COSMIC

Non-Functional & Project Requirements with COSMIC: Experts Guide

Version 2.

September 2020

Version control

Date	Reviewer(s)	Modifications / Additions
September 2020	See Acknowledgements	Minor editing

Acknowledgements

Authors and reviewers who have contributed to the development of v1.0 (alphabetical order)		
Alain Abran, École de Technologie Supérieure, Université du Québec, Canada	Diana Baklizky TI Metricas, Brazil	Carol Buttle Safety and Security Engineering Council, United Kingdom
Jean-Marc Desharnais École de Technologie Supérieure, Université du Québec, Canada	Peter Fagg Pentad Ltd, United Kingdom	Cigdem Gencel DEISER, Spain
Barbara Kitchenham, Keele University, United Kingdom	Arlan Lesterhuis The Netherlands	Roberto Meli Data Processing Organization, Italy
Dylan Ren Measures, China	Luca Santillo Agile Metrics, Italy	Hassan Soubra German University in Cairo, Egypt
Charles Symons*, United Kingdom	Frank Vogelezang METRI, Netherlands	Steve Woodward Canada

* Author of this guideline

Foreword

The COSMIC method aims to measure a 'functional size' of software based on its Functional User Requirements (FUR). In simple terms these specify 'what' a software product must do.

The main uses of COSMIC-measured 'functional sizes' are in:

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

This apparently simple process may be useful in practice because the 'functional sizes' are usually by far the largest driver of effort of software development projects. However, the success of this simple process depends heavily on what is meant by 'similar'.

Clearly many other factors than just the size of the functional requirements can affect project performance and may need to be taken into account in order to ensure fair, or 'like-for-like', comparisons. These same other factors may arise when estimating the project effort to develop some new software. Examples of such 'other factors' are:

- varying ease-of-use or system response time requirements (system quality factors);
- varying numbers of users that the system must serve (environmental factors)
- varying requirements for the technology to be used or for the technical architecture (technical factors);
- varying skill-levels of the project teams or project constraints such as schedule compression factors (project factors).

The Guideline begins by referring to these various system quality, environmental and technical requirements and constraints as 'non-functional requirements', abbreviated as 'NFR', and 'project requirements and constraints' abbreviated as 'PRC'.

In the late 1970's when functional size measurement was first invented, few NFR were recognized and they did not vary much for all the projects within a given company, or even within the same domain e.g. of business applications. The first methods of functional sizing attempted to account for a few NFR by an adjustment to the functional size. For example, Albrecht's 'Value Adjustment Factor' for FPA recognized 10 factors, later increased to 14 factors by IFPUG.

Since then, many more types of requirements are recognized as non-functional. With the enormous varieties of technology and software, taking them into account in the activities of project performance measurement, benchmarking and estimating can be much more difficult.

The purposes of this Guideline are

- to help understand and define the concepts of NFR and PRC;
- to propose a standard Glossary of terms and classification system for NFR and PRC;
- to provide practical guidance to users of the COSMIC FSM method on how to deal with NFR and PRC, as well as FUR, when making software project performance measurement comparisons, when developing benchmarks, and when estimating for new projects.

The focus of the Guideline is on the NFR for the software deliverables of a project and on the PRC. (A project may also have to deliver other related products such as the hardware on which the software executes, business processes and training for human users of the software, and such-like, but these are only considered in passing.)

The Guideline is structured as follows.

- Chapter 1 introduces the need for a coherent terminology and for methods of measuring or recording FUR, NFR and PRC consistently across the activities of software system project performance measurement, benchmarking and estimating, and starts to discuss how FUR, NFR and PRC affect measured software size and project effort.
- Chapter 2 introduces a coherent set of definitions of all types of requirements, in a hierarchical structure.
- Chapter 3 introduces a classification system for NFR and PRC and gives the criteria for deciding which NFR and PRC terms were included in the Glossary. The classification system should also help understanding and make it easier to search for a particular NFR or PRC term. The classification and lists of NFR and PRC terms should be valuable as a checklist when defining requirements for a new software project.
- Chapter 4 aims to give practical advice on how to deal with NFR and PRC in recording project data, comparing project performance, establishing internal benchmarks and estimating for new projects.
- Chapter 5 gives examples of quality requirements for a software system or product that initially appear as non-functional, but that evolve as a project progresses to requirements for software functionality. Most such functionality can be measured by the standard COSMIC method.
- Chapter 6 (to be written mostly in a later release) introduces standard ways of recording and measuring each of the NFR and PRC terms.
- The Glossary of Chapter 7 lists NFR and PRC terms and their definitions, selected using the criteria of Chapter 3.

The reader is assumed to have a general understanding of functional size measurement and of the COSMIC method. Much information about COSMIC and all documentation on the COSMIC method is available for free download from www.cosmic-sizing.org.

The COSMIC Measurement Practices Committee.

Table of Contents

Acronyms

The following acronyms are used in this Glossary

CMMI®	Capability Maturity Model Integration
COBIT	Control Objectives for Information and Related Technology, http://www.isaca.org/cobit/pages/default.aspx
COSMIC	Common Software Measurement International Consortium
FSM	Functional Size Measurement
FUR	Functional User Requirements
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electrotechnical Commission
IFPUG	International Function Point Users Group
ISBSG	International Software Benchmarking Standards Group
ISO	International Organization for Standardization
NFR	Non-Functional Requirements
PMI®	Project Management Institute
PRC	Project Requirements and Constraints
PRINCE2	PRojects IN a Controlled Environment, Version 2
ROI	Return on Investment
SPICE	Software Process Improvement and Capability Determination
SQuaRE	System and software product Quality Requirements and Evaluation

1

INTRODUCTION AND TERMINOLOGY

The purpose of this Guideline is to establish common understanding and terminology across the activities of software sizing, software project performance measurement, benchmarking and software project estimating.

The common subject linking these four activities are projects that develop or enhance software-intensive 'systems' (comprising software, hardware, data and maybe other deliverables) or just software products. For simplicity, when we do not need to be more precise, we will refer to all of these as 'software system projects' and their output as 'software systems or software products' (shortened to 'software system/product' where convenient).

1.1 The need for coherent and consistent data across four activities

Figure 1.1 shows the dependencies across the four activities of recording and measuring requirements for a software system project, deriving project performance measures, using these to develop project benchmarks and estimating for a new project based on its requirements and comparable benchmark data from previous projects (broad arrows indicate dependency of activities).

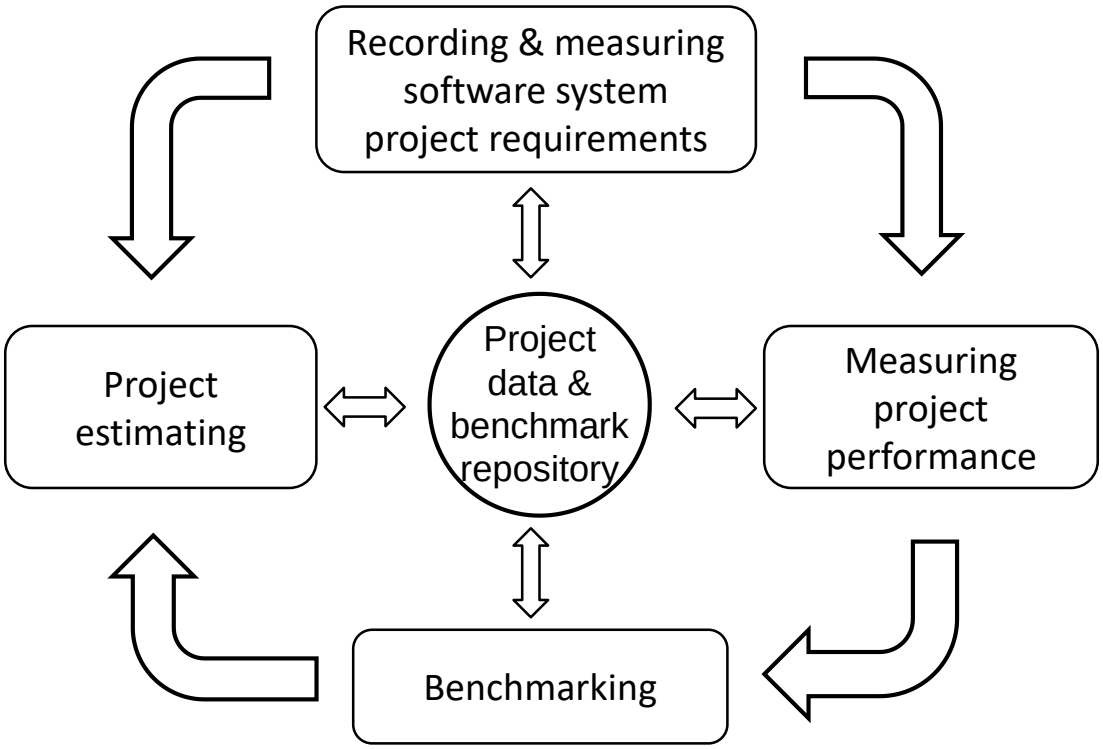

Figure 1.1 - Inter-relationships of four activities that require coherent and consistent data and terminology

We need coherent terminology to be used consistently across these four activities because typically:

- measures of the size of software system/product requirements are used with project effort data to produce project performance measures;
- these size and effort data are recorded together with the other characteristics of the project and of the software system/product that are needed to ensure like-for-like performance comparisons in a central project data repository;
- project performance measures are used to derive benchmark data for projects which are also classified according to their most significant project and software system/product characteristics;
- when an estimate of effort is needed for a new project, the software size is measured or estimated from its requirements and is combined with benchmark data for projects that had similar characteristics to produce the new project effort estimate,

To fully understand these four activities, we must first recognize that a software system project has to consider three types of requirements, which affect the software size and the project effort in different ways:

- Functional User Requirements (FUR) may be roughly defined as 'what the software must do'. They clearly affect the software size, which in turn affects project effort.
- Non-Functional Requirements (NFR) which are sometimes defined as 'how the software must do it'. Whether or not NFR affect software size is not immediately clear; they certainly affect project effort.
- Project Requirements and Constraints (PRC). These clearly affect project effort directly but do not affect software size.

As regards measurements, in practice the only measures of software size that are used consistently across these four activities are either counts of Source Lines of Code (SLOC) or measurements of the FUR by a 'Functional Size Measurement' (FSM) method such as the ISO standard COSMIC method.

SLOC size measures have an advantage that they measure a software size that is the result of meeting all the FUR and NFR, but they have so many disadvantages we will not consider them further. Conventionally, FSM Methods do not now measure NFR. (Past attempts to do so, e.g. IFPUG's 'Value Adjustment Factor' are now rarely used). In general, there are no widely-accepted standards on if and how NFR should be recorded and/or measured.

For project requirements and constraints, there are standards from benchmarking organizations such as the ISBSG that define how to measure or record the most important parameters.

Very commonly, a FSM method is used to measure a 'functional size' of the FUR, which is used as a measure of work output of a software system project. This approach can lead to satisfactory consistency across our four activities *provided* any NFR have a relatively low effect on project effort, or account for the same proportion of effort for all projects being studied. However, if NFR cause a high or varying amount of effort on the projects being studied, use of a functional size as the sole measure of project work-output may be unreliable

Consequently, to ensure coherence across our four activities, we would ideally use a consistent measure of functional size and we would measure, or at least record, NFR and PRC in a consistent way.

However, a survey [4] of terms from nine sources ([13] to [21] inclusive) that might be candidates for a Glossary of NFR and PRC terms relevant to these activities found that only one term ('Interfaces') was common to all nine sources (from 108 unique terms found in the nine sources). It seems that each source has defined what it thinks are the main NFR and PRC for its purpose (or has simply defined the NFR and PRC it can easily measure).

This lack of consistency of the factors to be considered across our four activities make it inherently difficult for an organization to decide what NFR and project data to capture for projects, along with the functional size of the software delivered, for use in project performance measurement and analysis, subsequent benchmarking and future project estimating activities.

1.2 Towards a coherent terminology

This section considers some basic terminology questions that must be properly understood before we can develop reliable definitions.

1.2.1 The relationship between 'requirements' and 'constraints'

The terms 'requirements' and 'constraints' are often used inter-changeably, which can be confusing.

In ordinary English, a requirement is a necessary condition whilst a constraint is simply a limiting condition. It follows that all requirements are constraints, but not all constraints are requirements. Figure 1.2 illustrates this difference with examples for NFR and PRC by means of a Venn diagram.

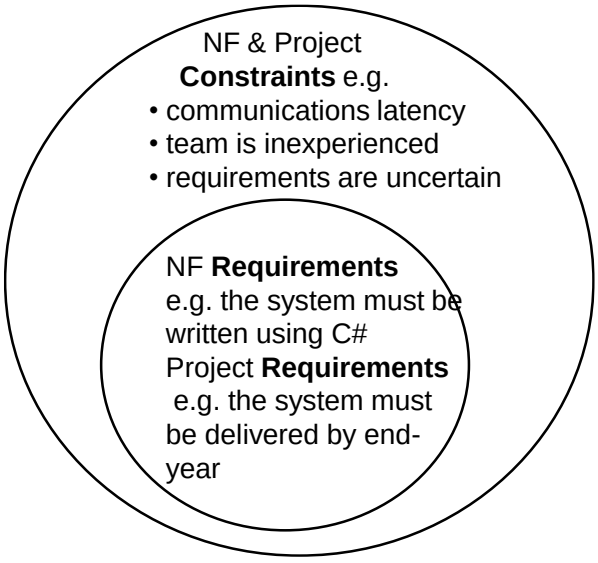

Figure 1.2 - The relationship between 'constraints' and 'requirements'

EXAMPLE 1: a requirement that the software shall be written in C# is also a constraint. But a situation where it happens that the requirements are uncertain and very difficult to establish, is a constraint, not a requirement.

EXAMPLE 2: Some terms can be either a constraint or a requirement depending on the context. Achieving a certain 'latency' target could be a requirement for real-time processing of audio signals, or latency could be a 'design constraint' for a space communications system.

Constraints that are not requirements are often only recognized after a project is completed, e.g. in a post-project review. The importance of this point is that if we are to understand project performance properly, then we must take into account project constraints that were not requirements.

EXAMPLE 3: A project might be measured as poorly-performing and a post-project review determined that this was due to the constraint that the team was inexperienced with the

technology used. (Conversely, if the team was very experienced with the technology, this should have been a positive factor for the project, not a 'constraint'.)

In this Guideline, for simplicity we will mostly use 'requirement' and only use 'constraint' when really needed. This is particularly necessary for 'project requirements and constraints'.

1.2.2 What do requirements apply to?

Requirements may apply to any of the following 'things', as shown in Figure 1.3 (in bold).

- The **software**; typically these are Functional User Requirements ('what the software must do') and software quality attributes, e.g. for usability, portability, maintainability, etc.;
- The **data** that the software system/product will maintain or use;
- The **technology** to be used, e.g. 'the system must execute on a Unix platform', the requirements must be captured using CASE tool XYZ', 'the software must be written in Java', etc.;
- **Other deliverables**, e.g. special documentation or training;
- The combined **hardware/software system**[1], e.g. a response time or an availability requirement will apply to the hardware/software system as a whole (not just to the software);
- The **project**, e.g. it must use a team with a particular skill-set; must be completed by the end of the year, must use an Agile approach, etc.;

In addition, the business or organizational **environment** may effectively impose certain requirements or constraints, e.g. the software must be developed off-shore in a specific country, or be subject to specific industry regulations, such as for banking or for safety, or will be used by a wide population.

The relationship between these various 'things to which requirements can apply is shown in Figure 1.3. (The 'crows-foot' symbol indicates that a project may develop and/or enhance one or more systems. Examples in italics are commonly thought of as 'Non-Functional Requirements')

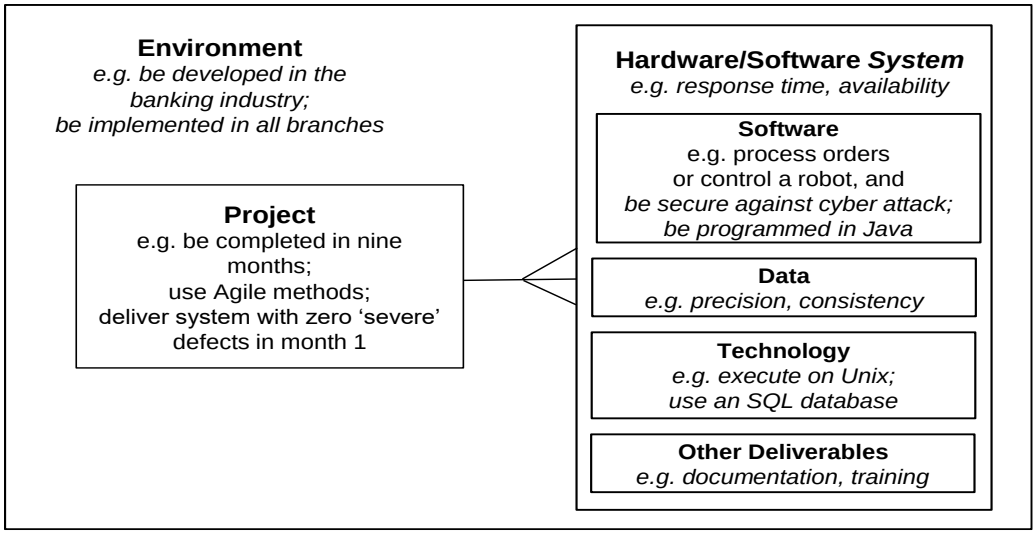

Figure 1.3 - The 'things' (in bold) that requirements apply to, and the Environment which may impose requirements.

[1] A 'system' may well be considered to include human business processes and support activities, or the totality of a machine that includes software (e.g. a vehicle 'power train' system), but in this Guideline we limit 'system' to mean a 'computer hardware-plus-software' system.

Two important observations from Figure 1.3:

- From their definition (see 2.1) Functional User Requirements apply only to the software.
- Other types of requirements (shown in italics in this diagram) that are commonly thought of as 'non-functional' can apply to the hardware/software system, to the software, to the data, to the technology and to other deliverables, or may be determined by the environment.

Note: The interest in considering the scope of the things to which an NFR can apply is that it helps us define the scope of what we will consider as an NFR. For the purpose of defining what we mean by a non-functional requirement, it does not matter which of the things are impacted by a particular NFR when it is implemented.

EXAMPLE 1: A requirement to use a particular programming language may apply to a particular piece of software or may be expressed as an organizational policy for all software system projects, and it is clearly a requirement of the technology to be used. Regardless, it is an NFR.

EXAMPLE 2: A Non-functional requirement for a three-layer architecture may apply only to the software or it may impact the hardware as well if the components must execute on different technical platforms. Either way, this NFR applies to the hardware/software system.

EXAMPLE 3: A Non-functional requirement for a particular response time might be achieved by using only fast hardware, or by using only a low-level programming language, or by a software architectural choice, or by a combination of hardware and software features. Regardless of how the NFR is implemented, the response time is normally measured only at the system level, not separately for the software or the hardware. It is clearly a system NFR.

There is one further limitation on what we will consider as a Non-functional requirement. The principal concern of this Guideline is the effect of NFR on the effort to deliver the software component of a hardware/software system or to deliver a software product. Hence when discussing NFR in Chapter 2 and their influence in practice for performance measurement, benchmarking and estimating in Chapter 3, we will not discuss NFR that apply exclusively to any other 'thing' that a project might have to deliver in addition to the software system/product. These 'Other deliverables' include documentation and training, (as in Fig. 2.1), and requirements for hardware to be acquired and installed, business processes to be established, etc. The variety of possible 'other deliverables' is potentially endless.

1.2.3 NFR often evolve into FUR as a project progresses

When first stated as 'high-level' (or 'outline') requirements, many cannot be classified clearly as FUR or as NFR – though PRC are usually clearly distinguishable. Some requirements may appear initially as non-functional but, as a project progresses and more of their details become known, it becomes clear that the requirement can be satisfied wholly or partly by software functionality. For simplicity in this Guideline, we say that such NFR may 'evolve'[2] at least partly into functional requirements for software [7].

The importance of this observation is that when a requirement that is initially stated as a Non-functional requirement evolves wholly or partly as a project progresses into FUR for software, the size of this 'additional' functionality can potentially be measured using the COSMIC method in the same way as any other requirement that was always a functional requirement for software from when it was first stated.

[2] Experienced software developers may know that such requirements will be satisfied by software functionality so never consider them as NFR in the first place. This, perhaps, explains why there is some confusion about distinguishing NFR and FUR.

Any part of a requirement originally stated as non-functional that does not evolve into FUR is referred to as a 'true' NFR in this Guideline

This evolution is illustrated in Figure 1.4 against the background of the software life-cycle. (Note: the software life-cycle illustrated in Fig. 1.4 may be part of a system life-cycle which may include activities to decide on the allocation of requirements to hardware or software. This aspect is not shown in Fig. 1.4.)

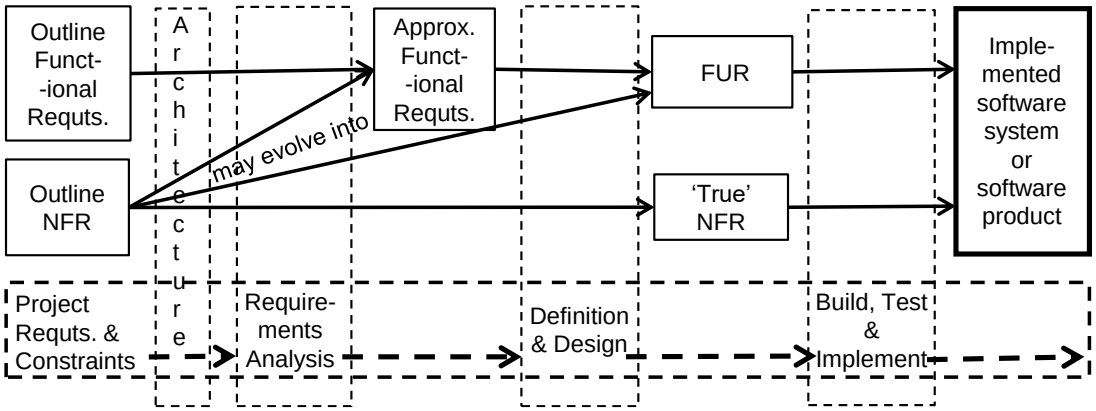

Fig. 1.4 - NFR may evolve, wholly or partly, into FUR as a project progresses

Examples will illustrate this evolution.

EXAMPLE 1: 'Maintainability' as defined in ISO 9126 and ISO 25000 series is a quality requirement and therefore an NFR according to the ISO 14143/1 definition of a Functional User Requirement (see below). But a personal computer (PC) supplier's requirement to be able to maintain its operating system software remotely on all its users' PC's may be implemented entirely in software, at least as seen by the PC user. When worked out in detail, this system NFR evolves almost entirely into FUR for PC software functionality. This leaves only the pure statement of the requirement to be able to maintain the operating system software remotely on all users' PC's remaining as a 'true' NFR.

EXAMPLE 2: In contrast to EXAMPLE 1, implementing a maintainability NFR for a space satellite system that must operate 24/7 will probably be achieved by a combination of multiple hardware processors (which might also be required for other reasons) as well as special software. Whatever hardware and/or software is chosen, the statement that 'the system must be maintainable 24/7' will remain a 'true' NFR of the hardware/software system.

EXAMPLE 3: A system response time requirement may evolve into and/or strongly influence the FUR for some specific software. But any part of the response time requirement that evolves into a requirement to use fast hardware, or, say, a low-level programming language, will remain a 'true' NFR. Similarly, the required response time statement (e.g. 'the response time during peak hour may not exceed one second') always remains a 'true' NFR for the combined hardware/software system'.

EXAMPLE 4: A requirement that an application must be readable by people with poor sight could be implemented in various ways. If the human user must be able to change the font size on screens used by the application, then this approach will have evolved into a FUR for the application. Alternatively if the application is to be implemented on a tablet with a touch-sensitive screen and built-in 'pinch/stretch' functionality, then there may be no impact at all for the application. Either way, the statement that the application must be usable by poor-sighted people remains a 'true' NFR.

1.3 Current practices in dealing with NFR in a system development project

Practices in specifying NFR seem to vary sharply with domain and with system and software development processes. The following are examples of differing practices.

In the domain of business application developments, in an organization following a '**waterfall'** development process, during the early part of the requirements phase, the focus is usually on capturing the software functional requirement. NFR may only be captured at a very global level. Specifying the NFR in more detail may be left until later, when the architecture and software design are considered.

In these later phases, NFR may be viewed, defined, interpreted, and evaluated differently by different people, particularly if they have been stated only briefly in the requirements phase. This creates problems for project management purposes and makes it challenging to take them into account in software estimation and software productivity benchmarking.

> *EXAMPLE: In a particular Bank's systems development department, many institutional NFR, such as for security, banking regulations, auditability, traceability, etc. were never stated in requirements documents. It was always assumed that everyone knew about these requirements. This could be quite risky.*

In organizations using the **Agile** development processes, it has been reported that the focus on delivering functionality can result in NFR that apply across the system being ignored until far too late in the project. This may arise due to the difficulty of allocating a Non-functional requirement to a single sprint or iteration.

> *EXAMPLE: A major European government project using Agile methods was stopped for a major re-think after two years of development when it was realized that security and privacy requirements had not been properly considered. These NFR had profound consequences for the system architecture.*

NFR become increasingly important when developing a 'smart system' or a **'system of systems**' (e.g. a system to enable automatic payment for use of car parking from portable phones), where architecture, portability and interface NFR become very important.

The common lesson from this experience is that NFR are as important as FUR, and should be taken into account early in the life of a new software system/product both for general efficiency and risk-control reasons.

NFR play a critical and therefore better-understood role in the development of **mission-critical and safety-critical systems**, e.g. systems for financial-market trading, air traffic control, major process plant control, etc. It has been reported [3] that in the Statements of Requirements sent to prospective suppliers of such systems, the NFR may account for 50% of the documentation. (It was also stated that most of the NFR are eventually allocated to and implemented as software.)

1.4 Summary and conclusions from this chapter

We have found that there is limited commonality in understanding and use of NFR and PRC across software project performance measurement, benchmarking and estimating activities. It is also not widely recognized that system non-functional requirements may evolve as a project progresses at least partly into software functional requirements. Further, in practice the importance of NFR is frequently not recognized and they are not specified early enough in software projects.

There is a need for a new coherent set of definitions of NFR and PRC, and for processes for how to deal with them consistently in software projects.

2

DEFINITIONS OF THE VARIOUS TYPES OF REQUIREMENTS

This section aims to give a coherent set of definitions for all the types of requirements that a software project may have to satisfy. All definitions in this chapter have been developed and agreed by COSMIC and IFPUG [29] except the ISO/IEC of FUR [1].

2.1 Functional User Requirements (FUR)

ISO/IEC 14143/1 DEFINITION – Functional User Requirements (FUR)
A sub-set of the user requirements. Requirements that describe what the software shall do, in terms of tasks and services. NOTE: Functional User Requirements relate to but are not limited to: • data transfer (for example Input customer data; Send control signal) • data transformation (for example Calculate bank interest; Derive average temperature) • data storage (for example Store customer order; Record ambient temperature over time) • data retrieval (for example List current employees; Retrieve latest aircraft position) Examples of user requirements that are not Functional User Requirements include but are not limited to: • quality constraints (for example usability, reliability, efficiency and portability) • organizational constraints (for example locations for operation, target hardware and compliance to standards) • environmental constraints (for example interoperability, security, privacy and safety) • implementation constraints (for example development language, delivery schedule)

NOTE: From v4.0.1 of the COSMIC method, we use the term ‘FUR’ to mean the functional user requirements that: contain all the information needed for a detailed COSMIC Functional Size Measurement.

Earlier in the life of a project, before that detail is known, we refer simply to ‘functional requirements’. (See Figure 1.4.)

2.2 Non-Functional Requirements (NFR)

There is no good accepted definition of NFR. The ISO 25756 definition for NFR is particularly unhelpful[3] in COSMIC's opinion, so we do not use it. (Indeed the term 'NFR' is not universally used. A software architect's view is that a better expression for NFR would be 'other stakeholder concerns' [4].)

COSMIC has therefore agreed a definition with IFPUG of NFR that may be easily compared and contrasted with the ISO definition of FUR, to make it easier to distinguish FUR and NFR.

DEFINITION – Non-Functional Requirements (of a software system/product)
Any requirement for a hardware/software system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software. Non-functional requirements concern: • the software system or software product quality; • the environment in which the software system or software product must be implemented and which it must serve; • the processes and technology to be used to develop and maintain the software system or software product, and the technology to be used for their execution.

It follows from the above that we can define three main classes of NFR.

DEFINITION – Quality Requirements
Requirements for the quality or for the architecture or design of a delivered software system or software product.

DEFINITION – System Environment Requirements
Characteristics of the environment in which a software system or software product is developed and maintained and which it must support in operation, e.g., its user base, etc.

DEFINITION – Technical Requirements
Requirements for how a software system or software product will be built, such as the programming language to be used and for the technology (hardware and communications) that the software system or software product will need in operation.

[3] The ISO 25756 definition defines a Non-functional requirement as: *"A software requirement that describes not what the software will do but how the software will do it. Example: software performance requirements, software external interface requirements, software design constraints, and software quality attributes.*

The main problem with this definition is that 'how the software will do it' is both vague and incompatible with three of the four examples of NFR given in the definition. For example, a 'software performance requirement' says nothing about how the software will do it. Only 'software design constraints' say anything about 'how' the software will 'do it'. Further, this ISO 25756 definition is not compatible with the ISO 14143/1 definition of Functional User Requirements.

Recalling section 1.2.3 which describes how some NFR may evolve wholly or partly as a project progresses into FUR for software, we can now see that Quality Requirements may evolve this way, but not System Environment or Technical Requirements.

Note that in addition to quality NFR for the software system/product, ISO/IEC has published a 'Data Quality Model' [39] which includes 15 terms that describe data quality from two perspectives, 'inherent' and 'system dependent', which partly overlap.

- 'Inherent' terms concern data quality attributes which describe whether the data itself meets needs, e.g. Accuracy, Completeness, Consistency, and Credibility.
- System dependent' terms concern data quality attributes that depend on the software system/product environment in which the data are used (hardware devices, etc.), e.g. Accessibility, Availability, Portability, and Precision.

Several of the terms in this Guideline with definitions relevant to NFR of the software system/product also occur in the ISO/IEC 'Data Quality Model' with definitions relevant to the quality of data used or maintained by the software.

This first version of this Guideline does not include the definitions that are relevant to data quality. They will be considered for inclusion in a future version of the Guideline.

2.3 Project Requirements and Constraints (PRC)

DEFINITION – Project Requirements and Constraints
Requirements that define how a software system project should be managed and resourced or constraints that affect its performance: Requirements may include: • the targets the project should achieve (e.g. budget, delivery date, product quality); • the project management processes that should be used; • how the project should be governed and resourced. Constraints may include: • limitations on the project resources planned or needed; • dependencies on other projects outside the control of the project concerned.

2.4 Summary model of requirements for a software systems project

Figure 2.1 shows the overall classification scheme for the various types of requirements that may arise in a software systems project, as used in this Guideline.

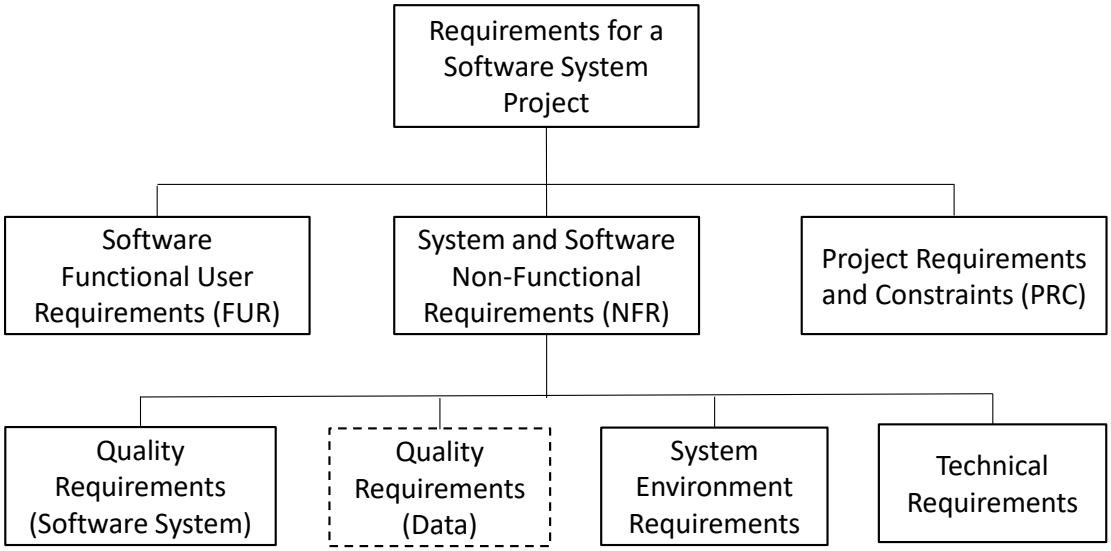

Figure 2.1 - Summary model of requirements for a software systems project

3

SELECTION AND CLASSIFICATION OF NFR AND PRC TERMS

3.1 Selection of NFR terms

Because there are so many possible NFR terms and because of the overlaps, subtleties and variations of definition, we are far from having a complete universally-accepted list. (In addition to the survey already mentioned [4] that found 108, terms, another study [40] listed 'at least 161' NFR terms, and a third study [8] found 122 terms and structured them into a hierarchy.)

The list of NFR terms in this Glossary was developed starting from the list in [4], expanded by including a wider range of ISO or IEEE terms, then followed by some rationalization. The list has been checked against 'A taxonomy of software projects productivity impact factors' [31] and has also benefited from a collaboration between COSMIC and the International Function Point Users Group (IFPUG) to produce a common Glossary of NFR terms [29]. The resulting list comprises 60 terms, which aims to be reasonably comprehensive and a useful practical starting point.

Readers should feel free to modify this list for their own purposes and, if they feel strongly, suggest changes to the list in the Glossary, using the procedure of Appendix A.

The 60 terms are divided into three Main Classes (corresponding to the three bullet points of the NFR definition) to make them more manageable and easier to find, as shown in the tables below. A very important factor in deciding on this structure was to try to reconcile the NFR classification with the structure of the 'product quality model' of the ISO/IEC 25010:2011 'System and software product Quality Requirements and Evaluation' ('SQuaRE') standard [5].

Selection of terms involved many pragmatic judgments. First we included only terms that are 'elementary' NFR, i.e. terms that are not composites of, or derived from, or classes of other NFR terms. This is because the number of terms derivable from two or more elementary NFR is huge. If needed, these can be defined locally.

EXAMPLE 1: The term 'performance' is one of the parameters of the SQuaRE Quality Model [5]. 'Performance' is defined as 'the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage'. The term 'performance' is therefore a composite non-functional requirement covering several possible parameters. To state a non-functional requirement of 'performance' would be meaningless without further information; you must specify a specific performance parameter, such as 'response time ', or 'transaction rate'. So 'performance' is not included.

EXAMPLE 2: All terms concerned with costs have been excluded, e.g. 'cost of ownership', 'ROI', etc., as they can all be derived from other data. Cost comparisons are, of course, extremely important but to understand them properly may require knowledge of cost accounting conventions (e.g. whether staff-rates are fully-loaded with all overheads, or only partially loaded), cost inflation (if comparing historic data) currency exchange rates, etc. These factors can be considered locally.

EXAMPLE 3: The ISBSG requirement to record Programming language 'level', (i.e. 2GL. 3 GL, 4 GL, etc.) [8], was omitted since these classes of 'programming language' are not well defined. Instead, 'programming language' and 'programming paradigm' are included.

Second, we excluded terms that are 'sub-sorts' of other terms included in the Glossary.

EXAMPLE 4: 'Maintainability' is included but it has many sub-sorts, e.g. modularity, modifiability, extendibility, flexibility, testability, etc. These sub-sort terms are not included. Maintainability might also be made easier by using re-usable software, hence 're-usability' may also be considered as a sub-sort of maintainability. However a requirement for 'reusability' is different from a requirement for 'maintainability', with different consequences for project activities, so both these terms are included.

Third, we avoided terms with strongly overlapping definitions.

EXAMPLE 5: 'Modifiability', 'Evolvability' and ''Extensibility' overlap with 'Adaptability'. Only the latter was included.

Some decisions on what to include or exclude were marginal.

EXAMPLE 6: The ISO/IEC 25010:2011 *definition of 'adaptability' (defined as 'degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments') overlaps strongly with that of 'portability' (defined as '(1) ease with which a system or component can be transferred from one hardware or software environment to another (*ISO/IEC/IEEE *24765:2010) (2) capability of a program to be executed on various types of data processing systems without converting the program to a different language and with little or no modification (ISO 2382-1:1993)). However, given the ISO/IEC 25010:2011 definition, 'adaptability' could also be applied to evolving business requirements. We therefore included both portability and adaptability.*

A few terms that have ISO definitions were excluded for reasons that might not be clear. See section 7.4 of the Glossary on 'Terms that have been excluded', where the reasons for their exclusion are given.

3.1.1 Quality Requirements

The table below shows the 42 NFR terms in this class quite closely mapped to the eight groups of the ISO/IEC 25010:2011 'System/Software Product Quality' model [5]. A ninth group ('Related to system or software architecture or design') of terms that are not mentioned in ISO/IEC 25010:2011 also seems to fit naturally into this main class.

	Quality Group	**NFR terms**
1	Related to the quality of the data maintained by the software	Accountability Accuracy Auditability Precision Validation (of data)
2	Related to system performance	Response time Transaction rate
3	Related to compatibility	Co-existence Compatibility Interoperability

	Quality Group	NFR terms
4	Related to the ease of use by the intended user	Accessibility Aesthetics (of the UI) Customer satisfaction (software) Learnability Multi-lingual support Operability Usability
5	Related to system reliability	Availability Back-up Dependability Diversity Failure management Fault tolerance Recoverability Reliability Safety
6	Related to control of access	Authenticity Confidentiality Non-repudiation Privacy Security Usage mode (live vs training/testing)
7	Related to maintainability	Adaptability Maintainability Reusability Re-use type
8	Related to ease of deployment	Installability Portability
9	Related to system or software architecture or design	Architecture/Design Interfaces Open source Operational processing mode

3.1.2 System Environment Requirements

The six System Environment Constraints that characterize the environment that the software system/product must support are taken mainly from the ISBSG Data Collection forms [19].

	System Environment Group	NFR terms
1	Context	Industry
2	Application Domain	Application type Application sub-type
3	Implementations	Implementations (number of)
4	User Base	Distinct users - maximum number Concurrent users - maximum number

3.1.3 Technical Requirements

The 12 Technical Requirements are taken mainly from the ISBSG Data Collection forms [19].

	Technical Group	**NFR terms**
1	Operational Platform	Operational platform type Operational platform physical distribution Operational platform volatility
2	Database	Database management system Database size
3	Operational Platform constraints	Communications network Operational processor memory Operational processor speed Operational storage capacity
4	Development requirements	Methods and tools Programming language Programming paradigm

3.2 Selection of terms for Project Requirements and Constraints

The 19 terms are mostly taken from ISBSG and PMI® Terminology [38].

	PRC Group	**Project Requirements and Constraints terms**
1	Project Type	Project type (e.g. new vs enhancement)
2	Project Resources	Effort Skills and experience level Staffing level Team relationships Work breakdown structure
3	Project Quality	Customer satisfaction (project) Defect count
4	Project Risk	Dependencies on other parties Post-project review findings Risk Scope change
5	Project Processes	Development environment Governance Location Process maturity Project management method
6	Project Duration (Schedule)	Duration Schedule compression / expansion

4

DEALING WITH NFR AND PRC FOR PROJECTS WITHIN AN ORGANIZATION

The purpose of this chapter is to advise on how to take account of individual NFR and PRC for comparing project performance, developing benchmarks and estimating within an organization, i.e. the most common practical situations where COSMIC functional sizes will be used.

[If your organization wishes to participate in an *external* benchmarking exercise, then some environmental data may need to be recorded in addition to that recorded for internal purposes to enable the external benchmark service supplier to make like-for-like comparisons across, for example, different countries. However, as these data are usually common for all projects to be benchmarked, they do not need to be routinely recorded for internal purposes.]

The contents of this Chapter are relevant to relatively simple software projects. For projects to develop large 'mission-critical' systems, where NFR can be dominantly important, a more sophisticated approach for dealing with NFR will be needed than is described here.

This Chapter takes no account of the use of automated tools such as for storing requirements, project data repositories, or for estimating. These tools may require some other data to be gathered than the data listed here.

It is recommended that organizations first study and decide what data they need to collect for their internal needs before selecting tools to meet these needs.

4.1 Commonality of NFR and PRC across an organization

The combined total of 79 types of NFR and PRC is long but it should not be necessary to record any of these for each system and project if they are common across the organization. In any one software development environment, most NFR and many PRC will be the same for all systems and projects.

> *EXAMPLE. If all systems in your organization are from the domain of 'insurance', and the systems must all be able to operate in disaster recovery mode, must all be auditable, etc., there is no point in recording these constraints for every system. But a constraint that varies by system such as that some systems must operate in on-line and some in batch mode should be recorded as this constraint affects the development tools that can be used and hence project performance.*

The NFR and PRC that should be recorded are those that vary significantly across projects, and hence may need to be considered when comparing performance across multiple projects and for building estimating models.

The set of NFR and PRC for a specific domain, perhaps in a specific organization, may be described as a 'NFR/PRC Pattern'.

[Obviously, if you work in a very complex business, e.g. aircraft manufacturing, then your systems will come from multiple domains, but you will probably not want to compare software project performance across multiple domains.]

4.2 Typical NFR and PRC to be recorded for business application projects

The following set illustrates the limited number of NFR and PRC that may be worth recording in any one organization because they typically vary across its projects. These NFR and PRC may need to be taken into account when making 'like-for-like' comparisons of project performance, developing internal benchmarks and when estimating for new projects.

The NFR and PRC in the table below are listed in the order of the Classification presented in Chapter 2.

NFR and PRC Class	Terms
Quality NFR	Response time, Transaction rate Security, Privacy Maintainability, Reusability Interfaces, Operational processing mode
System Environment NFR	Application type, sub-type Implementations (numbers of) Maximum number of concurrent users
Technical NFR	Operational platform type Programming language DBMS software
Project Requirements and Constraints (PRC)	Many, but not all of the 19 project requirements and constraints are worth recording. Example: if staff experience levels, processes and methods and tools are the same for all projects, then they need not be recorded for internal purposes.

Note that advanced software organizations may not need to record so many NFR for each project if they build certain functional implementations of NFR into the basic development environment

EXAMPLE: a project may not need to specify usability NFR if the development environment has built-in standard usability functionality (either in terms of re-usable standardized GUIs or standard human interface development procedures), unless a new usability NFR is needed.

Software functional size data must of course also be collected and the parameters of the size measurement that will enable anyone to understand the measurement in the future.

Generally, it is best to start with collecting data on a carefully-selected minimum number of NFR and PRC to avoid unnecessary work. Other NFR and PRC can be added at a later date if experience shows they are necessary.

4.3 Typical NFR and PRC to be recorded for real-time embedded software projects

As for business application projects discussed in section 4.2, advanced software organizations may not need to record many NFR separately for each project if they build certain functional implementations of NFR into the basic development environment.

Probably most of the NFR listed below will be common for a 'smart system' (or a 'system of systems'), i.e. a system formed by linking components each of which is a system with its own embedded software. For such systems, the NFR may be recorded only at the overall system level.

NFR and PRC Class	Terms
Quality NFR	Precision Response time, Transaction rate Interoperability Dependability, Safety Security, Maintainability, Reusability Architecture/Design, Interfaces
System Environment NFR	Application type, sub-type Implementations (numbers of)
Technical NFR	Operational platform type Operational processor memory Operational processor speed Operational storage, capacity Programming language DBMS software
Project Requirements and Constraints (PRC)	Many, but not all of the 19 project requirements and constraints are worth recording. Example: if staff experience levels, processes and methods and tools are the same for all projects, then they need not be recorded for internal purposes.

4.4 Establishing internal benchmarks

Internal benchmark data can be very valuable as the 'organization's standard performance levels' against which the performance of individual projects may be compared, and for use in project effort estimating.

Establishing such benchmarks normally involves sorting the data from completed projects into clusters of similar types, defined by their common NFR and PRC. Normally, projects would first be sorted by the common NFR and PRC of:

- environmental requirements (e.g. application domain or type), and then by
- project type (development, enhancement, etc.), and then by
- technical requirements, i.e. combinations of operational platform type and development methods & tools and programming language that are common in the organization.

Benchmarks for the main performance indicators can then be derived from the completed project data in these clusters by plotting the relationships of:

- software functional size versus actual project effort, to establish the productivity benchmark;
- software functional size versus actual project duration, to establish a project speed benchmark;

- defects found in the first month of operation versus functional software size to establish defect density, further sub-divided by defect severity;
- Actual versus estimated effort, and actual versus estimated duration, etc.

When these benchmarks are established, the performance parameters of an individual project may be compared against the benchmarks. Project constraints will then likely remain as the main causes of variations in individual project performance relative to the benchmarks.

Of course, many other benchmark data can be established and many other data analyses carried out depending on the organization's particular goals and concerns.

4.5 Dealing with NFR and PRC in software project estimating

The purpose of this section is to describe how NFR and PRC should be taken into account in an estimating process (but not to describe software project estimating processes in detail). An estimating process is very straightforward in principle, though the details will vary greatly across different types of projects.

Figure 4.1 shows how the measurement of functional size may evolve, super-imposed on the process shown in Fig. 1.4, and assuming a waterfall project management approach.

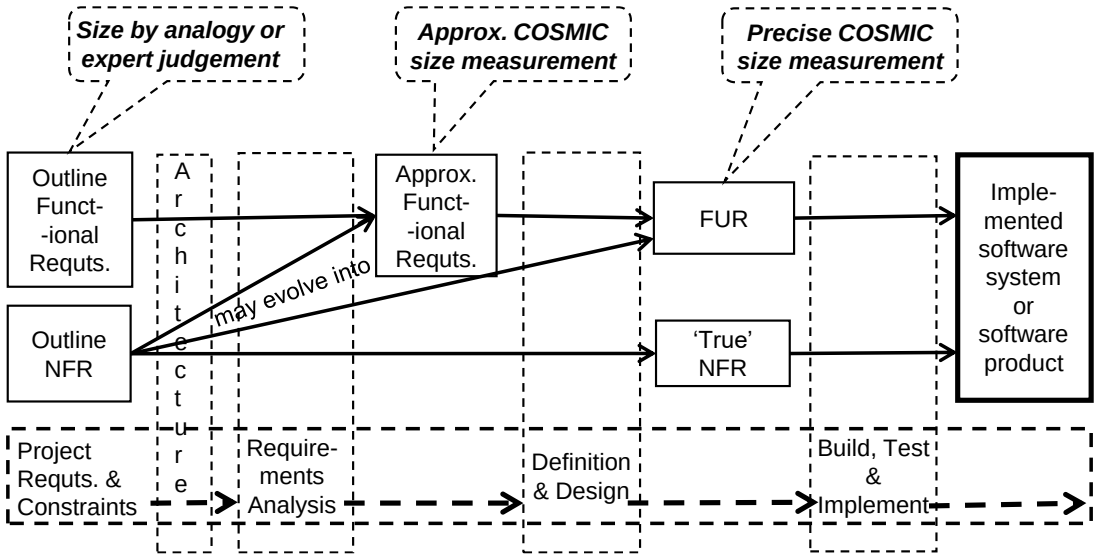

Figure 4.1 - The measurement of functional size as a project evolves

In an ideal world, the estimating process proceeds as follows:

Estimating project effort from outline FUR and NFR

The process for estimating project effort very early in a project will be largely independent of how the project will be managed once it starts, e.g. using a waterfall versus agile process.

a) When developing the initial outline or 'high-level' statement of requirements for the software to be developed by a new project, the focus will be on the functional requirements – what the software must do and on the PRC concerned with targets (budget, delivery date, etc.) and risk.

b) In spite of this focus, the classification and the Glossary of NFR terms should be used to elicit and document other requirements that may appear as NFR at this stage.

c) When there is an outline statement of functional requirements and the principal NFR have been identified, we have reached the starting point of the process shown in Figure 4.1. If a software project effort estimate is needed at this stage, it can only be made by using an analogy or by expert judgment, including a contingency allowance,

d) The outline functional requirements and NFR feed decisions about the system/software architecture which begin to help identify the Quality NFR that evolve into FUR [9].

Estimating project effort and cost from approximate functional requirements and NFR

e) As the project progresses, a contribution to the total software functional size may be added in for the Quality NFR that evolve into FUR. This contribution may be made on the basis of experience of adding a contingency or allowing for 'scope creep', or by using specific experience of sizing Quality NFR, e.g. using cases such as described in Chapter 5 of this Guideline (Quality NFR are the only types of NFR that should automatically be considered as possibly affecting the functional size.) The contingency for further scope creep may be updated.

f) When the requirements analysis phase is considered complete, it is unlikely that the software functional requirements are detailed enough for a precise COSMIC measurement. Nevertheless, the functional size may be estimated using a COSMIC approximate size measurement method [6].

g) The estimated total functional size may be converted to estimated project effort using benchmark data established (as in section 4.4) for projects with a functional size and a profile of NFR as close as possible to that of the new project. This 'closeness of fit' may be determined by analogy or by statistical analysis of completed past projects.

Figure 4.2 shows the various ways by which NFR 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 'true' NFR's of the programming language and hardware platform on which the software will execute; a NFR for significant re-use of existing code may also affect the effort calculation. (The model of Figure 4.2 for estimating project costs can be applied as soon as an approximate functional size is known and any time thereafter, and is independent of the development process e.g. waterfall versus agile.)

The estimated project effort obtained by this process may be further refined by the PRC of the project (see further below).

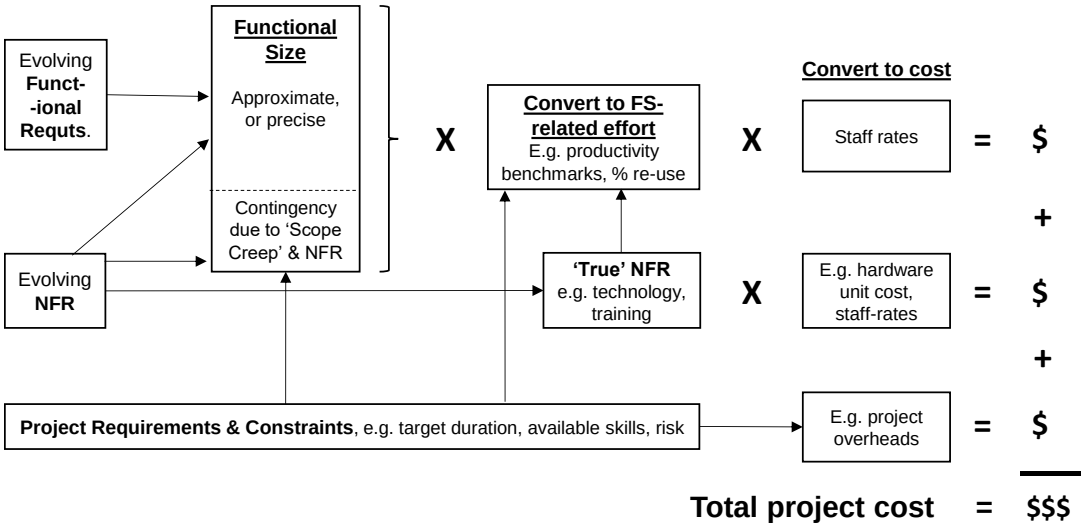

Figure 4.2 - The various ways by which evolving functional requirements, NFR and PRC contribute to functional size and affect the determination of project effort and cost

Estimating from detailed functional requirements

h) As the project progresses into the Design phase, the functional requirements will be worked out in more detail and the Quality NFR will evolve into FUR that can be measured. The size of the FUR (those originating as functional requirements plus those originating as NFR) will become progressively more accurate, hopefully falling within the contingency established in steps e) and f). The process of converting functional size and PRC to effort remains essentially the same but will be better informed.

Estimating in iterative or agile projects

i) Alternatively, if an 'Agile' or iterative process is followed, the functional requirements of each sprint or iteration can be measured. Contributions to the total functional size, and hence total software project effort can then be refined and made more accurate. [10] NFR must not be forgotten. NFR that evolve into FUR can be taken into account in separate sprints (or iterations) or in sprints that comprise FUR from various origins. The results of tracking actual progress against the PRC and any need for re-work will also need to be taken into account in refining the effort estimates.

Taking account of PRC

j) A project constraint such as a firm budget limit may directly affect the functional size that can be delivered. This may happen in agile developments where the prioritization processes may result in a project being stopped when the budget limit is reached; low-priority FUR are never implemented.

However, project requirements and constraints most often directly affect the project effort, but have no effect on the software size. Examples are staff experience, duration constraints, project processes, and risk. Such PRC may not need to be taken into account separately if the project to be estimated is considered to be typical of those included in the benchmark dataset.

If the project is not typical of those in the benchmark dataset, the additional (or reduced) effort for any of these PRC must be estimated separately. The approach will vary depending on the parameter. Examples:

- to take account of a duration constraint, most Estimators will rely on one of the effort/duration trade-off models;
- to take account of staff experience, the Estimator will allow for training or extra effort in consultation with the project manager who knows his staff (or if the staff are much more experienced in the problem area than average, the initially-estimated effort might be reduced);
- to take account of varying project management methods, e.g. agile versus waterfall, would need a special study to sort project effort versus size data by the two approaches;
- a measure of risk could affect the estimate of contingency effort needed;
- in some organizations, a fixed project start-up overhead adds to the cost.

5

EXAMPLES OF FUNCTIONAL SIZE MEASUREMENT OF NFR

5.1 Measuring the FUR of application software arising from Quality NFR

As explained earlier, all Quality NFR may evolve as a project progresses, wholly or partly into FUR for software. Most of these FUR can be measured at least to some degree by the COSMIC method. The table below illustrates possible evolutions for each of the 42 Quality NFR terms. The table shows:

- (Column 2) requirements initially expressed as NFR
- (Column 3) examples of functionality of application software that may evolve from the requirement initially expressed as non-functional (in column 2), whose functional size may be measured by COSMIC and added into the total size.
- (Column 4) the part of the requirement initially expressed as non-functional (in column 2) that remains a non-functional requirement throughout the system/software product life, and that can never contribute to functional size.

'Not measurable' means 'very unlikely to be measurable with the standard COSMIC method'. This could be due, for example, to the FUR resulting from the NFR being at a finer level of granularity than can be measured by the COSMIC method.

EXAMPLE: A Quality NFR for calculating the precision of an individual data attribute is very unlikely to be measurable by the COSMIC method.

Abbreviations: E = Entry, X = Exit, R = Read, W = Write.

Quality NFR Group	The initial Non-Functional Requirement (NFR)	Example functionality that may evolve from the NFR, whose size may be measured by COSMIC	The remaining part of the NFR that that cannot contribute to Functional Size
Data Quality	Accountability	Functionality that links a person responsible for maintaining data or software to changes	The specific accountability requirements statement
	Accuracy	Validation checks by R's of stored data. GUI elements that accept only valid data	The quantified accuracy targets
	Auditability	Functionality needed by auditors, e.g. X or W of hash totals; special enquiries	Adequate documentation
	Precision	Not measurable	The quantified precision targets
	Validation (of data)	See 'Accuracy'	E.g. legal requirements for independent audits. Need to store data within national boundary or 'safe haven'

Quality NFR Group	The initial Non-Functional Requirement (NFR)	Example functionality that may evolve from the NFR, whose size may be measured by COSMIC	The remaining part of the NFR that that cannot contribute to Functional Size
System Performance	Response time	Functionality to import external data so that it is available for immediate use.	The quantified response time target; Fast hardware; Low-level programming languages
	Transaction rate	Functionality to measure the transaction rate (usually measured by the OS, not by the application to which the NFR applies)	The quantified transaction rate target
Compatibility	Co-existence	Achieved via the OS?	The specific co-existence requirements
	Compatibility	Functionality of interfaces, achieved either via X/E exchanges or via shared persistent data, or via conversion functionality	The specific compatibility requirements
	Interoperability	See 'Compatibility'	The specific inter-operability requirements
Ease of use	Accessibility	See 'Usability'	The requirements for accessibility by specific classes of people; Provision of braille keyboards
	Aesthetics (of the UI)	Not measurable	The specific aesthetic requirements
	Customer satisfaction (software)	Not measurable	The customer satisfaction target, as measured by a specific technique.
	Learnability	Functionality to assist users to learn the system	The specific learnability targets
	Multi-lingual support	Functionality for the users to select the language and to enter E's and receive X's in multiple languages	The requirement to support named languages and their character sets
	Operability	Functionality to assist operators (maybe provided by the OS)	The specific operability requirements
	Usability	Graphical User Interface functions; Functionality to assist users, e.g. 'Help' functions	The specific usability requirements (e.g. 'must be usable by public with no training')

Quality NFR Group	The initial Non-Functional Requirement (NFR)	Example functionality that may evolve from the NFR, whose size may be measured by COSMIC	The remaining part of the NFR that that cannot contribute to Functional Size
System reliability	Availability	Functionality to exploit multiple parallel processors to ensure high availability	The specific, quantified availability target; High reliability or fault-tolerant hardware
	Back-up	Back-up functionality, if not provided by the OS	The specific requirements for back-up of named files
	Dependability	See 'Availability', 'Fault tolerance', Recoverability'	The specific Dependability target
	Diversity	The diverse software solutions resulting from a Diversity NFR	The specific Diversity target
	Failure management	Functionality to assist failure management	Manual processes; Specific quantified failure management targets
	Fault tolerance	Functionality to exploit multiple parallel processors to ensure continuity when one or more processors fail	The specific target for continuity in spite of faults. Fault-tolerant hardware
	Recoverability	Functionality to recover data and resume processing following faults or interruptions	The specific recoverability target
	Reliability	See 'availability', 'security', maintainability', etc.	The specific Reliability target
	Safety	Functionality (as in safety-critical systems) to ensure human safety	The specific safety targets
Access control	Authenticity	Functionality that enables the identity of a person or a resource to be proven	The specific authenticity target. Finger-print recognition hardware
	Confidentiality	Functionality that protects data against unauthorized access	The specific confidentiality target
	Non-repudiation	Functionality that helps prove that an event or action did take place	The specific non-repudiation target
	Privacy	Functionality to control access to personal data	The specific privacy requirements
	Security	Functionality to control access to systems and/or data	The specific security requirements
	Usage mode (live vs training)	Functionality to provide both modes and to allow the user to access both modes	The specific requirements for different modes

Quality NFR Group	The initial Non-Functional Requirement (NFR)	Example functionality that may evolve from the NFR, whose size may be measured by COSMIC	The remaining part of the NFR that that cannot contribute to Functional Size
Maintainability	Adaptability	Functionality to enable the system to be adapted to different hardware, operational, etc., environments	The requirements for the specific environments for which the software must be adaptable
	Maintainability	Functionality to enable the system to be maintained without re-programming, e.g. via parameterization	The specific maintainability requirements
	Reusability	The functionality that must be made re-usable	The specific re-usability requirements
	Re-use type	Functionality of re-used artefacts	The specific re-use types required
Ease of deployment	Installability	Functionality to facilitate ease of installation	The specific installability requirements
	Portability	'Middleware' functionality to enable portability across multiple DBMS or OS software	The requirements for the specific environments across which the software must be portable
System or software architecture or design	Architecture/ Design	Software can be measured in any layer of an architecture	Design requirements that do not affect FUR
	Interfaces	Functionality achieved either via X/E exchanges or via shared persistent data	The specific interface requirements
	Open source	It is irrelevant to FSM whether the software is open source or not but the functionality of such software may be measured	The specific open source requirements
	Operational processing mode	Any functionality can be measured regardless of its Operational Processing Mode	The specific operational processing mode requirement

5.2 A simple security NFR

A requirement may be stated as 'The application A will require its own user identification and password sign-on procedure. Only employees authorized by the System Administrator will be allowed access'.

As 'security' is listed in the ISO 14143/1 definition of FUR as a quality requirement, it may be interpreted as non-functional. In fact this requirement is commonly implemented entirely by software functionality which can be measured with the COSMIC method. The following illustrates a possible simple implementation.

First the System Administrator will need to maintain a file of employees who may access application A. (For simplicity, we ignore how the System Administrator maintains the security of this file.) The attributes of each employee record could be: Employee_ID, Employee_name, Employee_password (PW), Date of last change of PW, plus maybe a history of, say, the previous 10 PW's,

The System Administrator will need functional processes (FP's) to Create, Retrieve and Delete employee records, and probably to List employee records. The Create employee functional process will generate an e-mail to send an initial PW to a newly-authorized employee. Two more FP's will be needed a) to send a new PW to an employee who has forgotten his/her PW and b) to force an employee to change the current PW at, say, monthly intervals. So far we have identified the need for six FP's.

For the employee sign-on process, the application must be able to access the System Administrator's employee PW file. Two sign-on FP's will be needed, first for an initial sign-on when the employee will have to create his own personal PW, second to change a PW.

In total this simple example needs eight FP's. Guessing that each FP would need an average of 4 – 5 data movements, the total size of this simple access control functionality would be about 8 x 5 = 40 CFP.

5.3 A portability NFR

The paper 'A standards-based reference framework for system portability requirements' [23] analyses how various standards define what they mean by 'portability' requirements at the system level, i.e. before these requirements are allocated to hardware or software. At this level these are non-functional requirements.

The standards considered are from the European Cooperation for Space Standardization [24], and for IEEE 830 [25], ISO 9126 [17], ISO 24765 [26] and ISO 2382-1 [27].

The paper synthesizes the system portability NFR from these various standards into a generic model of portability requirements of which there are four main types. A system may be required to be independent of, i.e. portable across:

1. software components (operating system software, middleware, etc.);
2. data components (DBMS, etc);
3. hardware components (client, server, storage, etc);
4. the 'isolating software system calls function' (a function used by a software system to request a service from the OS which isolates the calling software system from the OS).

The generic portability model is analyzed as if it were being implemented by a set of SOA services. The approach of the COSMIC 'Guideline for sizing SOA software' [28] is used to measure the functional size of all the possible interfaces to the services which would be needed to implement the entire generic portability model. These interfaces require 240 CFP.

The lesson from this paper for this Guideline is that a single, apparently simple NFR such as that a piece of software must be portable across various technical hardware/software environments can evolve, when worked out in detail, into a considerable addition to the functional size of the piece of software.

5.4 NFR for a mobile e-mail system

The paper 'A meta-model for the assessment of non-functional requirement size' [32] contains a fully worked-out example of measuring part of the 'operationalization' of an e-mail system.

6

MEASUREMENT OF NFR

6.1 Sizing NFR collectively

When Allan Albrecht of IBM first defined Function Point Analysis [11], the size scale was defined to measure the 'functions in an application'. This size was adjusted by a 'complexity adjustment' which took account of the 'degrees of influence' of 10 factors that nowadays would be called NFR. The weights of all the components of this method were derived from IBM methods for estimating project effort [12]. Later the 'complexity adjustment' was extended to account for 14 NFR and was re-named as the 'Value Adjustment Factor' (VAF). [13].

The VAF gives higher values for on-line systems (typically ~ 1.1 to 1.2) compared with batch-processed systems (typically ~ 0.75 to 0.85). In the early 1980's this was understandable; at that time it often took more effort to develop an on-line system relative to a batch-processed system. For example, each on-line application might have to develop its own back-up and recovery processes, and making on-line software easy-to-use was difficult; GUI interfaces had not yet become generally available.

Later, Symons noted that 'the restriction to 14 factors 'seems unlikely to be satisfactory for all time' [14]. He proposed a 'Technical Complexity Adjustment' (TCA) which extended Albrecht's VAF list to 19 factors with the possibility of adding more factors locally. The weights of the 'degrees of influence' were also re-calibrated and related to development effort by a Delphi technique.

The experience of these early attempts to account for NFR by a collective size scale is that such measures might be useful for performance measurement comparisons and estimating for projects within a very limited software domain, with a limited range of NFR. But a collective size measure for all possible NFR cannot be generally valid across all types of software, for all time. Any such measure is soon out of date, given the continuously evolving technology.

Furthermore, such constructs as the VAF and the TCA are mathematically invalid[4], so the resulting number 'feels good' but has no mathematical validity (and it is therefore invalid to multiply such a number by a size in 'unadjusted' function points).

In addition, a collective size measure for NFR is semantically meaningless unless it is calibrated so that the weights applied to the various NFR are related to a common scale such as the relative effort to implement each of them. But since:

- there are so many possible NFR (over 100 according to some studies, with overlapping definitions);
- NFR can sometimes interact, e.g. satisfying a NFR for portability may or may not cause extra effort to meet a NFR for response time and another NFR for security;

[4] 1: The 14 General Systems Characteristics (GSC) are on a 'nominal' scale type. 2: Each GSC is subdivided into six categories, each with an increased 'ranking', at irregular intervals. 3: Each of these rankings is next 'labelled' from 0 to 5 (and while these appear as numbers they are merely ordered labels on which no mathematical operations are allowed). 4: Each of these 'labels' is multiplied by the same 'factor' of 0.1; it is mathematically invalid to add or multiply labels of an ordinal scale [33].

- the effort for the various NFR will itself vary with several factors, e.g. the technology used, the type of software, the re-use of existing software, etc.,

it is impossible to establish a standard set of NFR and a standard set of weights that will be applicable in all circumstances and for all time.

As one commenter remarked, 'non-functional' is a 'troublesome' term ... it 'bundles together many things that are otherwise unrelated to one another. … There's no reason to assume that such diverse concerns as design-time modifiability, run-time performance, product time-to-market and architectural consistency are all amenable to the same treatment'. [4]

Finally, even if the collective NFR size measure had some meaning, the fact that NFR evolve, as a project progresses, into FUR makes it extremely difficult to capture the associated effort data which would be needed to calibrate the collective NFR size measure and to use it in practice for estimating future projects. Figures 4.1 and 4.2 show the variety and complexity of how NFR and PRC affect effort as a project progresses and requirements evolve.

6.2 Recording and measuring individual NFR and PRC

In contrast to the conclusion of section 6.1, it could be beneficial for the software industry if a standard size scale could be established for each individual type of NFR and PRC for which a standard definition has been agreed as, for example, in the Glossary of this Guideline. This would at least facilitate comparing performance across projects subject to different NFR and PRC, and the use of this data for defining benchmarks and for new project estimating.

The task of developing a size scale for all NFR and PRC listed in the Glossary is however well beyond the scope of this first version of this Guideline.

At this stage we can only illustrate the various possible measurement scale types with examples.

Scale Type	Examples of possible NFR and PRC for the scale type
Nominal (labelling, classifying entities)	Programming language, Application domain, Project type
Ordinal (monotonically increasing)	A scale of project risk, e.g. (low, average, high, very high) Project process maturity e.g. CMM-I level
Interval (ratios not valid)	Target project completion date, Customer satisfaction expressed on a numerical scale
Ratio (the scale, or 'unit of measure' can vary)	Project duration, Database size (measured in e.g. Mbytes), Response time, Availability (measured as MTBF).
Absolute (counting entities)	Numbers of Implementations, Interfaces, Users, Defects,

Note that the parts of most Quality NFR that remain after separating out the parts that evolve into FUR for software (column 4 of the table in section 5.1) can only be measured on a nominal or ordinal scale.

6.3 ISO/IEC standards for measuring individual Quality NFR

ISO/IEC has published standards [34, 35] that list some measures for individual Quality NFR. The 'Quality Model' of this 9126 series of standards has, however, been updated to the SQuaRE 'Product Quality' model of the ISO/IEC 25010:2011 standard [5], which has been used as the basis for the selection and classification of NFR terms in this Guideline and in the joint COSMIC.IFPUG Glossary [29].

The ISO/IEC 25023 standard [36], still under development in 2015, will eventually replace the 9126 standards [34, 35]. It is sub-titled ‘Measurement of system and product quality’. Its contents will define measures for most of the Quality NFR listed in this Guideline.

7

GLOSSARY OF NFR AND PRC TERMS

This Glossary is divided into three lists: NFR terms, PRC terms, and excluded terms. Each list is in alphabetic order, selected as described in sections 3.1 and 3.2.

Readers who wish to add to or amend the lists are asked to use the Change Request and Comment Procedure in the Appendix to this Guideline.

The contents of this chapter were produced jointly with IFPUG (The International Function Point Users Group) and are identical to the corresponding chapter 4 of the joint COSMIC/IFPUG Glossary [29].

7.1 Sources of ISO standard and other definitions

Definitions of the terms in the Glossary are taken from the sources listed below.

Doc. Reference No.	Document Title
Chambers	The Chambers Dictionary
COSMIC/IFPUG	Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating, joint publication of COSMIC and IFPUG, September 2015 [29]
IEC 60050-191	International Electrotechnical Vocabulary, Chapter 191, Dependability and quality of service, 1990.
IEEE 982.1-2005	IEEE Standard Dictionary of Measures of the Software Aspects of Dependability
IEEE 1012-2004	IEEE Standard for Software Verification and Validation
ISBSG	Glossary of terms for software project development and enhancement v5.16a, 22/08/12
ISO 5725-1:1994	Accuracy (trueness and precision) of measurement methods and results -- Part 1: General principles and definitions
ISO 9241-110: 2006	Ergonomics of human-system interaction – Part 110: Dialogue principles
ISO 9241-171:2008	Ergonomics of human-system interaction – Part 171: Guidance on software accessibility
ISO/IEC 2382-1: 1993	Information technology–Vocabulary–Part 1: Fundamental terms
ISO/IEC 2382-20:1990	Information technology–Vocabulary–Part 20: System development
ISO/IEC 10746-2:2009	Information technology – Open Distributed Processing – Reference Model: Foundations
ISO/IEC 12207:2008	Systems and software engineering–Software life cycle

	processes
ISO/IEC 15026-1:2013	Systems and software engineering--Systems and software assurance--Part 1: Concepts and vocabulary
ISO/IEC 15288:2008	Systems and software engineering–Software life cycle processes
ISO/IEC 20000-1:2011	Information technology–Service management–Part 1: Service management system requirements
ISO/IEC/IEEE 24765:2010	Systems and software engineering–Vocabulary
ISO/IEC 25010:2011	Systems and software engineering–Systems and software Quality Requirements and Evaluation (SquaRE)–System and software quality models
ISO/IEC 25062:2006	Software engineering – Software product Quality Requirements and Evaluation (SquaRE)
ISO/IEC 42010:2011	Systems and software engineering – Architecture description
ISBN-13: 893-7485908328	Project Management Body of Knowledge, PMI
Wikipedia	www.wikipedia.org

7.2 Glossary of Non-Functional Requirement terms

The following abbreviations are used in the Glossary for the Main Classes of NFR:

Qual. = Quality Requirements; 'Env.' = System Environment Requirements; 'Tech.' = Technical Requirements;

N.B. The classification given for each term is not absolute. Some NFR terms could be classified under more than one heading.

NFR Term	Class	Group	Definition
Accessibility	Qual.	Ease of use	(1) Usability of a product, service, environment or facility by people with the widest range of capabilities (25062:2006) (2) degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use (25010:2011). Note: Although "accessibility" typically addresses users who have disabilities, the concept is not limited to disability issues. The range of capabilities includes disabilities associated with age. (ISO 9241-171:2008)
Accountability	Qual.	Data quality	Degree to which the actions of an entity can be traced uniquely to the entity (25010:2012)
Accuracy	Qual.	Data quality	A qualitative assessment of correctness, or freedom from error (24765:2010). (2) a quantitative measure of the magnitude of error (24765:2010). The proximity of a result or a measure to the true value (ISO 5725-1:1994) See also: precision
Adaptability	Qual.	Maintain-ability	Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments (25010:2011). Note: Adaptability includes the scalability of internal capacity, such as screen fields, tables, transaction volumes, and report formats. Adaptations include those carried out by specialized support staff, business or operational staff, or end users. If the system is to be adapted by the end user, adaptability corresponds to suitability for individualization as defined in ISO 9241-110. See also: 'portability' Related concept: modifiability, evolvability, extensibility, flexibility

NFR Term	Class	Group	Definition
Aesthetics (of the user interface)	Qual.	Ease of use	Degree to which a user interface enables pleasing and satisfying interaction for the user (25010:2011). Note: refers to properties of the product or system that increase the pleasure and satisfaction of the user, such as the use of color and the nature of the graphical design.' Related concept: 'Customer experience (software)'
Application Domain	Env.		See 'Application Type (or Software Type)'
Application Sub-Type	Env.	Application domain	A type of software within each of the four ISBSG 'Application Types'. (The ISBSG Glossary lists 20 sub-types of Business Applications, 8 sub-types of Real-time Applications, 7 sub-types of Mathematically-intensive software and 6 sub-types of Infrastructure software.)
Application Type (or 'Software Type')	Env.	Application domain	A classification of software into four groups: business applications, real-time applications, mathematically-intensive software, infrastructure software (ISBSG)
Architecture-/Design	Qual.	System or software architect-ture or design	Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution. (42010:2011). Examples: requirement to conform to the ISO 7-layer model, or to the AUTOSAR architecture [37]. Related concept: structure
Auditability	Qual.	Data quality	Facility of a software system or software product to enable an auditor to examine whether data is processed correctly so as to meet requirements and internal or external audit standards (COSMIC/IFPUG). Related concepts: assurance, compliance to regulations
Authenticity	Qual.	Access control	The degree to which the identity of a subject or resource can be proved to be the one claimed (25020:2011)
Availability	Qual.	System reliability	(1) Ability of a service or service component to perform its required function at an agreed instant or over an agreed period of time (20000-1:2011) (2) the degree to which a system or component is operational and accessible when required for use (25010:2011) Note: Availability is normally expressed as a ratio or percentage of the time that the service or service component is actually available for use by the customer to the agreed time that the service should be available. Availability is a combination of maturity (which reflects the frequency of failure), fault tolerance and recoverability (which reflect the length of downtime following each failure).

			See also: fault tolerance, reliability, recoverability
NFR Term	**Class**	**Group**	**Definition**
Back-up	Qual.	System reliability	(1) A system, component, file, procedure, or person available to replace or help restore a primary item in the event of a failure or externally caused disaster (24765:2010). (2) to create or designate a system, component, file, procedure, or person as a replacement (24765:2010)
Co-existence	Qual.	Compati-bility	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product (25010:2011)
Communica-tions network	Tech.	Operational Platform Constraints	The data communication protocols that a software system or software product must observe, e.g. none, standard LAN/WAN protocols, special open protocols, proprietary or classified protocols, etc.' (COSMIC/IFPUG)
Compatibility	Qual.	Compati-bility	(1) degree to which a product, system or component can exchange information with other products, systems or components, or perform its required functions, while sharing the same hardware or software environment (25020:2011) (2) the ability of two or more systems or components to exchange information (24765:2010) (3) the capability of a functional unit to meet the requirements of a specified interface without appreciable modification (2382-1:1993)
Concurrent users (maximum number)	Env.	User base	The maximum number of users that a system can support concurrently under specified conditions (COSMIC/IFPUG)
Confidentiality	Qual.	Access control	Degree to which a product ensures that data is accessible only by those authorized (25010:2011) Example degrees: 'internal use only', 'secret', 'top secret'. See also 'Privacy'
Customer Satisfaction (software)	Qual.	Ease of use	The degree to which the customer of a software system or software product is satisfied with the system/product (COSMIC/IFPUG)
Database management system software	Tech.	Database	Software system that is used by an application to efficiently manage the access control, storage and retrieval of persistent data used by the application. Sometimes regarded as part of the infrastructure software (COSMIC/IFPUG)
Database size	Tech.	Database	(1) A measure of the physical storage space needed for a database, usually measured in units such as 'megabytes'. (2) A measure of the size of a database in units relevant to the business application software that will use the database, e.g. no. of customers, no. of employees (COSMIC/IFPUG)

NFR Term	Class	Group	Definition
Dependability	Qual.	System Reliability	(1) trustworthiness of a computer system such that reliance can be justifiably placed on the service it delivers (IEEE 982.1-2005 IEEE. (2) availability performance and its influencing factors: reliability performance, maintainability performance and maintenance support performance (ISO/IEC 15026-1:2013) 3) the ability to perform when required (IEC 60050-191:1990) Note: Dependability characteristics include availability and its inherent or external influencing factors, such as availability, reliability (including fault tolerance and recoverability), security (including confidentiality and integrity), maintainability, durability, and maintenance support. (taken from the definition of 'Reliability' in ISO/IEC 25010:2011)
Disaster recovery	Qual.		See 'Recoverability''
Distinct users (maximum number)	Env.	User base	The maximum number of distinctly identifiable users that a system can support (COSMIC/IFPUG)
Distributed Processing	Tech.	Operation-al Platform	See 'Operational Platform type'
Diversity	Qual.	System Reliability	In fault tolerance, realization of the same function by different means (ISO/IEC 24765:2010) Example: use of different processors, storage media, programming languages, algorithms, or development teams
Ease of use	Qual.		See 'Usability'
Emotional Factors	Qual.		See 'Aesthetics' (of the user interface)
Failure Management	Qual.	System reliability	The management of failures from their occurrence until resolution (COSMIC/IFPUG), where 'failure' is defined as (1) termination of the ability of a product to perform a required function or its inability to perform within previously specified limits (25000:2005) (2) an event in which a system or system component does not perform a required function within specified limits (24765:2010)
Fault tolerance	Qual.	System reliability	(1) The ability of a system or component to continue normal operation despite the presence of hardware or software fault (25010:2010). (2). pertaining to the study of errors, faults, and failures, and of methods for enabling systems to continue normal operation in the presence of faults. cf. error tolerance, fail safe, fail soft, fault secure, robustness. (24765:2010) See also 'Operational Platform type'

NFR Term	Class	Group	Definition
Industry	Env.	Context	The type of business that a software system or software product must support, as identified by a Standard Industry Code. SIC codes 'are assigned based on common characteristics shared in the products, services, production and delivery system of a business'. (Wikipedia)
Implement-ations (number of)	Env.	Implement-ations	The number of times that a software system or software product must be installed. See also 'Installability', as 'installation' is virtually a synonym of 'implementation'. (COSMIC/IFPUG) Note: Normally, the effort for a development project includes only the first implementation.
Installability	Qual.	Ease of deploy-ment	Degree of effectiveness and efficiency with which a product or system can be successfully installed or uninstalled in a specified environment (25010:2010) 'Installation' is defined as 'system development phase at the end of which the hardware, software and procedures of the system become operational (2382-20:1990) See also 'Implementations (number of)'. Related concept: configurability
Interfaces	Qual.	System or software architect-ure or design	Shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical signal exchanges, and other characteristics (25010:2010) Related concepts: autonomy, inter-process communication For project interfaces, see 'Dependencies on other parties' (a PRC term)
Interoperability	Qual.	Compati-bility	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged (25010:2011)
Learnability	Qual.	Ease of use	Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use (25010:2011) Note: Can be specified or measured either as the extent to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use, or by product properties corresponding to suitability for learning as defined in ISO 9241-110. Related concept: teachability

NFR Term	Class	Group	Definition
Maintainability	Qual.	Maintain-ability	Ease with which a software system or component can be modified to change or add capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment (24765:2010). Note: Maintainability includes installation of updates and upgrades. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users. See also: adaptability Related concepts: comprehensibility, modularity, supportability
Methods and Tools	Tech.	Develop-ment require-ments	Procedures for carrying out tasks, and supporting software aids used by the project team. (COSMIC/IFPUG) NOTE: methods and tools used should be recorded by the principal software activities, i.e. requirements determination, analysis, design, programming, testing, implementation, maintenance and support. See also 'Project Management methods'.
Multilingual support	Qual.	Ease of use	Requirement for a system to be usable in two or more natural languages (COSMIC/IFPUG)
Non-repudiation	Qual.	Access control	Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later. (ISO/IEC 250101:2011)
Open source	Qual.	System or software architect-ture or design	Requirement to use open source software or not. (COSMIC/IFPUG) Open-source software is defined as 'computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose'. (Wikipedia)
Operability	Qual.	Ease of use	Degree to which a product or system has attributes that make it easy to operate and control (ISO/IEC 250101:2011) Note: Operability corresponds to controllability, (operator) error tolerance, and conformity with user expectations as defined in ISO 9241-110.
Operational platform physical distribution	Tech.	Operation-al platform	An indicator of whether the platform on which a software system or software product is required to execute is located at a single site or is distributed over multiple sites. (COSMIC/IFPUG) Note: not to be confused with a requirement to implement the software system or software product on a single platform at multiple sites.

NFR Term	Class	Group	Definition
Operational platform type	Tech.	Operation-al platform	The hardware/software environment on which a software system or software product executes. Examples: shared utility (e.g. 'cloud'); main-frame; mid-range; PC; embedded; mobile; multi-platform (for a distributed system); parallel (or 'array') processor, communications network processor (e.g. a router) (ISBSG, COSMIC/IFPUG)
Operational platform volatility	Tech.	Operation-al platform	An indicator of whether the operational platform (hardware or software) is stable or changes often. (COSMIC/IFPUG)
Operational processing mode	Qual.	System or software architect-ure or design	An indicator of whether a software system or software product is required to execute transactions on-demand ('i.e. 'on-line'); in batches; mixed on-line and in batches; or subject to real-time constraints. (COSMIC/IFPUG)
Operational processor speed	Tech.	Operation-al platform constraints	The speed of the processor on which a software system or software product executes. (Used to indicate whether the processor speed is limited, thus requiring special effort when developing the software system or software product.) (COSMIC/IFPUG)
Operational processor memory	Tech.	Operation-al platform constraints	The memory capacity of the processor on which a software system or software product executes. (Used to indicate whether the processor memory is limited, thus requiring special effort when developing the software system or software product.) (COSMIC/IFPUG)
Operational storage capacity	Tech.	Operation-al platform constraints	The on-line storage capacity available to an executing software system or software product. (Used to indicate whether the storage capacity is limited, thus requiring special effort when developing the software.) (COSMIC/IFPUG)
Portability	Qual.	Ease of deploy-ment	(1) Ease with which a system or component can be transferred from one hardware or software environment to another (24765:2010) (2) capability of a program to be executed on various types of data processing systems without converting the program to a different language and with little or no modification (2382-1:1993)
Precision	Qual.	Data quality	The degree of exactness or discrimination with which a quantity is stated (24765:2010) Example: a precision of 2 decimal places versus a precision of 5 decimal places
Privacy	Qual.	Access control	Ability of a software system or software product to protect personal data from unauthorized or unwarranted disclosure (COSMIC/IFPUG). See also 'Confidentiality'
Programming language	Tech.	Develop-ment require-ments	The computer languages in which a software system or software product is required to be programmed e.g. C, C#, Java (COSMIC/IFPUG)

NFR Term	Class	Group	Definition
Programming paradigm	Tech.	Develop-ment require-ments	A fundamental style of computer programming, a way of building the structure and elements of computer programs (Wikipedia), e.g. procedural, object-oriented, imperative, literate, declarative, functional, logic, symbolic, synchronous, etc.
Recoverability	Qual.	System reliability	Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system (25010:2011)
Reliability	Qual.	System reliability	(1) The ability of a system or component to perform its required functions under stated conditions for a specified period of time (24765:2010) (2) degree to which a system, product or component performs specified functions under specified conditions for a specified period of time (25010:2011) Note:
Response time	Qual.	System perform-ance	The elapsed time between the end of an inquiry or command to an interactive computer system and the beginning of the system's response (24765:2010). Related concept: 'latency'
Reusability	Qual.	Maintain-ability	Degree to which an asset can be used in more than one system, or in building other assets (25010:2010)
Re-use type	Qual.	Maintain-ability	Types of re-usable assets, e.g. requirements, designs, code (modules, object classes), test suites, documentation. (COSMIC/IFPUG)
Safety	Qual.	System reliability	Expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered (24765a:2011)
Scalability	Qual.		See 'Adaptability'
Security	Qual.	Access control	(1) Protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them (12207:2008) (2) The protection of computer hardware or software from accidental or malicious access, use, modification, destruction, or disclosure. Security also pertains to personnel, data, communications, and the physical protection of computer installations. (1012-2012) Related concepts: accountability, authenticity, confidentiality, integrity, non-repudiation, privacy

NFR Term	Class	Group	Definition
Transaction rate	Qual.	System perform-ance	A transaction rate is the rate at which a defined mix of transactions is processed on a defined operational platform; it may be a target rate or an actual rate and it may be the average rate over a defined time-period, a maximum rate or a percentile rate (e.g. 90% of the transactions shall complete faster than the target rate). Synonym: 'throughput rate'. Note: A transaction is the implementation of a software system or software product requirement that may correspond to a part of, or a whole, or more than one COSMIC functional process, or similarly correspond to an IFPUG elementary process. (COSMIC/IFPUG)
Usability	Qual.	Ease of use	(1) degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use (25010:2011) (2) ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component (24765:2010). See also appropriateness, recognizability, learnability, operability, user error protection, user interface aesthetics (qv), accessibility (qv)
Usage modes	Qual.	Access control	Requirement for a software system or software product to be able to be used in different modes, i.e. live, test, training, or combinations thereof (COSMIC/IFPUG)
User numbers	Env.		(See 'Distinct user maximum numbers' and 'Concurrent user maximum numbers')
Validation (of data)	Qual.	Data quality	Process of controlling that the data entered into a software system or software product satisfies requirements allocated to software in terms of format, range and type of permitted data values. The process should not allow invalid data to enter a data store and should inform the user of the nature of any defects. (COSMIC/IFPUG, partly based on IEEE 1012-2004)

7.3 Glossary of Project Requirement and Constraint terms

In this section, the word 'project' in any of the terms means a project of any 'Project type' as defined below. Where 'locally' appears in a definition, this could mean 'within your organization' or 'for a given benchmarking exercise', i.e. whatever is appropriate depending on the context.

All term definitions are proposed by COSMIC and IFPUG [29], unless another source is given explicitly.

Term	Project Group	Definition
Customer Satisfaction (project)	Project Quality	The degree to which the customer of a software system or software product is satisfied with the project that developed or enhanced it.
Defect count	Project Quality	The number of defects, within a defined period starting from the date of first implementation of a software system or software product. (Defect counts should be classified by severity and may be target or actual.) Note: 'Defect count' is an attribute of the delivered software system or software product. However, is it not a quality r*equirement* of the product, so it is classified as a Project Requirement and Constraint term, i.e. as an attribute of a project, together with other project performance-related characteristics, such as effort and duration.
Dependencies on other parties	Risk	Dependencies of the project activities on activities that are performed by other parties, e.g. other projects or decision-making bodies, which may affect the progress of the project.
Development environment	Processes	The hardware/software platform used by the development project. To be recorded if different from the Operational Platform. See also the classification of 'Operational Platform type'
Duration (Schedule)	Duration	The elapsed time for a project from Project Start Date (when a project is given resources and starts work) until Project Finish Date (the end of first site implementation). NOTE: Both the estimated and actual duration should be recorded, the latter excluding periods when the project was inactive. See also 'Schedule compression/expansion'
Effort	Resources	The amount of work (in labor units such as staff-months) required to complete a project. Note 1: Effort must be further clarified locally, e.g. it may: • be estimated, planned or actual; • be for a whole project or broken down by activity (see 'work breakdown'); • define whether users, customers, or support staff (e.g. DB specialists, project management office staff, etc.),are included in or excluded from the project team. Note 2: A 'project team' is defined as 'The people who report either directly or indirectly to the project manager. (PMI)

Term	Project Group	Definition
Governance	Processes	(1) The management framework within which project decisions are made, e.g. COBIT, PRINCE, etc. (2) The organization that is accountable for the project, e.g. a Steering Committee, Change Control Board.
Location	Processes	The country(ies) or site(s) where the project takes place. NOTE: Project location may be classified as e.g.: On-site, Multi-site, Near-shore, Multi-country, Off-shore etc. Off-shore: The practice of hiring external organizations to perform work in a country other than the one where the products or services are required or will be used; Near-shore: The practice of hiring external organizations to perform work in neighboring countries.
Post-project review findings	Risk	(1) Measures of actual project performance, e.g. actual productivity, customer satisfaction (project), defect counts, etc. (2) Factors (positive and negative) identified in a post-project review that affected the project outcome, such as unanticipated staff turnover, scope or technology changes, experienced team, etc. Note 1: Ideally the impact on the planned effort and/or schedule should be estimated for each factor. Note 2: see also 'Scope change (Scope creep)'
Process maturity	Processes	The level of adherence of the project processes to a quality standard, e.g. as per CMMI®, SPICE or similar assessment.
Project management method(s)	Processes	A method for dividing project activities into distinct phases (or stages, or iterations) for the purposes of planning and control. NOTE: Common project management methods include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming and agile methods. (Also known as software development methodology, or software development life cycle.)
Project type	Type	A class of software project dependent on its purpose in relation to the software. A project type may be New development, Enhancement, Maintenance, Re-development (ISBSG), where 'Maintenance' includes Adaptive, Corrective, Perfective and Preventive maintenance. Notes: 1. The criterion for when an activity is considered as a maintenance activity and when it is an enhancement project should be defined locally. 2. Maintenance may also be defined as a continuing activity to evolve a system and not as a project

Term	Project Group	Definition
Risk	Risk	(1) The aggregate probability of the project not succeeding in meeting its goals. (COSMIC/IFPUG) (2) An uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives (PMI®).) Risk is usually derived from other data that encompasses the size of the software to be delivered, anticipated requirements stability & validity, staff skills and experience in the problem area, stakeholders cohesion, etc. Risk analysis may also take into account the impact of failing to meet project goals and the uncertainty in the risk assessment.
Schedule compression / expansion	Duration	The degree to which a target project duration ('schedule') is compressed or expanded compared with the estimated duration that is ideally or optimally estimated as needed. (COSMIC/IFPUG) Note: The PMI defines 'schedule compression' as 'taking actions to decrease the total project duration after analyzing number of alternatives to determine how to get the maximum duration compression for the least cost.'
Scope change ("Scope creep")	Risk	Any change to the project's scope. A scope change almost always requires an adjustment to the project cost or schedule (PMI)
Skills and experience level	Resources	The degree to which the human resources who perform the project as defined by the plan have the necessary skills and expertise to perform or support the processes they are assigned to (after CMMI®)
Staffing level	Resources	The number of staff employed on the project. Note: Need to distinguish the average number over the life of the project from the peak number of staff, and planned versus actual.
Team relationships	Resources	Any factor that affects the team's ability to work effectively, e.g. team stability, culture (single/multi-culture) and cohesion, physical working conditions, relationships with non-project staff, e.g. other development teams, users, customers, specialist staff, etc.
Work-breakdown structure	Resources	A deliverable-oriented grouping of project work elements that organizes and defines the total work scope of the project. (PMI)

7.4 Terms that have been excluded from the Glossary

This Chapter contains terms that:

- were considered for inclusion but were NOT included in the Glossary. The reasons for their exclusion are given;
- or are used in the analysis of project performance data, but are not NFR nor project requirements.

Complexity	Composed of more than one or many parts; not simple or straightforward; intricate, difficult. (Chambers). Note that there can be several types of complexity of software: algorithmic, architectural, data, process, operational, semantic, etc. Excluded because the term is ill-defined, with many possible types
Control*	(1) In engineering, the monitoring of system output to compare with expected output and taking corrective action when the actual output does not match the expected output (24765:2010). (2) A requirement that a software system or software product must operate, regulate or direct some other device or process, probably in real-time (COSMIC/IFPUG) Excluded because the requirement to control is really a functional user requirement
Criticality	A requirement that is decisively important for some imperative goal such as the organization's mission, or for human safety (COSMIC/IFPUG) Excluded because it is a very high-level NFR that in practice would be elaborated in more detail
Programming language maturity	A classification of programming language maturity levels of historical development used by the ISBSG (2GL, 3GL, 4GL). Excluded because the distinctions between the three classes are not well defined. (See: 'programming paradigm'.)
Quality	The degree to which a set of inherent characteristics fulfills a set of requirements. (ISO 9000) (See 'customer satisfaction', 'defect level') Excluded because it is too general to be a non-functional requirement.

* Note: When measuring business application software using the COSMIC method, 'control commands' are not measured. The term 'control commands' is used only in the context of business application software [30].

REFERENCES

For the sources and definitions of terms in the Glossary, see section 7.1

(COSMIC publications are all available from the portal of www.cosmic-sizing.org).

[1] ISO/IEC 14143/1:2011, 'Information Technology - software measurement – functional size measurement', 2011.

[2] Butcher, C., 'Delivering mission-critical systems', British Computer Society, Central London Branch meeting, 18th November 2010.

[3] Symons, C.R., 'Accounting for non-functional requirements in productivity measurement, benchmarking and estimating', UKSMA/COSMIC International conference on software metrics and estimating', 27/28 October 2011, www.uksma.co.uk.

[4] Lago, P., Avgerieu, P., Hilliard, R.,. 'Software architecture: farming Stakeholders' concerns, IEEE Software, November/December 2010

[5] ISO/IEC 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SquaRE) – System and software quality models.

[6] COSMIC 'Guideline for early or rapid COSMIC functional size measurement using approximation approaches', July 2015, http://cosmic-sizing.org/publications/guideline-for-early-or-rapid-cosmic-fsm/.

[7] Al-Sarayreh, K., Abran, A., Cuadrado-Gallego, J., 'A Standards-based model of system maintainability requirements', Journal of Software: Evolution and Process, 2013, Vol. 25, no. 5, pp. 459-505.

[8] Saito, Y., Monden A., Matsumoto K., 'Evaluation of non-functional requirements in a request for proposal (RFP)', Nara Institute of Science & Technology, Japan, at International Workshop on Software Measurement (IWSM), Nara, 2012.

[9] Poort, E., van der Vliet, E., 'Estimating the cost of heterogeneous solutions', International Workshop on Software Measurement (IWSM) & MENSURA Conference, Rotterdam, 2014.

[10] COSMIC 'Guideline for the use of COSMIC FSM to measure Agile projects', September 2011, http://cosmic-sizing.org/publications/guideline-for-sizing-agile-projects-with-cosmic/

[11] Albrecht, A.J., 'Measuring application development productivity', IBM application development symposium, Monterey, CA, October 1979

[12] Albrecht, A.J., 'Where function points (and weights) came from', IBM Corporation, 19th February 1986.

[13] Albrecht, A.J., 'AD/M productivity measurement and estimate validation – Draft', IBM Corporate Information Systems & Administration Guideline, AD/M Improvement Program, Purchase, NY, May 1 1984.

[14] Symons, C.R., 'Function point analysis, difficulties and improvements', IEEE Transactions on Software Engineering, Vol. 14, No. 1, January 1988,

[15] IFPUG 'Software Non-functional Assessment Process', www.ifpug.org (as defined in 2011).

[16] IEEE 804, 'Recommended Practices for Software Requirements Specifications', 1983.

[17] ISO/IEC 9126: 1991. 'Software engineering – Software product evaluation – Quality characteristics and guidelines for their use', www.iso.org

[18] Wikipedia entry for 'Non-functional Requirements'.

[19] International Software Benchmarking Standards Group, 'Data Collection Questionnaire …using IFPUG or NESMA Function Points', 5.12, 2009,

[20] Software Engineering Institute, 'A Data Specification for Software Project Performance Measures: Results of a Collaboration on Performance Measurement', Carnegie Mellon University, CMU/SEI-2008-TR-012, July 2008.

[21] 'COCOMO II Model Definition Manual', version 2.1, Centre for Software Engineering, USC, 2000

[22] ISO/IEC 25020:2007, 'Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement reference model and guide', www.iso.org.

[23] Abran A., Al-Sarayreh, KT., Cuadrado-Collego, JJ., 'A standards-based reference framework for system portability requirements', Computer Standards and Interfaces, Elsevier, Vol 35, 2013, pp. 380-395. http://dx.doi.org/10.1016/j.csi.2012.11.003.

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

[25] IEEE 830, 1998, 'Recommended practice for software requirements specifications'.

[26] ISO 24765:2008. 'Systems and software engineering–Vocabulary'.

[27] ISO 2382-1:1993, 'Information technology–Vocabulary–Part 1: Fundamental terms.

[28] COSMIC 'Guideline for sizing Service-Oriented Architecture software', COSMIC method v4.0.1, 2014, http://cosmic-sizing.org/publications/guideline-for-sizing-service-oriented-architecture-software/.

[29] COSMIC and IFPUG 'Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating', joint publication of COSMIC and IFPUG, September 2015. http://cosmic-sizing.org/publications/glossary-of-terms-for-nf-and-project-requirements/

[30] COSMIC 'Measurement Manual: The COSMIC implementation guide for ISO/IEC 19761:2011', v4.0.1, April 2015, http://cosmic-sizing.org/publications/measurement-manual-401/

[31] 'A taxonomy of software projects productivity impact factors', v1.1, Gruppo Utenti Function Point Italia – Italian Software Metrics Association, 15/2/2011, www.gufpi-isma.org/sbc/tassonomia

[32] Kassab, M., Daneva, M., and Ormandjieva, O., 'A meta-model for the assessment of non-functional requirement size', 34th Euromicro conference on software engineering and advanced applications, Parma (Italy) 2008.

[33] Abran, A., 'Software metrics and software metrology', John Wiley and Sons and IEEE CS, 2010, Chapter 8, ISBN: 978-0-470-59720-0.

[34] ISO/IEC TR 9126-2:2003, Software engineering -- Product quality -- Part 2: External metrics. [35] ISO/IEC TR 9126-3:2003, Software engineering -- Product quality -- Part 3: Internal metrics.

[36] ISO/IEC CD 25023.3, Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Measurement of system and product quality, 14 Feb 14.

[37[AUTOSAR (AUTomotive Open System ARchitecture), www.autosar.org

[38] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Fifth Edition, http://www.pmi.org/pmbok-guide-and-standards/pmbok-guide.aspx

[39] ISO/IEC 25012:2008, 'Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Data Quality Model'.

[40] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-functional Requirements in Software Engineering," Kluwer Academic Publishing, 2000

APPENDIX: COSMIC CHANGE REQUEST AND COMMENT PROCEDURE

The COSMIC Measurement Practices Committee (MPC) is very eager to receive feedback, comments and, if needed, Change Requests for this guideline. This appendix sets out how to communicate with the COSMIC MPC.

All communications to the COSMIC MPC should be sent by e-mail to the following address:

mpc-chair@cosmic-sizing.org

Informal general feedback and comments

Informal comments and/or feedback concerning the guideline, such as any difficulties of understanding or applying the COSMIC method, suggestions for general improvement, etc should be sent by e-mail to the above address. Messages will be logged and will generally be acknowledged within two weeks of receipt. The MPC cannot guarantee to action such general comments.

Formal change requests

Where the reader of the guideline believes there is a defect in the text, a need for clarification, or that some text needs enhancing, a formal Change Request ('CR') may be submitted. Formal CR's will be logged and acknowledged within two weeks of receipt. Each CR will then be allocated a serial number and it will be circulated to members of the COSMIC MPC, a world-wide group of experts in the COSMIC method. Their normal review cycle takes a minimum of one month and may take longer if the CR proves difficult to resolve. The outcome of the review may be that the CR will be accepted, or rejected, or 'held pending further discussion' (in the latter case, for example if there is a dependency on another CR), and the outcome will be communicated back to the Submitter as soon as practicable.

A formal CR will be accepted only if it is documented with all the following information.

- Name, position and organization of the person submitting the CR.
- Contact details for the person submitting the CR.
- Date of submission.
- General statement of the purpose of the CR (e.g. 'need to improve text…').
- Actual text that needs changing, replacing or deleting (or clear reference thereto).
- Proposed additional or replacement text.
- Full explanation of why the change is necessary.

A form for submitting a CR is available from the www.cosmic-sizing.org site. The decision of the COSMIC MPC on the outcome of a CR review and, if accepted, on which version the CR will be applied to, is final.

Questions on the application of the COSMIC method

The COSMIC MPC regrets that it is unable to answer questions related to the use or application of the COSMIC method. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org site for further detail.