls are also called 'messages'. Each message may consist of one or more data movements" (COSMIC-SOA, 2010).
- When "a functional process of an application service in application A requires data that are available via an

> application service in application B, the former application service calls upon a functional process of the intermediary service. This service functionality is also needed by other applications in the overall Service-Oriented Architecture (SOA) framework, as it may itself be realised in the form of a utility service" (COSMIC-SOA, 2010).

The COSMIC functional size measurement method (ISO/IEC-19761, 2011) is supported by the COSMIC. In the measurement of software functional size using COSMIC, the software functional processes and their triggering events must be identified. The unit of measurement in this method is the data movement, which is a base functional component that moves one or more data attributes belonging to a single data group. Data movements can be of four types: Entry (E), Exit (X), Read (R) or Write (W).

The functional process is an elementary component of a set of user requirements triggered by one or more triggering events. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The subprocesses of each functional process constitute sequences of events (see Figure 2 for an illustration of the generic flow of data groups through software from a functional perspective).

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

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

4.2 System and COSMIC modelling for system operations requirements

This section identifies the function types that may be allocated to software-FUR for system operations, and the relationships between them.

4.2.1 Functional type 1: System operational control

Operational control is defined as the use of an element or elements of the control flow to perform a specific mission. System operational control directs the assigned authority to accomplish specific missions or tasks, which are usually limited by the function. System operational control includes: OPCI and IOPFs, which are considered part of the system operations mode.

A system modelling view (i.e., a high-level view) of the data movements for system operations control (Functional Type 1):

- *OPCI*: used by interacting users or devices to send data flows to IOPFs in the system operations mode
- *IOPF*: receives information and transforms it into variables, and then classifies the variables with their data to send the data to the next functional type, see Figure 3
- the individuals in a set of IOPF contact one another through intermediary services to allow different types of data movement (symbol ⊕ in Figure 3)
- a set of IOPF in Functional Type 1 (see Figure 3) sends additional data flows to a set of OPFEs in Functional Type 2 in Figure 3.

COSMIC modelling view of the data movements for the system operational control (Functional Type 1):

- a functional user (which may include a device) may send a data group (i.e., an ENTRY) to an OPCIF
- an OPCIF, based on the number of commands, may send a data group (i.e., an ENTRY) for each IOPF from 1 to *n*
- IOPFs (IOPF1 to *n*) may receive a set of data groups (i.e., an ENTRY) from an OPCIF
- IOPFs (IOPF1 to *n*) may interact (i.e., an EXIT–ENTRY) with each other through intermediary services
- Each IOPF (IOPF1 to *n*) in functional type 1 (see Figure 3) may send–receive a data group (i.e., an EXIT–ENTRY) to-from an OPFE (OPFE1 to *n*) in Functional Type 2 – see also Figure 3.

4.2.2 Functional type 2: System operational data

System operational data are designed to integrate the data from multiple sources into a single instruction for the system through an OPDI from-to the OPFE. System operational data include information about: the OPDI and the stored information used by an OPFE.

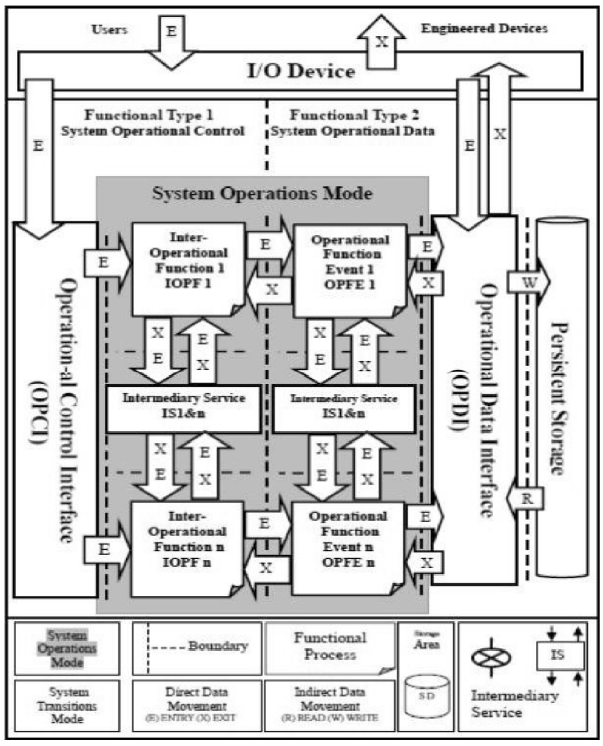
A system modelling view (i.e., high-level view) of the data movements for the system operational data (Functional Type 2):

- an OPFE in functional type 2 in Figure 3 receives a request from an IOPF from functional type 1 (see Figure 3), to provide output in a useful format.
- an OPFE reads and writes, through an OPDI
- (OPDI) data information based on the IOPF request.

A COSMIC modelling view of the data movements for the system operational data (Functional Type 2):

- an OPFE (OPFE 1 to *n*) in functional type 2 (Figure 3) may send data groups to an IOPF from (IOPF 1 to *n*) in functional type 1 – Figure 3
- OFEs (OPFE 1 to *n*) may use intermediary services to link some of their added information
- an OPFE (OPFE 1 to *n*) may send and receive a data group through an OPDIF
- an OPDIF may read and write the requested data from a storage device.

Figure 3 COSMIC reference model of operations requirements allocated to software (Functional Process Level)



4.3 Integrated view of the functional relationships for system operations

This section presents an integration of the system and COSMIC modelling views presented in the previous sections (see Figure 3):

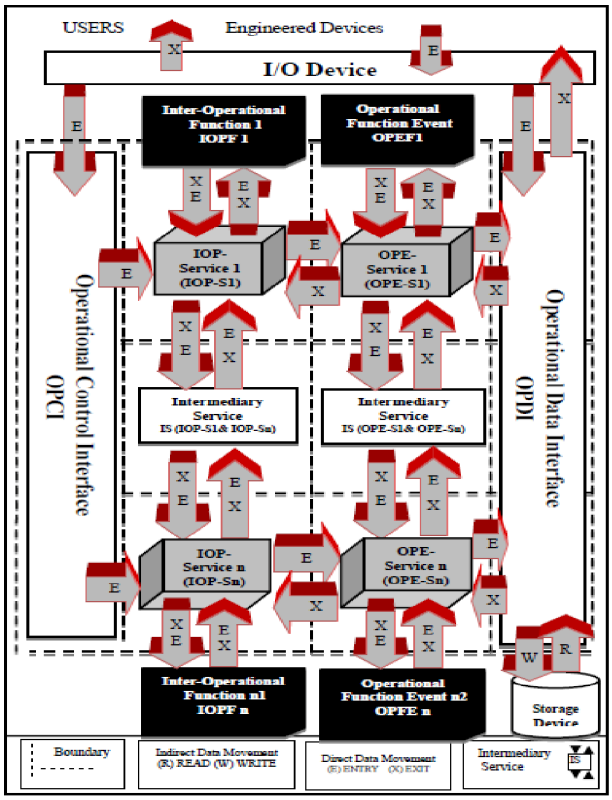
- the shaded area represents the functionality in the system operations mode
- the white area represents the functionality in the system transition model.

This integrated model is referred to in this paper as a generic model of software FUR for system operations. It can be used to specify the functional requirements derived from the system operations requirements, as well as to measure their functional size with the COSMIC ISO 19761 standard.

4.4 A standard generic measurement model of software FUR using an SOA for system operations

In this section, a COSMIC reference architectural model using an SOA in Figure 4 is built based on Figure 3 to show a more complete picture, which includes showing what is involved in instantiating the modelled entities in practice. Figure 4 illustrates a COSMIC reference architectural model using an SOA for system operations requirements. This model is built based on Figure 3 and the role of the COSMIC-SOA explained in Figure 2.

Figure 4 COSMIC reference model of operations requirements allocated to software (Functional Process and Services Levels) (see online version for colours)



5 A standards-based framework of system operations requirements (measurement phase)

5.1 COSMIC-SOA measurement context

The specification of software FUR for system operations in any specific project is a specific instantiation of the proposed generic model described in Figure 4. When the software specification document is at the level of the movement of data groups, then these functional requirements can be directly measured using the COSMIC measurement rules. The measurement example presented next is illustrative of a reference instantiation of the generic COSMIC specification and measurement model of software.

FUR for system operations in an SOA context for a single data group for all the identified possible flows of data groups. The measurement example in this section explains how to use the proposed reference model of system operations to size a hypothetical framework composed of all of the kinds of software FUR described in the framework.

5.2 Measurement of exchange services for system operations functionality

There are two types of system operations functionality, with their own services interacting, for the measurement of exchange services for system operations functionality using COSMIC-SOA:

- IOPFs from IOPF1 to IOPF*n* interacting with inter-operational services from IOP-S1 to IOP-S*n*
- OPFEs from OPFE1 to OPFE*n* interacting with operational event services from OPE-S1 to OPE-S*n*
- each inter-operational service must have the same corresponding operational event service; for instance, IOP-S1 must have OPE-S1
- according to COSMIC-SOA, each functional process may interact with its own service by sending and receiving data movements, i.e., Entry and Exit.

Table 3 illustrates the COSMIC-SOA measurement results for interactions between the IOPF1 with its own service, i.e., inter-operational service 1 (IOP-S1). The measurement result for this operation is equal to 4 CFP for each interaction of a data group between IOPF1 and IOP-S1.

Table 3 COSMIC-SOA measurement for the interactions between IOPF1 and IOP-S1

Type		*Data movement description*	*DMT*
IOPF1	IOP-S1	Inter-operational function 1 (IOPF1) sends a data group to inter-operational service 1 (IOP-S1)	X
		Inter-operational service 1 (IOP-S1) receives a data group from inter-operational function 1 (IOPF1)	E
		Inter-operational service 1 (IOP-S1) sends a data group to inter-operational function 1 (IOPF1)	X
		Inter-operational function 1 (IOPF1) receives a data group from inter-operational service 1 (IOP-S1)	E
The Total Functional size for One Operation			*4CFP*

Table 4 presents the COSMIC-SOA data movements considered for measuring the interactions between the OPEF 1 and its own service or operational event service *n* (OPE-S1). The measurement result for this operation is equal to 4 CFP for each interaction between OPEF *n* and OPE-S*n* (*n* represents each time a functional process interacts with a functional service).

Table 4 COSMIC-SOA measurement for the interactions between OPEF1 and OPE-S1

COSMIC-SOA types		*Data movement description*	*DMT*
OPEF1	OPE-S1	Operational event function 1 (OPEF1) sends a data group to operational event service 1 (OPE-S1)	X
		Operational event service 1 (OPE-S1) receives a data group from operational event function 1 (OPEF1)	E
		Operational event service 1 (OPE-S1) sends a data group to operational event function 1 (OPEF1)	X
		Operational event function 1 (OPEF1) receives a data group from operational event service 1 (OPE-S1)	E
The Total Functional size for One Operation			*4 CFP*

5.3 Measurement of intermediary services for system operations services

In this section and based on Figure 4, when a functional process service requires data that is available via another functional process service, the former calls upon a functional process of the intermediary service. Specifically:

- Each two inter-operational services from IOP-S1 to IOP-Sn can call upon one intermediary service between them. For example, with IOP-S1 and IOP-S2 or IOP-S2 and IOP-S3, each pair can call upon one intermediary service between them.
- Each two operational event services from OPE-S1 to OPE-Sn can call upon one intermediary service between them. For example, with OPE-S1 and OPE-S2 or OPE-S2 and OPE-S3, each pair can call upon one intermediary service between them.
- According to the COSMIC-SOA model of measurement for system operations, there are no intermediary services between inter-operational services and operational event services.
- According to the COSMIC-SOA model of measurement for system operations, the types of data movements for using the intermediary service must be Entry and Exit.

Table 5 illustrates the COSMIC-SOA measurement results for one intermediary service between IOP-S1 and IOP-S2. This table presents an instantiation of a single data group for all possible flows of the data groups identified earlier. The measurement results are equal to 8 CFP.

Note: Intermediary service 1&2: represents a single intermediary service (i.e., between functional service 1 and functional service 2, there is an Intermediary service 1&2.)

Table 5 COSMIC-SOA measurement for an intermediary service between two inter-operational services

COSMIC-SOA types		*Data movement description*	*DMT*
IOP-S1	IOP-S2	Inter-operational service 1 (IOP-S1) sends a data group to intermediary service 1&2 (IS1&2)	X
		Intermediary service 1&2 receives a data group from inter-operational service 1 (IOP-S1)	E
		Intermediary service 1&2 sends a data group to inter-operational service 2 (IOP-S2)	X
		Inter-operational service 2 (IOP-S2) receives a data group from intermediary service 1&2	E
		Inter-operational service 2 (IOP-S2) sends a data group to intermediary service 1&2	X
		Intermediary service 1&2 receives a data group from inter-operational service 2 (IOP-S2)	E
		Intermediary service 1&2 sends a data group to operational service 1 (IOP-S1)	X
		Inter-operational service 1 (IOP-S1) receives a data group from intermediary service 1&2 (IS1&2)	E
The Total Functional size for One Operation			*8 CFP*

5.4 Measurement of the direct and indirect data movements for system operations services using COSMIC-SOA

This section is based on Figure 4, which illustrates the possible flows of data between components in the same layer, i.e., between peer components (where a component may be an application or a service). Specifically, Table 6 illustrates COSMIC-SOA measurement results for the exchange of data movements between the system operations requirements model in a functional process or in service architecture layers – see Figures 3 and 4. This table presents an instantiation of this operation. The measurement results are equal to 21 CFP.

Table 6 COSMIC-SOA measurements for direct and indirect data groups for system operations

Types	*Data movement description*	*DMT*
User or engineered device	User or engineered device sends a data group to an I/O device	E
	User or engineered device receives a data group from an I/O device	X
I/O device	I/O device sends a data group to the operational control interface	E
I/O device	I/O device sends a data group to the operational data interface	E
	I/O device receives a data group from the operational data interface	X
Operational Control Interface Function (OPCIF)	Operational control interface sends a data group to the inter-operational service 1(IOP-S1)	E
	Operational control interface sends a data group to the inter-operational service n (IOP-Sn)	E
Inter-Operational Service 1 (IOP-S1)	Inter-operational service 1 (IOP-S1) sends a data group to operational event service 1 (OPE-S1)	X
	Operational event service 1 (OPE-S1) receives a data group from inter-operational service 1 (IOP-S1)	E
Operational Event Service 1 (OPE-S1)	Operational event service 1 (OPE-S1) sends a data group to inter-operational service 1 (IOP-S1)	X
	Inter-operational service 1 (IOP-S1) receives a data group from operational event service 1 (OPE-S1)	E
Inter-operational service n (IOP-Sn)	Inter-operational service n (IOP-Sn) sends a data group to operational event service n (OPE-Sn)	X
	Operational event service n (OPE-Sn) receives a data group from inter-operational service n (IOP-Sn)	E
Operational Event Service n (OPE-Sn)	Operational event service n (OPE-Sn) sends a data group to inter-operational service n (IOP-Sn)	X
	Inter-operational service n(IOP-Sn) receives a data group from operational event service n (OPE-Sn)	E
Operational Data Interface Function (OPDIF)	Operational data interface receives a data group from operational event service 1 (OPE-S1)	E
	Operational data interface receives a data group from operational event service n (OPE-Sn)	E
	Operational data interface sends a data group to operational event service 1 (OPE-S1)	X
	Operational data interface sends a data group to operational event service n (OPE-Sn)	X
	Operational data interface reads and writes a data group from/to persistent storage	R and W
Total Cosmic Functional Size		*21 CFP*

6 Measurement example context

The specification of software FUR for system operations requirements in any specific project is a specific instantiation of the proposed generic model described in Figure 4. When the software specifications document is at the level of the movement of data groups, then these functional requirements can be directly measured using the

COSMIC measurement rules. This section presents a specific measurement example of the use of the COSMIC generic model of system operations requirements allocated to software.

Example: The set of functional requirements allocated to software for the system operations requirements for a specific instantiation is the following:

- the OPCI sends one data group to IOP-Service 1
- the OPCI sends one data group to IOP-Service 2
- IOP-Service 1 sends a data group to OPE-Service 1
- IOP-Service 2 sends a data group to OPE-Service 2
- OPE-Service 1 sends a data group to the OPDI
- OPE-Service 2 sends a data group to the OPDI.

On the basis of Figures 3 and 4 for the COSMIC generic model of system operations requirements allocated to software, and on the COSMIC-SOA for specifying data movements, the measurement procedure to determine the functional size for operations on Functional Types 1 and 2 for this example is presented in the following sections.

6.1 COSMIC-SOA measurement of exchange messages

The functional process interacting with the functional service in this example includes the following data movements:

- IOPF1 in the application layer sends a data group to IOP-Service 1, which means that one functional process will interact with one functional service; size = 1 CFP.
- IOPF2 in the application layer sends a data group to IOP-Service 2, which means that one functional process will interact with one functional service; size = 1 CFP.
- OPEF1 in the application layer sends a data group to OPE-Service 1, which means that one functional process will interact with one functional service; size = 1 CFP.
- OPEF2 in the application layer sends a data group to OPE-Service 2, which means that one functional process will interact with one functional service; size = 1 CFP.

The number of functional services = 4 (based on measurement example);

- each functional process in the application layer will interact with each service; the data movements between each functional process and service = 4 CFP (see Section 5.2).
- therefore, the subtotal functional size for all services = 4 functional services × 4 CFP = **16** CFP.

6.2 COSMIC-SOA measurement of intermediary services

The intermediary services interacting with each functional service in this example include the following data movements:

- IOP-Service 1 has one functional service, and IOP-Service 2 also has one functional service; IOP-Services 1 and 2 need one intermediary service to contact each other – see Figure 4
- OPE-Service 1 has one functional service, and the OPE-Service 2 also has one functional service; OPE-Services 1 and 2 need one intermediary service to contact each other – see Figure 4.

Therefore, each intermediary service includes 8 data movements or 8 CFP. In this example, two intermediary services are needed – see Figure 4;

- Therefore, the functional size for the two intermediary services is 8 CFP each, for a subtotal of **16** CFP.

6.3 COSMIC-SOA measurement of data exchange between components

Table 7 presents the list of direct data movements identified by the measurer for this example (see Section 5.4).

Table 7 COSMIC-SOA measurements for direct data movements for system operations

Types	*Direct data movement description*	*DMT*
User or engineered device	User or engineered device sends a data group to an I/O device	E
	User or engineered device receives a data group from an I/O device	X
I/O device	I/O device sends a data group to an operational control interface	E
	I/O device sends a data group to an operational data interface	E
	I/O device receives a data group from an operational data interface	X
Operational Control Interface Function (OPCIF)	Operational control interface sends a data group to inter-operational service 1 (IOP-S1)	E
	Operational control interface sends a data group to inter-operational service n (IOP-S2)	E
Inter-Operational Service 1 (IOP-S1)	Inter-operational service 1(IOP-S1) sends a data group to operational event service 1(OPE-S1)	X
	Operational event service 1(OPE-S1) receives a data group from inter-operational service 1(IOP-S1)	E
Operational Event Service 1 (OPE-S1)	Operational event service 1(OPE-S1) sends a data group to inter-operational service 1(IOP-S1)	X
	Inter-operational service 1(IOP-S1) receives a data group from operational event service 1(OPE-S1)	E

Table 7 COSMIC-SOA measurements for direct data movements for system operations (continued)

Types	*Direct data movement description*	*DMT*
Inter-Operational Service 2 (IOP-S2)	Inter-operational service 2 (IOP-S2) sends a data group to operational event service 2 (OPE-S2)	X
	Operational event service 2 (OPE-S2) receives a data group from inter-operational service 2 (IOP-S2)	E
Operational Event Service 2 (OPE-S2)	Operational event service 2 (OPE-S2) sends a data group to inter-operational service 2 (IOP-S2)	X
	Inter-operational service 2 (IOP-S2) receives a data group from operational event service 2 (OPE-S2)	E
Operational Data Interface Function (OPDIF)	Operational data interface receives a data group from operational event service 1 (OPE-S1)	E
	Operational data interface receives a data group from operational event service 2 (OPE-S2)	E
	Operational data interface sends a data group to operational event service 1 (OPE-S1)	X
	Operational data interface sends a data group to operational event service n (OPE-S2)	X
Total COSMIC Functional Size		*19 CFP*

Table 8 presents the list of indirect data movements identified by the measurer for this example (see Section 5.4).

Table 8 COSMIC-SOA measurements for indirect data movements for system operations

Types	*Indirect data movement description*	*DMT*
Operational Data Interface Function (OPDIF)	Operational data interface reads a data group from persistent storage for OPE-S1	R
	Operational data interface writes a data group to persistent storage for OPE-S1	W
	Operational data interface reads a data group from persistent storage for OPE-S2	R
	Operational data interface writes a data group to persistent storage for OPE-S2	W
Total Cosmic Functional Size		*4 CFP*

6.4 The Total Functional Size (FSM)

From the measurement steps A, B and C above:

- the functional measurement size for the functional services = 16 CFP
- the functional measurement size for the intermediary services = 16 CFP
- the functional measurement size for the direct and indirect data movements for the functional services = 19 CFP + 4 CFP = 23 CFP.

Therefore, the total functional size for this example, based on Figure 4 and Sections 5.2–5.4, is equal to 55 CFP. This size is considered as a new functional size for the NFR that should be added to the size of the functional requirements defined for the system functionality allocated to the software to improve the software estimation purposes.

7 Discussion and conclusion

Operations requirements are typically described initially as NFRs at the system level, and system engineers must subsequently apportion these system requirements very carefully as either software or hardware requirements to conform to the operations requirements of the system. Within the ECSS and IEEE standards, a number of views and concepts are provided to describe various types of candidate operations requirements at the system, software and hardware levels.

This paper has introduced a standards-based reference framework for specifying and measuring software requirements for the functions needed to address the system's operations requirements. The main contribution of this paper is our proposed generic model of software FUR for system operations. This generic model can be considered as a kind of reference model for the identification of system operations requirements, and can be used for their allocation to software functions implementing such requirements as well as the paper deals with the operations quality attribute and proposes an approach to allow the identification and specification of the NFRs related to the operations and their allocation in the software system as specific operations functions, and a method, based on the COSMIC – ISO 19761 model, to estimate a measure of the functional size of operations. One of the main contributions is to provide the researchers' community (and the practitioner community) with this knowledge and measurement approach.

System requirements allocated to hardware have not been addressed in this paper. Since the structure of the general model is based on the generic model of software adopted by the COSMIC measurement standard, the necessary information for measuring their functional size is readily available, and an example has been presented of a specific instantiation of this reference model. Specifically, the generic model of operations presented in this paper is based on:

- the ECSS standards for the description of the NFR for system operations
- the COSMIC measurement model of functional requirements.

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

- a specification model for each type, or all types, of operations requirements: for example, the requirements to be allocated to software for the system operational control, modes and data
- a specification measurement model for each type, or all types, of operations requirements.

The generic model of system operations requirements proposed in this paper can provide system engineers with:

- an integrated reference view of system operations requirements that they can use to select the operations requirements necessary for a specific system to be developed (hardware-software-manual)
- a methodology to specify these operations NFR: with this reference model, beginners would not require years of training before being able to specify these at the levels of detail illustrated in the work reported in this paper
- an integrated model to be used as an input to make decisions on which of these detailed operations NFR will be allocated to 1: hardware, or 2: software, or 3: combinations of these for a specific context.

For software engineers, the proposed generic model of system operations requirements can also provide them with:

- A reference model that they can use to verify whether the system engineers have provided them with the right selection of system NFR-derived FUR, and at the necessary level of detail. This means that this standard-based reference model can be used as a quality technique for the following:
  - verification of system operations requirements coverage and descriptions
  - as a technique, at the software requirements phase, to elicit such requirements, referred to as 'both NFRs and emergent properties' in the SWEBOK Guide – ISO/IEC19759 (2004).
- To achieve this level of detailed inputs-requirements up front in the project life cycle; i.e., at the software requirements phase, rather than much later, at the software testing phase, which is the common practice.
- As a way to measure these FURs with COSMIC – ISO 19761, and take them into account in Function-Points-based software estimation models, thereby avoiding late discovery of mandatory FUR that lead to budget overruns and missed deadlines.

The measurement aspects presented in this paper have been limited to the system requirements allocated to software.

It will be interesting in future work to investigate whether this measurement approach can be extended to all such requirements at the system level (i.e., to all hardware-software-manual requirements, and not only to software requirements) as well as a number of related issues have not been tackled yet and will require further work along two research avenues:

- The proposed reference model accounts for only the definitions of the operations-NFR as defined by the standards. The resulting model may be incomplete for some application domains and further work is required to improve the model proposed, as well as the standards themselves.
- Even though the standards-based reference model is generic, the measurement results for each specific application will be distinct: each one will have its own specific instantiation of the reference model.
- Estimation models using the size of software-FUR derived from systems-FUR as their key independent variable may be improved by taking into account the size of these other software-FUR derived from system-NRF (operations and other types of NFR). Of course, empirical work is needed to collect data and investigate whether the performance of such expanded size-based estimation models is indeed improving.
- Research on estimation models using as inputs the size of software-FUR derived from both systems-FUR and system-NFR should investigate not only the initial development phase but also the system maintenance phase.

References

Abran, A. and Al-Sarayreh, K.T. (2010b) 'Standards-based model for the specification of system design and implementation constraints', *Industrial Proceedings, 17th European Systems & Software Process Improvement and Innovation, EuroSPI 2010 Conference*, Grenoble, France, Delta, Denmark, pp.4.7–4.16.

Abran, A. and Al-Sarayreh, K.T. (2010a) 'Measurement of software requirements derived from system operations requirements', *20th International Workshop on Software Measurement & International Conference on Software Measurement*, IWSM/ MetriKon/Mensura, Stuttgart, Germany, pp.101–114.

Abran, A., Al-Sarayreh, K.T. and Cuadrado-Gallego, J.J. (2010) '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, pp.153–158.

Abran, A., Al-Sarayreh, K.T. and Cuadrado-Gallego, J.J. (2013) 'A standards-based reference framework for system portability requirements', *Computer Standards and Interface*, Elsevier, http://dx.doi.org/10.1016/j.csi.2012.11.003.

Al-Khasawneh, A., Al-Sarayreh, K.T. and Meridji, K. (2013) 'A standards-based framework of software configuration derived from system requirements', *International Journal of Computer Applications in Technology (IJCAT)*, http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijcat (this issue).

Al-Sarayreh, K.T., Abran, A. and Cuadrado-Gallego, J.J. (2012a) 'A standards-based model of system maintainability requirements', *Journal of Software Maintenance and Evolution: Research and Practice*, John Wiley and Sons, 14 March, DOI: 10.1002/smr.1553.

Al-Sarayreh, K.T., Abran, A. and Cuadrado-Gallego, J.J. (2010) 'Measurement model of software requirements derived from system portability requirements', *9th International Conference on Software Engineering Research and Practice (SERP 2010)*, Las Vegas, USA, pp.553–559.

Al-Sarayreh, K.T., Al-Oqily, I. and Meridji, K. (2012b) 'A standard based reference framework for system adaptation and installation requirements', *Proceedings of the 6th International Conference on next Generation Mobile, Applications, Services and Technologies (NGMAST 2012)*, IEEE-CS Press, Paris, France, DOI. 10.1109/NGMAST.2012.19.

Andrew, J.R. (2000) 'An approach to quantitative non-functional requirements in software development', *Systems Engineering – A Key to Competitive Advantage for All Industries: 2nd European Systems Engineering Conference (EuSEC 2000)*, Munich, September 13–15, Herbert Utz Verlag.

Chung, L. and Leite, J.P. (2009) 'On non-functional requirements in software engineering', *Conceptual Modeling: Foundation and Applications, Essays in Honor of John Mylopoulos*, Springer-Verlag, Berlin, Heidelberg, pp.363–379.

Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. (2000) *Nonfunctional Requirements in Software Engineering*, Kluwer Academic Publishing, New York.

COSMIC-SOA (2010) The COSMIC method v3.0.1, *Guideline for Sizing SOA Software, v1.4*, The Common Software Measurement International Consortium.

ECSS-E-40-Part-1B (2003) *Space Engineering: Software – Part 1 Principles and Requirements*, European Cooperation for Space Standardization, the Netherlands.

ECSS-E-40-Part-2B (2005) *Space Engineering: Software – Part 2 Document Requirements Definitions*, European Cooperation for Space Standardization, The Netherlands.

ECSS-E-ST-10C (2009) *Space Engineering: System Engineering General Requirements*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-E-ST-32 C-Rev.1 (2008) *Space Engineering: Structural General Requirements*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-E-ST-33-01 (2009) *Space Engineering: Mechanisms*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-E-ST-70 (2008) *Space Engineering: Ground Systems and Operations*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-Q-80B (2003) *Space Product Assurance: Software Product Assurance*, European Cooperation for Space Standardization, The Netherlands.

ECSS-Q-ST-20 (2008) *Space Product Assurance: Quality Assurance*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-Q-ST-80 (2009) *Space Product Assurance: Software Product Assurance*, Requirements & Standards Division Noordwijk, The Netherlands.

ECSS-S-ST-00 (2008) *ECSS System: Description, Implementation and General Requirements*, Requirements & Standards Division Noordwijk, The Netherlands.

Fernández-Sanz, L. and Misra, S. (2012) 'Analysis of cultural and gender influences on teamwork performance for software requirements analysis in multinational environments', *Software*, *IET*, Vol. 6, No. 3, pp.167, 175, June 2012, DOI.10.1049/iet-sen.2011.0070.

Glinz, M. (2005) 'Rethinking the notion of non-functional requirements', *3rd World Congress for Software Quality*, Munich, Germany.

IEEE-Std-830 (1998) *IEEE Recommended Practice for Software Requirements Specifications (Revision of IEEE Std 830-1998)*, IEEE Computer Society.

ISO/IEC-14143-1 (2008) *Information Technology – Software Measurement – Functional Size Measurement Part 1: Definition of Concepts*, International Organization for Standardization: Geneva, Switzerland.

ISO/IEC19759 (2004) *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, International Organization for Standardization, Geneva.

ISO/IEC-19761 (2011) *Software Engineering – COSMIC v 3.0 – A Functional Size Measurement Method*, International Organization for Standardization, Geneva, Switzerland.

Jindan, F., Dechen, Z., Lanshun, N. and Xiaofei, X. (2012) 'A feature-oriented approach to platform-specific modelling of coarse-grained components', *Int. J. of Computer Applications in Technology*, Vol. 44, No. 1, pp.46–60, DOI: 10.1504/IJCAT.2012.048207.

Kaiya, H., Osada, A. and Kayjiri, K. (2004) 'Identifying stakeholders and their preferences about NFR by comparing use case diagrams of several existing systems', *Proceedings of the Requirements Engineering Conference, 12th IEEE International*, Washington DC, USA, pp.112–121.

Meridji, K., Al-Sarayreh, K.T. and Al-Khasawneh, A. (2013) 'A generic model for the specification of software reliability requirements and measurement of their functional size', *International Journal of information Quality (IJIQ)*, http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijiq

Moreira, A., Araujo, J. and Brito, I. (2002) 'Crosscutting quality attributes for requirements engineering', *14th International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy, pp.167–174.

Mylopoulos, J. (2006) 'Goal-oriented requirements engineering, part II', *Keynote at the 14th IEEE International Conference on Requirements Engineering (RE'06)*, IEEE Computer Society Press, Minneapolis/St. Paul, Minnesota, USA.

Paech, B., Dutoit, A., Kerkow, D. and Von Kneth, A. (2002) 'Functional requirements, non-functional requirements and architecture specification cannot be separated – a position paper', *Requirements Engineering: Foundations for Software Quality (REFSQ)*, Essen, Germany.

Park, D. and Kang, S. (2004) 'Design phase analysis of software performance using aspect-oriented programming', *5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004*, Lisbon, Portugal.

Rosa, N.S., Cunha, P. and Justo, G. (2002) 'Process NFL: a language for describing non-functional properties', *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, Hawaii, USA, Vol. 9, p.282b.

Tiako, P.F. (2011) 'Process modelling, delegation and control in global software development', *International Journal of Computer Applications in Technology (IJCAT)*, Vol. 40, No. 3, pp.160–169, DOI: 10.1504/IJCAT.2011.039137.