



The COSMIC Functional Size Measurement Method

Version 4.0.1

**Guideline for Early or Rapid
COSMIC Functional Size
Measurement**
by using approximation approaches

July 2015

Acknowledgements

Guideline authors and reviewers 2015 (alphabetical order)

Alain Abran École de Technologie Supérieure, Université du Québec Canada	Maya Daneva University of Twente the Netherlands	Jean-Marc Desharnais École de Technologie Supérieure, Université du Québec Canada
Cigdem Gencel Deiser Spain	Harold van Heeringen Metri the Netherlands	Arlan Lesterhuis the Netherlands
Roberto Meli DPO Italy	Luca Santillo Agile Metrics Italy	Hassan Soubra Ecole Supérieure des Techniques Aéronautiques et de Construction Automobile France
Charles Symons United Kingdom	Sylvie Trudel École de Technologie Supérieure, Université du Québec Canada	Francisco Valdés Souto Spingere Mexico
Frank Vogelesang * Ordina the Netherlands	Chris Woodward CW Associates United Kingdom	

* Editor of this Guideline version1.0

DOI 10.13140/RG.2.1.4195.0567

Copyright 2015. All Rights Reserved. The Common Software Measurement International Consortium (COSMIC). Permission to copy all or part of this material is granted provided that the copies are not made or distributed for commercial advantage and that the title of the publication, its version number, and its date are cited and notice is given that copying is by permission of the Common Software Measurement International Consortium (COSMIC). To copy otherwise requires specific permission.

Public domain versions of the COSMIC documentation, including translations into other languages can be found on the internet at www.cosmic-sizing.org.

Version Control

The following table gives the history of the versions of this document

DATE	REVIEWER(S)	Modifications / Additions
2007	COSMIC Measurement Practices Committee	The first version of the contents of this Guideline appeared in Chapters 1 and 2 of the 'Advanced & Related Topics' document of the COSMIC method version 3.0, December 2007. This Guideline replaces those Chapters for v4.0.1 of the COSMIC method.
Oct. 2013 Sept. 2014	Subject matter experts	Preliminary version of the Guideline issued for review by authors of the 2013 IWSM paper [1] and international experts Extended draft of the Guideline is issued for new review
July 2015	COSMIC Measurement Practices Committee	First public version of this document

Foreword

The COSMIC method provides a standardized way of measuring a functional size of software from the domains commonly referred to as 'business application' (or 'MIS'), 'real-time' and infrastructure software, certain types of scientific/engineering software and hybrids of these.

In practice it is sometimes sufficient or necessary to measure a functional size approximately¹. Typical situations where such a need arises are:

- early in the life of a project, before the Functional User Requirements (FUR) have been specified down to the level of detail where the precise size measurement is possible;
- when a size measurement is needed, but there is insufficient time or resources to measure using the standard method; a quick approximate size measurement will be acceptable;
- when the quality of the documentation of the actual requirements is not good enough for precise measurement.

Purpose of this Guideline

The purpose of this Guideline is to describe the current state of the art with regard to early or rapid COSMIC functional size measurement using approximation approaches. All proposed COSMIC approximation approaches rely on an approximation of the size and/or numbers of the functional processes of the software to be measured. A consequence of the fact that the size of a single functional process has no finite limits is that multiple COSMIC approximation approaches have been developed. Therefore this document describes a number of approximation approaches with their pros and cons, their recommended area of application and their validity, rather than document, say, one particular COSMIC approximation approach.

The reader is assumed to be fully familiar with the standard COSMIC functional size measurement method.

Chapter 1 describes reasons why it may be necessary to measure functional sizes approximately, how actual requirements are often expressed at varying levels of detail (known as 'levels of granularity') and some general principles of how to recognise and apply ways of measuring such actual requirements approximately.

Readers of this Guideline who are new to approximate sizing are strongly advised to first read Chapter 1 on the General Principles of approximate sizing. For details on the background of approximate sizing, see Chapters 9, 10 and 11.

Chapters 2 – 7 describe six different approaches to measuring functional sizes approximately.

If a quick start to approximate sizing is needed, we recommend first try using one of the approaches described in Chapters 2, 3 or 4, for which there is reported practical experience.

Chapter 8 describes some approximation approaches that cannot be applied directly with the information in this Guideline, but for which we foresee promising application.

Chapter 9 gives more details on how to compare requirements at different levels of granularity and different levels of decomposition.

Chapter 10 gives more details on how to adapt or calibrate the approaches of Chapter 2 – 7 to local needs.

Chapter 11 deals with approximate sizing of changes to software and with 'scope creep'

Chapter 12 draws general conclusions on approximate sizing.

COSMIC Measurement Practices Committee

July 2015

¹ Instead of describing this subject as approaches to 'approximate sizing', it might be more accurate to describe it as approaches to 'estimating sizes'. However, the word 'estimating' is strongly associated with methods of estimating project costs, effort or duration, etc. To avoid confusion, we therefore prefer to write about 'approximate sizing'.

Table of Contents

Acknowledgements	2
Version Control.....	3
Foreword	4
1. GENERAL PRINCIPLES OF APPROXIMATE SIZING	7
1.0 Note on Terminology	7
1.1 When is approximate COSMIC sizing needed?.....	7
1.2 Approaches to approximate functional size measurement	8
1.2.1 <i>The localization principle</i>	8
1.2.2 <i>General Principles</i>	8
1.2.3 <i>Measurement scaling</i>	9
1.2.4 <i>Approximate sizing by classification</i>	9
1.2.5 <i>Accuracy of approximate sizing</i>	9
1.3 Applicability of approximation approaches described in this Guideline	10
1.4 Levels of Granularity of Actual Requirements.....	10
1.5 Quality of Actual Requirements.....	11
2. THE AVERAGE FUNCTIONAL PROCESS APPROXIMATION.....	12
2.1 Origin and approximation mechanism.....	12
2.2 Applicability and reported use	12
2.3 Strengths and weaknesses	12
2.4 Recommended area of application	13
2.5 Research developments	13
2.6 Practical use in enhancement projects	13
3. THE FIXED SIZE CLASSIFICATION APPROXIMATION.....	14
3.1 Origin and approximation mechanism.....	14
3.2 Applicability and reported use	15
3.3 Strengths and weaknesses	15
3.4 Recommended area of application	15
3.5 Research developments	15
4. THE EQUAL SIZE BANDS APPROXIMATION	16
4.1 Origin and approximation mechanism.....	16
4.2 Applicability and reported use	16
4.3 Strengths and weaknesses	17
4.4 Recommended area of application	18
4.5 Research developments	18
5. AVERAGE USE CASE APPROXIMATION.....	19
5.1 Origin and approximation mechanism.....	19
5.2 Applicability and reported use	19
5.3 Strengths and weaknesses	19
5.4 Recommended area of application	20
5.5 Research developments	20

6.	EARLY & QUICK COSMIC APPROXIMATION	21
6.1	Origin and approximation mechanism.....	21
6.2	Applicability and reported use	22
6.3	Strengths and weaknesses	23
6.4	Recommended area of application	23
6.5	Research developments	23
7.	EASY FUNCTION POINT APPROXIMATION	24
7.1	Origin and approximation mechanism.....	24
7.2	Applicability and reported use	25
7.3	Strengths and weaknesses	25
7.4	Recommended area of application	25
7.5	Research developments	25
8.	EMERGING APPROXIMATION APPROACHES.....	26
8.1	Approximation from informally written textual requirements	26
8.2	Approximation using fuzzy logic – the EPCU model.....	26
9.	DIFFERENT LEVELS OF GRANULARITY AND DECOMPOSITION	28
9.1	The evolution of requirements in the early stage of a large software project	28
9.2	Measuring at varying levels of granularity - the ‘Everest’ system	31
9.3	Measuring at varying levels of granularity and decomposition in a software architecture	32
	9.3.1 <i>Description and analysis of the software architecture</i>	32
	9.3.2 <i>Sizing the Logical Network Element</i>	34
	9.3.3 <i>Discussion of the analysis of the LNE example</i>	34
	9.3.4 <i>Computing ‘scaling factors’ for the LNE example</i>	35
9.4	Functional size measurements and standard levels of decomposition.....	35
10.	LOCALIZATION PRINCIPLES.....	38
10.1	Establishing a locally-defined approach.....	38
11.	APPROXIMATE SIZING OF CHANGES OF FUNCTIONALITY AND SCOPE CREEP	40
11.1	Approximate sizing of changes to functionality	40
11.2	Approximate sizing and scope creep	40
12.	CONCLUSIONS ON APPROACHES TO APPROXIMATE SIZING	42
	REFERENCES	43
	GLOSSARY OF TERMS.....	45
	APPENDIX A: COSMIC CHANGE REQUEST AND COMMENT PROCEDURE	46

GENERAL PRINCIPLES OF APPROXIMATE SIZING

1.0 Note on Terminology

The COSMIC method measures the 'Functional User Requirements' (or FUR) of software – see the 'Measurement Manual' v4.0.1 for the full definition. From v4.0 onwards, COSMIC uses this term to apply to requirements that are specified in sufficient detail for a precise COSMIC Functional Size Measurement.

Approximate sizing approaches are designed to be applied when this level of detail is not (yet) available. In this Guideline, we will therefore refer to the 'actual requirements' as the subject that approximate sizing approaches are designed to measure. The term 'actual requirements' includes, of course, non-functional requirements. But many actual requirements that appear initially as non-functional evolve, as a project progresses into functional requirements that can be measured by the same approximate sizing approaches.

For the definition of general COSMIC terms used in this Guideline, see the Measurement Manual [2]. For terms specific to this Guideline, see the Glossary.

1.1 When is approximate COSMIC sizing needed?

There are three main circumstances in which only an approximate² COSMIC functional size may be possible:

- when a size measurement is needed rapidly and an approximate size measurement is acceptable if it can be measured much faster than with the standard method. This is known as 'rapid sizing';
- early in the life of a project before the actual requirements have been specified in sufficient detail for a precise size measurement. This is known as 'early sizing';
- in general, when the quality of the documentation of the actual requirements is not good enough for a precise size measurement.

Rapid sizing can be valuable when a very large piece of software or, say, a whole software portfolio needs to be measured but it would take too much time and money to measure precisely, and approximate sizes are acceptable.

Early in a project, an approximate size measurement may be needed for project effort estimating purposes, well before the FUR have been worked out in the detail needed for a precise measurement. In such cases the actual requirements would typically exist at various levels of detail and are laid down in artefacts that are not standardized in any way. For example, some actual requirements may exist at the Use Case level whilst others have been worked out in more detail. But Use Cases themselves may express requirements at different levels of detail. These different levels of detail are known as different 'levels of granularity'.

² Instead of describing this subject as approaches to 'approximate sizing', it might be more accurate to describe it as approaches to 'estimating sizes'. However, the word 'estimating' is strongly associated with methods of estimating project costs, effort or duration, etc. To avoid confusion, we therefore prefer to write about 'approximate sizing'.

Whether measuring precisely or approximately, Measurers should always try to get as much information and details on the description of the actual requirements. Assumptions can then be used to make the functional size measurement as precise as possible. In practice, software functional size measurements are often needed of actual requirements that are still incomplete and unclear. A mix of approximate and precise measurements may be the only possibility in these circumstances.

In practice, approximate sizing is probably needed more often than precise sizing.

1.2 Approaches to approximate functional size measurement

1.2.1 The localization principle

The fact that approximate sizing approaches are based on artifacts that may vary from place to place implies that the scaling factors need to be calibrated locally. However, it is difficult to define 'locally' exactly. 'Locally' can range from a single development or maintenance team to a state, nation or even a continent. In this document, 'locally' implies that the environment in which the scaling factors for the approximation approach have been defined is representative of the environment the approximation approach is to be used in.

Scaling factors should be established locally because the artifacts at level(s) of granularity higher than the functional process that will be measured and used to establish the scaling factors can only be defined (or illustrated by standard examples) locally.

More guidance on the localization principle is given in chapter 10.

1.2.2 General Principles

Steve McConnell described three levels of determining software size [3]:

- **Count:** If detailed information is available, the most accurate way of determining the size is to count. Adapting this to the COSMIC method means applying the standard method at the functional process level of granularity and counting the data movements, i.e. following the standard measurement process.
- **Compute:** If not enough information is available at the desired level of detail, count something that is available and then compute the answer by using calibration data. Adapting this to the COSMIC method means applying an approximation approach to a higher level of granularity and *scaling* this measurement to the functional process level of granularity.
- **Judge:** Experts can give an approximation of the size, based on a mental model established on experience. This expert judgment is the least accurate means of approximation. The accuracy can be strengthened if the expert judgment can be tied to concrete size information. Adapting this to the COSMIC method means *classifying* some available 'objects' (e.g. high-level statements of actual requirements or a list of Use Cases) and assigning a size based on the classification and knowledge of scaling factors established at the 'Compute' level.

The approaches described in this guideline in the chapters 2 to 7 are at the 'Compute' level of determining size, and the approaches described in this guideline in the chapter 8 are at the "Judge" level of determining size, as described by McConnell.

The approximation approaches described in chapters 2 to 7 use scaling, or a combination of classification and scaling. For one or more software artifacts for which actual requirements exist at a higher level of granularity than is needed for a precise COSMIC functional size measurement (i.e. the actual requirements are not in sufficient detail for a precise measurement):

- **Scaling:** Count or measure each actual requirement and multiply the count or measurement by a number, the 'scaling factor' to determine its COSMIC functional size. A scaling factor is determined by a calibration process.
- **Classification:** classify each actual requirement and assign a size to it (i.e. apply a scaling factor) that represents the COSMIC functional size for that class of the software.

1.2.3 Measurement scaling

The general principle of any scaling approach is to find some way of measuring the approximate size of locally-defined artifacts of actual requirements at a high level of granularity and then to measure the same requirements in units of CFP when they are known at the functional process level of granularity. A 'scaling factor' is a ratio that is used to convert measurements on locally-defined high-level artifacts to sizes expressed in CFP. This is illustrated in Table 1.1 below.

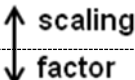
Level of granularity of the Actual Requirements	Measurement method	Measurement standard
Actual requirements at a high level of granularity derived from e.g.: <ul style="list-style-type: none">• high-level statement of actual requirements for the software• architecture artifacts• high-level view of existing software expressed in locally-defined (countable) units e.g. Use Cases, or in CFP	An 'Approximate approach' to the COSMIC measurement method. Calibrated locally	The size of the locally defined unit, expressed in local units or in CFP 
The functional process level of granularity	COSMIC measurement method	The CFP

Table 1.1 – Scaling of sizes between different levels of granularity

Scaling factors should be established locally. For more guidance, see chapter 10.

1.2.4 Approximate sizing by classification

The general approach of classification is that each part of the actual requirements to be sized approximately is allocated to a pre-defined class (or reference piece) of requirements whose size has been calibrated in CFP, i.e. each class has its own scaling factor. A size is thus assigned to each part of the actual requirements, based on its classification.

The scaling factors for each class can only be calibrated locally because the reference(s) that are used to establish the set of classes can only be defined (or illustrated by standard examples) locally.

It is highly desirable that an approximation approach that uses classification provides objective rules or criteria, or typical examples to assist the correct classification.

1.2.5 Accuracy of approximate sizing

Any approach to approximate sizing is the result of a trade-off between ease and speed of measurement versus loss of precision. Therefore the accuracy of each approach should be established and reported. See section 10.1 for guidance on establishing the accuracy.

Before selecting an approximation approach described in this Guideline, the Measurer should first check whether the scaling factors and/or classification systems described here can be applied to the local measurement task. Best practice is to finalize the chosen approach by recalibrating the scaling factors and/or refining the classes of the approach as described here, using local data, illustrated with local examples.

1.3 Applicability of approximation approaches described in this Guideline

Most of the practical experience described in this Guideline has been obtained from applying the approximation approaches described in Chapters 2, 3 and 4 to measuring the size of the actual requirements for new business application software.

For real-time software, an example in Chapter 4 describes the result of applying an approximation approach successfully to some very complex real-time avionics software. Another example in section 9.3 describes the approximate sizing of the functionality of a complex telecoms software architecture at various levels of granularity.

The approaches described in Chapters 5 – 7 have no publicly reported practical use as far as we are aware at the time of writing. The approaches described in Chapter 9 are still in the research phase.

The approximation approaches described here are applicable to the actual requirements:

- for new software and for enhancements to existing software that require whole new additions of functionality
- for software from any domain, e.g. business, real-time or infrastructure.

We are not aware of any reported experience of applying these approximation approaches to size the actual requirements for enhancements that involve many changes (adds, modifies and deletes) to existing software. However, these approaches may be applied to size such enhancements, provided great care is taken for the calibration process. See Chapter 11 for more.

1.4 Levels of Granularity of Actual Requirements

In the circumstances in which only an approximate COSMIC functional size may be possible, Measurers should be aware of the level of granularity of the software artifacts that are used to approximate the functional size.

A problem for all approximation approaches is that there is no way of unambiguously defining standard levels of granularity higher than the functional process level. A set of higher levels of granularity might be named as, for example, the 'Use Case level', the 'Component' level, the 'Sub-system' level. But these levels can only be properly defined locally, usually with the aid of examples.

Furthermore, research has shown [15] that Measurers, especially if inexperienced, often do not realize that actual requirements are expressed at different levels of granularity and/or fail to distinguish the levels. This is one of the commonest problems faced when intending to measure functional sizes, whether precisely or approximately.

Chapter 9 is devoted to the aspects that must be considered when measurements are made at different levels of granularity of actual requirements. Measurers are encouraged to take note of the aspects discussed in this chapter, before applying any of the approaches described in this Guideline.

1.5 Quality of Actual Requirements

Classifying the quality of the various parts of actual requirements by using the scheme of the 'Guideline for assuring the accuracy of measurements' [16] can help determine the accuracy of an approximate size measurement. This quality classification scheme defines five levels for the quality of actual requirements of the functionality, to which we have added a sixth level for this Guideline: *Not mentioned (An 'unknown unknown')*.

Table 1.2 below shows the six levels and their definitions and which approximation approaches (as defined in the following Chapters) can be applied.

Functional Process Quality Level	Quality of the functional process definition	Approximate sizing approaches that can be used
Completely defined	Functional process and its data movements are completely defined	Use standard COSMIC FSM method
Documented	Functional process is documented but not in sufficient detail to identify the data movements	See Chapters 2 – 8
Identified	Functional process is listed but no details are given of its data movements ³	See Chapters 2, 5, 6 and 8
Counted	A count of the functional processes is given, but there are no more details ³	See Chapters 2, 5, 6 and 8
Implied (A 'known unknown')	The functional process is implied in the actual requirements but is not explicitly mentioned	See Chapters 2 – 7, or use expert judgment (see 1.2.1)
Not mentioned (An 'unknown unknown')	Existence of the functional processes is completely unknown at present	Add a contingency for 'scope creep' on the basis of expert judgment (see 1.2.1), or past experience (see 11.2)

Table 1.2 – Functional Process Quality levels related to the approximation approaches

Given this guidance, we **strongly recommend** that Measurers do not use any of the approximation approaches described in this Guideline as simple 'recipe books'. Always:

1. examine the actual requirements to be measured closely so that you understand the level of granularity, the completeness and quality of the actual requirements before starting to use an approximation approach;
2. try to obtain more detail than is given in the actual requirements from an expert in the software so that you can at least list and name the functional processes;
3. check an approximation approach using local requirements and measurements to ensure it produces reasonably accurate sizes in your local environment and, if necessary, calibrate the approach locally before using it for your own real measurements. See Chapter 10 for more on localization.

³ The approaches described in chapters 2-7 are at the 'Compute' level of determining size and the approaches described in chapter 8 are at the 'Judge' level of determining size, as described in section 1.2.2.

THE AVERAGE FUNCTIONAL PROCESS APPROXIMATION

2.1 Origin and approximation mechanism

The average functional process approximation was first introduced in version 2.2 of the COSMIC method [4].

This is the simplest process for obtaining an approximate size of a piece of software. It may be used when the actual requirements a piece of software are known only to the level of functional processes but not to the level of data movements.

A Sampling and calculation of the size of an average functional process

1. Identify a sample of actual requirements whose functional processes and data movements have been defined in detail, with characteristics similar to the actual requirements of the software to be measured.
2. Identify the functional processes of these sample requirements.
3. Measure the sizes of the functional processes of these sample requirements precisely using the standard COSMIC method.
4. Determine the average size, in CFP, of the functional processes of these sampled requirements (e.g. average size = 8 CFP). '8' is then the scaling factor for this approach.

B Approximation using the calculated average of the sample

1. Identify and count all the functional processes of the actual requirements of the software to be measured (e.g. = 40 functional processes = the scaling factor).
2. The approximate functional size of the actual requirements of the software to be measured is estimated to be (number of functional processes x scaling factor) = $40 \times 8 = 320$ CFP.

2.2 Applicability and reported use

In organizations that have established a COSMIC measurement practice this approach is used to produce a first ball-park approximation of the size.

In 2005 Vogelezang reported [5] that in different industry sectors different sizes were measured for an average functional process, i.e. for the scaling factor. This supports our recommendation that the use of this approximation approach should always be calibrated locally.

2.3 Strengths and weaknesses

Strength:

- It is easy to use.

Weaknesses:

- This approach requires sampling and calculation of an average functional process, based on detailed measurements from within the same localization (see chapter 10). This data is often not (yet) available.
- This approach is (assumed to be) domain dependent.

2.4 Recommended area of application

This approximation is valid as long as there is sufficient reason to assume that the sample used to calculate the size of the average functional process is representative for the software of which the functional size is approximated. See also chapter 8.

2.5 Research developments

In 2009 Van Heeringen, Van Gorp and Prins, carried out measurements to compare the accuracy of the average functional process approximation with precise measurements using the standard COSMIC measurement method. In [6] they compared approximations of 24 pieces of software from different organizations against the precise measurements.

In 2013 De Marco, Ferrucci and Gravino reported good results with this approach to estimate the development effort for web application development [7].

2.6 Practical use in enhancement projects

Large proportions of projects must not only create new functional processes, but must also modify existing functional processes. In practice the following approach has been observed:

1. While measuring projects, each functional process is marked as either 'New' or 'Modified'.
2. The average size for a new and for a modified functional process were established separately. The average values obtained varied depending on the domain. But typically, the average size of a modified functional process was about half the average size of a new functional process.
3. When approximating the size of a project at an early stage, the new and modified functional processes must be identified, counted and multiplied by their respective average sizes.

THE FIXED SIZE CLASSIFICATION APPROXIMATION

3.1 Origin and approximation mechanism

The fixed size classification approximation was first introduced in the 'Advanced and Related Topics' document in version 3.0 of the COSMIC method [8].

The approach depends on identifying a typical size classification of the functional processes in the piece of software to be measured. A corresponding size, or scaling factor, is then assigned to each functional process.

A statement of actual requirements must be analysed to identify the functional processes and to classify each of them according to their size in one of three or more size classes called, for instance, Small, Medium and Large. The table below shows an example of a set of size classes that is in actual use in a specific business organization. The rows of the table show the three possible size classes for this organization and the total number of CFP that must be assigned to a functional process in each group (for instance, if one small functional process is identified, it is assigned a scaling factor of 5, so that its size is 5 CFP). To force the Measurer to make a deliberate choice of size, the step size between the classes is taken to be fairly wide, at 5 CFP.

The four columns #E, #X, #R and #W explain why the functional process of a given size is assigned the number of CFP. For instance, a small functional process is assumed to consist of 1 Entry, 1 Read, 1 Write and 1 Exit data movements, because only one object of interest is accessed. The fifth column 'Error messages' adds in one Exit for error/confirmation messages.

Classification	Size (CFP)	#E	#X	#R	#W	Error messages
Small	5	1	1	1	1	1
Medium	10	2	2	3	2	1
Large	15	3	3	4	4	1
...						

Table 3.1 – Fixed size classification from [8]

If the functional size of some actual requirements must be estimated early in the development process, each actual requirement is assigned one or more functional processes, together with their appropriate size classification and corresponding size approximation. Use of a table such as 3.1 above helps the Measurer to make a quicker decision on the assignment of the size class for each functional process. If necessary, the table may be extended to accommodate one or more additional sizes, such as 'very large' of 20 CFP. When well calibrated, this approach should give a more accurate functional size than the average size approach of Chapter 2 (although the observations of the equal size bands approach in Chapter 4 should also be considered).

3.2 Applicability and reported use

This approach has been used extensively by a large business organization in the Netherlands. The approach was successful within that organization. There is no public information on the use and accuracy of this approximation other than in this organization.

3.3 Strengths and weaknesses

Strengths:

- It is easy to use.
- It can be implemented in a simple way.
- As the approximate sizes are based on an expected number of objects of interest to be accessed (hence the data movements), knowledge of this factor helps the Measurer to decide which classification to assign to a functional process.

Weaknesses:

- This approach is (assumed to be) domain dependent.
- Assigning functional processes to a size class is a subjective element of this approximation approach, which reduces the strength of the approximation. See also Chapter 1.

3.4 Recommended area of application

The numbers used in the example in section 3.1 suggest that this approach is only applicable to software with small, relatively simple functional processes of limited size range. The approach can easily be extended to account for functional processes with more data movements. For software with larger functional processes in a more extended size range, other sizes of the classes will apply with a different accuracy.

This approximation is valid as long as there is sufficient reason to assume that the assigned size classification is representative for the software of which the approximate functional size is to be measured. See also chapter 10.

It is highly desirable that objective local rules are determined to assist Measurers in assigning the correct classification.

3.5 Research developments

There are currently no known research developments for this approximation approach.

THE EQUAL SIZE BANDS APPROXIMATION

4.1 Origin and approximation mechanism

The equal size bands approximation was first introduced in version 2.2 of the COSMIC method [4].

This approximation approach can be refined to give a more accurate result if sufficient size data are available for an accurate calibration relevant to the local measurement need. In the 'Equal Size Bands' approach, the functional processes are classified into a small number of size bands. The boundaries of the bands are chosen in the calibration process so that the total size of all the functional processes in each band is the same for each band. (So if, for example, the choice is to have three bands, then the total size of all the functional processes in each band will contribute 33% to the total size of the software being measured.) This approximation approach can be refined to give a more accurate result if sufficient size data are available for a more accurate calibration.

4.2 Applicability and reported use

Vogelezang and Prins have reported on using this approach for early sizing, having carried out a calibration using measurements on 37 business application developments, each of total size greater than 100 CFP. [9]

It was decided to use four size bands to make a distinction between relatively small processes, medium-sized processes, large and very large ones. The average sizes of each band when the 2,427 functional processes of the 37 applications were distributed over the four bands are:

Band	Average size of a Functional Process	% of total Functional Size	% of total number of Functional Processes
Small	4.8	25%	40%
Medium	7.7	25%	26%
Large	10.7	25%	19%
Very Large	16.4	25%	15%

Table 4.1 – Equal size bands from 37 business applications

This same approach was used to calibrate one component of a major real-time avionics system (of total size 10,875 CFP), giving the following results for the four bands.

Band	Average size of a Functional Process	% of total Functional Size	% of total number of Functional Processes
Small	5.5	25%	49%
Medium	10.8	25%	26%
Large	18.1	25%	16%
Very Large	38.8	25%	7%

Table 4.2 – Equal size bands from a major component of an avionics system

Note the similarity between the *numbers* of functional processes in each of the four bands in spite of the totally different types of software. However, the average size of the functional processes in each band is quite different, especially for the large and very large bands. This emphasizes the need for local size calibration.

To size a new piece of software the functional processes of the new piece are identified, they are classified as 'Small', 'Medium', 'Large' or 'Very Large'. In the next step, the average sizes of each band (such as listed above but preferably calibrated locally) are then used to multiply the number of functional processes of the new piece of software, in each band respectively to get the total estimated approximate size.

The advantage of this approach is that at the end of a new approximate size measurement for some new software, the Measurer can check if the contribution to the total size of the functional processes of the new software in each size band is close to the 25% assumed by this 'Equal Size Band' approach. If so, the calibration will have been suitable for this new measurement. If not, the Measurer should consider whether the calibration has been accurate enough.

The accuracy of any calibration of classification sizes for this approach is critically important for accurate sizing since functional processes typically exhibit a skewed size distribution, as illustrated by both sets of data given above. In other words, software systems typically have many functional processes of small size and few of larger sizes. More attention must therefore be paid to accurate sizing of the few 'large' and the even fewer 'very large' functional processes to get an accurate total size.

4.3 Strengths and weaknesses

Strengths:

- It is easy to use.
- The approach has been shown to be applicable for software in both the business application and real-time domains.
- Tool support is available⁴.

⁴ This approximation approach is integrated in the freeware sizing tool SIESTA that is available from Sogeti in the Netherlands. See the vendor section on www.cosmic-sizing.org.

Weaknesses:

- Special care should be taken to ascertain that the number of bands is chosen in such a way that the bands are significantly far enough apart. This can be tested using variance analysis.
- When carrying out an approximate size measurement, assigning functional processes to a size class is a subjective element of this approximation approach, which reduces the strength of the approximation. See also Chapter 1.
- When there are a few functional processes in the Very Large band, the average size of this band must be measured with great care, and similarly the allocation of functional processes to the band requires care.

4.4 Recommended area of application

This approach is recommended for approximate sizing of software that has a significantly skewed distribution of the size of functional processes.

This approximation is valid as long as there is sufficient reason to assume that the assigned size classification is representative for the software of which the approximate functional size is to be measured. See also chapter 10.

It is highly desirable that objective local rules are determined to assist Measurers in assigning the correct classification.

The conclusion we can draw from these results is that the greater the skew, the greater the advantage of this approach for accuracy over the approaches of Chapters 2 and 3.

4.5 Research developments

In 2009 Van Heeringen, Van Gorp and Prins, carried out measurements to compare the accuracy of the equal size bands approximation with precise measurements using the standard COSMIC measurement method. In [6] they compared approximations of 24 pieces of software from different organizations with the precise measurements. The results were that on average the difference between the approximated size and the size that was determined with the standard FSM procedure was only 1.26%. This means that for 90% of all results in the study, the Equal Size Bands approach gave a result within a reach of -15% to +25% of the corresponding result found with the standard COSMIC approach.

In 2012 the results of the study done by Vogelesang and Prins in 2005 [9] were tested using a fuzzy logic model to approximate the functional size of the C-registration system Case Study by Valdes Souto and Abran [10]. This test showed that the equal size bands approximation was a better approximation approach than the EPCU fuzzy logic model used in the experiment when the FP are known and fully documented as in the C-registration Case Study. See section 8.2 for a description of the EPCU model.

In 2013, in his PhD thesis on the development of a scaling factors framework to improve the approximation of software functional size, Almakadmeh asserted that a solid approximation framework could be designed by combining the equal size bands approximation with the quality rating mechanism from the COSMIC Guideline for Assuring the Accuracy of Measurement [17].

AVERAGE USE CASE APPROXIMATION

5.1 Origin and approximation mechanism

The average use case approximation was first introduced in the 'Advanced and Related Topics' document in version 3.0 of the COSMIC method [8].

The principle of the approximation is similar to the average functional process approximation from chapter 2, but on a higher level of granularity, namely the use case.

Local calibration might determine that a (locally-defined) use case comprises, on average, 3.5 functional processes, each of average size 8 CFP (as in the example in Chapter 2). Hence the average size of a use case according to this local definition, is $3.5 \times 8 = 28$ CFP per use case.

For a new project with 12 use cases, the software size would be $12 \times 28 = 236$ CFP.

Thus, with this calibration, identifying the number of use cases early in a development project's life will provide a basis for making a preliminary estimate of software size in units of CFP. The uncertainty on this approximate size will be greater than that with the approach discussed in e.g. Chapter 2. This is because the scale factor 28 is the product of two other scale factors (8 and 3.5) which are themselves estimated. (The result might therefore be better expressed as, for example, 240 plus or minus x%, where the 'x%' has been obtained by appropriate analysis).

5.2 Applicability and reported use

In literature there is no reported use of this approximation.

5.3 Strengths and weaknesses

Strengths:

- It is easy to use if there is a local standard on what is a Use Case, more specifically describing the expected level of granularity of a Use Case.

Weakness:

- There is evidence that the approach would not work unless the organization producing Use Cases adopts some sort of standard to ensure consistency in their size. Unfortunately, the definition of a Use Case in the Unified Modeling Language does not ensure that any two analysts working on the same set of actual requirements will agree on its representation by the same set of Use Cases.
- The scaling factor is the product of two other scaling factors which are themselves estimated. This increases the uncertainty of the approximation result.

- Sufficient historical data are required for localization of the size of an average use case. A study by Gencel and Symons [11] of practices in a very large software house showed that different parts of the software house had quite different ideas on what a Use Case is, so it is of vital importance to verify the homogeneity of the historical data as well.

5.4 Recommended area of application

This approximation is valid as long as there is sufficient reason to assume that the assigned size classification of an average use case is representative for the software of which the functional size is approximated.

5.5 Research developments

A guideline for mapping the concepts of UML to those of the COSMIC method is under development.

A study by Gencel and Symons [11] of practices in a very large software house showed that different parts of the software house had quite different ideas on what a Use Case is. In one part of the software house there was a fairly consistent ratio of functional processes per Use Case. In another part, this ratio varied widely. This finding should be taken into account in the local calibration when this approximation approach is used.

EARLY & QUICK COSMIC APPROXIMATION

6.1 Origin and approximation mechanism

The Early & Quick COSMIC approximation approach is an adaptation of the Early & Quick Function Points approach [12]. The Early & Quick FP approach was originally proposed in 1997 [13] for IFPUG Function Point Analysis, in order to help Measurers in sizing software systems starting from non-detailed information, typically with reduced effort and time compared to that needed for standard measurements. In 2007 [14] the approach was published as an open specification under the Creative Commons Attribution-NoDerivatives 4.0 Public License [15]. Later, the Early & Quick approach has extended its applicability domain to the COSMIC measurement method [16]. This was established by taking advantage of enhancement opportunities derived from local or global measurement data sets, like the ISBSG benchmarking data base and others [5][9].

The Early & Quick COSMIC approximation approach combines scaling and classification approaches. It permits the use of different levels of granularity for different branches of the system on different levels of decomposition. The overall size approximation (which is a 3-point estimate of a minimum, most likely, and maximum size) is the sum of the individual components' size approximations.

The Early & Quick COSMIC approximation approach is based on the capability of the Measurer to classify a part of the actual requirements as belonging to a particular functional category. An appropriate reference table then allows the Measurer to assign a CFP average value for that item (this is applied for software in each identified layer of the software architecture separately, as per the standard COSMIC method). Each function can be categorized, in order of increasing magnitude and decreasing number of composing elements, as a Functional Process, Typical Process, General Process, or Macro-Process.

- a) In the Early & Quick approach a Functional Process⁵ (FP) is the smallest process, performed with the aid of the software system, with autonomy and significance characteristics. It allows the user to attain a unitary business or logical objective at the operational level. It is not possible, from the user's point of view, to proceed to further useful decomposition of a Functional Process without violating the principles of significance, autonomy and consistency of the system. A Functional Process can be Small, Medium, Large or Extra Large, depending on its estimated number of data movements.
- b) A Typical Process (TP) is a set of the four basic user operations: Create, Retrieve, Update and Delete (CRUD) on data describing a particular object of interest. These Typical Processes are frequently found in business application software.
- c) A General Process (GP) is a set of medium Functional Processes and may be thought as an operational sub-system of the application. A GP can be Small, Medium or Large, based on the estimated number of Functional Processes that it contains
- d) A Macro-Process (MP) is a set of medium General Processes and may be thought as a relevant sub-system of the overall Information System of the user's organisation. A MP can be Small, Medium or Large, based on the estimated number of General Processes that it contains.

⁵ This description of a functional process differs from the COSMIC method standard definition. It is planned that in a future version the COSMIC standard definition is adopted.

Each level is built up on the basis of the lower one. An appropriate reference table then allows the Measurer to assign a CFP average value for that item. See section 6.2 for the reference table.

In order to make an estimate the Measurer (after having gone through the preliminary steps of the standard method - defining boundaries of applications, layers and scope of measurement)⁶ has to classify each part of the actual requirements as belonging to one level of the proposed categories. An assignment table will give the related size measure of that requirement. In this way not only leaves of the functionality tree may be directly quantified but also intermediate nodes.

6.2 Applicability and reported use

The Early & Quick COSMIC approximation approach is based on the capability of the Measurer to classify a part of the actual requirements as belonging to a particular functional category. Each part of the actual requirements is to be classified, in order of increasing magnitude and number of composing elements at one of four levels, as a Functional Process, Typical Process, General Process, or Macro-Process. The reference table then allows the Measurer to assign a CFP value for that part of the actual requirements (this is applied for each identified level separately). The most recently published values are depicted in table 6.1 below [16]. They are regarded as release candidate values by the authors and should not be used without local testing of the calibration.

Type	Level	Ranges / COSMIC Equivalent	min CFP	most likely	max CFP
Functional Process	Small	1 - 5 Data movements	2.0	3.9	5.0
	Medium	5 - 8 Data movements	5.0	6.9	8.0
	Large	8 - 14 Data movements	8.0	10.5	14.0
	Very large	14+ Data movements	14.0	23.7	30.0
Typical process	Small	CRUD (Small/Medium processes) CRUD + List (Small processes)	15.6	20.4	27.6
	Medium	CRUD (Medium/Large processes) CRUD + List (Medium processes) CRUD + List + Report (Small processes)	27.6	32.3	42.0
	Large	CRUD (Large processes) CRUD + List (Medium/Large processes) CRUD + List + Report (Medium processes)	42.0	48.5	63.0
General process	Small	6 -10 Generic FP's	20.0	60.0	110.0
	Medium	10 - 15 Generic FP's	40.0	95.0	160.0
	Large	15 - 20 Generic FP's	60.0	130.0	220.0
Macro process	Small	2 - 4 Generic GP's	120.0	285.0	520.0
	Medium	4 - 6 Generic GP's	240.0	475.0	780.0
	Large	6 - 10 Generic GP's	360.0	760.0	1,300

Table 6.1 – Release candidate values for the functional categories of the Early & Quick approach

Assigning each part of the actual requirements to a specific category higher than the Functional Process level is quite subjective. The detailed description of the approach gives guidance on assigning the proper category [14]. The precision of the approach is thus strongly dependent on the training and capability of the Measurers who use it to understand the categories at the higher levels of granularity.

In literature there is no reported use of this approximation approach at present.

⁶ N.B. This approach's use of 'preliminary steps' corresponds to the COSMIC method's 'Measurement Strategy' phase, but does not appear to be as rigorous as the process defined by COSMIC.

6.3 Strengths and weaknesses

Strengths:

- The approach is applicable when a significant part of the actual requirements is not yet known to a level of detail to allow functional processes to be identified.
- It can handle different levels of granularity and decomposition within the actual requirements.

Weaknesses:

- Assigning functional processes to a size class is a subjective element of this approximation approach, which reduces the strength of the approximation. See also Chapter 1.
- The definitions of a General Process ('an operational sub-system of the application') and of a Macro Process ('a relevant sub-system of the overall Information System of the user's organization') are vague and will vary from one organization to another.
- Due to the subjectivity involved in assigning each actual requirement to a specific category higher than the functional process, the precision of the approach is strongly dependent on the training and capability of Measurers.
- Domain dependent.
- Calibration required. The COSMIC adaptation of the approach has not been officially released.
- Not designed for approximating enhancements.
- Lack of reported use of the COSMIC adaptation of the approach.

6.4 Recommended area of application

This approach is most suited when (a part of) the actual requirements is not detailed enough to identify functional processes. COSMIC does not recommend this approach unless the Measurer receives further assistance from the method proprietor.

6.5 Research developments

The proprietor periodically researches the approach in order to calibrate the weights of the categories' elements and for reporting the results of application.

Research done by Almakadmeh to evaluate the reproducibility and accuracy of this approach, concluded poor reproducibility when the same approximate measurement was carried out by different Measurers [17]. Although a part of the observed inaccuracies in the estimations may have been caused by the way the experiment had been set up, this research clearly shows that this approach should be used with caution and only after proper training in the correct use of the approach.

EASY FUNCTION POINT APPROXIMATION

7.1 Origin and approximation mechanism

The EASY (EARly & SpeedY) approximation approach was first introduced in 2012, as a simplified instantiation of a more generic Software Measurement Approximation Rapid Technique (SMART) to size fuzzy actual requirements [23]. In the 'SMART' approach, on any function the Measurer is free to assume one or more value 'possibilities' based on his/her understanding of the actual requirements describing the functionality.

Example: "This report is 'most probably' 5-data-movements (60%), but it 'might have' 2 additional data movements (30%), or even 4 additional data movements (to be confirmed) (10%)." The approximate value for the function is the weighted sum of all possible values (where the weights are the corresponding probabilities. In the example, this would mean an approximate size of $5 \times 0.6 + 7 \times 0.3 + 9 \times 0.10 = 6.0$ CFP). (All probabilities of options for one function must sum to 100%.)

Note that the most probable value is not necessarily always the 'middle' one. It is up to the Measurer to assign the probabilities on the possible values. This is different from any 'average' approach depicted in previous chapters, where average or middle values are taken as being 'always' the most likely values. However, the 'SMART' approach might be time-consuming, for the Measurer to assign more than one possible value to each function being sized, and moreover a corresponding probability to each value per function.

The EASY approximate approach provides most typical probability distributions for the Measurer to pick from, and allows for approximate sizes and accurate sizes to be mixed. (Accurate sizes, that is sizes measured according to the standard measurement method, correspond to sizes where a value is assigned with a close-to-100% probability).

Table 7.1 below depicts typical probability distributions of approximate values for most common cases in the Business domain (FP stands for 'Functional Process') [23].

Classification of the FP	Specification level	CFP (min)	CFP	CFP (max)	Approximate CFP	Probability
Small FP	Little unknown	2 (10%)	3 (75%)	5 (15%)	3.2	>80%
Small FP	Unknown (No FUR)	2 (15%)	4 (50%)	8 (35%)	5.1	<50%
Medium FP	Little unknown	5 (10%)	7 (75%)	10 (15%)	7.25	>80%
Medium FP	Unknown (No FUR)	5 (15%)	8 (50%)	12 (35%)	8.95	<50%
Large FP	Little unknown	8 (10%)	10 (75%)	12 (15%)	10.1	>80%
Large FP	Unknown (No FUR)	8 (15%)	10 (50%)	15 (35%)	11.45	<50%
Complex FP	Little unknown	10 (10%)	15 (75%)	20 (15%)	15.25	>80%
Complex FP	Unknown (No FUR)	10 (15%)	18 (50%)	30 (35%)	21	<50%

Table 7.1 – Probability distributions of approximate values in the business domain

Different choices of probability distributions, as well as minimum and maximum CFP values for the several cases of Functional Process above, lead to different instantiation of the EASY approximation approach. A similar case can be made for the Real-time domain.

A similar approach, with different values, is easily built for approximating the sizes of 'small' to 'large' functional changes (for enhancement projects).

7.2 Applicability and reported use

In literature there is no reported use of this approximation.

7.3 Strengths and weaknesses

Strengths:

- It can be mixed with standard measures.
- It can be scaled to different levels of granularity.
- Preliminary calibration is not required.
- It works for enhancement projects (size of changes, instead of sizes of functions).

Weaknesses:

- It might be time-consuming (depending on how close the approximation is required to be accurate with the standard size).
- It relies on the choice of the 'typical' cases to map the fuzzy actual requirements onto.

7.4 Recommended area of application

This approximation is valid throughout the evolution of the actual requirements, as their description evolves in time.

7.5 Research developments

Usage data is being collected for validation and improvement purposes.

EMERGING APPROXIMATION APPROACHES

In the previous chapters a number of approximation approaches have been described that can be directly applied in practice if the recommendations at the end of chapter 1 have been properly understood and implemented.

The approximation approaches in this chapter are still in an early stage of development. We believe that these approaches have the potential to evolve into approximation approaches or tools that can be used for estimation purposes in the near future. Therefore we have included short descriptions of these approaches.

8.1 Approximation from informally written textual requirements

Application of the standard COSMIC method requires FUR at a level of granularity where external interactions with the software are visible to the Measurer. In agile software development usually the actual requirements are not at this level, but are informally written textual requirements that can be understood by the development team. At Concordia University in Canada an estimation approach has been developed to extract a size approximation from informally written textual requirements [18].

The approximation approach builds on work in the area of automating COSMIC functional size measurement from formal requirement specifications, but used the approach on informal textual requirements to produce an approximate functional size. To make this approach work, first a number of informal textual requirements must be selected to describe a single functional process and then be manually measured. The textual requirements and the corresponding sizes are stored per functional process in a database to act as reference. Then the measured functional processes are divided into four fuzzy size classes based on the quartile boundaries of the total dataset. Then with text mining, linguistic features are extracted from the dataset to train a text classification algorithm that can automatically classify a new set of textual requirements belonging to one of the four fuzzy size classes.

The strength of this approach is that after the preparation stage it can be fed with textual requirements that can automatically provide a size estimate. In the experiment, textual requirements from various sources have been used to test the approach. However, we believe that a weakness of this approach could lie in different linguistic characteristics in different environments, and the easy replication for distinct languages, meaning that local calibration would be required for each environment.

8.2 Approximation using fuzzy logic – the EPCU model

In 2012, Valdés et al. proposed a solution using a fuzzy logic model, referred to as the Estimation of Projects in a Context of Uncertainty – the EPCU model, to create an approximate sizing approach without the need to use local historical data [10]. This approach was developed for the early phases of a software development project, where most of the actual requirements are written in natural language, and more often than not the estimates are developed in an environment of uncertainty, because the actual requirements are not fully known.

In 2014, the solution was tested with a case study for an industry project where the actual requirements were made available for the Measurer only as a list of use cases, in which is typical for the early phases of the software life cycle, i.e. the actual requirements were not detailed.

The EPCU model takes into account:

- the experience-based linguistic variables used by estimation experts in the domain of estimation (approximation in this case) and
- the way experts combine these linguistic variables to approximate the functional size.

In practical experiments, Valdés reported that the EPCU estimation process for most of the projects was significantly better than the use of “expert judgment” estimation approach [19] In addition, it can be noted that the EPCU model enables a systematic replication: whatever the skill-level of the people that assign values for the input variables, the EPCU model generates estimates with less dispersion than the “expert judgment” approach that is especially useful when there is inherent subjectivity in assigning a size class.

Applying the EPCU model has six steps:

1. Identification of the input variables
2. Specification of the output variable, i.e. the estimated functional size
3. Generation of the inference rules
4. Fuzzification
5. Inference rule evaluation
6. Defuzzification

The first three steps are related to the configuration of the estimation process and generate an estimation model, or “EPCU context”. An EPCU context is "a set of variables (inputs and output) and the relations that affect a specific project or a set of similar projects" [10].

Two input variables were considered in the EPCU context for approximate sizing:

1. Variable 1: the functional process size, and
2. Variable 2: the number of objects of interest about which data is moved by the functional processes.

The strength of this approach is that it does not need local historical data to provide a size estimate. Although in the experiment the estimates were significantly better, we believe that a weakness of this approach could lie in the practical use of the approach⁷. Applying this approach involves a number of steps that require trained input, which makes it in the current state challenging for use in industrial software engineering projects. Also this model needs to be analyzed further to validate assumptions about the upper boundary cutoff, which is based on earlier research from the Netherlands [9].

⁷ This approximation approach is supported by a tool from the Mexican company Spingere. For more details, see the vendor page on www.cosmic-sizing.org.

DIFFERENT LEVELS OF GRANULARITY AND DECOMPOSITION

The purpose of this chapter is to discuss some aspects of functional size measurement that must be considered to ensure that measurements made on different sets of actual requirements are comparable with respect to the levels of granularity and decomposition. The ideas described here can arise when functional size measurements must be made in the early stages of a large software project and in general when it is necessary to ensure the comparability of size measurements across the various parts of the actual requirements.

These aspects of measuring functional sizes need to be considered in the Measurement Strategy phase of the process described in the Measurement Manual. The ideas are not specific to the COSMIC method, but in principle are relevant to any functional size measurement method.

9.1 The evolution of requirements in the early stage of a large software project

In the early stage of a large software development project, when the actual requirements are first being established, one of the two following approaches may be followed.

- First, the actual requirements are being defined in ever-more detail.
- Second, the actual requirements may be split into smaller, well-demarcated, more manageable, 'chunks', so that separate teams can work on them in parallel. These separate 'chunks' may later be developed as separate pieces of software, e.g. as separate 'sub-systems'.

The result may be (and this has been observed on several occasions in practice) that when a first measurement of size is required, the actual requirements to be measured exist at various 'levels of granularity' and at various 'levels of decomposition'. It is easy to confuse these two concepts. Therefore both concepts are described below.

DEFINITION – Level of decomposition

Any level resulting from dividing a piece of software into components (named 'Level 1', for example), then from dividing components into sub-components ('Level 2'), then from dividing sub-components into sub-sub components ('Level 3'), etc.

NOTE 1: Not to be confused with 'level of granularity'.

NOTE 2: Size measurements of the components of a piece of software may only be directly comparable for components at the same level of decomposition.

When faced with actual requirements documents that may or may not be at the same level of granularity and/or at the same level of decomposition, the Measurer must clearly examine these levels before establishing any scaling factors, as described in chapter 1.

DEFINITION – Level of granularity

Any level of expansion of the description of a single piece of software (e.g. a statement of its requirements, or a description of the structure of the piece of software) such that at each increased level of expansion, the description of the functionality of the piece of software is at an increased and uniform level of detail.

NOTE: Measurers should be aware that when requirements are evolving early in the life of a software project, at any moment different parts of the required software functionality will typically have been documented at different levels of granularity.

Precise COSMIC functional size measurements require that the actual requirements to be measured exist at a level of granularity at which functional processes and their data movements can be identified (i.e. the level at which we refer to them as 'Functional User Requirements' (FUR). The 'functional process level of granularity' is defined as follows.

DEFINITION - Functional process level of granularity

A level of granularity of the description of a piece of software at which

- the functional users are individual humans or engineered devices or pieces of software (and not any groups of these) AND
- single events occur that the piece of software must respond to (and not any level at which groups of events are defined)

NOTE 1: In practice, software documentation containing functional requirements often describes functionality at varying levels of granularity, especially when the documentation is still evolving.

NOTE 2: 'Groups of these' (functional users) might be, for example, a 'department' whose members handle many types of functional processes; or a 'control panel' that has many types of instruments; or 'central systems'.

NOTE 3: 'Groups of events' might, for example, be indicated in a statement of FUR at a high level of granularity by an input stream to an accounting software system labeled 'sales transactions'; or by an input stream to an avionics software system labeled 'pilot commands'

RULES - Functional process level of granularity

- a) Precise functional size measurement of a piece of software requires that its FUR are known at a level of granularity at which its functional processes and data movement sub-processes may be identified.
- b) If some requirements must be measured before they have been defined in sufficient detail for a precise measurement, the requirements can be measured using an approximate approach. These approaches define how requirements can be measured at higher level(s) of granularity. Scaling factors are then applied to the measurements at the higher level(s) of granularity to produce an approximate size at the level of granularity of the functional processes and their data movement sub processes. See the 'Guideline for Early or Rapid Functional Size Measurement'.

This guideline describes several approaches to implement rule b).

As described in section 1.4, a problem for all approximation approaches is that there is no way of unambiguously defining standard levels of granularity higher than the functional process level. Furthermore Measurers, especially if inexperienced, often do not realize that actual requirements are expressed at different levels of granularity and/or fail to distinguish the levels. This is one of the commonest problems faced when intending to measure functional sizes, whether precisely or approximately.

The following are examples of statements of requirements at different levels of granularity and how they should be analyzed before applying an approximate approach to functional size measurement.

Statement of Requirements	The Level of Granularity and how to analyze the Requirements
"The software system shall control all processes of the washing machine, including washing cycles, heating, filling, emptying, user control panel interface, etc."	A very high level of granularity. Very difficult for someone without experience to measure even approximately without further detail. However, an experienced Measurer with knowledge of the washing machine's hardware, should be able to at least list the number of functional processes and then use an approximate sizing approach.
The "software shall enable personnel officers to maintain data about all permanent employees."	The word 'maintain' usually implies at least functional processes to create, read, update and delete data (remember the acronym 'CRUD'). It is reasonable to measure using an approximation approach, but the Measurer must check if the list of functional processes is complete. (Several types of enquiry and update functional processes could be needed.)
"I want to be able to enquire on the order backlog which must be up-to-date at any time." (Example of a possible 'User Story' for an Agile development)	This actual requirement appears to define a single enquiry functional process. It may even be that enough is known from the context that the functional process can be measured precisely. But the actual requirement is not clear. The Story may imply other functional processes. The Measurer must ask, e.g. a) what does 'order backlog' mean? How detailed is the enquiry – by order, by customer, by product, by time since ordered, etc? b) what functionality is needed to maintain the order backlog up-to-date at any time?
"Access of customers to the system over the web shall be subject to industry-standard security, requiring e-mail address as the ID and a password." ⁸	Most Measurers should be able to list the functional processes for this actual requirement, e.g. to allow new and existing customers to access the system, to handle forgotten passwords, a facility to change a password, etc. So it should be possible to measure at least an approximate functional size.
"Monthly reports shall be produced for the Sales Managers at Branch, Region and National levels."	This actual requirement is unclear. It might specify three functional processes, but because no detail of the data movements is given, it also might only indicate a different sorting. Without further details, the approximate sizes could have a wide range of uncertainty.
"Every 10 seconds the software shall read and display the external temperature and update the temperature history log."	A fully specified functional process with its data movements. Can be measured precisely

⁸ This security requirement might be considered as a 'Non-Functional Requirement' (NFR). But like many other NFR's, the requirement is satisfied by software and this software can be measured by the COSMIC method.

To illustrate the type of difficulties faced by Measurers, we provide two examples.

1. First, in section 9.2, we briefly re-consider the example of a well-known system for ordering goods over the Internet, which is referred to as the 'Everest' system in the Measurement Manual where it is discussed in detail. This case illustrates the difficulties of measuring actual requirements at different levels of granularity.
2. Second, in section 9.3, we consider an example from a telecoms software architecture. This example illustrates the difficulties of measuring actual requirements that are being defined to lower and lower levels of granularity and are being decomposed into smaller 'chunks' at the same time, in parallel.

9.2 Measuring at varying levels of granularity - the 'Everest' system

The case of the Everest system is described in version 4.0.1 of the Measurement Manual, section 2.4.3. The description of the part of the Everest system that is given and analyzed is highly simplified and "covers only the functionality available to Everest's customer users. It thus excludes functionality that must be present so that the system can complete the supply of goods to a customer, such as functionality available to Everest staff, product suppliers, advertisers, payment service suppliers, etc."

If we were to describe the total Everest application at its highest levels of granularity, we might show it as a set of functional areas, of which 'customer ordering' would be only one area. The other areas might be: internal processes (e.g. accounting); product supply; management; advertising; payment services; system maintenance; etc. We could 'decompose' the total application at this level, and then consider each functional area independently.

The task for someone who must measure the actual requirements of any one area would therefore be to understand the actual requirements as they evolve at lower and lower levels of granularity. The measurement scope would be defined as the 'customer ordering functional area'. The Measurer would not have to think about 'decomposition' within this scope.

Recalling some observations on this case study, as we 'zoom-in' to lower levels of granularity of the actual requirements of the customer ordering functional area:

- the scope of the area to be measured does not change,
- the functional users (individual customers who place orders) do not change. A customer can 'see' the whole functionality of the area at all the levels of granularity of the analysis.

A further, and most important observation was that "in practice when some functionality is analyzed *in a top-down approach*, it cannot be assumed that the functionality shown at a particular 'level' on a diagram will always correspond to the same 'level of granularity' as this concept is defined in the COSMIC method. (This definition requires that at any one level of granularity the functionality is 'at a comparable level of detail'.)"

As the diagrams in the Measurement Manual showing a possible analysis of the Everest ordering system illustrate, in practice functional processes can occur at various levels in such diagrams. The Measurer must therefore examine each main branch, minor branch, or leaf of the system 'tree' to develop a scaling factor appropriate to that part. As in practice at any given moment all parts of a functional model will not have evolved to the same level of granularity, the same one scaling factor cannot be applied to each part.

9.3 Measuring at varying levels of granularity and decomposition in a software architecture

The example in this section illustrates an approach to sizing the actual requirements of software as they are defined at lower and lower levels of granularity that differs from the approach described for the 'Everest' system in section 9.2. This example is also from a different software domain, namely from a complex, real-time telecoms software architecture. The example was provided by a major manufacturer of telecommunications equipment, as an illustration of their current practice. **The description below uses the manufacturer's terminology** [8].

The purpose is to measure the functional size as the actual requirements evolve, as input to a project estimating method.

9.3.1 Description and analysis of the software architecture

Figure 9.1 a) shows a 'Logical Network Element' (or LNE) within the software architecture, and the analysis of its functionality into two lower levels of granularity, namely the 'System Component' (or SC) level in Figure 9.1 b) and the 'Sub-system' (or SS) level in Figure 9.1 c).

Models such as shown in Figure 9.1 are produced in the telecoms company in three stages (at each level of granularity) and the goal is to be able to estimate the project effort to develop the whole LNE at each stage. The analysis described here is therefore a form of 'early approximate sizing'. But this case and its analysis also helps illustrate other issues.

A first key difference between the way this telecoms architecture is described and analysed compared with that of the 'Everest' example in section 9.2 is that at each level of granularity the measurement scope is sub-divided so that each 'component' revealed at each level is measured separately. (Remember, in the 'Everest' example, the measurement scope was unchanged as the analysis zoomed-in on the lower levels of granularity.) This approach therefore involves 'decomposition' of the functionality as it is analysed, in addition to the 'zooming-in'.

An inevitable consequence of decomposing a piece of software (and hence of decomposing its actual requirements and its measurement scope) is that new functional users appear with each decomposition. Example: if a piece of software is decomposed into two inter-related components, then the two components must become functional users of each other and will exchange data movements. Hence, if the total size of several components is needed, the Measurer will have to consider the rules on aggregating size measurements (see the Measurement Manual)

The different measurement scopes of the LNE are shown in Fig. 9.1 by the solid line at the LNE level, the dashed lines at the SC level and the dotted lines at the SS level.

Logically, at each level of granularity, the components appear to communicate with each other directly. (In practice, of course, the components communicate via an operating system; this becomes obvious in practice at the lowest SS level of granularity, which is the level at which physical components are developed.) For sizing purposes, therefore, the components of the architecture at each level of granularity may be considered as functional users of each other.

Figure 9.1 a) shows, at the highest level of granularity, the single functional process of Logical Network Element 1 (LNE1). As far as this functional process is concerned, LNE1 has two functional users at the same level of granularity, namely LNE2 and LNE3. These users are peer pieces of software. Some data enters LNE1 from LNE2 and some data is sent by LNE1 to LNE2 and to LNE3. Some data is also sent to and retrieved from persistent storage by LNE1.

At one lower level of granularity, Figure 9.1 b) shows that LNE1 is decomposed into four System Components, namely SC1 to SC4. At this level, the functional users of each System Component are either other System Components in LNE1 or are System Components within LNE2 and LNE3 (the Figure does not illustrate this latter aspect).

Figure 9.1 b) shows that the single functional process at the LNE level has been decomposed into three functional processes, one in each of the System Components SC1, SC2 and SC4. (We now see that SC3 does not participate in the functional process at the LNE level.)

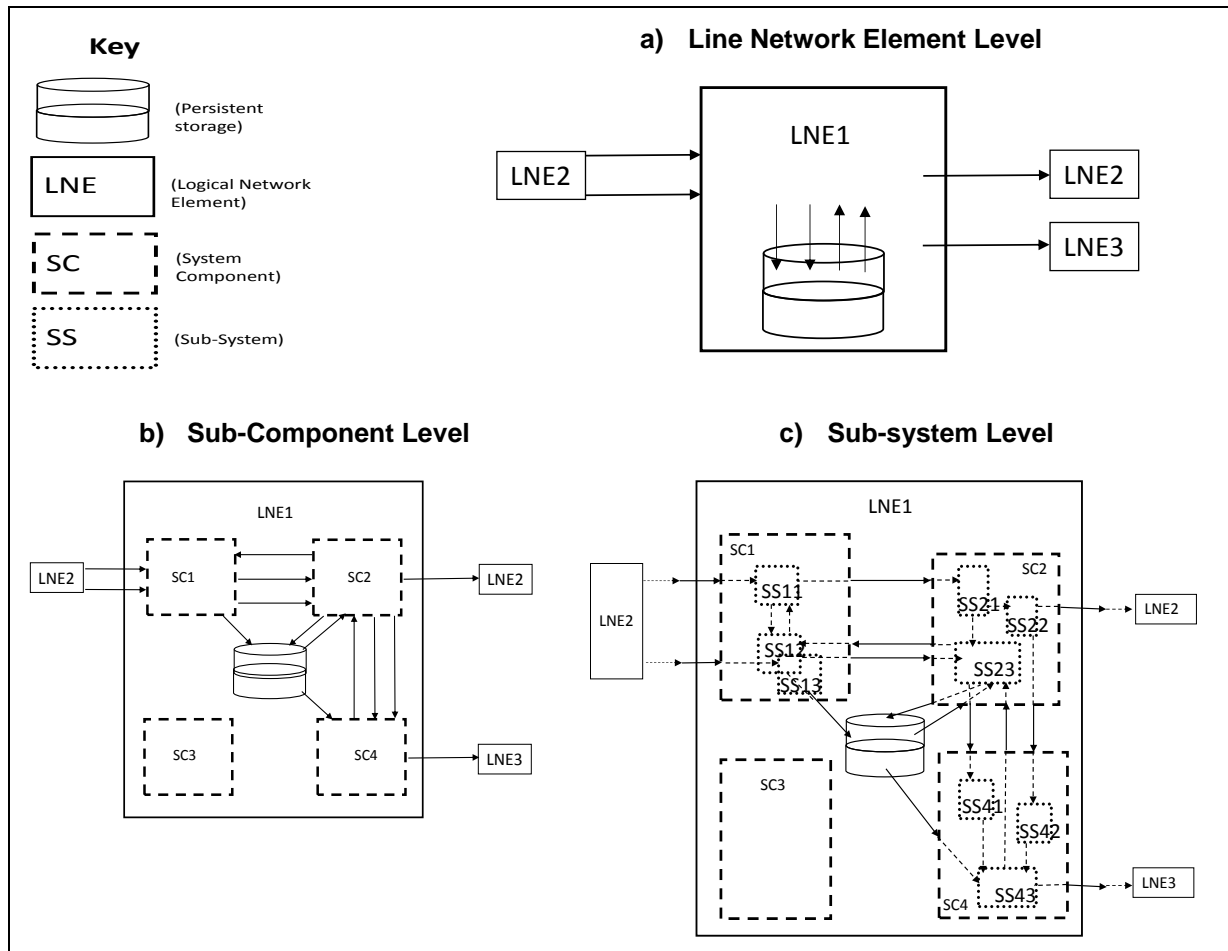


Figure 9.1 - A Line Network Element and its analysis into two lower levels of granularity

At one lower level of granularity, Figure 9.1 b) shows that LNE1 is decomposed into four System Components, namely SC1 to SC4. At this level, the functional users of each System Component are either other System Components in LNE1 or are System Components within LNE2 and LNE3 (the Figure does not illustrate this latter aspect).

Figure 9.1 b) shows that the single functional process at the LNE level has been decomposed into three functional processes, one in each of the System Components SC1, SC2 and SC4. (We now see that SC3 does not participate in the functional process at the LNE level.)

At the lowest level of granularity, Figure 9.1 c) shows that each System Component is decomposed into a number of Sub-systems (SS's). At this level, the functional users of any one Sub-system are either other Sub-systems within LNE1 or are Sub-systems in other LNE's (the latter is not illustrated in the Figure). The single functional process at the LNE level has now been decomposed into nine functional processes at this lowest level of granularity, one in each Sub-system.

At each level of granularity, some data is moved to persistent storage and some is retrieved from persistent storage. Figure 9.1 shows which components of LNE1 are involved in this functionality as we decompose to lower levels of granularity. (For diagramming convenience, persistent storage is shown as a 'common resource', irrespective of the levels of granularity and decomposition. Strictly speaking, as this Figure shows that at each level of granularity the functionality is also decomposed,

resulting in new measurement scopes, persistent storage should be shown within each scope where it is used.)

For information, the physical sequence of Data Movements (DM) at the Sub-system level of granularity in LNE1 is as follows:

- a) The triggering Entry to SS11 (of SC1 of LNE1) comes from LNE2 (sent by one of its SS's within one of its SC's)
- b) After exchanges of DMs between SS's (which may be part of the same or different SC's), LNE1 sends an Exit (by SS22 of SC2) to a SS within LNE2 (e.g. for requesting more information from the original initiator of the functional process)
- c) An SS within LNE2 then responds with another Entry (different from the triggering Entry) to LNE1 (actually to SS13 inside SC1)
- d) Again after some internal DMs, LNE1 sends (by SS43 inside SC4) a final Exit to a SS within LNE3
- e) In addition Reads and Writes take place during the process.

As stated above, it is only at the Sub-system level that project teams actually start to develop software; Sub-systems are autonomous applications. This is important because it is at this level of granularity that the telecoms software company that provided this example wishes to carry out individual project estimating.

9.3.2 Sizing the Logical Network Element

With this analysis approach, the size of the functionality shown in Figure 9.1 apparently increases as more components and functional processes are revealed at the lower levels of granularity.

This 'growth' is analogous to what we see in road maps. As we move from a large-scale map to one of smaller scale showing more roads, so the size of the road network appears to increase, even though the unit of measure for both maps (e.g. the kilometre) is the same for each scale.

The sizes of the functionality on Figure 9.1 at each level of granularity are as follows.

- At the LNE level of granularity (one functional process) = 8 CFP
- At the SC level of granularity (three functional processes) = 20 CFP
- (At the SS level of granularity (eight functional processes) = 32 CFP

Note that as a check on the measurements in this example, the size at any one level of granularity can be obtained from the sizes at the immediately lower level by eliminating all the inter-component Entries and Exits for the components at the lower level.

9.3.3 Discussion of the analysis of the LNE example

The first main observation from the analysis of this example is that when described in the terminology of the telecoms equipment manufacturer, it suggests a weakness in the definition of the 'functional process level of granularity' given in chapter 1, which states:

"A level of granularity of the description of a piece of software at which

- the functional users are individual humans or engineered devices or pieces of software (and not any groups of these) AND (etc.)"

The difficulty in interpreting this definition in this context is that *all* the functional users in the LNE case are 'pieces of software', and it is impossible to define what is a piece of software in any way that is generally-applicable for our purpose.

The analysis approach and terminology of the telecoms equipment company illustrated here results in the measurement scope being re-defined at each level of granularity, and functional processes being defined at three levels of granularity, rather than the one standard level assumed by the definition. .

This problem can be avoided by changing the telecom company's terminology so that the term 'functional process' would be used only at the Sub-system level. At the higher levels, terms such as 'super-process' and 'super-super-process' could be used.

But there is a more fundamental issue that whatever names are used, the definition of the 'functional process level of granularity' will be specific to this company. For the company this definition is relatively easy to interpret because it is the level at which they set project teams to work to develop 'individual pieces of software (and not any groups of these)', as per the definition. However, this approach does not guarantee that what this company means by 'a piece of software' will be comparable to that of another company. Software can be aggregated or decomposed to multiple levels of granularity, and the types of the software's components can vary with the technology used. One company's 'piece of software' might be a complete Logical Network Element. Another company's piece might be a single re-usable object (perhaps a saleable product), which would clearly be a different level of granularity.

The problem with the definition of the 'functional process level of granularity' should not arise when the context includes functional users that are 'individual human and/or engineered devices', which should always be precisely identifiable. In such a context, if there are also functional users that are 'pieces of software', then the level of granularity of those software users and the data exchanges with the software being measured should be the same as that of the human or engineered device functional users.

9.3.4 Computing 'scaling factors' for the LNE example

Supposing that in the LNE example we are at the stage of having completed the specification partly at the highest level of granularity of the LNE's and partly at the SC level, and wish to determine the size of the eventual software at the lowest level of granularity of the Sub-systems for input to a project estimation method. For this, we need scaling factors to multiply the measured sizes of an LNE or a SC to obtain the size measured at the lowest SS level.

If we were using the size measurements given in section 9.3.2 on this LNE1 to calibrate an approximate approach to sizing other LNE's and their SC's at the SS level, we would conclude that the scaling factors to be used would be as follows.

- To scale a size measured at the LNE level to a size measured at the SS level, multiply the LNE size by 4.0 (32 / 8)
- To scale a size measured at the SC level to a size measured at the SS level, multiply the SC size by 1.6 (32 / 20).

In practice, it is likely that at the moment when a project estimate is required necessitating a functional size measurement as input, the actual requirements will have been developed at varying levels of granularity. In such circumstances the Measurer will have to exercise judgment when estimating the size at the required level of granularity by using a mixture of actual and scaled measurements.

9.4 Functional size measurements and standard levels of decomposition

The issue to be briefly dealt with in this section has already been referred to above, namely that to ensure comparability of measurements there is a need to standardize levels of decomposition as well as to use the standard functional process level of granularity.

The relationship between these two 'views' of levels is complex and is best carried out in four steps, as follows:

Step 1. We wish to measure a size of *software* for purposes such as project performance measurement or estimating. Identify the *levels of decomposition* of the software and determine the measurement of each piece of software to be measured.

Examples of levels of decomposition of software, hence of possible measurement scopes:

- The telecoms system, the subject of section 9.2, could be measured at any of the LNE, system component or sub-system levels (as described in Fig. 9.1) or even at lower levels of decomposition such as that of sub-system major components, re-usable components, etc.
- In the business application domain, software can be measured at the level of a whole application, or of a major application component (e.g. of a 'n-tier' architecture) or an object class, or re-usable component, etc. (But note that in industries such as banking, where software systems have grown and evolved over decades, it can be difficult to define and distinguish an 'application'.)

Step 2. Identify the *level of granularity* of the actual requirements of a piece of software to be measured can be expressed at any *level of granularity*.

Step 3. Identify the functional users that *cannot be decomposed* (individual humans or pieces of hardware) . The individual humans or pieces of hardware interact with pieces of software at only a limited number of software levels of decomposition of practical interest for measurement purposes. (Example, it may be useful to measure the size of business application software as seen by a human user at only two levels of decomposition, namely that of a whole application, or the user interface component of a multi-tier application.

We can therefore define precisely the few combinations of human or hardware functional users and levels of decomposition of the software with which they interact. It follows that we can precisely identify the functional process level of granularity for these combinations. Hence we can identify the functional processes precisely and produce functional size measurements that can be reliably compared from different sources for these combinations. It also follows that we can apply the approximation approaches described in this Guideline to actual requirements at higher levels of granularity than the functional process level.

Step 4. Identify functional users that are pieces of software, these *can exist at multiple levels of decomposition*. There are no *standard* levels of decomposition of software. In addition, the actual requirements of both the software being measured and of its software functional users can be expressed at multiple levels of granularity. It is therefore intrinsically difficult in practice to define a universal standard for a functional process level of granularity when the software being measured and all its functional users are pieces of software. It follows that it is equally difficult to identify and measure functional processes in a way that ensures measurements from all sources are comparable for this 'software/software' combination. Similarly, it is difficult to apply the approximation approaches described in this Guideline for this 'software/software' combination.

These difficulties can be overcome within an organization or between collaborating organizations that can define local standards for software levels of decomposition, and for levels of granularity if needed for approximate sizing.

To enable greater size measurement comparability, COSMIC recommends that suppliers of services or tools that use functional size measurements specify standard levels of decomposition of software for which their service or tool can accept the size measurements. It is common practice for suppliers of benchmarking services, for example, to specify standard programming language levels (e.g. 3GL, 4GL, etc.) and technology platforms (e.g. main-frame, mid-range, PC), but it is not common practice to specify standard levels of decomposition of software.

COSMIC suggests the following as example candidates for standardization of levels of decomposition in the domain of business application software.

- A 'whole application'
- A major component of a whole application
- A re-usable object-class

The permitted functional users of software at all these levels of decomposition can be humans or other pieces of software at any of these levels of decomposition, Example: a whole application can be a functional user of a SOA component, and vice versa. But the case of a human user of a re-usable object-class is not likely to be of interest.

These standard 'levels of decomposition' are also typical examples of measurement scopes, as given in the Measurement Manual.

LOCALIZATION PRINCIPLES

10.1 Establishing a locally-defined approach

The following general guidance can be given to an organization when establishing a locally-defined approach to approximate COSMIC sizing.

- a) The local organization should define one or more (types of) artifact at a higher level of granularity than the level at which functional processes and their data movements are known, that describe the software functionality in a way that can be measured (i.e. at least identified and counted)
- b) Artifacts selected in step a) might be, in *ascending* order of level of granularity, documents describing actual requirements at the functional process level, the 'use case' level, the sub-system level, etc. (Note that there is no standard terminology for levels of granularity above the level of a functional process⁹.) Great care is therefore needed when defining standard artifacts suitable for approximate measurement above the functional process level of granularity.
- c) The high-level artifacts selected for the measurements to calibrate the scaling factor must be representative of the software that needs to be sized in the future by the approximate approach
- d) The artifacts selected above should all be of roughly similar size, or be classifiable in bands (classes) of similar size (see the approximation approaches described in Chapters 3, 4, 6 and 7), so that it is reasonable to assign an average local scaling factor to each artifact or class of artifacts
- e) After applying an approximation approach to measure some high-level actual requirements, it is important to learn from the experience by establishing the accuracy of the approximate measurements when the detailed actual requirements of the same software become known. This is done by first measuring the precise sizes for at least a sample of the detailed actual requirements. Then compare the approximate sizes with the precise measurements to check that the scaling factor(s) used were reasonable. For instance, a result in a particular project could be that the actual total size when the final actual requirements are precisely measured turns out to be significantly greater than the measurements obtained by the approximate approach. The inaccuracy might be due to using inappropriate scaling factors and/or 'scope creep' on the project concerned. This result could be used to adapt the approximate measurement approach to take account of such factors in the future. (See section 11.2 for more on taking into account 'scope creep'.)
- f) Given the uncertainty in approximate sizing, a range or some indication of the precision should be given when reporting an approximate size measurement, based on the established accuracy by comparing with the detailed size measurement as described under e).

⁹ Terms like 'Use Case' are of course defined, but in spite of such definitions, practice shows there is no guarantee that for a given actual requirement, two analysts will analyse the same number of Use Cases. Each organization must therefore establish its own understanding of what constitutes one Use Case.

The procedures to establish the precision of an approximate size measurement should also be established locally. The precision of any particular measurement will depend on the following two factors.

- The level of detail and uncertainty of the requirements, which obviously varies with the state of progress of the project (See section 1.5 on the quality of requirements, the examples of section 9.1, and the guidance on 'scope creep' in section 11.2);
- The particular approximate sizing approach used for the measurement. (Example: the more detailed approach of Chapter 4 should, if well calibrated, give a more accurate size measurement than the approach of Chapter 2, for the same quality of the requirements.)

Best practice is then to produce a 'three point' estimate of the size, where the three points are the minimum size estimate, the most likely size estimate and the maximum size estimate. Presenting these three figures to show a range of uncertainty on the approximate size measurement is much more valuable to decision-makers and to anyone who must estimate project effort than just reporting the most-likely estimate.

The Early and Quick approach described in Chapter 6 includes a three-point size estimation element.

For more on the subject of three-point estimation, see for example [Wikipedia](#).

APPROXIMATE SIZING OF CHANGES OF FUNCTIONALITY AND SCOPE CREEP

11.1 Approximate sizing of changes to functionality

To estimate the approximate size of an actual requirement to change some existing software, the general guidance follows the same approach as for new software. If there is an existing catalogue of functional processes and their sizes (or size classification), then one of the two following approaches may be chosen.

- a) If possible, based on the Functional User Requirements for the changes, judge which data movements of the relevant functional processes will be impacted and count these data movements;
- b) Otherwise, estimate for each functional process the number or proportion of data movements that has to be changed. For example, if a functional process to be changed is sized as 12 CFP and it is estimated that the change affects 25% of the data movements, then the size of the change is 3 CFP.

Use these numbers or proportions instead of the total sizes of the functional processes to be changed to adapt one of the approximation approaches described above.

If there is no catalogue of existing functional processes, the first task would be to identify the functional processes affected by the actual change requirements, and then follow one of the approximate approaches above.

11.2 Approximate sizing and scope creep

Experience shows that early in the life of a software development project, the functional size of the software tends to increase as the project progresses from outline actual requirements to detailed actual requirements, to functional specification, etc. This phenomenon, often referred to as 'scope creep', can arise because

- the scope expands beyond that originally planned to include additional areas of functionality
- and/or, as the detail becomes clearer, the required functionality turns out to be more extensive (e.g. to require more data movements per functional process) than was originally envisaged
- and/or Non-Functional Requirements may turn out to be (partly) implemented in software [20]

(It can also happen, of course, that the scope is reduced from the originally planned scope, e.g. due to budget cuts.)

The approximate sizing approaches described in this Guideline do not explicitly take into account scope creep. When using these approximation approaches for early sizing therefore, potential scope creep should be considered as an additional factor. If potential scope creep is ignored, there is a risk of under-estimating the final software size and hence the project effort.

Estimating the potential for scope creep on a particular project goes beyond the scope of this Guideline. However, it may be helpful to address the following questions:

- Are the actual requirements on this project particularly uncertain at the outset? If so, what correction (or 'contingency') to the approximate size should be made for possible scope creep?
- If scope creep is endemic within the organization, then we can use past measurements to help quantify this phenomenon. For instance, in a given organization and using a given development process for which many measurements exist, it may be possible to find a recurrent pattern such as 'by the end of phase 3, sizes are typically 30% greater than at the end of phase 1'.

CONCLUSIONS ON APPROACHES TO APPROXIMATE SIZING

Approaches to approximate sizing can be made to work and are valuable for use early in a new software project's life and/or can save time and effort compared with sizing precisely using the standard COSMIC measurement method. Approximate sizing may also be necessary when the actual requirements are unclear. But approximate sizing approaches need to be used with care.

Whatever the purpose of using an approach for approximate sizing, whenever further information becomes available enabling a more accurate and/or precise sizing, the Measurer should refine and update the measurement. This is especially required when using the measurement results as an input to estimation (such as effort prediction) – due to the phenomenon of error propagation [22].

For obvious and similar reasons, no approximate sizing should be accepted in deriving actual values in contractual situations, or analogous cases, where precise and accurate figures are required – any preliminary approximate sizing should be replaced by standard measurements in the final stages of projects subject to such constraints.

When there is a need for approximate sizing, the organisation should:

- choose an approach which is optimal for the purpose of the measurement, given the availability of data for the calibration, the time available for the measurement and the accuracy required of the approximate size;
- calibrate the approach using accurately-measured local data on software that is comparable to that for which the approximate sizes must be measured;
- when actual requirements are unclear or incomplete, seek help to try to at least identify all the functional processes
- pay particular attention to identifying any large functional processes and to determining good scaling factors for them, as they can make a large contribution to the total size although they are few in number;
- consider whether an allowance (or 'contingency') should be made for 'scope creep' and for the contribution that incorporating the Non-Functional Requirements may lead to when publishing an approximate size;
- estimate and report the plus or minus uncertainty on the approximate size, mentioning any contingency that has been made for scope creep; estimating the uncertainty on an approximate size is especially important in contractual situations.

REFERENCES

- [1] F.W. Vogelezang, C.R. Symons, A. Lesterhuis, R. Meli and M. Daneva, "Approximate COSMIC Functional Size: Guideline for approximate COSMIC Functional Size Measurement", 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, Ankara, Turkey
- [2] C.R. Symons and A. Lesterhuis, "The COSMIC Functional Size Measurement Method Version 4.0.1 Measurement Manual, the COSMIC implementation guide for ISO/IEC 19761:2003", April 2015
- [3] S. McConnell, "Software Estimation, Demystifying the Black Art", Microsoft Press 2006.
- [4] A. Abran, J-M. Desharnais, S. Oligny, D. Saint-Pierre and C.R. Symons, "COSMIC-FFP Measurement Manual version 2.2, the COSMIC implementation guide for ISO/IEC 19761:2003"
- [5] F.W. Vogelezang, "Early estimating using COSMIC-FFP", Proceedings of the 2nd Software Measurement European Forum, March 16-18, 2005, Rome.
- [6] H.S. van Heeringen, E.W.M. van Gorp and T.G. Prins, "Functional size measurement – accuracy versus costs – is it really worth it?", Software Measurement European Forum (SMEF 2009), Rome, Italy 2009.
- [7] L. de Marco, F. Ferrucci and C. Gravino, "Approximate COSMIC size to early estimate Web application development effort", 39th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2013, Santander, Spain, September 4-6, 2013
- [8] A. Lesterhuis and C.R. Symons, "The COSMIC Functional Size Measurement Method version 3.0 Advanced and Related topics", December 2007.
- [9] F.W. Vogelezang and T.G. Prins, "Approximate size measurement with the COSMIC method: Factors of influence", Software Measurement European Forum (SMEF 2007), Roma, Italia, 2007
- [10] F. Valdes Souto and A. Abran, "Case Study: COSMIC Approximate Sizing Approach Without Using Historical Data", 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, Assisi, Italy
- [11] C.R. Symons, and C. Gencel, 'From requirements to project effort estimates: work in progress (still?)', Requirements Engineering for Software Quality (REFSQ 2013) conference, Essen, Germany, April 2013
- [12] L. Santillo, "Early & Quick COSMIC-FFP Analysis Using the Analytic Hierarchy Process." Proceedings of the 10th International Workshop on Software Measurement, *Lecture Notes in Computer Science* 2006, (pp. 147-160), Berlin (Germany) 2000
- [13] R. Meli, "Early Function Points : a new estimation method for software projects", ESCOM 97, Berlin (Germany) 1997
- [14] T. Iorio, R. Meli and F. Perna, "Early & Quick Function Points® v3.0:enhancements for a Publicly Available Method", Software Measurement European Forum (SMEF 2007), Roma, Italia, 2007
- [15] Creative Commons, "Creative Commons Attribution-NoDerivatives 4.0 Public License, conditions and terms", creativecommons.org/licenses/by-nd/4.0/legalcode
- [16] M. Conte, T. Iorio, and L. Santillo, "E&Q: An Early & Quick Approach to Functional Size Measurement Methods", Software Measurement European Forum (SMEF 2004), Italy 2004.
- [17] K. Almakadmeh, "Development of a scaling factors framework to improve the approximation of software functional size with COSMIC - ISO19761", PhD thesis for the Ecole Technologie Supérieure of the Université du Québec, Montréal, June 11 2013
- [18] I. Hussain, L. Kosseim and O.Ormandjieva, "Approximation of COSMIC functional size to support early effort estimation in Agile", Data & Knowledge Engineering 85 (2013) 2-14

- [19] F. Valdés, A. Abran, "Comparing the Estimation Performance of the EPCU Model with the Expert Judgment Estimation Approach Using Data from Industry", chapter 15, in 'Software Engineering Research, Management and Application 2010, published in 'Studies in Computational Intelligence', Volume 296, Springer-Verlag, Berlin, pages 227-240.
- [20] F. AbuTalib, D. Giannacopoulos and A. Abran, "Designing a Measurement Method for the Portability Non-Functional Requirement", 23rd International Workshop on Software Measurement & 8th International Conference on Software Process and Product Measurement, October 2013, Ankara (Turkey)
- [21] F.W. Vogelezang, C.R. Symons and A. Lesterhuis, "Web Advice Module – COSMIC Case Study", published January 2014 by Nesma (www.nesma.org).
- [22] L. Santillo, "Error Propagation in Software Measurement and Estimation", 16th International Workshop on Software Measurement (IWSM), October 2006, Potsdam (Germany)
- [23] L. Santillo, "EASY Function Points – 'SMART' Approximation Technique for the IFPUG and COSMIC Methods", 22nd International Workshop on Software Measurement & 7th International Conference on Software Process and Product Measurement, November 2012, Assisi (Italy)

GLOSSARY OF TERMS

The terms in this Glossary are specific to this Guideline on Approximate COSMIC Size Measurement. For other COSMIC terms, see the main Glossary in the Measurement Manual v4.0.1.

Accuracy.

1. A qualitative assessment of correctness, or freedom from error (ISO/IEC 24765:2010)
2. a quantitative measure of the magnitude of error (ISO/IEC 24765:2010). *See also: **precision**.*

Approximate Sizing.

1. Approximate measurement of a size.
2. Measurement of a size by an approximate approach.

Calibration

Determining the relation or deviation of the scaling factors or classification values in the environment in which the approximation approach is used to the scaling factors or classification values published in reference documents like this Guideline, to ascertain an as accurate as possible result of the application of the approximation approach.

Classification

Allocating a part of the actual requirements to a defined class (or reference piece) of requirements whose size has been calibrated in CFP.

Localization

Calibrating scaling factors or classification values to an environment that is representative of the environment the approximation approach is to be used in.

Precision.

The degree of exactness or discrimination with which a quantity is stated (ISO/IEC 24765:2010)
Example: a precision of 2 decimal places versus a precision of 5 decimal places.

Scaling factor.

A constant that is used to convert a size measured under one set of conditions (e.g. one level of granularity of some actual requirements) to a size measured under another set of conditions (e.g. another level of granularity of the same requirements).

APPENDIX A: 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 an error 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 of the Guideline for approximate COSMIC functional size measurement 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 organisations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org website for further detail.