Non-Functional Requirements and COSMIC Sizing

Practitioner's Guide

May 2020

(minor update: December 2020)

Foreword

The COSMIC method measures a 'functional size' of software based on functional user requirements (FURs). COSMIC-measured functional sizes are mainly used in:

- estimating project effort, e.g. where 'project effort' = (new software estimated functional size) / (productivity from previous similar projects);
- measuring and comparing performance across projects of similar characteristics, e.g. using 'productivity' = (software functional size) / (project effort).

The purpose of this guide is to:

- illustrate that system non-functional requirements (system-NFRs) may be allocated in projects as software functional user requirements (software-FUR);
- demonstrate how to specify and measure, using COSMIC Function Points, the functional size of software-FURs that were system-NFRs; and
- provide an illustrative example.

This version updates the References and formatting style & layout for better readability.

Two additional documents on this topic are available from the COSMIC group:

- A repository of NFR functions described in a number of standards (ISO, IEEE and ECSS), together with additional examples and case studies (in progress) [1].
- "Non-Functional & Project Requirements with COSMIC: Experts Guide", v.2, 2020 [2].

Acknowledgements

Editors:

Alain Abran, École de technologie supérieure, Université du Québec, Canada.

Khalid Al-Sarayreh, Hashemite University, Jordan

Arlan Lesterhuis, The Netherlands

Reviewers:

J.M. Desharnais, École de technologie supérieure, Université du Québec, Canada

K.R. Jayakumar, Amitysoft, India

Dylan Ren, Measures Ltd., China

Luca Santillo, Agile Metrics, Italy

Hassan Soubra, German University in Cairo, Egypt

Frank Vogelezang, Metri, Netherlands

Jelle de Vries, Rubicon Consulting, Netherlands

Table of Contents

1. SYSTEM-NFRS EVOLVING INTO SOFTWARE-FURS

System non-functional requirements (system-NFRs) may first be stated as 'high-level' requirements. As a project progresses and more of its details become known, these system-NFRs may be allocated to software functions, hardware functions or a combination of both. This evolution is illustrated in Figure 1 across a project lifecycle.

Note 1 Depending on the maturity of the organization and the project strategy, the NFRs may become clear sooner or later in the lifecycle. By using a mature requirements engineering process, most NFRs will already be known in the design phase.

Note 2 For simplicity in this guide, Figure 1 shows system-NFRs that have 'evolved' into software functional user requirements which are then referred to as 'software-FURs' [1, 2, 5].

Note 3 System-NFRs are also often referred to as system quality requirements.

Note 4 The project lifecycle illustrated in Figure 1 may be part of a system lifecycle which includes activities for the allocation of system requirements to hardware, software and/or administrative procedures.

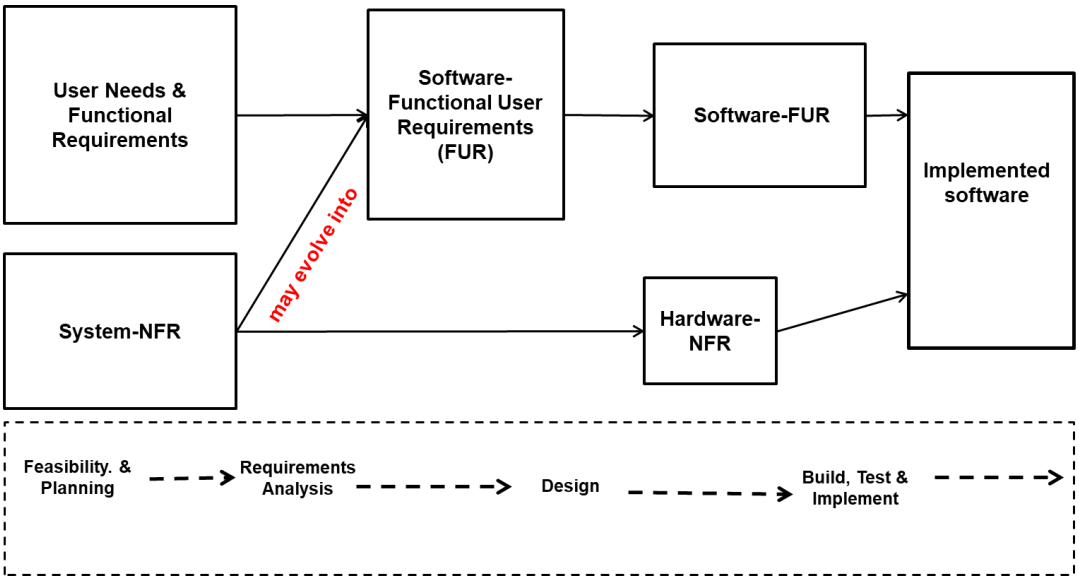


Figure 1. System-NFRs may evolve into software-FURs as a project progresses.

It is **important** to note from Figure 1 that when part or all of a requirement initially stated as a system-NFR evolves, as a project progresses, into a software-FUR, the additional software-FUR can be measured or approximated using COSMIC Function Points.

2. SYSTEM-NFR ALLOCATION TO SOFTWARE FUNCTIONS

There are four key steps once the system-NFRs are identified:

Step 1 Mapping systems-NFRs expressed at the high level to related system-NFR functions listed in international standards

Step 2 Allocation of system-NFRs to software functions as applicable

Step 3 Specification of the software functions into software-FURs within a software architecture

Step 4 Functional size measurement (or approximation) with COSMIC Function Points

Step 1. Mapping system needs to system-NFRs described in international standards

At the system level, the stakeholders identify the system needs in terms of NFRs.

The following aids in the COSMIC NFR Guides help identify system needs and map them to standards-based NFRs:

- List of standards-based NFR types (or system quality NFRs)
- Standards-based templates of NFR models and functions to identify which are needed within a specific system project

Note Some project-specific system-NFR needs may not be covered in the above Guides. Such needs must be identified and documented in addition to standards-based system-NFRs.

Step 2. Hardware, software, and procedure allocation

At the system design level, the system-NFR functions and sub functions need to be allocated to:

- Software functional level (within a software architecture)
- Hardware level
- Procedure level (e.g., instructions, training products)

Note The hardware functional level and the procedure level are beyond the scope of this guide and will not be further discussed.

Step 3. Specification of software functional requirements using functions in international standards

The system-NFR functions selected in Step 1 and allocated to software within a software architecture in Step 2 are then specified in terms of software functions. To this end, the relevant software functions and sub-functions are:

a) Identified and located within the software architecture

b) Specified within the software architecture
   - Ideally, including the identification of all required data movements of the necessary data groups
   - Otherwise, assumptions may be made about the number and sizes of functions and sub-functions (see the example in Section 4).

Step 4. Software size measurement (or approximation)

The NFR functions allocated to software can be measured using COSMIC measurement rules and guidelines.

Note 1 An estimate of size can be made using COSMIC guidelines for approximation when enough details are not present in software functions for accurate measurement.

Note 2 Not all system-NFRs lead to allocation of software-FURs. Examples for each NFR type with functionalities that may or may not be allocated to software functions are provided in Chapter 5 of the COSMIC Expert's Guide on Sizing NFRs [2].

3. EARLY SIZING OF SOFTWARE-FURS

When not all of the functional details are available, it is challenging to use any ISO measurement standard with high accuracy. To address this need for early sizing in the context of incomplete requirements, organizations such as the COSMIC Group have developed early sizing techniques to approximate software functional size.

The COSMIC Group proposes a number of early software sizing techniques, some of which are applicable at the feasibility stage, and others at the early requirements stage. Figure 2 shows COSMIC sizing techniques available as a project progresses.

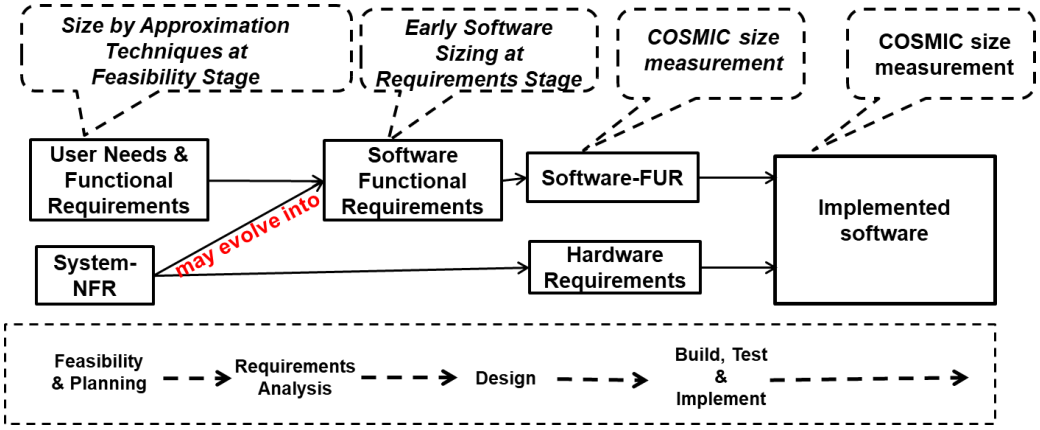


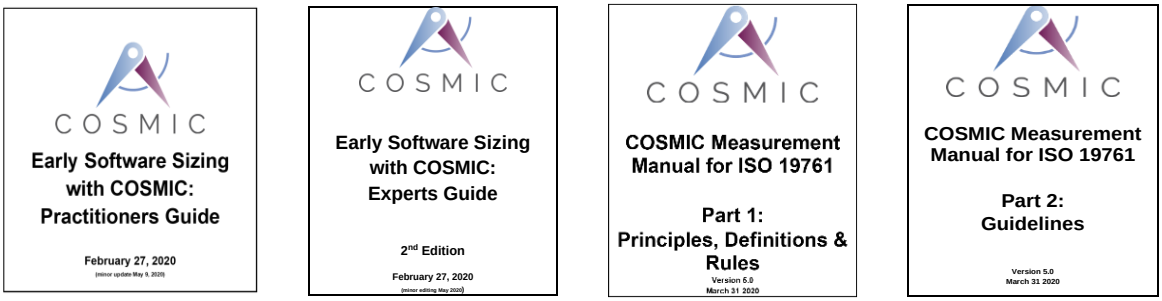
Figure 2. COSMIC sizing of software-FURs as a project progresses.

The following size approximation techniques are described in the 'Practitioners Guide' and 'Experts Guide' of the 'Early Software Sizing with COSMIC' documents [3, 4]:

A) For the feasibility stage:
- Early & Quick Approximation
- Software Iceberg Analogy
- Easy COSMIC Function Points

B) For the early software requirements stage:
- Average Size of Functional Processes
- Fixed Size Classification
- Equal Size Bands
- Average Use Cases

C) For the stages after requirements up to project completion:
Functional size measurement in compliance with the ISO 19761 standard rules and guidance provided by the COSMIC Measurement Manual [8, 9, 10].



4. EXAMPLE: FROM SYSTEM PERFORMANCE NFRS TO SOFTWARE FUNCTIONS

This section presents an illustrative example of the application of the four-step process to System Performance NFRs, where some performance functions are allocated to software functions.

Step 1 - System level: Stakeholder needs for system performance

Table 1 presents a number of system-NFR types in ISO, IEEE and ECSS [6] standards. This table can be used by stakeholders to identify the system-NFRs that are relevant to their own project. Table 2 illustrates a selection of three types of system-NFRs selected by stakeholders (in yellow): Portability, Performance and Security.

Table 1. NFR types in standards.

Id.	NFR	Standards used	Stakeholder needs
1	Reliability	ECSS-ISO-IEEE	
2	Maintainability	ECSS-ISO-IEEE	
3	Interfaces	ECSS-ISO-IEEE	
4	Portability	ECSS-ISO-IEEE	
5	Operations	ECSS-ISO-IEEE	
6	Configurations	ECSS-ISO-IEEE	
7	Data definitions & Database systems	ECSS-ISO-IEEE	
8	Adaptation and Installation	ECSS-ISO-IEEE	
9	Design & Implementation	ECSS-ISO-IEEE	
10	Performance	ECSS-ISO-IEEE	
11	Security	ECSS-ISO-IEEE	
12	Safety	ECSS-ISO-IEEE	
13	Resources	ECSS-ISO-IEEE	
14	Human factors	ECSS-ISO-IEEE	
15	Others….	In house	
16	Others…..	In house	

Table 2. A selection of 3 system-NFR.

Id.	NFR	Standards used	Stakeholder needs
1	Reliability	ECSS-ISO-IEEE	
2	Maintainability	ECSS-ISO-IEEE	
3	Interfaces	ECSS-ISO-IEEE	
4	Portability	ECSS-ISO-IEEE	X
5	Operations	ECSS-ISO-IEEE	
6	Configurations	ECSS-ISO-IEEE	
7	Data definitions & Database systems	ECSS-ISO-IEEE	
8	Adaptation & Installation	ECSS-ISO-IEEE	
9	Design & Implementation	ECSS-ISO-IEEE	
10	Performance	ECSS-ISO-IEEE	X
11	Security	ECSS-ISO-IEEE	X
12	Safety	ECSS-ISO-IEEE	
13	Resources	ECSS-ISO-IEEE	
14	Human factors	ECSS-ISO-IEEE	
15	Others….	In house	
16	Others…..	In house	

Note Within your organization, this and other templates can be expanded with columns for additional information with criteria associated with the NFRs, such as expected impact (high, medium, low), and the system architect's choice of a solution.

Step 2 – Hardware-software allocation

A template that can be used to allocate system-NFRs to hardware, software or administrative procedures is shown as Table 3.

Table 3. System-NFR allocation template.

Id.	NFR	Standards used	Stakeholder needs	To be allocated to hardware	To be allocated to software	To be allocated to admin. procedures
1	Reliability	ECSS-ISO-IEEE				
2	Maintainability	ECSS-ISO-IEEE				
3	Interfaces	ECSS-ISO-IEEE				
4	Portability	ECSS-ISO-IEEE				
5	Operations	ECSS-ISO-IEEE				
6	Configurations	ECSS-ISO-IEEE				
7	Data definitions & Database	ECSS-ISO-IEEE				
8	Adaptation & Installation	ECSS-ISO-IEEE				
9	Design & Implementation	ECSS-ISO-IEEE				
10	Performance	ECSS-ISO-IEEE				
11	Security	ECSS-ISO-IEEE				
12	Safety	ECSS-ISO-IEEE				
13	Resources	ECSS-ISO-IEEE				
14	Human factors	ECSS-ISO-IEEE				
15	Others….	In house				

Table 4 illustrates how the template in Table 3 is used for the case example under discussion.

Table 4. Completed System-NFR allocation template for the example.

Id.	NFR	Standards used	Stakeholder needs	To be allocated to hardware	To be allocated to software	To be allocated to admin. procedures
1	Reliability	ECSS-ISO-IEEE				
2	Maintainability	ECSS-ISO-IEEE				
3	Interfaces	ECSS-ISO-IEEE				
4	Portability	ECSS-ISO-IEEE	X	X		
5	Operations	ECSS-ISO-IEEE				
6	Configurations	ECSS-ISO-IEEE				
7	Data definitions and Database systems	ECSS-ISO-IEEE				
8	Adaptation & Installation	ECSS-ISO-IEEE				
9	Design & Implementation	ECSS-ISO-IEEE				
10	Performance	ECSS-ISO-IEEE	X	X	X	X
11	Security	ECSS-ISO-IEEE	X	X		
12	Safety	ECSS-ISO-IEEE				
13	Resources	ECSS-ISO-IEEE				
14	Human factors	ECSS-ISO-IEEE				
15	Others….	In house				
16	Others…..	In house				

Note Not all performance functions are allocated to software: Table 4 shows some are allocated to hardware, such as faster CPU, faster access to cache storage, etc., and an additional one is allocated to an administrative procedure.

Table 5 presents a template of a more detailed view of the standards-based model of system-performance-NFR. This NFR model consists of both dynamic and static performance requirements, related NFR sub-models indicators and the necessary NFR functions.

Table 5. System performance NFR template: Model, sub-models and functions.

NFR Model	NFR sub-models	NFR Functions
System Performance Dynamic Requirements (SPDR)	Throughput Time (TT),	[1] Bandwidth Function (BF) [2] Workload Function (WF)
	Response To Reference Signals (RRS)	[3] Response Time Function (RTF), [4] Settling Time Function (STF) [5] Tracking Error Function (TEF)
System Performance Static Requirements (SPSR)	Resource Consumption (RC)	[6] Main Memory Time Function (MMTF)
	Evaluation of accuracy (EA) - composed of • performance error (PE) • knowledge error (KE).	[7] Absolute Performance Error Function (APEF) [8] Performance Stability Error Function (PSEF)
	Evaluation of processing speed (EPS)	[9] System Scalability Function (SSF) [10] Concurrency Function (CF)

For this example, only system performance static requirements are allocated to software. Table 6 illustrates the specific allocation of the related sub-models and NFR functions.

Table 6. System performance indicators and functions allocated to software (highlighted in yellow).

NFR Model	NFR sub-models	NFR Functions
System Performance Dynamic Requirements (SPDR)	Throughput Time (TT),	[1] Bandwidth Function (BF) [2] Workload Function (WF)
	Response To Reference Signals (RRS)	[3] Response Time Function (RTF), [4] Settling Time Function (STF) [5] Tracking Error Function (TEF)
System Performance Static Requirements (SPSR)	Resource Consumption (RC)	[6] Main Memory Time Function (MMTF),
	Evaluation of accuracy (EA) - composed of • performance error (PE) • knowledge error (KE).	[7] Absolute Performance Error Function (APEF) [8] Performance Stability Error Function (PSEF).
	Evaluation of processing speed (EPS)	[9] System Scalability Function (SSF) [10] Concurrency Function (CF).

Note 1 Retrieving the data for calculating the performance error (PE) sub-model and the performance stability error (PSEF), and printing them in a performance report, is carried out by software for performance reporting and monitoring purposes.

Note 2 As Table 4 indicates, there may be additional related system performance functions allocated to hardware.

Step 3 – Software functional requirements – Specification in an SOA context

This section illustrates the distribution of software-FURs within a service oriented architecture (SOA), where some functions are allocated to a functional process within the application itself (Application A in Figure 3), and others to functional processes at the service level (Service S in Figure 3) [4, 7].

This is an example of data movements at the application and service levels in an SOA architecture. It assumes that there is a single function that interacts with a single service (see Figure 3).

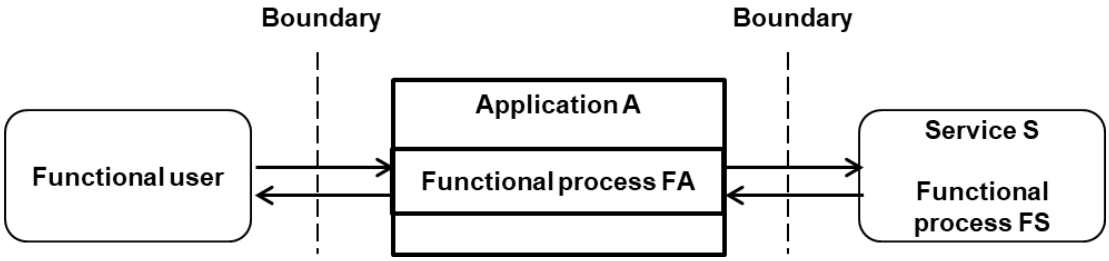


Figure 3. Data movements of a software functional process FA of Application A interacting with functional process FS in service S in an SOA architecture [7].

Table 7 presents the system performance NFR-functions 6 to 10 allocated to software functions in an SOA:

- Functions 6, 9 and 10 are allocated at the application level.
- Functions 7 and 8 are allocated at the SOA service level.

Table 7. Specifications at the application & service level within an SOA architecture.

Performance NFR Model	NFR sub-model	NFR Functions	Application level	SOA service level
System Performance Dynamic Requirements (SPDR)	Throughput Time (TT),	[1] Bandwidth Function (BF) [2] Workload Function (WF)		
	Response To Reference Signals (RRS)	[3] Response Time Function (RTF), [4] Settling Time Function (STF) [5] Tracking Error Function (TEF)		
System Performance Static Requirements (SPSR)	Resource Consumption (RC)	[6] Main Memory Time Function (MMTF),	X	
	Evaluation of accuracy (EA) - composed of • performance error (PE) • knowledge error (KE).	[7] Absolute Performance Error Function (APEF) [8] Performance Stability Error Function (PSEF).		X X
	Evaluation of processing speed (EPS)	[9] System Scalability Function (SSF) [10] Concurrency Function (CF).	X X	

Note The performance functions allocated to hardware, software and procedures in Table 4, and the additional reporting allocated to report performance sub-models, such as PE, APEF, PSEF, etc. as indicated in the note to Table 6, can be sized from a functional perspective with COSMIC.

Step 4 – Software size measurement (or approximation)

Functional size of all components can be measured using COSMIC measurement rules and guidelines.

Ignoring specific details of the measurement, the sizes for various software components for the example case in Table 7 are presented in Table 8:

- Size at the function level (functions 6, 9 & 10) = 10 CFP + 14 CFP + 6 CFP = 30 CFP
- Size at the service level (functions 7 & 8) = 12 CFP + 8 CFP = 20 CFP

The total functional size of the system static performance NFRs allocated to software in this example = 50 CFP.

Note In this example, other system performance NFRs have been allocated to hardware (Table 4).

Table 8. COSMIC sizing of system performance NFRs allocated to software.

Performance NFR Model	NFR sub-models	NFR Functions	Application - Size in in CFP	Service – Size in in CFP
System Performance Dynamic Requirements (SPDR)	Throughput Time (TT),	[1] Bandwidth Function (BF) [2] Workload Function (WF)		
	Response to Reference Signals (RRS)	[3] Response Time Function (RTF) [4] Settling Time Function (STF) [5] Tracking Error Function (TEF)		
System Performance Static Requirements (SPSR)	Resource Consumption (RC)	[6] Main Memory Time Function (MMTF)	10 CFP	
	Evaluation of accuracy (EA): which is composed of • performance error (PE) • knowledge error (KE).	[7] Absolute Performance Error Function (APEF) [8] Performance Stability Error Function (PSEF)		12 CFP 8 CFP
	Evaluation of processing speed (EPS)	[9] System Scalability Function (SSF) [10] Concurrency Function (CF).	14 CFP 6 CFP	

5. DEALING WITH NFRS IN PROJECT ESTIMATING

An estimating process can proceed as follows.

5.1 Estimating project effort from an outline of software-FURs & system-NFRs

The process for estimating project effort very early in a project is largely independent of how the project is managed once underway:

a) When developing the initial outline or 'high-level' statement of requirements for the software to be developed or modified, the focus is on the functional requirements: What must the software do, what constraints are there on targets (budget, delivery date, etc.) and what risks must be managed.

b) System-NFRs may also be stated by stakeholders at estimation time.

c) If a software project effort estimate is needed at this stage, it can only be made using estimation techniques relevant for the feasibility stage, including a contingency allowance.

   Note 'scope creep' is not included. Scope creep is the number one reason for project failures and should be handled as change requests.

d) The outline functional requirements and system-NFRs feed the system/software architecture decisions that help identify the system-NFRs, which evolve into software-FURs.

5.2 Estimating project effort & cost from approximate software-FURs & system-NFRs

a) When the requirements analysis phase is in its early stages, it is unlikely that the software functional requirements are detailed enough. Here, the functional size may be approximated using a COSMIC early sizing technique as described in [1,2].

   Note Software-FURs are unlikely to be detailed enough for less mature organizations. More mature organizations are more able to do an acceptable requirements engineering process.

b) As the project progresses, a contribution to the total software functional size may be added to the system-NFRs that will evolve into software-FURs. This contribution may be made by applying the experience gained in previous projects with regard to sizing system-NFRs allocated to software-FURs, and by adding a contingency.

c) The approximated total functional size may be converted to estimated project effort using benchmark data established for projects with a functional size and NFR profile as close as possible to that of the new project. This 'closeness of fit' may be determined by analogy or by statistical analysis of past completed projects within the organization, which of course will vary from one company to the next.

d) When size and effort data from previous projects is available, the effort for new projects can be estimated with the help of regression analysis.

Figure 4 shows the various ways system-NFRs may, as they evolve, contribute to the total functional size and/or may affect the benchmark figure used to convert size to effort and then to cost. For example,

- The benchmark figure used will probably depend on the programming language, the hardware platform, etc.;
- NFRs for significant re-use of existing code may also affect the effort calculation.

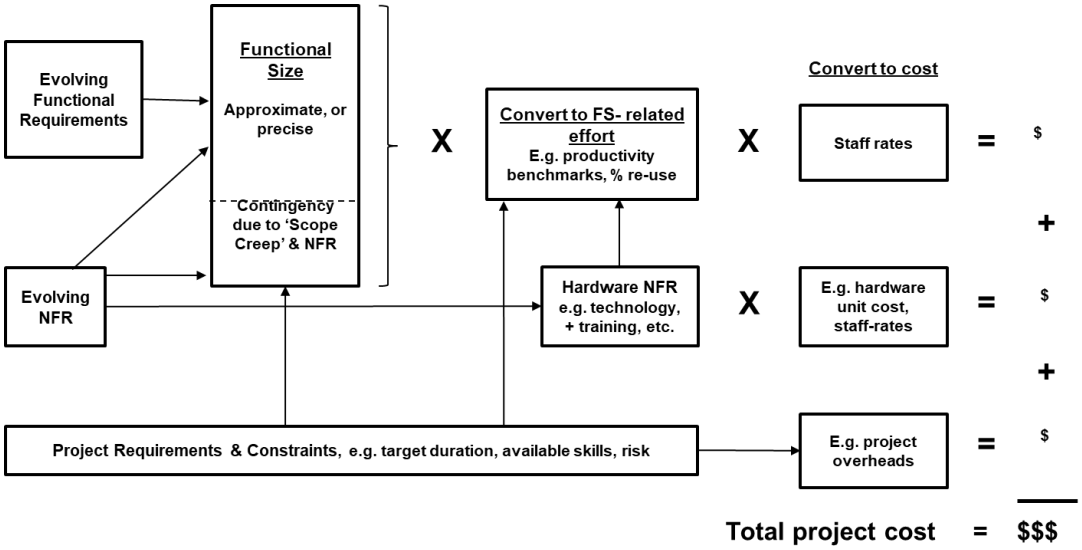


Figure 4. Determination of project effort and cost.

The model for estimating project costs in Figure 4 can be applied as soon as an approximate functional size is known and any time thereafter.

Note The estimation process itself is independent of the development process, e.g. waterfall versus agile. Productivity, on the other hand, may vary across development processes and organizations.

6. REFERENCES

(COSMIC publications are all available for free at www.cosmic-sizing.org).

[1] COSMIC, 'Repository of standards-based Non-Functional Requirements (NFR)', in progress, mid 2021.

[2] COSMIC, 'Non-Functional & Project Requirements with COSMIC: Experts Guide', v2.0, 2020.

[3] COSMIC, 'Early Software Sizing with COSMIC: Practitioners Guide', 2020.

[4] COSMIC, 'Early Software Sizing with COSMIC: Experts Guide', 2020.

[5] K.T. Al-Sarayreh, K. Meridji, A. Abran, S. Trudel, 'System performance requirements: A Standards-based model for early identification, allocation to software functions and their size measurement', submitted to e-Information Software Engineering journal.

[6] European Cooperation for Space Standardization, 'Space Engineering: Software – Part 1 - Principles and Requirements, The Netherlands, 2005.

[7] COSMIC 'Guideline for sizing Service-Oriented Architecture software', COSMIC method v4.0.2, 2019.

[8] COSMIC 'Measurement Manual v5.0: Part 1: Principles, Definitions and Rules', May 2020.

[9] COSMIC 'Measurement Manual v5.0: Part 2: Guidelines', May 2020.

[10] COSMIC 'Measurement Manual v5.0: Part 3: Examples', May 2020.

APPENDIX: MODELS, SUB-MODELS & FUNCTIONS IN STANDARDS-BASED NFRS

A- List of System-NFRs

Id.	NFR	Standards used
1	Performance	ECSS-ISO-IEEE
2	Maintainability	ECSS-ISO-IEEE
3	Portability	ECSS-ISO-IEEE
4	Security	ECSS-ISO-IEEE
5	Reliability	ECSS-ISO-IEEE
6	Interfaces	ECSS-ISO-IEEE
7	Operations	ECSS-ISO-IEEE
8	Adaptation & Installation	ECSS-ISO-IEEE
9	Safety	ECSS-ISO-IEEE
10	Resources	ECSS-ISO-IEEE
11	Human Factors	ECSS-ISO-IEEE
12	Data Definition & Data Bases	ECSS-ISO-IEEE
13	Configuration	ECSS-ISO-IEEE
14	Design	ECSS-ISO-IEEE

B. Models for NFRs

ID	System Performance Requirements		
	Model(s)	**Sub-Models**	**Functions**
1	System Performance Dynamic Requirements (SPDR)	Throughput Time (TT),	[1] Bandwidth Function (BF) [2] Workload Function (WF)
		Response To Reference Signals (RRS)	[3] Response Time Function (RTF), [4] Settling Time Function (STF) [5] Tracking Error Function (TEF)
	System Performance Static Requirements (SPSR)	Resource Consumption (RC)	[6] Main Memory Time Function (MMTF), [7] Storage Device Time Function (SDTF) [8] Processor Instruction Execution Function (PIEF).
		Evaluation of accuracy (EA) with: • performance error (PE) • knowledge error (KE).	[9] Absolute Performance Error Function (APEF) [10] Performance Stability Error Function (PSEF). [11] Absolute Knowledge Error Function (AKEF) [12] Relative Knowledge Error Function (RKEF).
		Evaluation of processing speed (EPS)	[13] System Scalability Function (SSF) [14] Concurrency Function (CF).

ID	System Maintainability Requirements		
	Model	**Sub-models**	**Functions**
2	System maintainability procedures (SMP)	System maintainability failure procedure (SMFP)	[1] System diagnostic function (SDF) [2] Failure data operation function (FDOF) [3] Failure data monitoring function (FDMF) [4] Failure data control function (FDCF) [5] System failure task function (SFTF)
		System registered failure procedure (SRFP)	[6] Failure detection function (FDF) [7] Failure isolation function (FIF)
		System malfunction procedure (SMP)	[8] Correct data faults function (CDFF) [9] Correct system defects function (CSDF)
		System stability procedure (SSP)	[10]Fault prevention of data control function (FPDCF) [11]Fault prevention of system function (FPSF)
		System testability procedure (STP)	[12]System time function (STF) [13]Fault allocation time function (FATF)

ID	System Portability Requirements		
	Model	**Sub-model**	**Functions**
3	System Portability Environment (SPE)	Software Component Portability	[1] Independence of the Operating System Function (IOSF) [2] Independence of the Middleware Function (IMF) [3] Independence of the Programming Language Virtual Machine Function (IPLVMF) [4] Independence of the Browser Function (IBF)
		Data Component Portability	[5] Independence of the Database Function (IDF) [6] Distributed Data Base Management System Function (DDBMSF)
		Hardware Component Portability	[7] Independence of the Client Function (ICF) [8] Independence of the Server Function (ISF) [9] Independence of the Storage Function (ISTF) [10]Independence of the Network Function (INF)
		Isolation of System Calls Portability	[11] Isolation of Software System Calls Function (ISSCF)

ID	System Security Requirements		
	Model	**Sub-models**	**Functions**
4	System Security Environment (SSE)	System confidentiality	[1] Identification function [2] Authentication function [3] Authorization function
		System availability	[4] Network redundancy function [5] Power redundancy function [6] Automatic restart function
		System integrity	[7] Backup data function [8] Firewall function [9] Antivirus function [10]External PKI function [11]Encryption\decryption function

ID	System Reliability Requirements		
	Model	**Sub-models**	**Functions**
5	System Reliability Environment (SRE)	System Failure mode and effect analysis (FMEA)	[1] Operational failure function (OPEF) [2] Failure mechanism function (FPF)
		System Fault tree analysis (FTA)	[3] Fault prevention function (FPF) [4] Fault detection function (FDF) [5] Fault removal function (FRF)
		System Reliability hazard analysis (RHA)	[6] Error in handling input function (EHIF) [7] Error in producing output function (EPOF) [8] Error in producing correct output function (EPCOF)

ID	System Interface Requirements		
	Model	**Sub-models**	**Functions**
6	User Functional Interfaces (UFI)	Control software interfaces	[1] External system Interface for applications Function [2] Internal system interface applications for each layer socket Function [3] System Interface configurationFunction
		Control Hardware interfaces	[4] Physical Interfaces Function [5] Thermal Interfaces Function [6] Electrical Interfaces Function
		Interface communicationsLinks	[7] External interfaces DataCommunications Function [8] Internal interfaces communications Links Function

ID	SystemOperations R equirements		
	Model	**Sub-models**	**Functions**
7	System Operations Environment (SOE)	System operations mode	[1] Inter-operational function (IOPF) [2] Operational function event (OPFE)
		System transitions mode	[3] Operational data interface function (OPDIF) [4] Operational control interface function (OPCIF)

ID	System Adaptation & Installation Requirements		
	Model	**Sub-models**	**Functions**
8	System Adaptation & installation Environment (SOE)	System Software Environment (SSE)	[1] Software Data Structure Function (SDSF) [2] Registered Data Transfer Function (RDTF) [3] Control Data Transfer Function (CDTF) [4] Set Data Transfer with System Resources Function (SDTF)
		System Integrated Environment (SIE)	[5] Operational Environment Function (OPEF) [6] Localizing I/O Resources Function (IORF)
		System Hardware Environment (SHE)	[7] Host-Target Platform Function (HTPF) [8] Memory Resources Function (MRF) [9] Storage Resources Function (SRF) [10]Transmission Resources Function (TRF)

ID	System Safety Requirements		
	Model	**Sub-models**	**Functions**
9	System Safety Environment (SSE)	Control system hazards	[1] Software operation risk function [2] Software design risk function [3] Software configuration risk function [4] System loss operation function [5] System failure detection function [6] System failure isolation function
		Critical system catastrophic	[7] System safety audit function [8] System redundancy status function

ID	System Resources Requirements		
	Model	**Sub-models**	**Functions**
10	System Resources Environment (SSE)	I/O resource addresses	[1] I/O port addresses function [2] I/O resource list function [3] I/O resource addresses function [4] I/O transmission addresses function [5] Block of bus relative memory addresses function
		Hardware resources	[6] Processor capacity for software item function [7] Memory capacity for software item function [8] Storage device capacity for software item function [9] Interrupt vectors function

ID	System Human Factors Requirements		
	Model	**Sub-models**	**Functions**
11	System Human Factors Environment (SHEE)	Cognitive ergonomics (CE) (performance of human factors)	[1] Human capabilities function (HCF) [2] Training function (TF) [3] Staffing function (SF) [4] Personal selection function (PSF)
		Environmental ergonomics (EE) (Safety of human factors)	[5] Mechanical safety function (MSF) [6] Electrical safety function (ESF) [7] Operational safety function (OSF) [8] Psychology and physiological safety function (PSF) [9] Environmental safety function (ESF)
		Human interface factors (HIF)	[10] Interface characteristics and task performance function (ICTPF) [11] Interface customization function (ICF) [12] Identification of safety related controls function (ISRCF)

ID	System Data Definitions & DataBase Requirements		
	Model	**Sub-models**	**Functions**
12	System Product Data (SPD)	System entity types (SET)	[1] Function to identify event (EF) [2] Function to identify parameter (PF) [3] Function to identify system element (SEF) [4] Function to identify reporting data (RDF) [5] Function to identify activity (AF)
		System value types (SVT)	[6] Function to identify simple value (SVF) [7] Function to identify record value (RVF)
		System data types (SDT)	[8] Function to identify simple type (STF) [9] Function to identify complex type (CTF)
		System data items (SDI)	[10] Function to identify configuration data (SCDF) [11] Function to identify monitoring data (SMDF) [12] Function to identify control data (SCDF1)