e-Informatica Software Engineering Journal, Volume 14, Issue 1, 2020, pages: 117–148, DOI 10.37190/e-Inf200105

System performance requirements: A standards-based model for early identification, allocation to software functions and size measurement

Khalid T. Al-Sarayreh[*], Kenza Meridji[**], Alain Abran[***], Sylvie Trudel[****]
[*] *Department of Software Engineering, Hashemite University*
[**] *Department of Software Engineering, University of Petra*
[***] *Department of Software Engineering and Information Technology , École de technologie supérieure (ETS), Université du Québec*
[****] *Département d'informatique, University of Quebec at Montreal*
Khalidt@hu.edu.jo, kmeridji@uop.edu.jo, alain.abran@etsmtl.ca, trudel.s@uqam.ca

To be or not? To be! – Wally Shakelance

Abstract

Background: In practice, the developers focus is on early identification of the functional requirements (FR) allocated to software, while the system non-functional requirements (NFRs) are left to be specified and detailed much later in the development lifecycle.
Aim: A standards-based model of system performance NFRs for early identification and measurement of FR-related performance of software functions.
Method: 1) Analysis of performance NFR in IEEE and ECSS standards and the modeling of the identified system/software performance functions using Softgoal Interdependency Graphs. 2) Application of the COSMIC-FSM method (e.g., ISO 19761) to measure the functional size of the performance requirements allocated to software functions. 3) Use of the COSMIC-SOA guideline to tailor this framework to service-oriented architecture (SOA) for performance requirements specification and measurement. 4) Illustration of the applicability of the proposed approach for specification and measurement of system performance NFR allocated to the software for an automated teller machine (ATM) in an SOA context.
Result: A standards-based framework for identifying, specifying and measuring NFR system performance of software functions.
Conclusion: Such a standards-based system performance reference framework at the function and service levels can be used early in the lifecycle by software developers to identify, specify and measure performance NFR.

Keywords: Non-functional requirements, (NFR) performance requirements, international standards, Softgoal Interdependency Graphs(SIGs), COSMIC-FSM, COSMIC-SOA

1. Introduction

Over the years, system non-functional requirements (NFRs) from a variety of stakeholders have significantly increased the urgency and effort required to deliver software systems with very high-quality levels. The large and diverse body of literature on software quality and NFR makes it challenging for practitioners to figure out detailed reference works to use as a baseline for early identification, specification and measurement of any of the large number of NFRs.

Submitted: 31 December 2019; Revised: 15 June 2020; Accepted: 22 June 2020; Available online: 7 August 2020

Developers must take into consideration both system functional user requirements (FURs) and non-functional requirements (NFRs) early in the system requirements analysis in order to then allocate them at the software/hardware FR level [1–6] (see Figure 1).

The success of a software project depends heavily on its ability to be executed with the required functionalities while under specific constraints. Software functionalities fall under the concept of functional user requirements (FURs) and refer to the set of functions or services required from the system and allocated to the software, while constraints fall under the concept of non-functional requirements (NFRs).

In practice, requirements are usually addressed at the system level [1–4] at the start of the project either as high-level system functional user requirements (system-FURs), or as high-level system-NFRs. The latter must typically be detailed, allocated, and implemented in either hardware or software, or both – see Figure 1.

Software engineers focus on software-FURs for the early development phases, while system-NFRs are typically discussed at later development phases, such as evaluation or testing phases. To distinguish between these types of requirements, the term system-FURs is used to describe the required functions in a system, while system-NFRs is used to describe how the required functions must behave in a system.

In the software requirements engineering phase, system-NFRs are analyzed and detailed, and some may be specified as Software-FURs to allow a software engineer to develop, test, and configure the final software deliverables to system users. It should be noted that a number of such system constraints, while referred to as system-NFRs by some authors, are referred to as quality aspects by other authors.

A number of researchers have investigated issues related to NFR, such as considering them as measurable inputs to effort estimation models [1], which, although based on a different point of view, can be used concurrently with FUR, including their procedures and approaches.

This paper specifically addresses system performance NFR and extends our previous research on three other types of NFR: security [2], portability [3] and maintainability [4]. A key strength of the approach in our previous work is that it is based on the consensus documented in international standards, such as the European Cooperation for Space Standardization (ECSS), the Institute of Electrical and Electronics Engineers (IEEE) and ISO on a number of such NFR, and our proposal for a standards-based reference model for specific types of NFR.

The contribution of this work is a standards-based measurement framework of system performance requirements to be used by de-

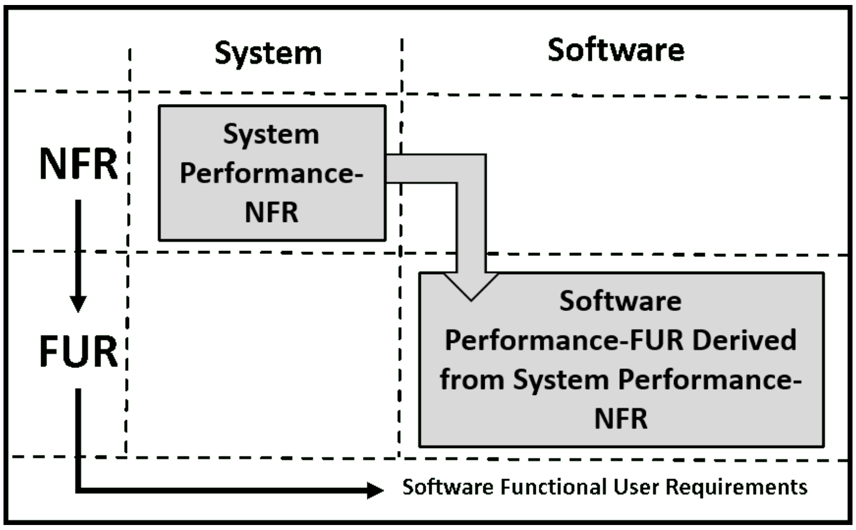

System
Software
NFR
FUR
System Performance-NFR
Software Performance-FUR Derived from System Performance-NFR
Software Functional User Requirements

Figure 1. System performance-NFR allocated to software performance-FUR

velopers in the early development stages as a generic model for the identification, specification and measurement of the system performance requirements allocated to software functions.

The proposed framework was developed in four main steps:

1. Identifying, analyzing, and categorizing into an integrated view the system performance requirement functions and services described from different perspectives into ECSS and IEEE standards. Then, modeling the identified system/software performance requirements and clarifying the relations between these requirements using SIGs.
2. Applying the COSMIC-FSM method to identify and measure the data movements derived from the allocated software performance requirements. This leads to handling the system performance requirements allocated to the measured software performance requirements as quantitative requirements.
3. Developing the proposed framework in the context of service-oriented architecture using COSMIC-SOA guidelines to support a distinct business domain.
4. Illustrating the applicability of the proposed approach for the specification and measurement of system performance NFRs allocated to the software for an automated teller machine (ATM) within an SOA context.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 discusses the system performance requirements identification and related software performance requirements in international standards. Section 4 details the proposed standards-based system performance reference framework at the function and service levels in the context of a service-oriented architecture (SOA). Section 5 presents an illustrative example using the proposed standards-based framework for identifying and specifying ATM banking system performance requirements, allocating them to the software performance functions in an SOA context and measuring them with COSMIC, an ISO-recognized measurement unit. Section 6 presents conclusions and further work.

2. Related work

2.1. Non-functional requirements in the literature and international standards

A number of proposals for identifying and specifying different types of NFR, including different methods, approaches, views and terminologies have been made [1–7].

To help software project teams make the best tradeoff decisions for conflicting NFRs, Zhang and Wang [8] proposed a tradeoff model for conflicting software non-functional requirements (CNFR) using a fuzzy ranking method to express stakeholder assessments of each NFR.

To help developers prioritize such kinds of requirements early in the project cycle, Shah et al. [9] proposed an approach for specifying the NFR conflicts from previous ontological representations of the NFRs.

Daclin et al. [10] analyzed interoperability as a single NFR as a part of a complex NFR domain, linking interoperability and its impacts on the system performance requirements into a collaborative system in a crisis management framework.

Cysneiros et al. [11] highlighted the challenges facing developers of capturing NFR simultaneously with FR at the early phases of software development. They suggested the integration of NFR with FR into conceptual models based on a goal oriented strategy aimed at reducing the cost of software development as well as increasing customer satisfaction.

Various studies have focused on NFR [12–15] within the software product line process. Tawhid and Petriu [16] for example, proposed a UML model transformation framework to determine and reuse the performance requirements for a specific product.

Siegmund et al.[17] proposed a holistic approach, named SPL Conqueror, for the optimization of the specification and measurement of NFR in the SPL domain. They also carried out an analysis of the quality attributes (i.e., NFR) in SPL as well as a verification of product satisfaction of the quality conditions and constraints.

Danylenko and Lowe [18] studied a context-aware recommender system with the objective to defer architectural decisions, thus permitting concentration on the core system functionality design. In early development phases this recommender system helps to ease the difficulties of NFR efficiency.

Kyo and Gil-Haeng [19] proposed a systematic software development process to support successful management and modeling for NFR. This process allows NFR to be systematically managed and efficiently modeled.

Industry, through its participation in international standards organizations, has also contributed by describing and categorizing NFR. For example: performance requirements are one of sixteen NFR types in ECSS [20–25], which have been categorized by IEEE [26] as one of thirteen NFR types, using different terms and views. Although in academia and industry NFR performance requirements are frequently discussed, there is a lack of a performance model that can be used in the early development stages. In the research reported in this paper, we propose a standards-based framework for early identification and measurement of system performance requirements by analyzing all the performance NFR related terms and views dispersed throughout the international standards, such as ECSS and IEEE.

2.2. System performance requirements in the literature

To develop new insights into performance, in this research, we analyzed related works in standards on performance in hardware domains where there is considerable, accumulated expertise. We looked for performance-related concepts and sub-concepts that were also relevant to software.

System performance requirements have been discussed from various viewpoints in the literature. Shang et al. analyzed [27] the VxWorks real time operating system used in the aerospace and medical fields including five significant performance indicators: task switching time, preemption time, interrupt latency time, message communication time and semaphore shuffling time.

Alwadi et al. [28] proposed a framework for the quality of service (QoS) attributes and included performance as one of the prime system NFR, allowing performance requirements to be decomposed and allocated to a set of the system's functionalities.

Zhiwei et al. [29] proposed an approach for improving the concurrent system performance on the dynamic weighted k-out-of-n system (DWKNS). Subsequent to the state possibility and request of system components over time, this approach was combined with the Markov process with the universal generating function (UGF) method and the state probability for the performance of the system components.

Al-Sarayreh [30] considered system performance requirements to be more comprehensive than typical hardware-centric systems and proposed that dynamic system performance requirements be included with maintainability, upgradability, interface interoperability, reliability, safety and security (MUIRSS). MUIRSS should be analyzed and visibly connected to ensure that they are included with development.

The system dynamic performance of Kai and Huamin [31] for control systems indicates a relational variance of the controller design method. Their proposed method is used for the first order plus delay time system.

Krishna and Abraham [32] discussed the importance of the analysis of performance and memory NFR in real time embedded systems. Based on agile using incremental development, their development approach helps system engineers track the system performance requirements and related parameters throughout the development cycle. Their results are taken as a reference for a systematic analysis approach for memory and system performance NFR parameters using the most suitable mathematical methods.

Vila et al. [33] presented an approach for estimating the radio resource requirements for RAN slice admission control in order to describe the interference conditions of resource estimation method influences on system performance requirements extracted from data analytics collected from management plans.

Ruberg et al. [34] proposed a data processing and cleaning method for a performance and energy consumption estimation approach to manage system performance requirements. Their approach links software component feature measurements (SCFMs) and software performance quality indicators (SPQIs) to diagnose the software and functional requirements.

2.3. COSMIC functional size measurement method

There are currently five functional size measurement (FSM) methods adopted by ISO: the COSMIC Function Points method is the only second generation of such FSM methods, and its design has corrected a number of the defects identified in the other four FSM methods of the first generation.

Measuring software functional size is an important factor for managing and estimating the project budget early in the software development lifecycle. The COSMIC functional size measurement (FSM) method conforms to the measurement requirements proposed in ISO 14143-1 [35] and has been adopted as ISO 19761 [36]. This subsection presents the COSMIC generic model for software requirements and how such a model can be used to measure software functionalities with an ISO-recognized measurement unit.

In the COSMIC-FSM method, the functional user requirements are decomposed into one or more functional processes, each of which may be comprised of sub-processes and include a number of data movements.

Figure 2 illustrates the COSMIC generic model of software FR. The front-end direction of the model shows that users access the software through input/output devices (such as mouse and microphone) or engineered devices (such as sensors). The back-end direction shows that the software is accessed by storage hardware (such as RAM memory). Figure 2 also illustrates the following four types of data movement: ENTRIES (E): exchanges data groups from users or engineered devices to software (left-hand side in Figure 2). EXITS (X): exchanges data groups from software to users or engineered devices (left-hand side in Figure 2). READS (R): exchanges data groups from hardware storage to software (right-hand side in Figure 2). WRITES (W): exchanges data groups from software to hardware storage (right-hand side in Figure 2).

The core principle of the COSMIC-FSM method is to measure the size of software FR by identifying the recognized data movements (E, X, R and W). Once the data movements are identified, each type of data movement is assigned the value of one COSMIC Function Point (e.g., 1 CFP). The functional size of the software to be measured is obtained by summing the sizes of all the corresponding data movements. Since the

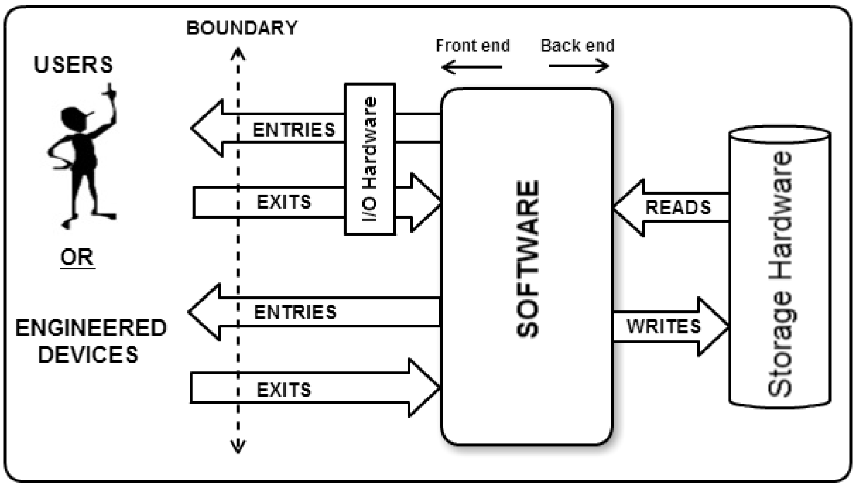


Figure 2. COSMIC generic model for software FR

COSMIC-FSM method aims to measure the size of the software, only the functional user requirements allocated to the software are considered in the measurement procedure.

Moreover, the COSMIC-FSM method is applicable to all the software development phases, from the analysis to implementation phases. Note that the COSMIC generic model in Figure 4 is not specific to any type of software nor to any particular method for describing functional user requirements. In the framework proposed in this paper, the COSMIC-FSM method is applied to measure the size of the software performance functional requirements with an ISO-recognized size unit.

2.4. Service-oriented architecture (SOA) and its COSMIC view

The service-oriented architecture (SOA) approach provides significant benefits to organizations, such as reducing software development and maintenance costs and increasing software quality by reusing services [37]. Various definitions have been introduced to define SOA, but none have been universally adopted. For instance, SOA has been defined as: 1) A process that involves the definition of the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements [36, 37]; 2) A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations [36]; 3) Utilization of loosely coupled software services to support business processes requirements and user requirements [37].

The COSMIC-SOA guidelines document illustrates how to measure the size of software services in an SOA context [37]. The term services in the COSMIC guideline refers to a suite of related functions of software FR and also to the separation of functions into distinct units, where these services are connected with each other by exchanging data, shared format or by coordinating activities between two or more services [37].

COSMIC-SOA guidelines offer three types of data movements – exchange services, intermediary services and data exchanges, which are described in more detail in the following sections.

2.4.1. COSMIC-SOA exchange messages

COSMIC-SOA exchanges messages (Figure 3) when an application needs information from a dif-

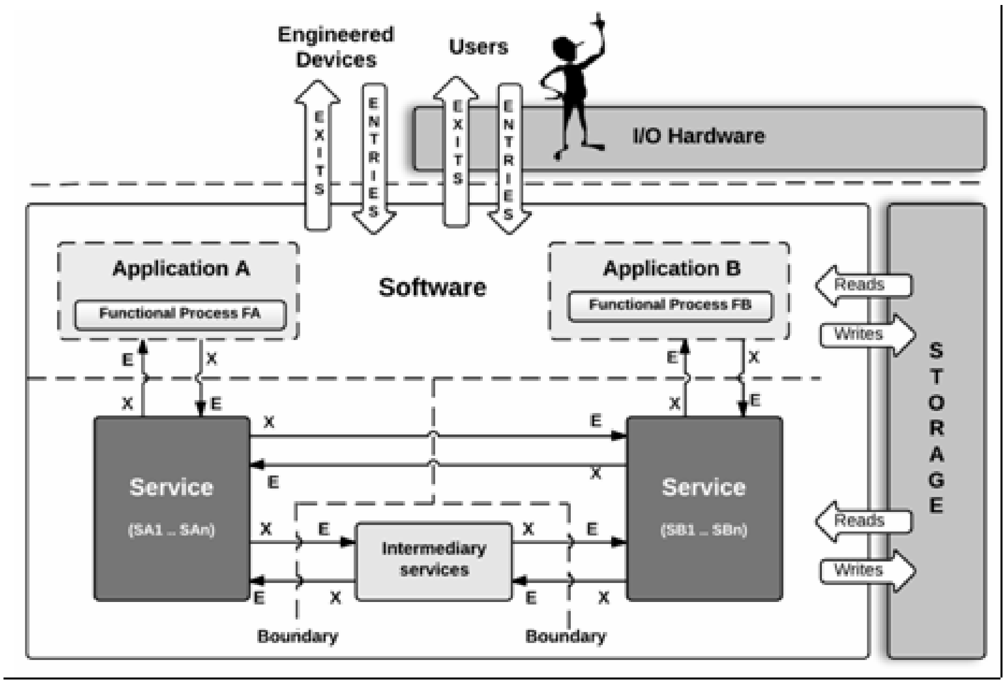


Figure 3. COSMIC-SOA guidelines for modeling data movements

ferent application. For instance, if application A needs to exchange data with application B, the services of application A will be invoked by the functional process of A to communicate with the services of application B to obtain the needed information. These calls between the functional processes of A and its services or between A services and B services are known as messages, where each message may involve one or more data movements [37].

2.4.2. COSMIC-SOA intermediary services

When services (Figure 3) of any application require data from another application in the overall SOA framework, the intermediary service will be used. For instance, if services of application A need data from services of application B, the services of application A will invoke the intermediary services to obtain the required data from the services of application B.

2.4.3. COSMIC-SOA data exchanges

For components in the same layer (e.g., in the application service layer) (Figure 3), two types of data movements can be used: direct and indirect message exchanges. For instance, in direct exchange, if the service of application A requires to exchange a message with a service of application B, it will use an Exit and/or an Entry for exchanging messages with the service of application B. While the indirect exchange occurs using storage, for instance, the service of application A writes the data in storage which is read later by service B [37].

2.5. Softgoal Interdependency Graphs

Softgoal Interdependency Graphs (SIGs) [38] have been proposed for analyzing and demonstrating NFR as softgoals. Each softgoal can be represented as decomposed into one or more specific goals using interdependency relations between the analyzed goals until arriving at solutions that satisfy the assigned NFR.

SIGs [39] illustrate three different types of goals at the high level: 1) Softgoals that satisfy the NFR with the software, 2) Claim softgoals which enhance the rationale between related softgoals, and 3) Operationalization of system softgoals (including a set of processes, data representations and system behavior).

These SIGs [39] at the low level (i.e., subgoals) provide both positive and negative contributions to the assigned softgoals at the high level.

Softgoals and subgoals can interact with each other using the following relations [39]: 1) AND means that each softgoal is decomposed into more than one related goal and is satisfied if all the related goals are satisfied. 2) OR means that each softgoal is decomposed into one or more related goals and is satisfied if at least one related goal is satisfied. 3) EQUAL means that each softgoal is decomposed into one related goal and is satisfied if the linked goal is satisfied.

The SIGs [38, 39] approach uses the terms goals and subgoals to represent the conditions or criteria that the system should meet (e.g., non-functional requirements or quality attributes) instead of more commonly used terms in software engineering, such as functions and software specifications. In addition, the SIGs approach does not distinguish between the system view and the software view.

This research provides a mapping between some of the SIGs terms to the standards-based terms used in this paper, as presented in Figure 1. Therefore, the expression function to be specified is used instead of a functional goal while both are encoded as is in the SIG approach.

3. Performance requirements identification

This section introduces and discusses performance terms and views for identifying system performance NFR and related software performance FR, which may then be used for specifying and measuring the system performance requirements. Numerous terms and views are found throughout the ECSS and IEEE international standards as well as previous works in academia. These have addressed software performance FR derived from system performance FR and NFR (see Figure 1).

Figure 1 also illustrates system performance requirements expressed as either system performance NFR or system performance FR.

3.1. ECSS concepts for performance requirements

ECSS standards [20–25] mention the importance of establishing performance requirements in detail at both system and software levels during the development phase so as to evaluate the consistency and cohesion of the control system within the required standards. This includes: 1) The objective(s) for each designed control system, which are normally created by the requirements engineering process; 2) The formal mathematical requirements, which are created by the requirements analysis.

Enhancing and regularly improving software applications requires system monitoring and evaluation of system performance. The performance monitor [23] provides information related to the use of processor instruction execution and storage control. For example, to provide information related to the period of time passed between events in a processing system.

The performance monitor can be used to debug the software application and analyze system faults and errors by defining a machine's state at a specific point in time. The information from the performance monitor helps system engineers to evaluate and improve the performance of a given system, or by developing enhancements of performance requirements in new system design.

ECSS standards [20–25] define the following concepts and views for system performance requirements allocated to software: 1) Frequency domain requirements such as throughput time, which includes: Workload and Bandwidth; 2) Response to reference signals for command profiles, which includes: Response time, Settling time, and Tracking errors; 3) Accuracy and stability errors in the presence of disturbances: Performance errors (absolute and stability errors) for evaluating the accuracy and Knowledge errors (absolute and relative errors) for evaluating accuracy; 4) Processing speed includes: System scalability, and System concurrency; 5) Resource consumptions include: Processor instruction execution, Main memory time, and Storage device time.

3.2. IEEE concepts for performance requirements

IEEE standard 830-1998 [26] describes the following terms and concepts for system performance requirements allocated to software as dynamic and static numerical requirements: 1) Dynamic numerical requirements, such as workload; 2) Static numerical requirements, such as capacity and concurrency. These two types of system performance requirements should be quantified with a measurable procedural method.

3.3. Describing system performance and related software functions

This section presents a brief description of system performance requirements and their allocated software performance requirements.

3.3.1. Performance dynamic requirements

Performance dynamic numerical requirements may involve the data amount, transaction number and tasks to be processed within a specific period of time for both normal and peak workload conditions [26]. The unified standards-based view includes two types of system requirements for dynamic requirements: the response to reference signals and throughput time.

Response to reference signals (RRS): Response to reference signals refers to the specific values that change to a new value in a relatively short period of time, including response time or settling time values. Enhancing the response to reference signals is reflected positively on the system performance level. The unified standards-based view includes three types of functions for response to reference signals [25]: response time function, settling time function and tracking error function.

– Response time function (RTF) The response time is widely defined as the period of time that the system takes to respond to the user after receiving the user task. This relation

between response time and performance is an inverse relation, since a decrease in response time leads to an increase in performance level.
- Settling time function (STF) The settling time refers to the time required for the system to recover from an overload and to reach steady state. STF has also been called recovery time or reaction time [40]. It is important to reach steady state in as little time as possible.
- Tracking error function (TEF) Tracking errors includes tracking performance error. The knowledge error (KE) or errors resulting from the central processing unit (CPU) and the main memory are also important to minimize system errors. System performance, therefore, can be enhanced by increasing system accuracy and speed.

Throughput time (TT): Throughput time is the number of event responses carried out by the system in a specific period of time [41]. Thus, maximizing the throughput time leads to increasing performance of the system. The unified standards-based view includes two types of functions for throughput time:
- Bandwidth function (BF) The bandwidth function refers to the maximum amount of data that can be carried over a network or data-transmission medium in a unit of time. The throughput time is limited by the bandwidth function. Large bandwidth leads to more event responses over time [42].
- Workload function (WF) The workload function measures the number of transactions performed by the system within certain periods of time. The performance level is good when the system workload is significantly lower than its capacity [43]. Otherwise system performance will be slow.

3.3.2. Performance static requirements

Performance static numerical requirements are sometimes specified under a separate section.as capacity. They may also involve information types, the amount of time handled, the number of simultaneous users and terminals supported [26]. Based on IEEE standards, the unified standards-based view includes three types of system requirements for static numerical requirements: resource consumption, evaluation of processing speed and evaluation of accuracy.

Resource consumption (RC): The efficiency of system resources (such as CPU, main memory and system storage) significantly affects the system performance. Heavy resource consumption can lead to the system's inability to effectively deal with its processes [44, 45], therefore, slowing down or crashing causing poor system performance. Proper utilization of resources leads to high system performance.

The unified standards-based view includes three types of functions for resource consumption: main memory time function, storage device time function and processor instruction execution function.
- Main memory time function (MMTF) The main memory is also known as the system internal memory or primary memory; it is used to store the data that is in use. When the CPU requires access to specific data from the storage device, the main memory will access the storage device and retrieve the required data to be processed by the CPU [46]. The time spent to access data in the main memory needs to be as small as possible in order to optimize system performance.
- Storage device time function (SDTF) Storage devices have a huge capacity to hold data in a permanent way. Fast storage devices are preferred to slower devices. Storage speed is impacted by two factors: Access time: the average time to locate data on the storage medium, and Data transfer rate: the amount of data transferred to or from the device per second [47].
- Processor instruction execution function (PIEF) Computer instructions are a set of commands executed by the processor to perform specific functions. Increasing the speed of executing such instructions can significantly contribute to improving the system performance level.

Evaluation of accuracy (EA): The developed system should achieve a high level of accuracy (i.e., precision). The definition of accuracy varies from one system to another. For instance: In satellite systems, accuracy refers to the positioning accuracy provided by the system [48]. In radio systems, accuracy refers to how closely the actual output frequency matches the set frequency [48].

System accuracy may be determined through measuring system error. Based on ECSS standards, the unified standards-based view includes two error types: performance error and knowledge error.

- Performance error (PE) is defined as the functions that quantify the difference between the system's desired state and the system's actual state [27, 28]. In the unified standards-based view, two common PE indices are used for measuring performance error:
  1. Absolute performance error function (APEF) The absolute performance error is defined as the instantaneous value of the performance error at any given time [25]. Applying a specific mathematical operator on the performance error function determines the APE. In addition, each system has a maximum APE value, which the calculated APE should not exceed.
  2. Performance stability error function (PSEF) The system stability is defined as the ability of the system to maintain a particular situation for a given time. The stability error is the peak-to-peak variation of the system attitude during the time period [25]. The PSEF is known as the change of error over a given time [25]. In addition, applying a specific mathematical operator to the performance error function and to the APE determines such a function. Just like the APE, each system has a maximum PSE value and the calculated PSE should not exceed the maximum APE.

  It is possible to use other performance error indices if the system so requires.
- Knowledge error (KE) is defined as the functions that quantify the difference between the system's estimated (or known) state and its actual state [27, 28]. In the unified standards-based view, two common knowledge error indices are used for measuring knowledge error:
  1. Absolute knowledge error function (AKEF) The absolute knowledge error (AKEF) is defined as the instantaneous value of the knowledge error at any given time [29]. Applying a specific operator to the KE function determines the AKE. Just like the performance error indices, each system has a maximum AKE value and the calculated AKE should not exceed the maximum APE.
  2. Relative knowledge error function (RKEF) The relative knowledge error (RKEF) refers to the difference between the instantaneous knowledge error at a specific time and its mean value over a time interval containing that time [29]. It is possible to use two other KE indices types should the system require it.

Evaluation of processing speed (EPS): The processing speed refers to how quickly the processor handles instructions. The processing speed for a CPU is measured by the CPU clock rate. A CPU with a high clock rate leads to high speed instruction processing. The unified standards-based view includes two types of functions for EPS: system scalability function and concurrency function.

1. System scalability function (SSF) System scalability function (SSF) is the system's ability to process increased workload while maintaining the required system performance level [48]. To make the system scalable, additional hardware is added, such as CPU or memory, without making any changes to the system architecture.
2. Concurrency function (CF) The concurrency function refers to executing several instructions simultaneously, which improves the use of system resources while also reducing the system response time [49, 50].

4. Measurement framework for performance requirements

Figure 4 shows the four main phases used to determine the proposed measurement framework for system performance requirements:

Phase 1 (Logical view): Identify and analyze the functions to be specified for system performance requirements. In this phase, the logical views are defined based on the functional user requirements view.

Phase 2 (Process view): Design and integration of the identified system performance requirements. In this phase, the process view is developed which includes default, rationale and component approaches. For more details, see Figure 4.

Phase 3 (Development view): Design a system performance requirements model using SIGs at the functional level. In this phase, the system performance model is designed and built by integrating the logical and process views [51].

Phase 4 (Deployment view): Design a measurement context at functional and service levels with COSMIC-SOA to measure the functional size of the software performance requirements. In this phase, the design measurement strategy is used to develop the proposed generic model of the system performance requirements allocated to software based on functional user requirements (FUR) views. Next, an architectural measurement context for the service levels is designed by applying the COSMIC-SOA guideline to develop a framework in an SOA context.

"For preliminary design of the performance requirements model, the SIGs tool is used. For the performance measurement model, we used another visualization tool called LibreOffice Draw Tool. This tool extends the preliminary performance model in SIGs by adding the generic COSMIC measurement procedure and adopting the detailed COSMIC-SOA to the proposed performance model."

4.1. Integration of system performance functions to be allocated to software (Phases 1 and 2)

The terms and views found in ECSS and IEEE to describe the performance NFR in Section 3.3 are combined and integrated using both their logical and process views. This leads to a dynamic view (Figure 5) and a static view (Figure 6) of the system performance functions and related

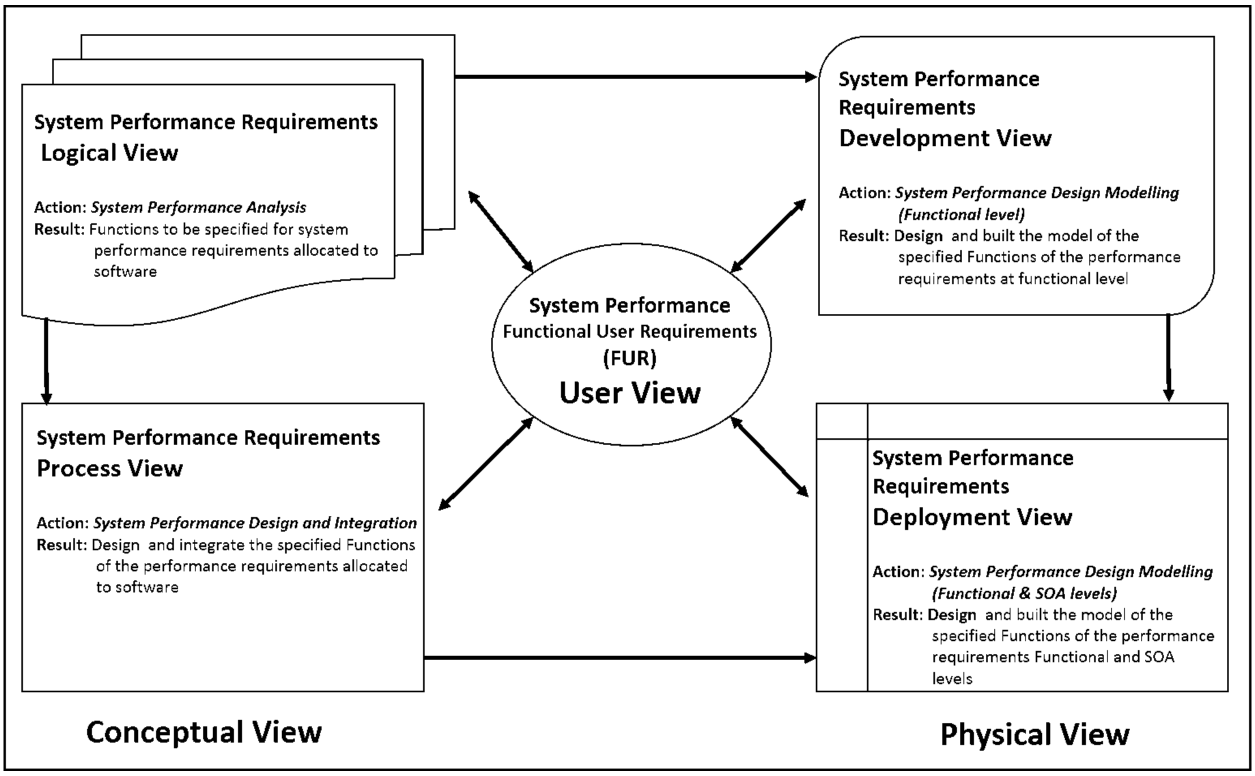

System Performance Requirements
Logical View
Action: System Performance Analysis
Result: Functions to be specified for system performance requirements allocated to software
System Performance Requirements
Development View
Action: System Performance Design Modelling (Functional level)
Result: Design and built the model of the specified Functions of the performance requirements at functional level
System Performance Functional User Requirements (FUR)
User View
System Performance Requirements
Process View
Action: System Performance Design and Integration
Result: Design and integrate the specified Functions of the performance requirements allocated to software
System Performance Requirements
Deployment View
Action: System Performance Design Modelling (Functional & SOA levels)
Result: Design and built the model of the specified Functions of the performance requirements Functional and SOA levels
Conceptual View
Physical View

Figure 4. System performance requirements from four different views

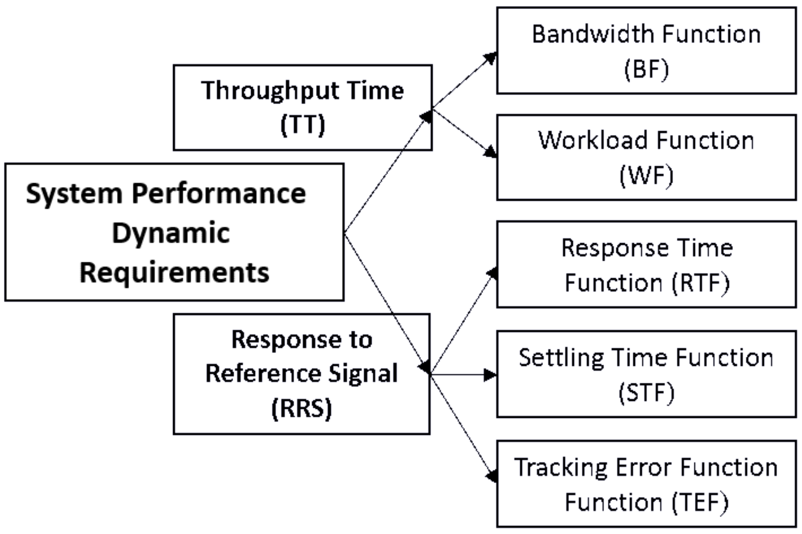

System Performance Dynamic Requirements
Throughput Time (TT)
Response to Reference Signal (RRS)
Bandwidth Function (BF)
Workload Function (WF)
Response Time Function (RTF)
Settling Time Function (STF)
Tracking Error Function Function (TEF)

Figure 5. Integrated model of ECSS and IEEE system performance dynamic requirements and related functions

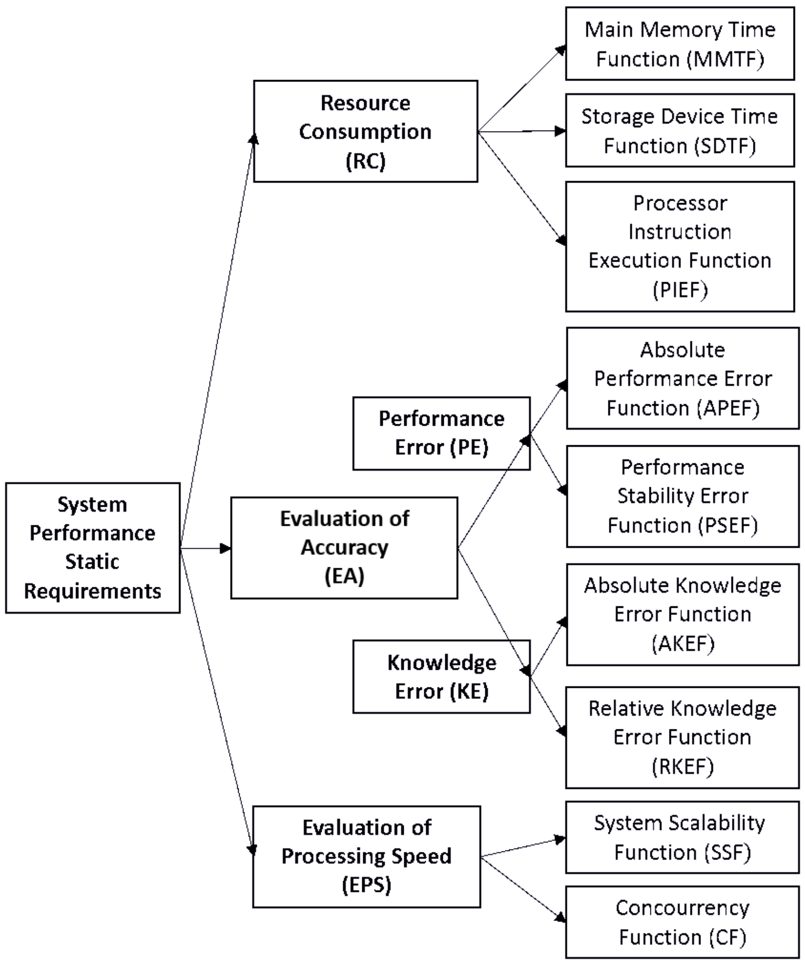

System Performance Static Requirements
Resource Consumption (RC)
Evaluation of Accuracy (EA)
Performance Error (PE)
Knowledge Error (KE)
Evaluation of Processing Speed (EPS)
Main Memory Time Function (MMTF)
Storage Device Time Function (SDTF)
Processor Instruction Execution Function (PIEF)
Absolute Performance Error Function (APEF)
Performance Stability Error Function (PSEF)
Absolute Knowledge Error Function (AKEF)
Relative Knowledge Error Function (RKEF)
System Scalability Function (SSF)
Concourrency Function (CF)

Figure 6. Integrated model of ECSS and IEEE system performance static requirements and functions

software functions, which can then be used to specify and measure them.

4.2. Design of system performance requirements at the functional level (Phase 3)

The proposed framework for system performance requirements is established at two main levels: the functions level and the services level. This section illustrates and describes in detail the framework at the functional level. In this section, the software interdependency goals (SIGs) and ISO 19761 are used to design the framework of the system performance NFR allocated to software at the functional level, divided into four sub-models, see Figure 7.

4.2.1. System performance dynamic requirements (SPDR)

The functions for the system performance dynamic requirements must address:

The throughput time (TT): The throughput time (TT), which involves two functions: the bandwidth function and the workload function. Figure 7 illustrates the interdependency relationships between these functions:

1. The bandwidth (BF) and the workload (WF) functions may exchange data in a direct way with each other, and/or
2. may exchange data in an indirect way through the persistent storage, and as well
3. may require data from any function in the overall performance framework through intermediary services.

The response to reference signals (RRS): The response to reference signals (RRS) involves three functions: the response time function (RTF), the settling time function (STF) and the tracking error function (TEF). Figure 7 shows the interdependency relations between these functions:

1. The response time, settling time and tracking error functions may exchange data in a direct way with each other;
2. may exchange data in an indirect way through the persistent storage;
3. may require data from any function in the overall performance framework through intermediary services.

4.2.2. System performance static requirements (SPDR)

The system performance static requirements include four function types – see Figure 7.

Resource consumption (RC): Resource consumption (RC) which involves three functions: the main memory time function (MMTF), the storage device time function (SDTF) and the processor instruction execution function (PIEF). Figure 7 illustrates the interdependency relationships between these functions:

1. They may exchange data in a direct way with each other, and/or
2. may exchange data in an indirect way through the persistent storage, and
3. may require data from any function in the overall performance framework through intermediary services.

The evaluation of accuracy (EA): The evaluation of accuracy (EA) which is composed of performance error (PE) and knowledge error (KE). Performance error involves two specified functions: the absolute performance error function (APEF) and the performance stability error function (PSEF). The knowledge error also involves two specified functions: the absolute knowledge error function (AKEF) and the relative knowledge error function (RKEF). Figure 7 illustrates the interdependency relationships between these functions:

1. They may exchange data in a direct way with each other;
2. may exchange data in an indirect way through the persistent storage;
3. may require data from any function in the overall performance framework through intermediary services.

The evaluation of accuracy (EA): The EPS which is composed of two functions: the system scalability function (SSF) and the concurrency function (CF). Figure 7 illustrates the interdependency relationships between these functions:

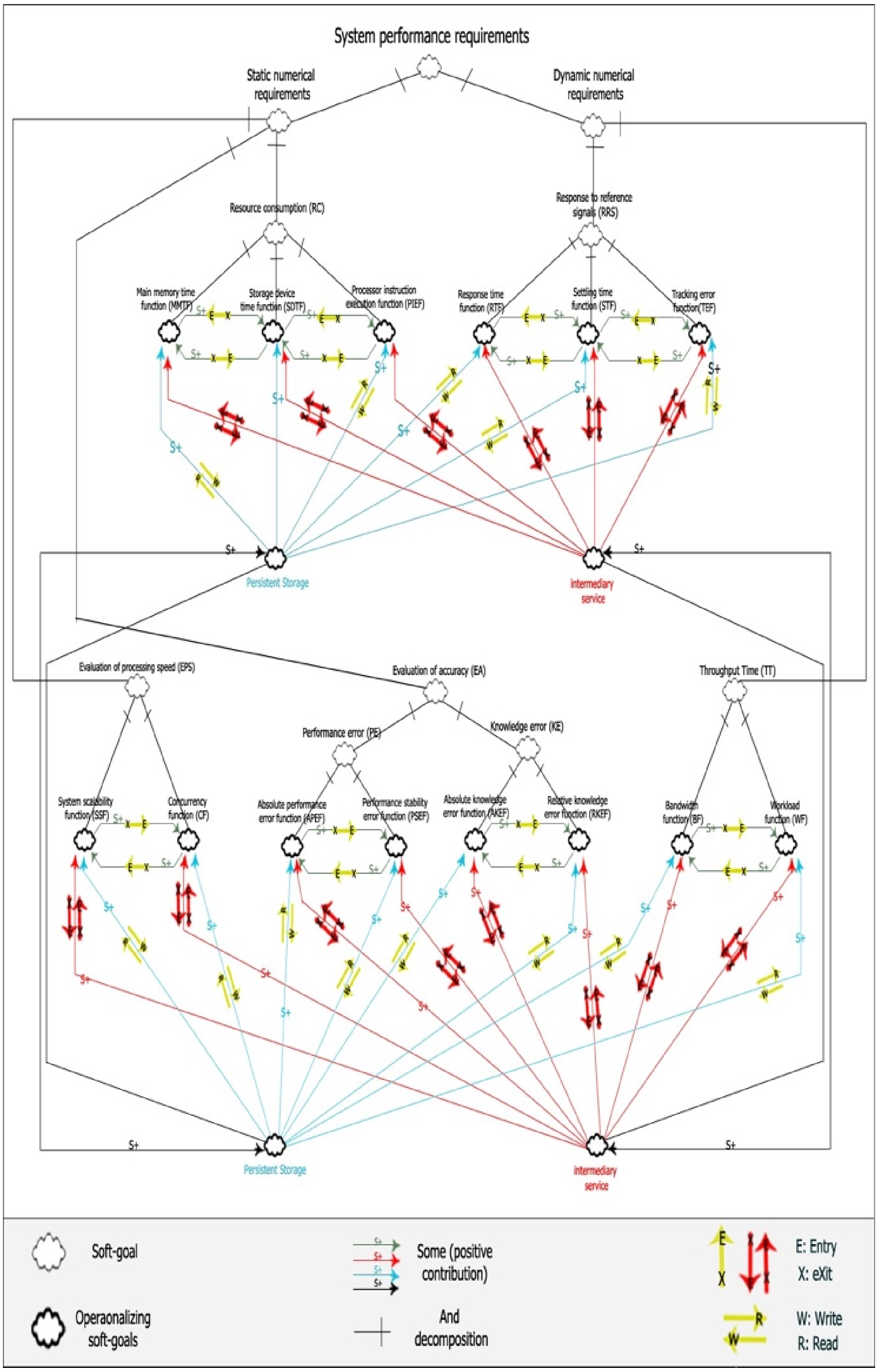

System performance requirements
Static numerical requirements
Dynamic numerical requirements
Resource consumption (RC)
Response to reference signals (RRS)
Main memory time function (MMTF)
Storage device time function (SDTF)
Processor instruction execution function (PIEF)
Response time function (RTF)
Settling time function (STF)
Tracking error function(TEF)
Persistent Storage
intermediary service
Evaluation of processing speed (EPS)
Evaluation of accuracy (EA)
Throughput Time (TT)
Performance error (PE)
Knowledge error (KE)
System scalability function (SSF)
Concurrency function (CF)
Absolute performance error function (APEF)
Performance stability error function (PSEF)
Absolute knowledge error function (AKEF)
Relative knowledge error function (RKEF)
Bandwidth function (BF)
Workload function (WF)
S+
E
X
R
W
Soft-goal
Operaonalizing soft-goals
Some (positive contribution)
And decomposition
E: Entry
X: eXit
W: Write
R: Read

Figure 7. Full view of the system performance NFR at the functional level

1. They may exchange data in a direct way with each other;
2. may exchange data in an indirect way through the persistent storage;
3. may require data from any function in the overall performance framework through intermediary services.

4.3. Design of the measurement framework for system performance NFR (Phase 4)

This section introduces the framework in an SOA context using the COSMIC-SOA guidelines. For clarity, the proposed framework is divided into the seven sub-models illustrated in Figures 8 to 13. Figure 14 shows the full view in the SOA.

Figure 8 shows that all the derived functions from resource consumption have their own services. The interdependency relations between these functions and their services are:

– The main memory time function, the storage device time function and the processor instruction execution function may require data from their services using EXIT and ENTRY data movements.
– The main memory time service may exchange data at a service layer either in a direct way with the storage device time service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
– The storage device time service may exchange data at a service layer either in a direct way with the main memory time service and the processor instruction execution service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
– The processor instruction execution service may exchange data at a service layer either in a direct way with the storage device time service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
– The main memory time service, the storage device time service and the processor instruction execution service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

Figure 9 shows that all the derived functions from the evaluation of processing speed have their own services. The interdependency relations between these functions and their services are:

– The system scalability function and the concurrency function may require data from their services using EXIT and ENTRY data movements.
– The system scalability service may exchange data at a service layer either in a direct way with the concurrency service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
– The concurrency service may exchange data at a service layer either in a direct way with the system scalability service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
– The system scalability service and the concurrency service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

As mentioned in Section 4.1, the evaluation of accuracy (EA) is comprised of two error types: performance error (PE) and knowledge error (KE). Figure 10 shows that all the derived functions from performance error have their own services. The interdependency relations between these functions and their services are:

– The absolute performance error function and the performance stability error function may require data from their services using EXIT and ENTRY data movements.

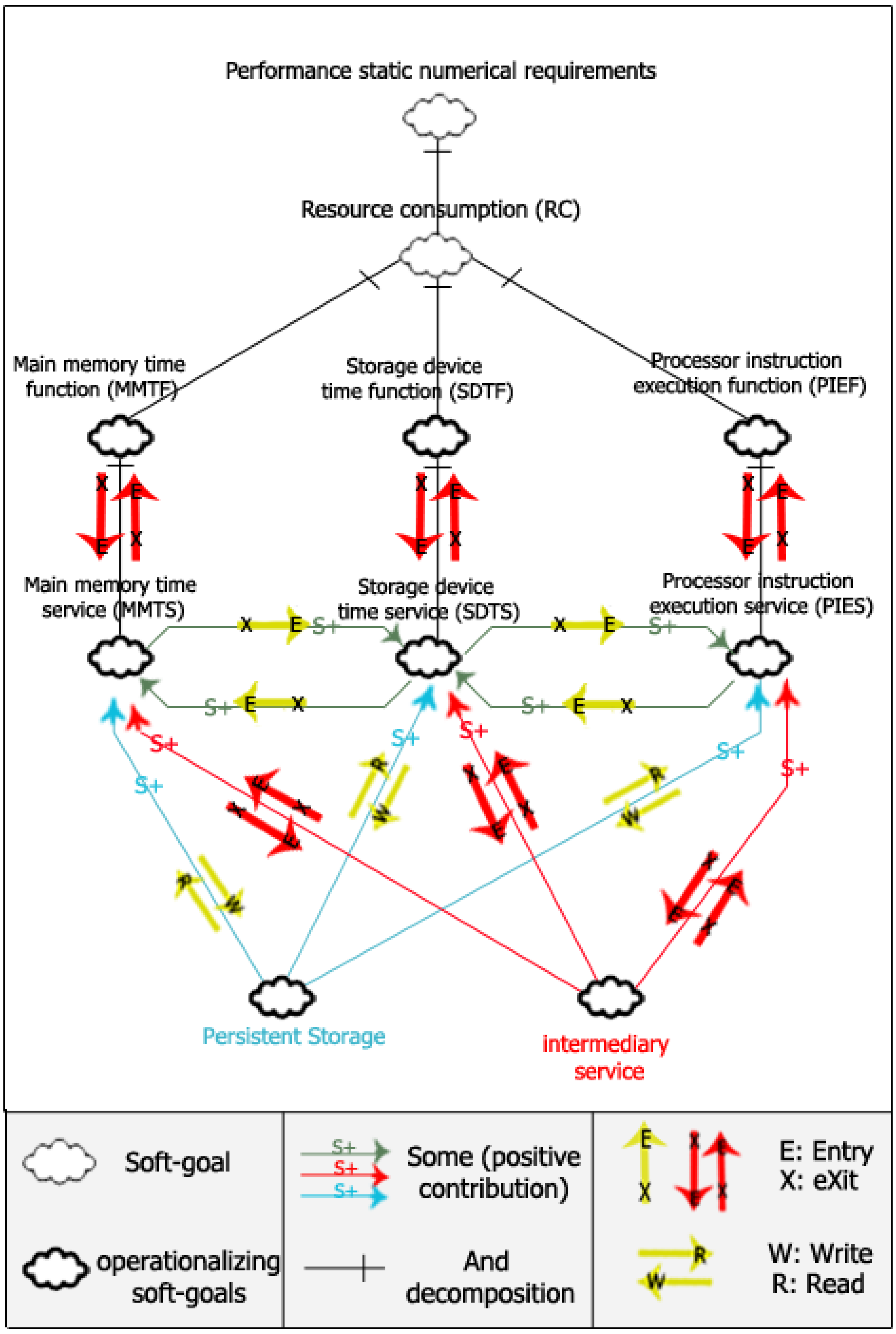

Performance static numerical requirements
Resource consumption (RC)
Main memory time function (MMTF)
Storage device time function (SDTF)
Processor instruction execution function (PIEF)
Main memory time service (MMTS)
Storage device time service (SDTS)
Processor instruction execution service (PIES)
Persistent Storage
intermediary service
Soft-goal
operationalizing soft-goals
Some (positive contribution)
And decomposition
E: Entry
X: eXit
W: Write
R: Read

Figure 8. Resource consumption sub-model in an SOA context

– The absolute performance error service may exchange data at a service layer either in a direct way with the performance stability error service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

– The performance stability error service may exchange data at a service layer either in a direct way with the absolute performance error service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

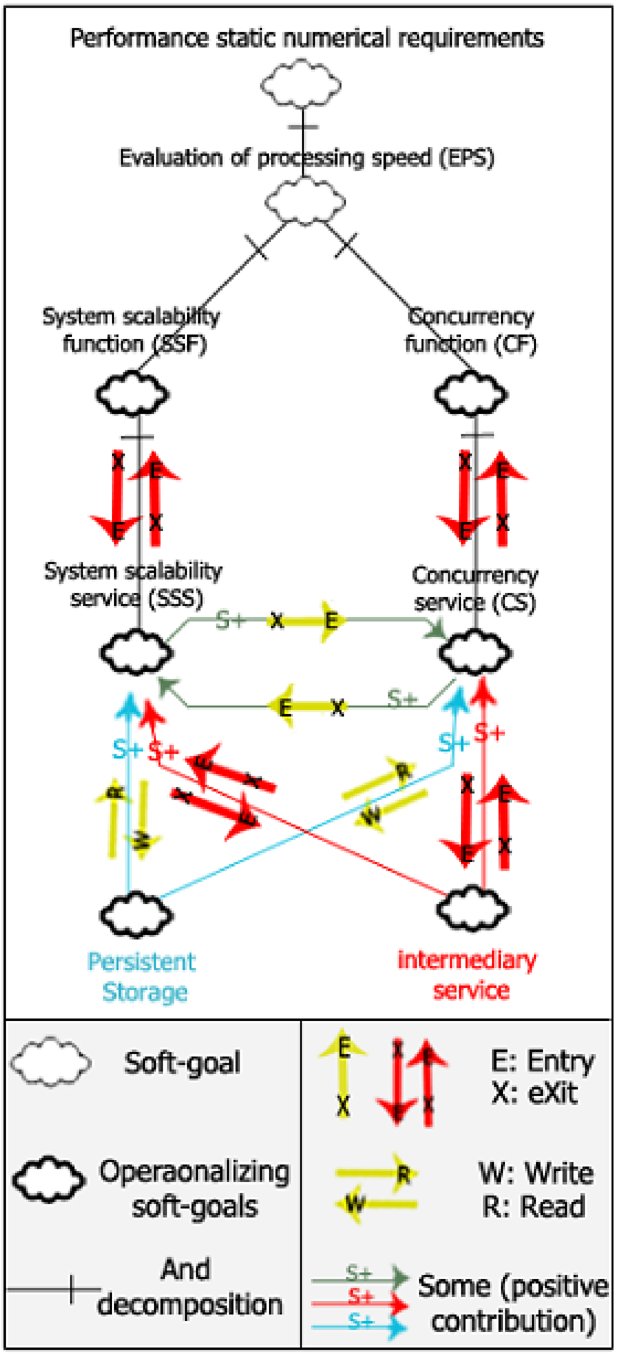

Performance static numerical requirements
Evaluation of processing speed (EPS)
System scalability function (SSF)
Concurrency function (CF)
System scalability service (SSS)
Concurrency service (CS)
Persistent Storage
intermediary service
Soft-goal
Operaonalizing soft-goals
And decomposition
E: Entry
X: eXit
W: Write
R: Read
Some (positive contribution)

Figure 9. Evaluation of processing speed sub-model in an SOA context

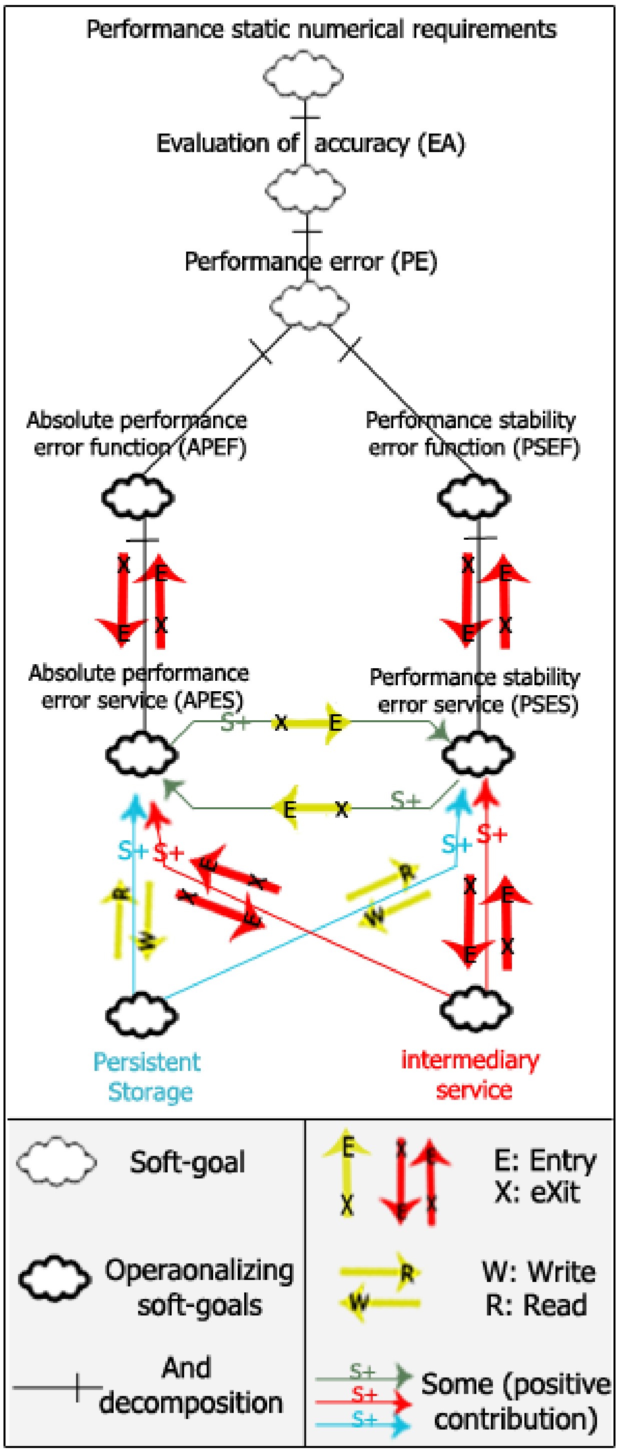

Performance static numerical requirements
Evaluation of accuracy (EA)
Performance error (PE)
Absolute performance error function (APEF)
Performance stability error function (PSEF)
Absolute performance error service (APES)
Performance stability error service (PSES)
Persistent Storage
intermediary service
Soft-goal
Operaonalizing soft-goals
And decomposition
E: Entry
X: eXit
W: Write
R: Read
Some (positive contribution)

Figure 10. Performance error sub-model in an SOA context

– The absolute performance error service and the performance stability error service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

Figure 11 shows that all the derived functions from the knowledge error (KE)have their own services. The interdependency relations between these functions and their services are:

– The absolute knowledge error function and the relative knowledge error function may require data from their services using EXIT and ENTRY data movements.

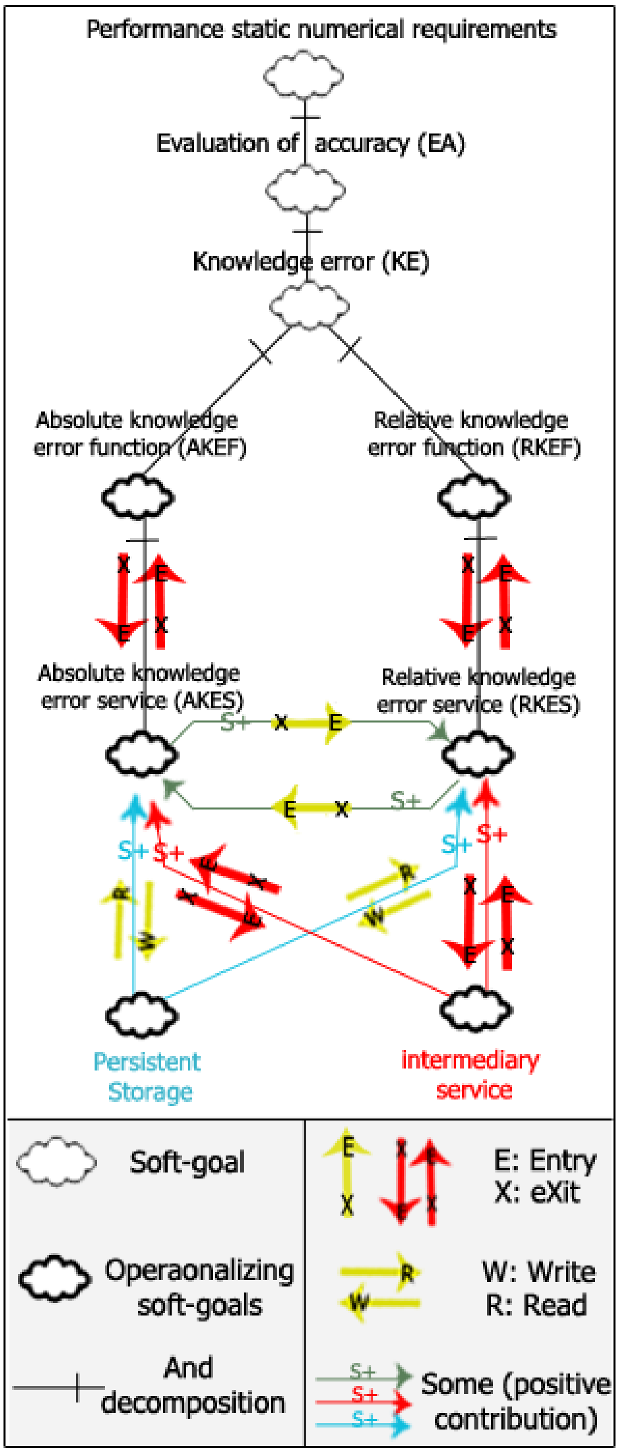

Performance static numerical requirements
Evaluation of accuracy (EA)
Knowledge error (KE)
Absolute knowledge error function (AKEF)
Relative knowledge error function (RKEF)
Absolute knowledge error service (AKES)
Relative knowledge error service (RKES)
Persistent Storage
intermediary service
Soft-goal
Operaonalizing soft-goals
And decomposition
E: Entry
X: eXit
W: Write
R: Read
Some (positive contribution)

Figure 11. Knowledge error sub-model in an SOA context

– The absolute knowledge error service may exchange data at a service layer either in a direct way with the relative knowledge error service using COSMIC EXIT and ENTRY data movements or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

– The relative knowledge error service may exchange data at a service layer either in a direct way with the absolute knowledge error service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

– The absolute knowledge error service and the relative knowledge error service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

Figure 12 shows that all the derived functions from the response to reference signals have their own services. The interdependency relations between these functions and their services are:

– The response time function, the settling time function and the tracking error function may require data from their services using EXIT and ENTRY data movements.

– The response time service may exchange data at a service layer either in a direct way with the settling time service using COSMIC EXIT and ENTRY data movements or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

– The settling time service may exchange data at a service layer either in a direct way with the response time service and the tracking error service using COSMIC EXIT and ENTRY data movements or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

– The tracking error service may exchange data at a service layer either in a direct way with the settling time service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

- The response time service, the settling time service and the tracking error service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

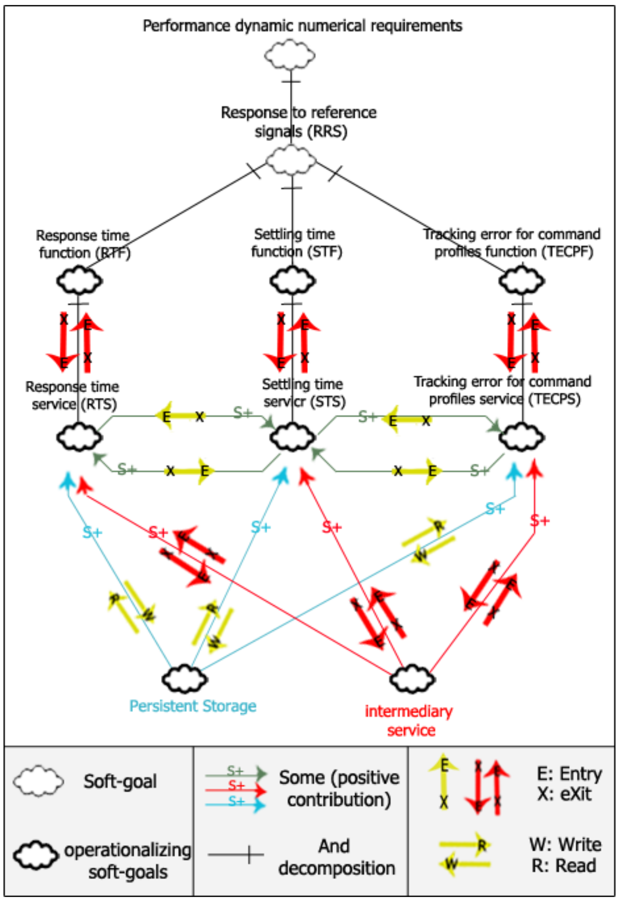

Performance dynamic numerical requirements
Response to reference signals (RRS)
Response time function (RTF)
Settling time function (STF)
Tracking error for command profiles function (TECPF)
Response time service (RTS)
Settling time servicr (STS)
Tracking error for command profiles service (TECPS)
Persistent Storage
intermediary service
Soft-goal
operationalizing soft-goals
Some (positive contribution)
And decomposition
E: Entry
X: eXit
W: Write
R: Read

Figure 12. Response to reference signals sub-model in an SOA context

Figure 13 shows that all the derived functions from the throughput time (TT) have their own services. The interdependency relations between these functions and their services are:

- The bandwidth function and the workload function may require data from their services using EXIT and ENTRY data movements.
- The bandwidth service may exchange data at a service layer either in a direct way with the workload service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.

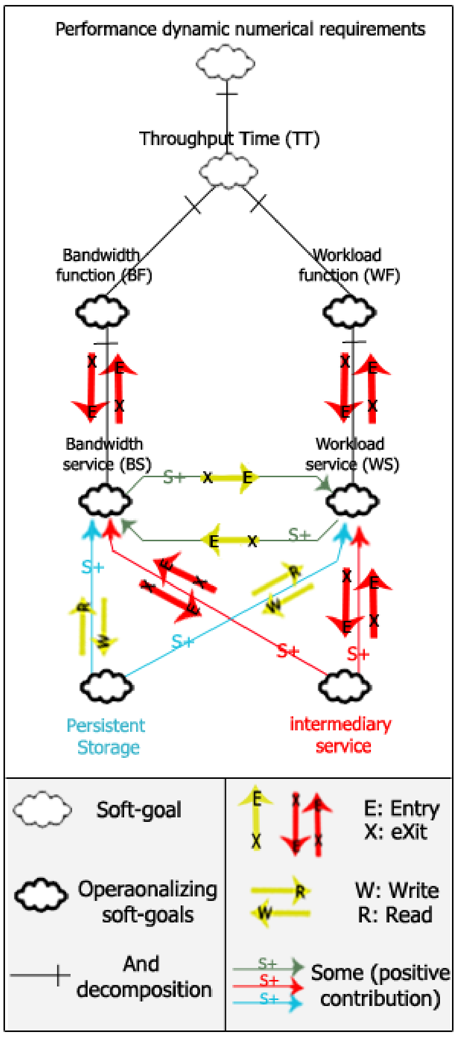


Figure 13. Throughput time sub-model in an SOA context

- The workload service may exchange data at a service layer either in a direct way with the bandwidth service using COSMIC EXIT and ENTRY data movements. Or it may exchange data in an indirect way through the persistent storage using COSMIC READ and WRITE data movements.
- The bandwidth service and the workload service may require data from any service in the overall performance framework through the intermediary service using COSMIC EXIT and ENTRY data movements.

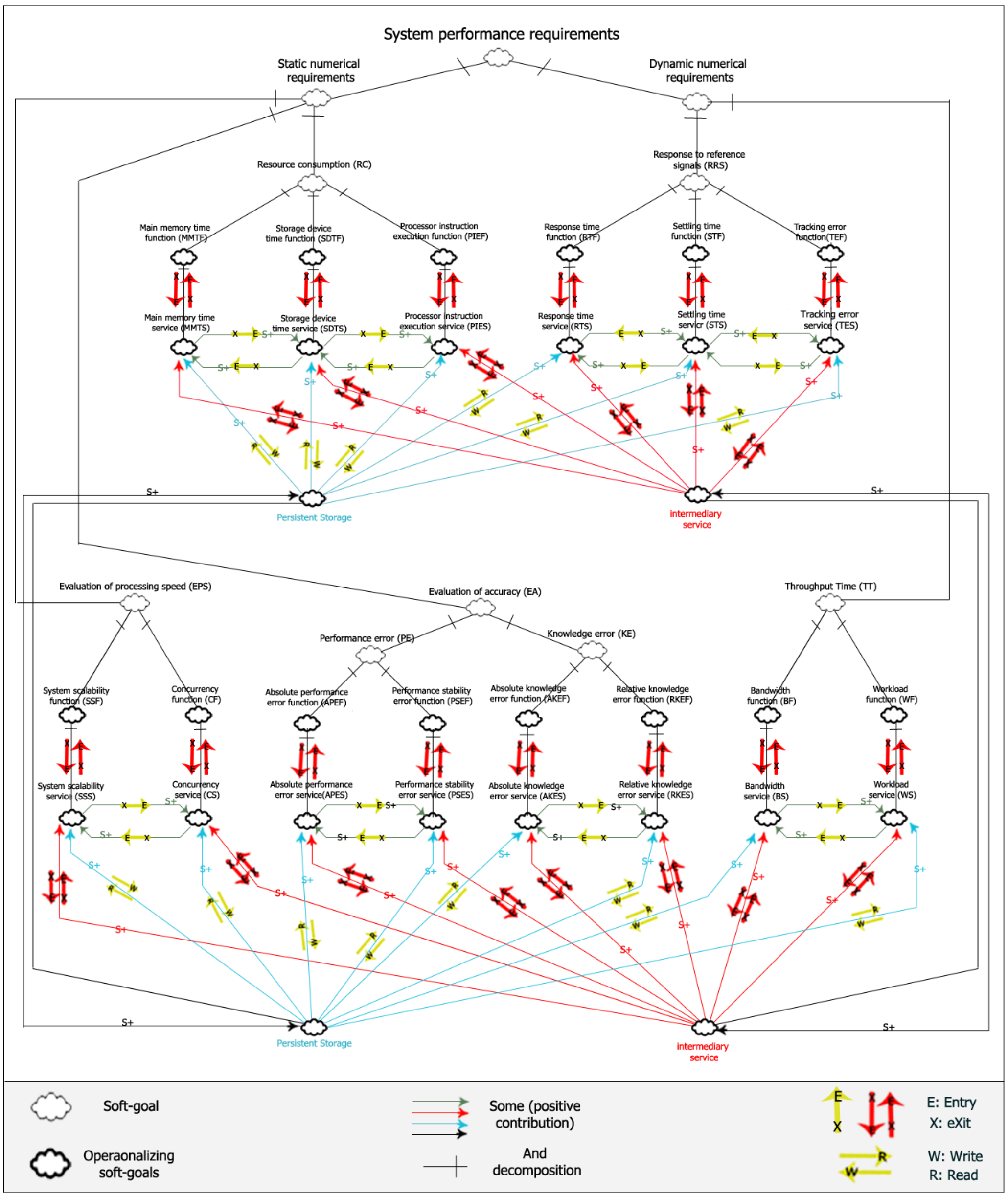

System performance requirements
Static numerical requirements
Dynamic numerical requirements
Resource consumption (RC)
Response to reference signals (RRS)
Main memory time function (MMTF)
Storage device time function (SDTF)
Processor instruction execution function (PIEF)
Response time function (RTF)
Settling time function (STF)
Tracking error function(TEF)
Main memory time service (MMTS)
Storage device time service (SDTS)
Processor instruction execution service (PIES)
Response time service (RTS)
Settling time servicr (STS)
Tracking error service (TES)
Persistent Storage
intermediary service
Evaluation of processing speed (EPS)
Evaluation of accuracy (EA)
Throughput Time (TT)
Performance error (PE)
Knowledge error (KE)
System scalability function (SSF)
Concurrency function (CF)
Absolute performance error function (APEF)
Performance stability error function (PSEF)
Absolute knowledge error function (AKEF)
Relative knowledge error function (RKEF)
Bandwidth function (BF)
Workload function (WF)
System scalability service (SSS)
Concurrency service (CS)
Absolute performance error service(APES)
Performance stability error service (PSES)
Absolute knowledge error service (AKES)
Relative knowledge error service (RKES)
Bandwidth service (BS)
Workload service (WS)
Persistent Storage
intermediary service
Soft-goal
Operaonalizing soft-goals
Some (positive contribution)
And decomposition
E: Entry
X: eXit
W: Write
R: Read

Figure 14. Full view of the system performance NFR model in an SOA context

Figure 14 illustrates the full view for the measurement framework of the system performance requirements on the basis of the previous sub-models in Figures 8–13 at functional level and in an SOA context.

From Figure 14, the following points can be observed for measurement purposes:

- In the direct data exchange situation, each EXIT and ENTRY data movement will be assigned a size of 1 CFP.
- In the indirect data exchange situation, each READ and WRITE data movement will be assigned 1 CFP.
- Data required through intermediary services that requires using 4 EXITS and 4 ENTRIES will be assigned 8 CFP.

5. Illustrative example: ATM banking system

5.1. Overview

Banking systems provide a variety of financial services to individuals, businesses and governments. An automated teller machine (ATM) is a computerized system found in a public location. Customers are identified by inserting a smartcard that contains a unique number and some information about the customer and their account status. The services typically provided by ATM banking systems include: accepting deposits, cash withdrawal, issuing balance statements, pre-paid mobile charges and money transfers. To achieve high customer satisfaction, such systems must demonstrate high quality levels including: excellent performance, security and reliability.

5.2. Purpose and process

The purpose of this example is to present a "proof--of-concept" on a small-scale of the concepts proposed in this paper by illustrating the use of the proposed measurement framework for system performance requirements allocated to software (as in Figure 14). More specifically, to derive the system performance requirements allocated to an ATM system, and to measure the functional size of these allocated requirements using the COSMIC method. This illustrative example was realized by applying the following steps:

- Analyze and specify the main components of the ATM internal structure in a physical view.
- Design the workflow scenario-based application for the customer view.
- Identify the ATM functional user requirements for the customer and system views.
- Specify the ATM system requirements allocated to software.
- Specify the ATM system performance requirements allocated to software as an extended view to step 4.
- Map the allocated system performance requirements with the proposed framework.
- Measure the functional size of the allocated system performance requirements to the specified banking system with the COSMIC ISO-recognized measurement unit.

5.3. ATM internal structure system

Here we detail the internal structure of the bank ATM and the relationships among its various parts. An ATM typically consists of several devices such as: central processor unit (CPU), crypto processor, memory, customer display, function key buttons (typically situated near the display), smart chip card reader, encrypting PIN pad, customer receipt printer, vault, and modem.

The vault stores all devices and parts that require limited access, such as:

- Cash dispensing mechanism (CDM),
- Deposit mechanism (DM),
- Security sensors (SS) (e.g., magnetic, thermal, seismic, gas),
- Electronic journal system (EJS) to keep system log,
- Cash dispenser (CD) which includes several removable cash cartridges, deposit mechanism and removable deposit cartridges.

The software specifications for the ATM system are: read the ATM card, count currency notes, connect to bank network, take input from user, validate user, dispense cash to user and receive deposit envelopes from the user through deposit slot.

5.4. ATM block diagram

The set of ATM system scenarios includes authentication of the PIN entered with the one encrypted on the card. Once the PIN is confirmed, the customer can access their bank account to make the chosen transaction. Or else the system shows a suitable message to clarify rejection of access. Figure 15 illustrates the first instantiation scenario for customer authentication for the ATM as follows:

- The client inserts his/her smartcard; the card reader processes the smartcard's data using the card transaction handler, and informs the system that the smartcard is valid.
- The card transaction handler displays a message on the ATM screen asking for the customer PIN number.
- The ATM screen asks the customer to enter the PIN and the customer enters PIN code which is passed on to the card transaction handler.
- The card transaction handler verifies and gives authorization if the PIN is correct; if not,

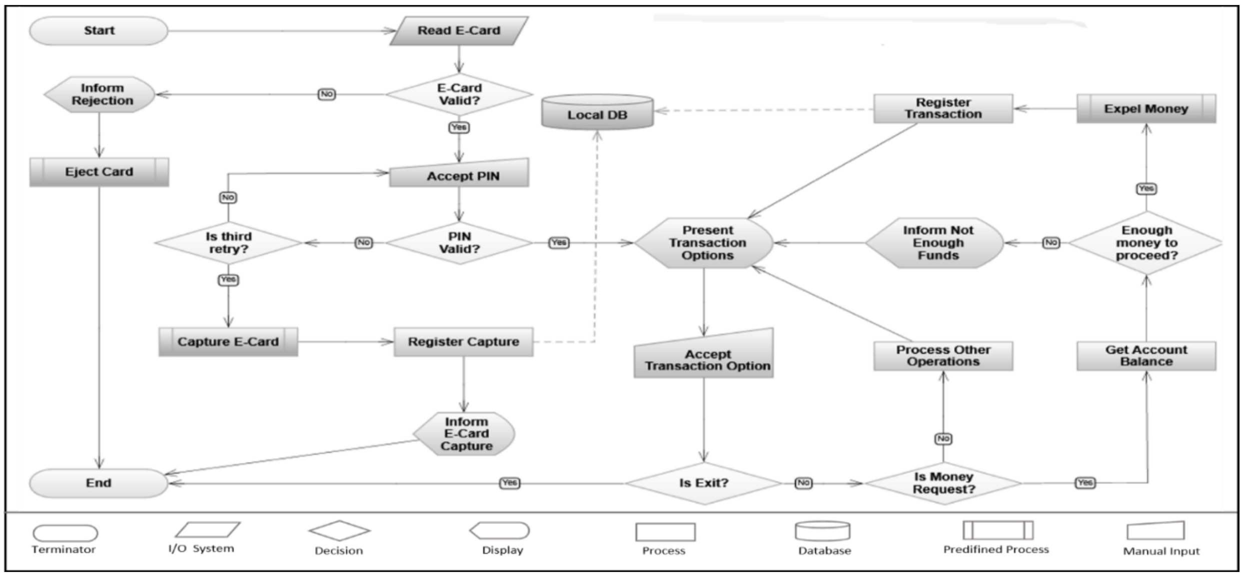


Figure 15. ATM functional requirements at the system level

a message appears on the screen to inform the customer that PIN number is invalid.

– The customer enters PIN again when the message appears on the screen to enter the PIN code number.
– If the customer has not provided the correct PIN in three iterations, the card reader will capture the customer smartcard and the session is terminated.

The second instantiation scenario after customer authentication for the ATM is as follows:

– The main menu that appears on the ATM system screen contains three types of transactions: ”get account balance inquiry” (choice 1), “cash withdrawal” (choice 2) and “money deposit” (choice 3). A choice to allow the user to exit the system (choice 4) appears as well.
– The user at that point chooses either to make a transaction by entering one of the three choices or exits the system.
– If the client enters “get account balance inquiry”, the ATM retrieves the balance from the bank’s database and the screen displays the client’s account balance.
– If the client enters “cash withdrawal”, the ATM screen displays a menu holding typical withdrawal amounts such as 50, 100, 200.
– The withdrawal menu also displays a choice to permit the customer to cancel the transaction.
– If the withdrawal amount selected is larger than the client’s account balance, the screen displays a message telling the client to select a smaller amount. The ATM then returns to the beginning of this scenario.
– If the withdrawal amount selected is less than or equal to the client’s account balance, the ATM proceeds and issues the client’s requested amount.
– Then the ATM subtracts the withdrawal amount from the client’s account in the bank’s database.
– The screen displays a message informing the user to take money.

5.5. ATM functional requirements (FR)

The functional requirements (FR) represent the system tasks from the stakeholder perspective and are typically derived from the context of use. The FR perspective can describe customer scenarios, system goals and objectives within the system environment and can connect these perspectives with assigned hardware resources.

Customer requirements are a subset of stakeholder requirements and can be collected in the stakeholder requirements specification document together with other perspective scenarios which have been derived from the system block diagram

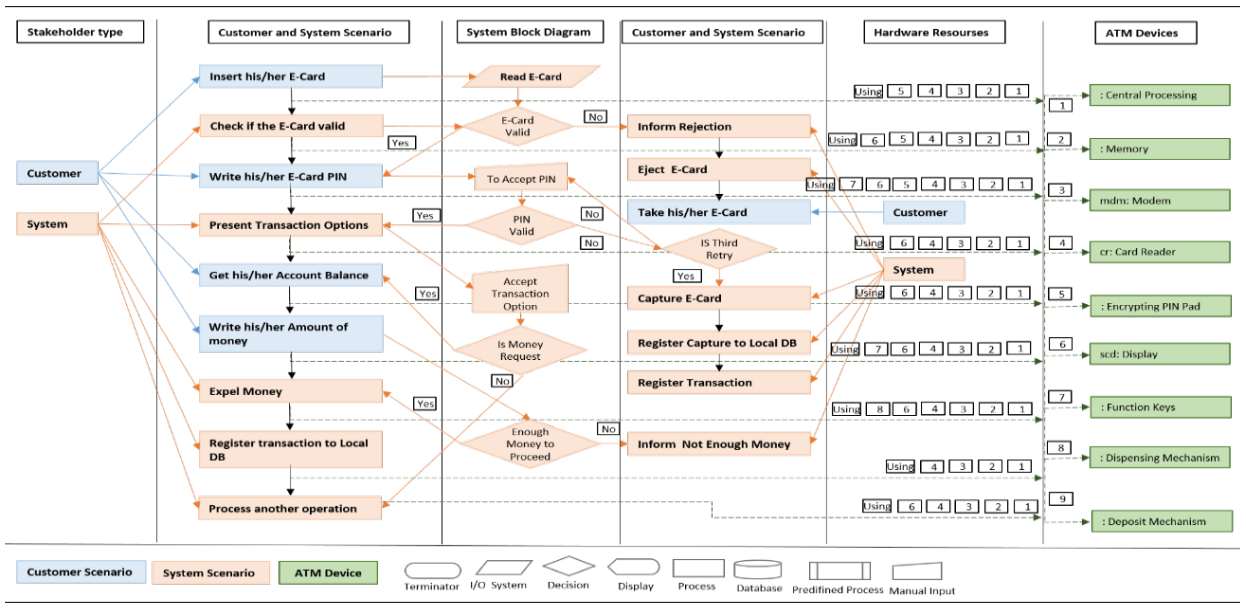


Figure 16. Customers and system scenarios identified from the ATM functional requirements

of Figures 15 and 16, including the identification of related hardware performance.

5.6. ATM system requirements allocated to software

Each system FR should be allocated to some specific software and/or physical component. Allocation should be defined in the early phases of the system development life cycle. At a high level in the system NFR this allocation impacts the design of the system architecture. Figure 17 illustrates the customer FR perspectives connected with software goals and sub software goals to derive the system requirements allocated to software.

5.7. System performance NFR allocated to software functions

This section presents an instantiation of some system performance requirements, and an example of their allocation to software. For this example, the following system performance NFR have been selected for the ATM system:

– Requirement 1: The maximum data transmission over the system network shall be 100 megabits per second.
– Requirement 2: The maximum time to respond to a customer transaction shall be one second.
– Requirement 3: The main memory access time in the system shall be 50 nanoseconds.
– Requirement 4: The system shall be scalable to handle the increased workload while maintaining the system performance level.
– Requirement 5: The system shall support the concurrent execution to execute the customer's transactions in a concurrent way.
– Requirement 6: The maximum time to recover the system from the instability state shall be two minutes.
– Requirement 7: The storage device access time in the system shall be 35 milliseconds and the data transfer rate shall be 156 megabytes per second.
– Requirement 8: The processor of the system shall be able to execute 2000 instructions per second.

For this example, these system performance NFR were allocated to the following software functions, see Figure 18:

– Requirement 1: to the bandwidth function and its services.
– Requirement 2: to the response time function services.
– Requirement 3: to the main memory time function services.
– Requirement 4: to the system scalability function services.

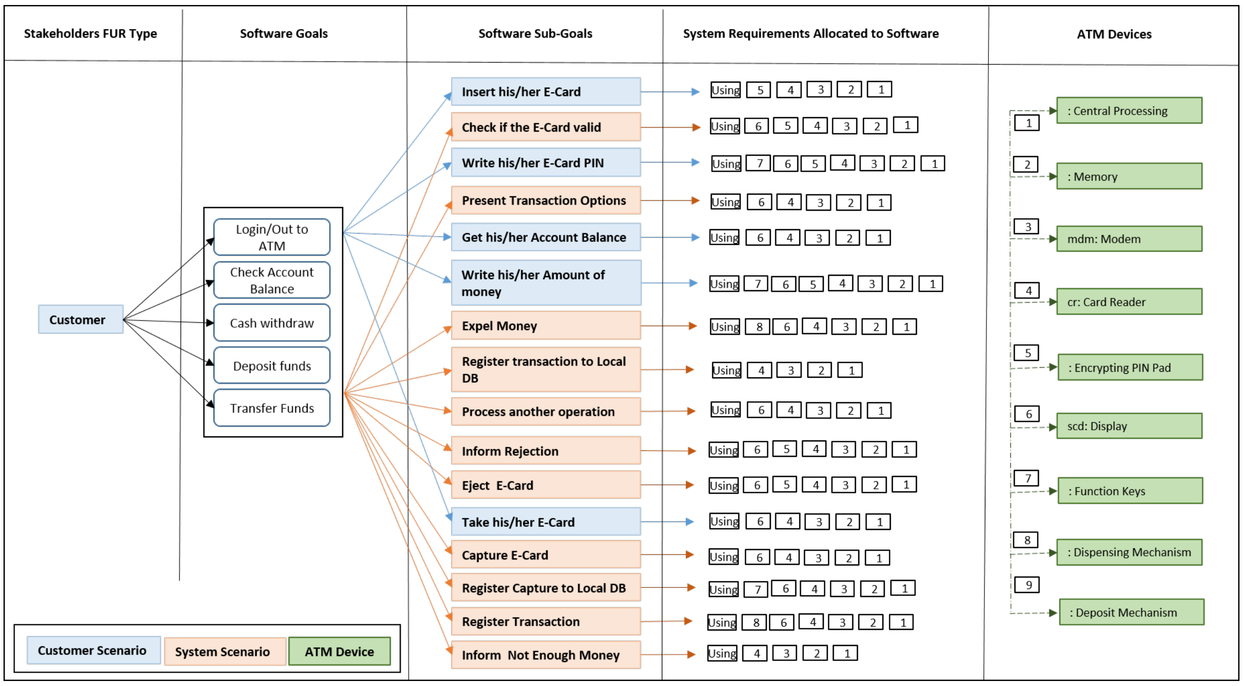


Figure 17. ATM customer and system scenarios allocated to software and ATM devices

– Requirement 5: to the concurrency function services.
– Requirement 6: to the settling time function services.
– Requirement 7: to the storage device time function services.
– Requirement 8: to the processor instruction execution function.

5.8. Mapping system performance NFR to an SOA context

This section maps the specified system performance requirements for the ATM system within an SOA context – see Figure 19. It can then be used to measure the instantiation case of the specified system performance NFR allocated to software for the banking ATM.

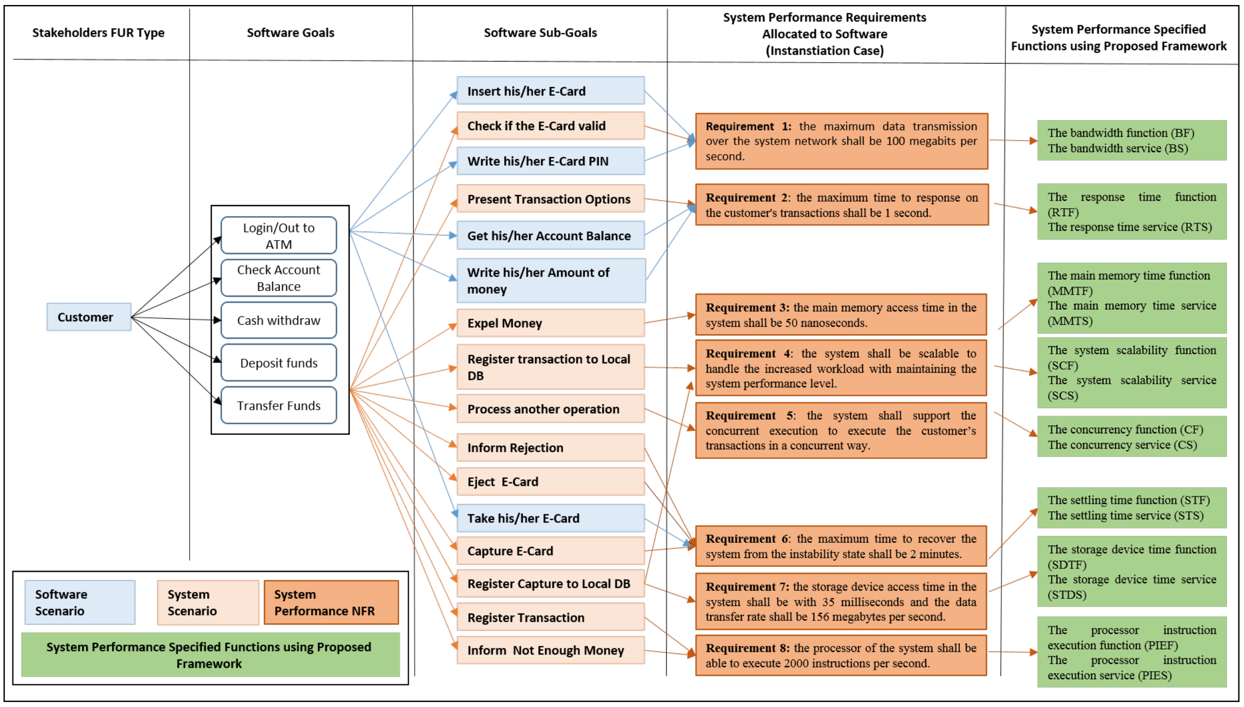
5.9. Measuring the specified system banking performance NFR (instantiation case)

This step identifies the detailed data movements for the allocated software performance functions in an SOA context. It is important to note that the performance NFR may require additional resources be added (i.e., hardware) to the system.

In this example, for illustrative purposes, a single data group was selected for each specified performance function, while in an industrial context these performance functions may require more than one data group. Table 1 shows the COSMIC measurement of the system performance NFR allocated to the software requirements at functional and service levels.

This example shows the corresponding COSMIC size for the selected specified requirements at the functional and services level (Figures 18 and 19). For measurement purposes they correspond to COSMIC data movements as follows:

Requirement 1 (R1): is allocated to one identified function (bandwidth function and its services) for the following customer-FR (Insert E-card, check if E-card is valid and Write E-card PIN). It extracts the identification data from the smartcard and invokes its own three services between three customer-FR, and then uses the persistence storage twice (once to check if the card is valid and second to check if the PIN number is correct. Here, we consider that the PIN is entered correctly).


Stakeholders FUR Type
Software Goals
Software Sub-Goals
System Performance Requirements Allocated to Software (Instanstiation Case)
System Performance Specified Functions using Proposed Framework
Customer
Login/Out to ATM
Check Account Balance
Cash withdraw
Deposit funds
Transfer Funds
Insert his/her E-Card
Check if the E-Card valid
Write his/her E-Card PIN
Present Transaction Options
Get his/her Account Balance
Write his/her Amount of money
Expel Money
Register transaction to Local DB
Process another operation
Inform Rejection
Eject E-Card
Take his/her E-Card
Capture E-Card
Register Capture to Local DB
Register Transaction
Inform Not Enough Money
The bandwidth function (BF)
The bandwidth service (BS)
The response time function (RTF)
The response time service (RTS)
The main memory time function (MMTF)
The main memory time service (MMTS)
The system scalability function (SCF)
The system scalability service (SCS)
The concurrency function (CF)
The concurrency service (CS)
The settling time function (STF)
The settling time service (STS)
The storage device time function (SDTF)
The storage device time service (STDS)
The processor instruction execution function (PIEF)
The processor instruction execution service (PIES)
Software Scenario
System Scenario
System Performance NFR
System Performance Specified Functions using Proposed Framework

Requirement 1: the maximum data transmission over the system network shall be 100 megabits per second.

Requirement 2: the maximum time to response on the customer's transactions shall be 1 second.

Requirement 3: the main memory access time in the system shall be 50 nanoseconds.

Requirement 4: the system shall be scalable to handle the increased workload with maintaining the system performance level.

Requirement 5: the system shall support the concurrent execution to execute the customer's transactions in a concurrent way.

Requirement 6: the maximum time to recover the system from the instability state shall be 2 minutes.

Requirement 7: the storage device access time in the system shall be with 35 milliseconds and the data transfer rate shall be 156 megabytes per second.

Requirement 8: the processor of the system shall be able to execute 2000 instructions per second.

Figure 18. ATM system performance NFR allocated to software functions

The measurement of the functional size using ISO 19761 (COSMIC) for R1 is calculated based on data movements between groups of processes as follows:

- Three identification functions (3 ENTRY and 3 EXIT).
- Two identification services (4 ENTRY and 4 EXIT).
- Functional and services processes use persistence storage twice to obtain information about the card (2 READ and 2 WRITE).

The total functional size for R1 is 18 CFP.

Requirement 2 (R2): is allocated to one identified function (response time function and its services) for the following customer-FR (Present transaction options, get account balance and Write amount of money).

A. The main menu on the ATM system screen displays three types of transactions options. The measurement of the COSMIC functional size for R2-A is as follows:

- One identification function (1 ENTRY and 1 EXIT).
- One identification service (2 ENTRY and 2 EXIT).
- Functional and services processes use persistence storage once, which appears on the main menu of the ATM application (1 READ and 1 WRITE).

The total functional size for R2-A is 8 CFP.

B. In this instantiation the customer enters "get account balance inquiry". The ATM retrieves the client's account balance from the bank's database which is then displayed on the screen. The measurement of the COSMIC functional size for R2-B is as follows:

- One identification function (1 ENTRY and 1 EXIT).
- One identification service (2 ENTRY and 2 EXIT).
- Functional and service processes use persistence storage once which appears on the main menu for the ATM application (1 READ and 1 WRITE).
- One intermediary service is needed to retrieve the customer account balance from the external database (4 Entry and 4 Exit).

The total functional size for R2-B is 16 CFP.

C. When the customer enters "cash withdrawal", the ATM screen displays a menu showing typical

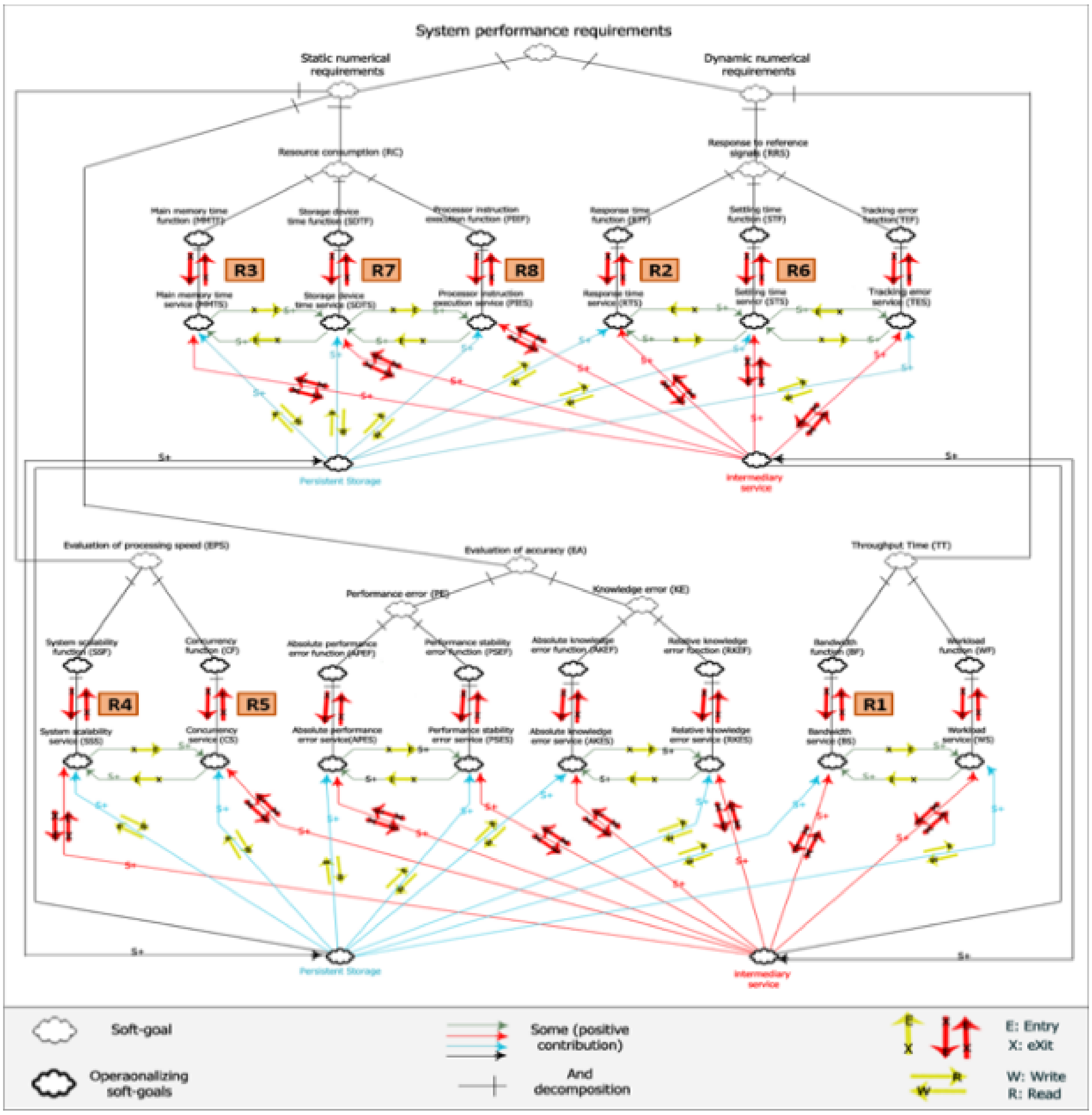

System performance requirements
Static numerical requirements
Dynamic numerical requirements
Resource consumption (RC)
Response to reference signals (RRS)
Main memory time function (MMTF)
Storage device time function (SDTF)
Processor instruction execution function (PIEF)
Response time function (RTF)
Settling time function (STF)
Tracking error function (TEF)
R3
R7
R8
R2
R6
Main memory time service (MMTS)
Storage device time service (SDTS)
Processor instruction execution service (PIES)
Response time service (RTS)
Settling time service (STS)
Tracking error service (TES)
Persistent Storage
intermediary service
Evaluation of processing speed (EPS)
Evaluation of accuracy (EA)
Throughput Time (TT)
Performance error (PE)
Knowledge error (KE)
System scalability function (SSF)
Concurrency function (CF)
Absolute performance error function (APEF)
Performance stability error function (PSEF)
Absolute knowledge error function (AKEF)
Relative knowledge error function (RKEF)
Bandwidth function (BF)
Workload function (WF)
R4
R5
R1
System scalability service (SSS)
Concurrency service (CS)
Absolute performance error service (APES)
Performance stability error service (PSES)
Absolute knowledge error service (AKES)
Relative knowledge error service (RKES)
Bandwidth service (BS)
Workload service (WS)
Persistent Storage
intermediary service
S+
Soft-goal
Operaonalizing soft-goals
Some (positive contribution)
And decomposition
E: Entry
X: eXit
W: Write
R: Read

Figure 19. ATM system performance requirements allocated to software functions within an SOA context

withdrawal amounts, such as 50, 100, 200. In this instantiation, the customer can enter their own desired amount. The measurement of the COSMIC functional size for R2-C calculated based on data movements between groups of processes is described as follows:

– One identification function (1 ENTRY and 1 EXIT).
– One identification service (2 ENTRY and 2 EXIT).
– One intermediary service is needed to subtract the amount of money from the customer account balance (i.e., the customer has 300, takes 50, the remainder is 250) (4 ENTRY and 4 EXIT).
– Functional and service processes use persistence storage twice to obtain information about the card (1 READ and 1 WRITE).

The total functional size for R2 (A, B, C) is (8 + 16 + 16) = 40 CFP.

Requirement 3 (R3): is allocated to one identified function (main memory time function and its services) for the customer-FUR (Expel money).

The measurement of the COSMIC functional size for R3 is calculated based on data movements between groups of processes and is described as follows:

- One identification function (1 ENTRY and 1 EXIT).
- One identification service (2 ENTRY and 2 EXIT).

The total functional size for R3 is 6 CFP.

Requirement 4 (R4): is allocated to one identified function (system scalability function and its services), twice for the customer-FUR (Register transaction to local DB and Register capture to local DB).

The measurement of the COSMIC functional size for R4 is calculated based on data movements between groups of processes and is described as follows:

- One identification function twice (2 ENTRY and 2 EXIT).
- One identification service twice (4 ENTRY and 4 EXIT).
- Twice the intermediary service while the system uses the database (8 ENTRY and 8 EXIT).

The total functional size for R4 is 28 CFP.

Requirement 5 (R5): is allocated to one identified function (settling time function and its services) for the customer-FR (Process another operation).

The measurement of the COSMIC functional size for R5 is calculated based on data movements between groups of processes and is described as follows:

- One identification function (1 ENTRY and 1 EXIT).
- One identification service (2 ENTRY and 2 EXIT).

The total functional size for R5 is 6 CFP.

Requirement 6 (R6): is allocated to one identified function (concurrency function and its services) for the customer-FUR (Inform rejection, Eject E-card, Take E-card and Capture E-card).

The measurement of the COSMIC functional size for R6 is calculated based on data movements between groups of processes and is described as follows:

- One identification function four times (4 ENTRY and 4 EXIT).
- One identification service four times (8 ENTRY and 8 EXIT).

The total functional size for R6 is 24 CFP.

Requirement 7 (R7): is allocated to one identified function (storage device time function and its services) for the customer-FUR (Register capture to local DB).

The measurement of the COSMIC functional size for this process is calculated based on data movements between groups of processes and is described as follows:

- One identification function (1 ENTRY and 1 EXIT).
- One identification service (2 ENTRY and 2 EXIT).
- Two intermediary services, one to use the local ATM database and the second to give order to storage to capture the customer smartcard (8 Entry and 8 Exit).

The total functional size for R7 is 22 CFP.

Requirement 8 (R8): Requirement 8 (R8): is allocated to one identified function (processor instruction execution function and its services) for the customer-FUR (Register transaction and Inform not enough money).

The measurement of the COSMIC functional size for R8 is calculated based on data movements between groups of processes and is described as follows:

- One identification function twice (2 ENTRY and 2 EXIT).
- One identification service twice (4 ENTRY and 4 EXIT).

The total functional size for R8 is 12 CFP.

5.10. Summary of findings

Table 1 lists the 17 software subgoals to be measured from the customer-FUR perspective. These software subgoals call eight specified functions and eight specified services processes of system performance allocated to software in R1, R2 to R8. These are identified and presented in columns 1, 2 and 3. For each identified functional process, the description of the measured resource represents a data performance group moved by one data movement type, each one measured as 1 CFP (COSMIC function point).

The total functional size for the performance requirements R1, R2 to R8 applied to this software in a functional level is 34 CFP and in SOA context is equal to 122 CFP, independent of the languages and technologies used to implement them. In further applications, this functional size number can be used for effort estimation models and for software benchmarking.

5.11. Limitations of the illustrative example

The ATM example illustrates how the proposed approach is applicable in a relatively simple context. Future research is needed to investigate its scalability to much larger contexts, such as complex systems that perform millions of operations per second. Naturally, for such large complex systems, organizations have more resources and can dedicate much more resources to make use of this approach. In such contexts, additional research on efficiency studies would be welcome.

5.12. Practical Implications

We think that this framework can be easily used by someone knowledgeable in performance standards, COSMIC measurement and SOA architecture. Someone familiar with other environments, such as cloud computing environments, could also use and adapt this framework to a context-specific environment. More specifically, requirement specialists, software architects, performance engineers and project managers can use the proposed standards-based performance framework.

The generic approach proposed in our research program on standards-based identification, specification and measurement of system NFRs allocated to software had already influenced the COSMIC Group when it published its initial strategy on how to handle NFRs and Project Requirements from a project management perspective [52]. To facilitate industry adoption of this approach, including the performance framework presented here, we recently edited, for the COSMIC Group, a practitioner's guide on Non-Functional Requirements and their Sizing with COSMIC [53]: it includes a set of templates based on the performance framework documented in more detail in this study. Organizations with a very large NFR knowledge base as well as experts in performance issues can also compare their own practices with this framework and identify gaps within their practices; such gaps can be addressed, fully or partially, by using this framework. If their own practices are not structured and explicit, they can use the structure of this framework to structure and document their own related practices.

5.13. Threats to validity

The quality of an experimental case study is important as it can substantiate the limits of the study and identify threats to its validity, which could impact the results. The key types of validity threats as proposed by Wohlin [54] are:

Internal validity is concerned with the reliability of the results and refers to the treatment that caused the outcome [54]. There can be other uncontrollable or not measured factors that influenced the results. The internal validity threat in this example includes the change of the design process for this example. Here, we believe that if different researchers use the same method on the same example, they will get the same results. In contrast, in the absence of a full description of some parts considered in this example, variability in the results would be expected.

External validity [54] here refers to the generalization of the results outside the scope of the example and whether or not the cause and effect established hold in other situations. An external validity threat is expressed at the outcomes level. The proposed measurements framework of system performance NFR was illustrated using only the performance requirements specifications of an ATM system. There is no claim in this study on generalizability and scalability to much larger contexts, such as complex systems that perform millions of operations per second. Scalability is outside the scope of the research presented here. To alleviate the risk of this validity threat

Table 1. ATM measurement of the system performance NFR allocated to software functions

Req. ID	Software sub-goals customer-FR & system-FR	Specified system performance NFR allocated to software functions		Data movements identified				
				E	X	R	W	Size in CFPs
R 1	Insert E-card	Bandwidth function	Bandwidth service	3	3	-	-	6
	Check if E-card valid Enter E-card PIN			4	4	2	2	12
R 2	Present transaction options	Response time function	Response time service	3	3	-	-	6
	Retrieve account balance amount of money Display			14	14	3	3	34
R 3	Expel money	Main memory time function	Main memory time service	1	1	-	-	2
				2	2	-	-	4
R 4	Register transaction to local DB	System scalability function	System scalability service	2	2	-	-	4
	Register capture to local DB			12	12	-	-	24
R 5	Process another operation	Concurrency function	Concurrency service	1	1	-	-	2
				2	2	-	-	4
R 6	Inform rejection	Settling time function	Settling time service	4	4	-	-	8
	Eject E-card Accept E-card Capture E-card			8	8	-	-	16
R 7	Register capture to local DB	Storage device time function	Storage device time service	1	1	-	-	2
				10	10	-	-	20
R 8	Register transaction	Processor instruction execution function	Processor instruction execution service	2	2	-	-	4
	Inform not enough money			4	4	-	-	8
Total functional size of system performance requirements at functional level				17	17	0	0	34 CFP
Total functional size of system performance requirements at services level				56	56	5	5	122 CFP
Total # functional size measurement at (functional and service levels)				73	73	5	5	156 CFP

additional examples and case studies could be conducted using requirement specifications of different types of software products (e.g., business application software, Telecom software, medical embedded software, etc.).

Construct validity [54] refers to the relation between the theory behind the experiment and the observation(s). The treatment and the results may not correspond to the cause and the effect controlled and measured. A limited number of standards and methods have been selected for this example. Nevertheless, there exist other standards and methods, and future research could investigate the use and relevance of such other standards and methods.

6. Conclusion and further work

This paper proposed a measurement framework of system performance NRF allocated to software functions. This work on system performance NFR extends our previous work on three types of NFR (security, portability and maintainability) to facilitate the early identification, specification and measurement of such kinds of NFR.

The suggested framework includes some of the consensual performance terms and concepts used by two sets of international standards (ECSS and IEEE) and some related works. They were analyzed and integrated using different design views beginning with the logical view, followed by process view, development view and ending with physical views. Next, the set of ISO 19761 (COSMIC) concepts and views were adopted for describing the framework functionality at a lower level to illustrate that the proposed framework is designed for measurement purposes as well as for capture of the performance concept. The proposed framework was designed using SIGs.

This research considered system performance requirements as both static and dynamic performance types, each with its own set of candidate sub-concepts.

Additionally, for a more complete software view of a complex environment (i.e., functional services in a service-oriented architecture), COSMIC-SOA was applied to the suggested framework.

Finally, an ATM example was presented to guide developers and software engineers to use the measurement of system performance NFR allocated as performance FUR at the software level.

The main contribution of this work is its ability to assist developers, system and software engineers to specify system performance requirements early in the life cycle in order to address the specified performance functions to be allocated to software as functional requirements.

The proposed framework can also be used for identification, specification and measurement of system performance NFR using ISO 19761 independently of any programming languages, and in addition can address software performance FUR early in their implementation.

In this paper, the proposed measurement aspects addressed the system requirements allocated to software. It will be interesting in further work to extend this measurement aspect to consider other types of requirements at the system level containing hardware requirements.

Some related issues were not addressed and additional work is required such as using these functional size measurement results of system performance requirements allocated to software-FURs as new input for estimating models in software engineering projects. Additional empirical work is required to verify that such expanded size can improve the estimation models, including for testing and maintenance effort.

The ATM illustrative example showed that the proposed measurement framework can help to specify and measure the functional size of system performance-NFR allocated to software functions. Consequently, this may improve planning, managing and development of software at different phases of the software development life cycle. Furthermore, the measurement results of the proposed framework may be used in benchmarking studies.

This example was not built to learn but to demonstrate that the framework was usable, and the purpose in building this example was not to evaluate the framework. This again, would require much more empirical work with practitioners to evaluate it in a number of contexts, and with a set of criteria for evaluation.

This proposed measurement framework is limited to measuring the system performance requirements allocated to software at the functional and service levels. It will be interesting in further work to focus on its applicability to different types of software products in order to generalize the results reported in this illustrative example. In addition, further work could focus on automating the measurement of software performance requirements through building an automated measurement tool (or enhancing an existing one).

Future work on the scalability of the framework proposed in this study would be valuable for industrial research where researchers look at such practical scalability issues, with financial resources much larger than the resources available to university researchers.

References

[1] K.T. Al-Sarayreh, "Model of early specifications of performance requirements at functional levels," *Recent Advances on Electroscience and Computers*, 2015, p. 236.

[2] K. Meridji, K.T. Al-Sarayreh, A. Abran, and S. Trudel, "System security requirements: A framework for early identification, specification and measurement of related software requirements," *Computer Standards and Interfaces*, Vol. 66, 2019, p. 103346.

[3] A. Abran, K.T. Al-Sarayreh, and J.J. Cuadrado--Gallego, "A standards-based reference framework for system portability requirements," *Comput. Stand. Interfaces*, Vol. 35, No. 4, Jun. 2013, p. 380–395. [Online]. https://doi.org/10.1016/j.csi.2012.11.003

[4] K.T. Al-Sarayreh, A. Abran, and J.J. Cuadrado--Gallego, "A standards-based model of system maintainability requirements," *journal of software: evolution and process*, Vol. 25, No. 5, 2013, pp. 459–505.

[5] K.T. Al-Sarayreh, A. Abran, and J.J. Cuadrado--Gallego, "Measurement model of software requirements derived from system portability requirements," in *9th International Conference on Software Engineering Research and Practice (SERP 2010)*, 2010, pp. 553–559.

[6] K.T. Al-Sarayreh, I. Al-Oqily, and K. Meridji, "A standard based reference framework for system adaptation and installation requirements," in *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*, 2012, pp. 7–12.

[7] M. Kassab, O. Ormandjieva, and M. Daneva, "An ontology based approach to non-functional requirements conceptualization," in *2009 Fourth International Conference on Software Engineering Advances*. IEEE, 2009, pp. 299–308.

[8] X. Zhang and X. Wang, "Tradeoff analysis for conflicting software non-functional requirements," *IEEE Access*, Vol. 7, 2019, pp. 156 463–156 475.

[9] M. Dewi and Z. Didar, "An ontological framework to manage the relative conflicts between security and usability requirements," in *2010 Third International Workshop on Managing Requirements Knowledge*. IEEE, 2010, pp. 1–6.

[10] N. Daclin, B. Moradi, and V. Chapurlat, "Analyzing interoperability in a non-functional requirements ecosystem to support crisis management response," *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*, 2018, pp. 429–434.

[11] L.M. Cysneiros, K.K. Breitman, C. Lopez, and H. Astudillo, "Querying software interdependence graphs," in *2008 32nd Annual IEEE Software Engineering Workshop*. IEEE, 2008, pp. 108–112.

[12] K.T. Al-Sarayreh, "Dependability model for decomposition and allocation of system safety integrity levels of software quality," *International Review on Computers and Software*, Vol. 10, No. 11, 2015.

[13] S. Al-Qudah, K. Meridji, and K.T. Al-Sarayreh, "A comprehensive survey of software development cost estimation studies," in *Proceedings of the international conference on intelligent information processing, security and advanced communication*, 2015, pp. 1–5.

[14] R.E. Al-Qutaish and K.T. Al-Sarayreh, "Software process and product ISO standards: a comprehensive survey," *European Journal of Scientific Research*, Vol. 19, No. 2, 2008, pp. 289–303.

[15] A. Abran and K.T. Al-Sarayreh, "Standards-based model for the specification of system design and implementation constraints," in *Industrial Proceedings, 17th European Systems and Software Process Improvement and Innovation, EuroSPI 2010 Conference.* Publisher: Delta, Denmark Grenoble (France), 2010, pp. 4–7.

[16] R. Tawhid and D. Petriu, "Automatic derivation of a product performance model from a software product line model," in *2011 15th International Software Product Line Conference*. IEEE, 2011, pp. 80–89.

[17] M. Noorian, E. Bagheri, and W. Du, "Non-functional properties in software product lines: A axonomy for classification." in *SEKE*, Vol. 12, 2012, pp. 663–667.

[18] A. Danylenko and W. Löwe, "Context-aware recommender systems for non-functional requirements," in *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*. IEEE, 2012, pp. 80–84.

[19] H.T. Jung and G.H. Lee, "A systematic software development process for non-functional requirements," in *2010 International conference on information and communication technology convergence (ICTC)*. IEEE, 2010, pp. 431–436.

[20] *Gyro terminology and performance specification*, European Cooperation for Space Standardization Std. ECSS-E-ST-60-21C, 2017.

[21] *System engineering general requirements*, European Cooperation for Space Standardization Std. ECSS-E-ST-10C Rev. 1, 2017.

[22] *Software product assurance*, European Cooperation for Space Standardization Std. ECSS-Q--ST-80C Rev.1, 2017.

[23] *Space Engineering: Control Performance*, European Cooperation for Space Standardization Std. ECSS-E-ST-60-10C, 2008.

[24] *Space engineering, Stars sensors terminology and performance specification*, European Cooperation for Space Standardization Std. ECSS-E--ST-60-20C Rev.2 DIR1, 2017.

[25] *Satellite attitude and orbit control system (AOCS) requirements*, European Cooperation for Space Standardization Std. ECSS-E-ST-60-30C:, 2013.

[26] *Recommended Practice for Software Requirements Specifications*, Institute of Electrical and Electronics Engineers Std. 830-1998, 1998.

[27] L. Mo, S. Zhigang, H. Quan, Y. Guizhi, L. Ya, and S. Fengli, "Analysis and testing of key performance indexes of vxworks in real-time system," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2018, pp. 369–374.

[28] A. Ahmad, N. Abdulrahman, B. Sascha, J. Naoum, and T. Klaus, "Toward a performance requirements model for the early design phase of IT systems," in *2018 Sixth International Conference on Enterprise Systems (ES)*. IEEE, 2018, pp. 9–16.

[29] Z. Chen, T. Zhao, J. Jiao, and H. Wu, "Availability analysis of multi-state weighted k-out-of-n systems with component performance requirements," in *2018 Annual Reliability and Maintainability Symposium (RAMS)*. IEEE, 2018, pp. 1–5.

[30] K.T. Al-Sarayreh, I. Ibrahim Al-Oqily, and K. Meridji, "A standard-based reference framework for system operations requirements," *International Journal of Computer Applications in Technology*, Vol. 47, No. 4, 2013, pp. 351–363.

[31] J. Kai, X. Ling, and Z. Huamin, "A parameter tuning method of proportional integral controller for the first-order plus delay time system based on the desired dynamical performance," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE, 2014, pp. 6110–6115.

[32] K. Arun and A. Sunil, "Statistical analysis of memory and performance non functional requirements in real time embedded system development for agile methodology," in *2015 International Conference on Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 300–305.

[33] I. Vila, J. Perez-Romero, O. Sallent, A. Umbert, and R. Ferrus, "Performance measurements-based estimation of radio resource requirements for slice admission control," in *90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[34] M. Anish, R. Anand, R. Srivaths, and J. Niraj, "Automated energy/performance macromodeling of embedded software," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 3, 2007, pp. 542–552.

[35] *Software measurement – Functional size measurement Part 1: Definition of concepts*, ISO/IEC Std. 14 143-1, 1998.

[36] *COSMIC v 3.0 – A Functional Size Measurement Method, I*, ISO/IEC Std. 19 761, 2003.

[37] P. Fagg, A. Lesterhuis, G. Rule, G. Ungerer, K. Galegaonkar, S.and Natarajan, L. Santillo, F. Vogelezang, P. Jain, M. O'Neill, and C. Symons, "Guideline for sizing SOA software," The Common Software Measurement International Consortium (COSMIC), Tech. Rep., 2010.

[38] L. Santillo, "Seizing and sizing SOA applications with COSMIC function points," *Proceedings of SMEF*, 2007.

[39] E. Marks, *Service-oriented architecture governance for the services driven enterprise*. John Wiley and Sons, 2008.

[40] L. Chung, B. Nixon, and E. Yu, "Dealing with change: An approach using non-functional requirements," *Requirements Engineering*, Vol. 1, No. 4, 1996, pp. 238–260.

[41] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Springer Science and Business Media, 2012, Vol. 5.

[42] K. Hemenway, M. Iff, and T. Calishain, *Spidering Hacks: 100 Industrial-Strength Tips and Tools.* " O'Reilly Media, Inc.", 2004.

[43] I. Lee, J. Leung, and S. Son, *Handbook of real-time and embedded systems.* CRC Press, 2007.

[44] K.T. Al-Sarayreh, L.A. Hasan, and K. Almakadmeh, "A trade-off model of software requirements for balancing between security and usability issues," *International Review on Computers and Software*, Vol. 10, No. 12, 2016, pp. 1157–1168.

[45] C.M. Kozierok, *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2005.

[46] A. Abran and K. Meridji, "Analysis of software engineering from an engineering perspective," *European Journal for the Informatics Professional*, Vol. 7, No. 1, 2006, pp. 46–52.

[47] K. Meridji, K.T. Al-Sarayreh, and A. Al-Khasawneh, "A generic model for the specification of software reliability requirements and measurement of their functional size," *International Journal of Information Quality*, Vol. 3, No. 2, 2013, pp. 139–163.

[48] J. Carr, *The technician's radio receiver handbook: wireless and telecommunication technology.* Elsevier, 2001.

[49] J.J. Parsons and D. Oja, *New Perspectives on Computer Concepts 2014: Comprehensive.* Cengage Learning, 2013.

[50] B. Parkinson and J. Spilker, "Progress in astronautics and aeronautics: Global positioning system: Theory and applications. american institute of aeronautics/astronautics," 1996.

[51] H. El-Rewini and M. Abd-El-Barr, *Advanced computer architecture and parallel processing.* John Wiley and Sons, 2005, Vol. 42.

[52] K.T. Al-Sarayreh and A. Abran, "Specification and measurement of system configuration non functional requirements," in *20th International Workshop on Software Measurement and International Conference on Software Measurement, IWSM/Metrikon/Mensura, Stuttgart, Germany*, 2010, pp. 141–156.

[53] A. Abran and K.T. Al-Sarayreh, "Non-functional requirements and their sizing with cosmic: Practitioner's guide," in *COSMIC Gruop*, 2020, pp. 1–14.

[54] W. Claes, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in software engineering.* Springer Science and Business Media, 2012.