The COSMIC Functional Size Measurement Method

Version 3.0

Advanced and Related Topics

December 2007

Acknowledgements

COSMIC Method Version 3.0 authors and reviewers 2007 (alphabetical order)		
Alain Abran, École de Technologie Supérieure, Université du Québec, Canada	Jean-Marc Desharnais, Software Engineering Lab in Applied Metrics – SELAM, Canada	Arlan Lesterhuis*, Sogeti, The Netherlands
Bernard Londeix, Telmaco, United Kingdom	Pam Morris, Total Metrics, Australia	Serge Oligny, Bell Canada
Marie O'Neill, Software Management Methods, Ireland	Tony Rollo, Software Measurement Services Ltd., United Kingdom	Grant Rule, Software Measurement Services Ltd., United Kingdom
Luca Santillo, Agile Metrics, Italy	Charles Symons*, United Kingdom	Hannu Toivonen, Nokia Siemens Networks, Finland
Frank Vogelezang, Sogeti, The Netherlands		

* Editors of this version 3.0

Copyright 2007. 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 Web at www.gelog.etsmtl.ca/cosmic-ffp

Version Control

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

DATE	REVIEWER(S)	Modifications / Additions
December 2007	COSMIC Measurement Practices Committee	First public version of this document

The COSMIC Method, Advanced and Related Topics - Copyright © 2007.
All rights reserved. The Common Software Measurement International Consortium (COSMIC)

Foreword

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

The COSMIC method was accepted by ISO/IEC JTC1 SC7 in December 2002 as International Standard ISO/IEC 19761 'Software Engineering – COSMIC-FFP – A functional size measurement method' (hereinafter referred to as 'ISO/IEC 19761').

Purposes of this document

This 'Advanced and Related Topics' document has two purposes.

First, the document contains material on the COSMIC method v3.0 that is intended for advanced practitioners, i.e. measurement specialists who have already mastered the concepts of the method as described in the 'Measurement Manual' (see below). This first edition of this document mainly addresses the topic of ensuring comparability of measurements made on functional user requirements ('FUR') early in the life of a large software project, when the requirements may typically be in the process of being sub-divided and worked out in more detail by different teams of requirements engineers working in parallel. When such partially-developed statements of FUR are encountered, the measurer must be aware of the verification checks that should be made to ensure that measurements made on different parts of the FUR are comparable. The same verification checks may be needed when measurements of FUR are received from different sources.

Second, the document contains material that is not part of the basic COSMIC method, but which is highly relevant to its practical use. In this first edition of this document, two related topics are covered, namely 'early or rapid approximate sizing using the COSMIC method' and 'convertibility' (i.e. methods of converting sizes measured using the COSMIC method to or from sizes measured using '1st generation' functional size measurement (FSM) Methods). A basic mastery of the concepts of the Measurement Manual is also necessary to understand both these topics.

The topic of 'approximate sizing' is closely related to the advanced topic of 'ensuring comparability'. Also, the former topic introduces the subject of 'measurement scaling' which is needed to discuss the latter topic. The topic of 'approximate sizing' is therefore dealt with first in this document.

- **COSMIC Method Documentation**

For a full account of the COSMIC method documentation and for the definition of all terms that are common to all COSMIC documents, please refer to 'COSMIC Method v3.0: Documentation Overview and Glossary of Terms'.. This document and all other COSMIC documents mentioned below may be down-loaded free-of-charge from www.gelog.etsmtl.ca/cosmic-ffp

The other principal documents of interest will be as follows.

- The ISO/IEC 19761 International Standard, which contains the fundamental normative definitions and rules of the method (obtainable directly from ISO – see www.iso.org.).
- The 'COSMIC Method v3.0: Method Overview', which provides a complete introduction to the method for those who are new to the subject or who do not need to understand all of its details.
- The 'COSMIC Method v 3.0: Measurement Manual', which provides all the principles, rules and definitions, and also aims to provide further explanation and many examples in order to help measurers to fully understand and to apply the method. This should be the main 'working document' that measurers will need in practice.

The COSMIC Method, Advanced and Related Topics - Copyright © 2007.
All rights reserved. The Common Software Measurement International Consortium (COSMIC)

Other COSMIC documentation available includes Guidelines for the method's application in specific software domains, various Case Studies illustrating the method's use, research papers, benchmark data, etc. Translations of the Measurement Manual into other languages are also available. All these can be found on www.gelog.etsmtl.ca/cosmic-ffp.

More general background information on functional size measurement and its uses, on the advantages of the COSMIC method, on the COSMIC organization and its activities, on suppliers of COSMIC-related services, COSMIC Newsletters, etc., can be found on www.cosmicon.com.

COSMIC Measurement Practices Committee

December 2007

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

Table of Contents

The COSMIC Method, Advanced and Related Topics - Copyright © 2007.
All rights reserved. The Common Software Measurement International Consortium (COSMIC)

1

'EARLY' OR 'RAPID' APPROXIMATE SIZING

The COSMIC functional size measurement method requires that measurements of the functional user requirements ('FUR') of a piece of software must be made at a standard level of granularity, known as the 'functional process level of granularity'. The rules for this standard level of granularity, as defined in the Measurement Manual, are as follows.

RULES - Functional process level of granularity
a) Functional size measurement should be made at the functional process level of granularity
b) Where a functional size measurement is needed of some FUR that have not yet evolved to the level where all the functional processes have been identified and all the details of their data movements have been defined, measurements should be made of the functionality that has been defined, and then scaled to the level of granularity of functional processes.

This chapter discusses approaches for implementing rule b), i.e. when a need arises in practice to measure a functional size approximately[1]. Typical situations where such a need arises are:

- early in the life of a project, for example as input to a project estimating process, before the FUR have been specified down to the level of detail where the precise size measurement is possible – known as 'early sizing',
- when a measurement is needed but there is insufficient time to measure the required size using the standard method; an approximate size will be acceptable if it can be measured faster than is possible with the standard method – known as 'rapid sizing'

In practice basically the same approaches to approximate sizing can be used, whether the need is for 'early sizing' or for 'rapid sizing'.

When the need is for early sizing, when only high-level artifacts such as (for example) 'use cases' exist, the task is to measure these use cases in some locally-defined way and to use a locally-calibrated 'scaling factor' to convert the use case size measurements to COSMIC Function Points (CFP).

The need for 'rapid sizing' may arise when it is uneconomic to spend the time and effort to identify the full detail of the functional processes and to count their data movements. Measuring an approximate size, requiring less time and effort would be more acceptable. The task then is to identify artifacts of the software at some higher level of granularity such as, say, 'functional process groups' and first to identify and to size these. Then again, using a locally-calibrated scaling factor, the size measurements of the artifacts at the high level of granularity can be converted to CFP.

[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'.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

This chapter gives some general guidance on establishing scaling factors and approximate sizing and then gives several examples of approaches to approximate sizing. In the examples, the motivation is usually given as 'early sizing' but similar approaches can be used for 'rapid sizing'.

The examples of this chapter assume, for simplicity, that all the high-level artifacts that must be locally identified and measured (where these local sizes are then scaled to give an approximate size in CFP) will be at the same ('high') level of granularity. In practice, when a measurement must be made early in the life of a software project, this may not be true. Chapter 2 deals with the factors to consider in more complex situations that can arise in practice.

1.1 General principles of approximate sizing by measurement scaling

The general principle of any approach to approximate sizing is that some type(s) of artifact(s) of the FUR of the software to be measured must be defined locally, that define functionality at a high level of granularity that is measurable in some way. The average size(s) of these artifact(s) must then be established locally, in units of CFP. These average size(s) are a form of 'scaling factors'. Scaling factors are ratios that can be used to convert measurements on the high-level artifacts to sizes expressed in CFP.

This principle is illustrated in Figure 1.1.

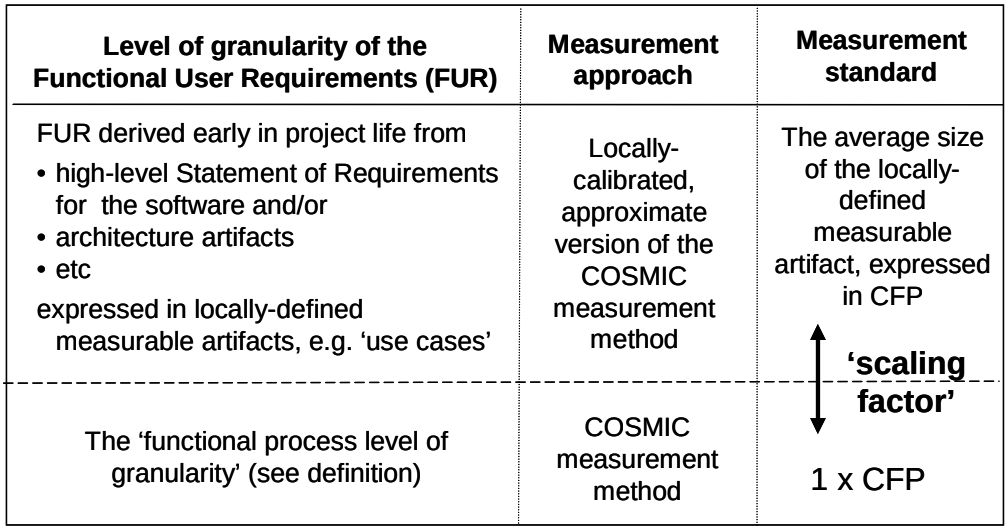

Figure 1.1 – Scaling of sizes between different levels of granularity

All approaches designed to measure approximate COSMIC sizes must be established locally. COSMIC can give general guidance on setting up an approach to approximate sizing, but the actual approach can only be finalized locally because:

- the artifacts at level(s) of granularity higher than the functional process that will be measured can only be defined (or illustrated by standard examples) locally
- the scaling factor(s) used to convert counts of the high-level artifacts to sizes in CFP should always be calibrated locally

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 data movement, that describe the functionality of the software in a way that can be measured (i.e. identified and counted), and for which the measurement result can be scaled (i.e. converted via a 'scaling factor') to CFP.

b) Artifacts above the level of granularity of the data movement that might be used as the basis for an approximate sizing method, in ascending order of level of granularity, might be documents

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

describing FUR at the functional process level, the 'use case' level, the sub-system level, etc. (Note that there is no standard terminology above the level of a functional process[2].) Due to the increasing difficulty in defining standard artifacts suitable for approximate measurement above the level of granularity of a functional process, it is not advisable to establish an approximate sizing method more than two levels above the functional process level

c) Clearly, 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 high-level artifacts should all be of roughly similar size, or defined in bands of similar size (see the examples of sections 1.2.2 and 1.2.3 below) at any locally-defined level of granularity, so that it is reasonable to assign an average local scaling factor to each artifact or group of artifacts

e) Each measurement assignment should begin by examining the high-level artifacts of the software, in co-operation with an expert on the requirements, to ensure that the approximation approach to be used is appropriate for this measurement or whether it needs fine-tuning – for example adjusting the scaling factors. This step will increase the reliability of the approximate sizing and give greater confidence in the results.

f) When the detailed requirements become known and it becomes possible to measure them precisely, a representative sample of the detailed requirements should also be measured. The approximate sizes should then be compared against the precise measurements in order to check that the scaling factor(s) used were reasonable and to learn from the experience. For instance, a result in a particular project could be that the actual total size when the final requirements are precisely measured turns out to be much 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 process to take account of such factors in the future. (See section 1.4 for more on taking into account 'scope creep'.)

g) Given the uncertainty in approximate sizing, a range or some indication of the uncertainty should always be given when stating a size measured this way.

1.2 Examples of approaches to approximate sizing

In this section, four examples of approaches to approximate sizing of new 'whole' sets of requirements are presented. The measurer can choose the most promising approach according to the situation. It could also be beneficial to use two or more approaches and to compare their results.

For instance, the real experience of one particular organization showed that sizing their functional processes using the 'Fixed Size Classification' approach of example 3 below, and then using the simpler 'Average Functional Process' approach of example 1, gave a difference of usually less than 10% in the total size of the software. If this can be confirmed for an organization and the loss of accuracy is acceptable, then the simpler 'Average Functional Process' approximation approach could be used routinely, to save measurement effort.

Some guidance is also given in section 1.3 on approximate sizing of functional *changes* to software.

Once again, it must be emphasized that COSMIC does not recommend that any of these approaches should be used literally, as described below. Any of these approximate sizing methods should first be re-calibrated locally, using accurately-measured local data on software that is comparable to that for which the approximate sizes must be measured.

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

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

1.2.1 Example 1, the 'Average Functional Process' approach

The first concept above the data movement is the functional process. The simplest process for obtaining an approximate size of a new piece of software is therefore as follows. Given a new piece of software:

A Sampling and calculation of an average functional process

1. Identify a sample of other pieces of software with characteristics similar to the new piece
2. Identify the functional processes of these other pieces
3. Measure the sizes of the functional processes of these other pieces accurately using the standard COSMIC method
4. Determine the average size, in CFP, of the functional processes of these other pieces (e.g. = 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 new piece of software (e.g. = 40)
2. Based on the sample, the approximate size of the new piece of software is estimated to be (Number of Functional Processes x average size from sample) = 40 x 8 = 320 CFP
3. Test the uncertainty of the approximate size of the new piece by precisely measuring a sample of its functional processes; express the uncertainly as a percentage of the approximate size, e.g. 'plus or minus 10%'.

If the task is to size some existing software quickly (i.e. the motive is 'rapid sizing'), then essentially the same process should be followed.

A. A representative sample of the software to be sized can be measured accurately and these sizes used to determine an average size of a functional process for the software to be measured.

B. The approximate size of the remainder of the software can then be obtained by simply identifying and counting the functional processes and applying the average size.

This approach is obviously much faster than if the data movements of each functional process must be identified, as in an accurate measurement.

1.2.2 Example 2, the 'Fixed Size Classification' approach

In this refinement of the approach of Example 1, a statement of 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. A corresponding scaling factor is then assigned to each functional process.

The table below shows an example of a set of size classes that is in actual use in a specific business organisation. The lines of the table give 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 given 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. The fifth column 'X-messages' adds in one Exit for all error and similar types of messages.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

Fixed Size Classification	Total CFP		#E	#X	#R	#W	X-messages.
Small	5		1	1	1	1	1
Medium	10		2	2	3	2	1
Large	15		3	3	4	4	1
…							

If the functional size of some requirements must be estimated early in the development process, each requirement (e.g. perhaps expressed as a 'use case') is assigned one or more functional processes, together with their appropriate size. Use of a table such as shown above helps the person who has to size the FUR 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 size than the approach of Example 1 although the observations of Example 3 should also be considered.

1.2.3 Example 3, the 'Equal Size Bands' approach

The previous approach can be refined further to give a more accurate result if sufficient size data are available for an accurate calibration. 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.) Two examples of the application of this approach are given.

EXAMPLE 3a: Vogelezang and Prins[3] 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[4].

It was decided to use four size bands. The average sizes of each band when the 2427 functional processes of the 37 applications were distributed over the four bands (and the names given to these bands) are:

'Small'	4.8 CFP
'Medium'	7.7 CFP
'Large'	10.7 CFP
'Very Large'	16.4 CFP

To interpret these figures, for instance:

- 25% of the total size of these 37 applications is accounted for by 'Small' functional processes whose average size is 4.8 CFP;
- another 25% of the total size by 'Medium' functional processes of average size 7.7 CFP;
- etc.

Another finding was that the distribution of the *number* of functional processes in each band, per application, was 40-45% Small, 25-30% Medium, 15-20% Large and 5-15% Very Large.

EXAMPLE 3b: The calibration using this approach of one component of a major real-time avionics system gave the following average sizes of the functional processes in each band[5]:

---

[3] Vogelezang F.W. and Prins, T.G., 'Approximate size measurement with the COSMIC method: Factors of influence', SMEF 2007 Conference, Rome, Italy. (Obtainable from: http://metrieken.sogeti.nl/Home/SEC/Publicaties.jsp.)

[4] 13 application projects of size below 100 CFP were excluded from the analysis because it was found that the 13 projects had an average of 10 functional processes, all of them small (average size 6.6 CFP). In these circumstances, use of four bands was found to be excessive and use of the 'average functional process' method would be more appropriate.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

'Small'	5.5 CFP
'Medium'	10.8 CFP
'Large'	18.1 CFP
'Very Large'	38.8 CFP

To size a new piece of software, using the steps B1 to B3 of the Average approach of Example 1 above, the functional processes of the new piece are identified and 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 sizing, the measurer can check if the contribution to the total size of the functional processes 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 scaling factors for this approach is critically important for accurate sizing since functional processes typically exhibit a skewed size distribution, as illustrated by the Vogelezang and Prins 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.

1.2.4 Example 4, the 'Average Use Case' approach

The approaches of Examples 2 and 3 have successively refined that of Example 1 with the aim of achieving higher accuracy of the approximate size. All three Example approaches, however, are applicable at the functional process level of granularity.

Thes approach of Example 4 extends that of Example 1 to a higher level of granularity, namely of 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 measured by the Example 1 approach). Hence the average size of a use case according to this local definition, is 3.5 x 8 = 28 CFP per use case.

For a new project with 12 use cases, the software size would be 12 x 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 of Example 1. 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).

1.3 Approximate sizing of changes to functionality

To estimate the approximate size of a requirement to change some existing software, the process follows the same principles. 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.

---

[5] In the Measurement Manual v2.2, the average sizes of these bands were given as 'Small' = 3.9 CFP, 'Medium' = 6.9 CFP, 'Large' = 10.5 CFP, and 'Very Large' = 23.7 CFP. In fact these are the average sizes of bands where each band contained 25% of the number of functional processes of the software, not 25% of the size of the software. We apologize for this error.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

a) If possible, on basis of each Functional User Requirement, 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 have to be changed. So 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.

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

1.4 Early 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 requirements to detailed requirements, to functional specification, etc. The 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

(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 above, even when calibrated locally, will not usually 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 possible 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 is related to risk analysis, a subject that goes beyond the scope of this Guideline. However, it may be helpful to address the following questions.

- Are the requirements on this project particularly uncertain at the outset? If so, what correction to the approximate size should be made for possible scope creep?
- Is scope creep endemic within the organization and can we use past measurements to help quantify this phenomenon? For instance, in a given organization and using a given development process for which a lot of 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'. Such a finding can then be used in a new project to take a measurement at the end of Phase 1 using an approximation approach, and to add 30% to the approximate size to predict a most-probable size at the end of phase 3.

1.5 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 accurately using the standard COSMIC measurement method. But approximate sizing methods need to be used with care.

When there is a need for approximate sizing , the measurer 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;

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

- pay particular attention to identifying the large functional processes and to determining good scaling factors for them, since they are few in number, but make a large contribution to the total size;
- consider whether an allowance should be made for ‘scope creep’ when publishing an approximate size;
- estimate and report the plus or minus uncertainty on the approximate size, mentioning any correction that has been made for scope creep; estimating the uncertainty on an approximate size is especially important in contractual situations.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

2

ENSURING THE COMPARABILITY OF SIZE MEASUREMENTS

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 functional user requirements ('FUR') are comparable. The circumstances in which it may be necessary to apply the ideas described here can arise when functional size measurements must be made in the early stages of a large software project when the FUR may be being sub-divided and worked out in varying levels of detail by different teams.

It may also be necessary to apply these ideas when measurements must be compared from different, independent sources, or when a measurement made in one organization must be used as input to a supplier's process such as to an estimating tool, or to control price/performance, or for benchmarking purposes.

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.

2.1 The evolution of functional user requirements in the early stage of a large software project

In the early stage of a large software development project, when the FUR are first being established, two processes may be happening. First, the FUR are being defined in ever-more detail. Second, the FUR 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 FUR to be measured exist at various 'levels of granularity' and at various 'levels of decomposition'. Since it is easy to confuse these two concepts, their definitions are reproduced below.

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.

DEFINITION – Level of decomposition
Any level of division of a piece of software showing its components, sub-components, etc.

When faced with FUR 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.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

To illustrate the type of difficulties faced by measurers, we provide two examples. First 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 difficulties of measuring FUR at different levels of granularity

Second, we consider an example from a telecoms software architecture. This example illustrates the difficulties of measuring FUR that are being defined to lower levels of granularity and are being decomposed into smaller 'chunks' at the same time, in parallel.

2.2 Measuring at varying levels of granularity in a business application - the 'Everest' system [6]

As stated in the Measurement Manual, the description of the part of the Everest system that is given and analyzed in that document 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 start again to describe the total Everest application at its highest levels of granularity, we might show it as a set of systems, of which 'customer ordering' would be only one system. The others 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 work on each system independently. The task for someone who must measure the FUR of any one system would therefore be to understand the FUR as they evolve at lower and lower levels of granularity. There would be no problem of further levels of decomposition to deal with.

Recalling some observations on this case study, as we 'zoom-in' to lower levels of granularity of the FUR of the customer ordering system:

- the scope of the system to be measured does not change
- the functional users (individual customers who place orders) do not change. A customer can 'see' the functionality of the system 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 diagram in the Measurement Manual showing a possible analysis of the Everest ordering system illustrates, in practice functional processes can occur at various levels of such a diagram. 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. In practice, it cannot be assumed (as assumed in the previous chapter 1 for the sake of simplicity) that at any given moment all parts of a functional model have evolved to the same level of granularity and that the same one scaling factor can be applied to each part.

[6] For a full discussion of the 'Everest' example, see the COSMIC Method v3.0, Measurement Manual.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

2.3 Measuring at varying levels of granularity and of decomposition in a pure software architecture – a telecoms system

This example illustrates an approach to sizing the Functional User Requirements ('FUR') of software as they are defined at lower and lower levels of granularity that differs from the approach described for the 'Everest' system given in section 2.2. The example in this section 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.

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

2.3.1 Description and analysis of the software architecture

Figure 2.3.1 shows a 'Logical Network Element' (or LNE) within the software architecture and the analysis of its functionality into components at two lower levels of granularity, namely the 'System Component' (or SC) level and the 'Sub-system' (or SS) level.

Models such as shown in Figure 2.3.1 are of course produced in the telecoms company in three stages (at each level of granularity) and the requirement 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', similar to the examples discussed in chapter 1. 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 2.2 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 analyzed, in addition to the 'zooming-in'.

An inevitable consequence of decomposing a piece of software (and hence of decomposing its FUR 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. 2.3.1 by the solid line at the LNE level, the dashed lines at the SC level and the dotted lines at the SS level. Note that there is *only one layer* involved in this case, namely the layer in which all components (Sub-Systems, System Components and Logical Network Elements) reside. The Sub-Systems are *not* subordinates of the System Components but *represent* the System Components when the latter are shown at a level of granularity with more detail and are sub-divided. The same holds for the System Components which represent the Logical Network Elements.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

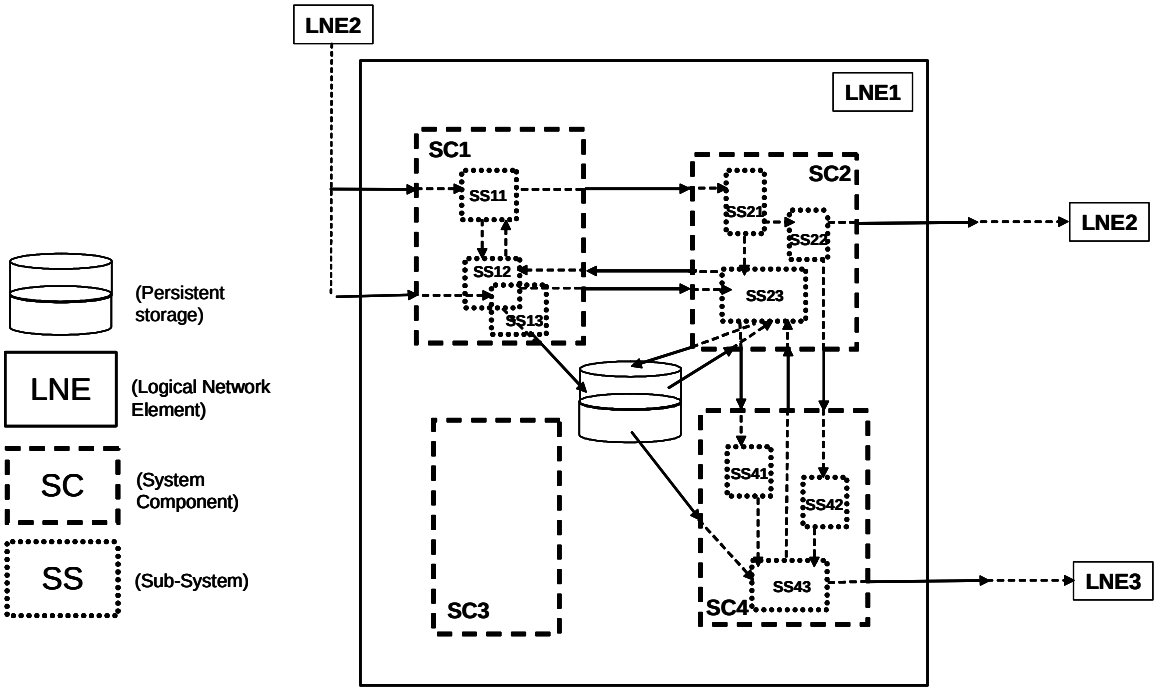

Figure 2.3.1 - A Line Network Element and its analysis into two lower levels of granularity

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 2.3.1 shows, at the highest level of granularity, a 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, the diagram 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 2.3.1 shows that there are several data movements between the System Components within LNE1. For such exchanges, the System Components are functional users of each other. 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 2.3.1 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. The diagram shows which components of LNE1 are involved in this functionality as we decompose to lower levels of granularity.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

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 project estimating.

2.3.2 Sizing the Logical Network Element

With this analysis approach, the size of the functionality shown in Figure 2.3.1 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 2.3.1 at each level of granularity are as follows. (Abbreviations used are FP = functional process, E = Entry, X = Exit, R = Read and W = Write.)

At the Logical Network Element level of granularity:

1. The size of the FP of LNE1 is 2E + 2X + 2R + 2W = 8 CFP

At the System Component level of granularity:

1. The size of the FP of SC1 is 3E + 2X + 0R + 1W = 6 CFP
2. The size of the FP of SC2 is 3E + 4X + 1R + 1W = 9 CFP
3. The size of the FP of SC3 = 0 CFP
4. The size of the FP of SC4 is 2E + 2X + 1R + 0W = 5 CFP

At the Sub-System level of granularity

1. The size of the FP of SS11 is 2E + 2X + 0R + 0W = 4 CFP
2. The size of the FP of SS12 is 2E + 1X + 0R + 0W = 3 CFP
3. The size of the FP of SS13 is 1E + 1X + 0R + 1W = 3 CFP
4. The size of the FP of SS21 is 1E + 2X + 0R + 0W = 3 CFP
5. The size of the FP of SS22 is 1E + 2X + 0R + 0W = 3 CFP
6. The size of the FP of SS23 is 3E + 2X + 1R + 1W = 7 CFP
7. The size of the FP of SS41 is 1E + 1X + 0R + 0W = 2 CFP
8. The size of the FP of SS42 is 1E + 1X + 0R + 0W = 2 CFP
9. The size of the FP of SS43 is 2E + 2X + 1R + 0W = 5 CFP

In summary, the total measured sizes of the FUR are:

- At the LNE level of granularity = 8 CFP
- At the SC level of granularity = 6+9+0+5 = 20 CFP

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

- At the SS level of granularity = 4+3+3+3+3+7+2+2+5 = 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.

2.3.3 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 does not neatly conform to the Rule (a) for the 'functional process level of granularity' given in chapter 1, which states: 'Functional size measurement should always be made at the functional process level of granularity' (where the 'functional process level of granularity' is defined as:

- '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)
- (etc)')

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 Rule.

One way of avoiding this apparent invalidation of the Rule would be to change the 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 this does not solve the whole problem. A more fundamental issue is 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 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 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.

2.3.4 Conclusions from the LNE example

Software may be sized at any level of granularity where the functional users of the piece of software can be identified at that level and functional processes can be identified.

The uncertainty about what is meant by a 'functional process level of granularity' arises only in cases where the functional users are only 'pieces of software', i.e. the measurer is faced with measuring components of a purely software architecture.

Within any one organization, it should always be possible to define (maybe by examples) what is meant by 'a piece of software' and to define standard level(s) of granularity, so as to ensure common levels of granularity for measurement. However if measurements of components of pure software architectures must be compared across different organizations, great care should be taken to ensure

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

that a common level of granularity is adopted. Without such a check, functional size measurements may not be comparable across the two organizations

2.3.5 Computing 'scaling factors' for the LNE example

The process of computing scaling factors between the three levels of granularity of the Line Network Element is the same as for the examples in chapter 1.

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 2.3.2 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 FUR 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.

2.4 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 a standard functional process level of granularity.

The example of Figure 2.3.1 shows that as software is decomposed to lower levels, the measured size increases due to inter-component data movements. (This particular example is rather extreme, but even decomposing a business application into its main peer components that are designed to execute on multiple technology platforms will typically reveal several inter-component data movements.) The sum of the sizes of the parts of a piece of software will always be greater than the size of the whole – see the rules on aggregation of sizes in the Measurement Manual.

COSMIC recommends, therefore, 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.

The COSMIC organization has no authority to specify such standard levels of decomposition, but suggests the following as example candidates for standardization.

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

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

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

3

CONVERTIBILITY OF COSMIC MEASUREMENTS

Users of '1st generation' functional sizing methods may wish to understand if their data can be converted to size measurements according to the COSMIC method. This chapter sets out some considerations on converting sizes measured with three '1st generation' functional sizing methods (IFPUG, MkII, and NESMA)[7] to the COSMIC method. The reader is assumed to know the details of the 1st generation method of interest and of the COSMIC method.

Note that all functional size scales are arbitrary in absolute terms. So whether a set of sizes measured on one scale happens to be bigger or smaller, on average, than when measured on the other scale is immaterial. Existing functional size measurement methods have not in general been designed to give similar sizes for the same piece of software, but it happens that the four methods of interest discussed here all give sizes that are of the same order of magnitude. The only questions to ask when considering convertibility are therefore:

a) do sets of measurements on any two size scales happen to correlate in any statistically significant sense, and if so, how?, and
b) what causes an individual pair of measurements to depart significantly from a best-fit regression curve?

Since the IFPUG and MkII methods were designed to measure functional sizes of software from the business application (or 'MIS') domain, convertibility is really only an issue for software from that domain.

Conversion of sizes measured with the COSMIC method in the 'other direction' to sizes according to a 1st generation method involves similar considerations. Such conversion might be relevant for the purposes of providing input to a process, e.g. project estimating or benchmarking, that accepts only sizes measured using a 1st generation method. However, in the long-term, re-calibrating the estimating or benchmarking process with COSMIC-based measurements would be preferable to backwards conversion.

3.1 Convertibility: Theory and practice

It would be ideal if sizes measured with other methods could be exactly converted to COSMIC sizes by widely-applicable mathematical formulae, but there are theoretical reasons why this is not possible. This leaves three possible approaches for conversion that we will describe here, namely

- Theoretical conversion within an empirical size range
- 'Manual conversion'
- Statistically-based conversion formulae

[7] When referring to 'IFPUG' and 'MkII' sizes here, we always mean 'Unadjusted Function Points'. The IFPUG and MkII 'Value Adjustment Factors' are not recognized by the NESMA or COSMIC methods. All the statistically-based results reproduced here use 'unadjusted' function point sizes.

The NESMA method is a variant of the IFPUG method developed by the Netherlands Software Metrics Association. For the purposes of this chapter, it may be assumed to have the same measurement rules and to give similar sizes to those of the IFPUG method, unless noted otherwise.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

First, understanding the reasons why exact conversion of sizes by mathematical formulae is impossible should help to understand the limitations of the other three approaches.

3.1.1 The impossibility of exact conversion by mathematical formulae

Measurements of sizes of FUR using 1st generation methods are almost always made assuming the 'users' are individual humans and/or other peer applications. This same assumption will normally be made for the 'functional users' when using the COSMIC method for software from the business application domain. So it is normally safe to assume that measurements of the size of the FUR of a business application made by a 1st generation method and by the COSMIC method are comparable in that they are both made at the same level of granularity of the FUR.

However, *exact* mathematically-based conversion formulae from sizes measured with a 1st generation method to COSMIC sizes are impossible because the Base Functional Components (or BFC's) of any of these three methods do not map exactly to the BFC's of the COSMIC method and because of differences in the measurement rules.

The following examples illustrate why exact conversion is not possible.

EXAMPLE 1: The IFPUG method includes both direct contributions to size from 'files' referenced by the software being measured, and indirect contributions (via the contribution of the 'number of files referenced' to the complexity classification of each elementary process). The COSMIC method only includes contributions to size from 'persistent data' via Read and Write data movements in each functional process. Furthermore, the IFPUG concept of a 'file' does not always map directly to a COSMIC 'object of interest'.

Hence a piece of software measured on the IFPUG method that refers to many files, especially 'complex' files, but has few references-per-file from its elementary processes will probably have a much larger IFPUG size than the corresponding COSMIC size.

The MkII method measures the contributions of references to stored data ('logical entities') in a way that is close to that of the COSMIC method but which is not identical in its detailed rules.

EXAMPLE 2: Whilst the definitions of an IFPUG 'elementary process' and a COSMIC 'functional process' appear to have the same intent, the detailed counting rules of the two methods differ in what they consider to be separate processes for various types of functionality. A principal example is that functions to maintain 'code tables'[8] are not considered as elementary processes, i.e. they are ignored altogether by the IFPUG method, whereas the COSMIC method may consider such functions as functional processes for functional users such as system administrators who use them to maintain the code tables. The MkII method has a similar but not identical convention to the COSMIC method. The NESMA method is similar, but not identical to the IFPUG method

EXAMPLE 3: The contributions to size of the input and output parts of IFPUG elementary processes (and of MkII 'logical transactions') take into account the number of data attributes in the input and output. The COSMIC method considers only the numbers of objects of interest about which data is moved in the input and output parts of a functional process.

EXAMPLE 4: The BFC's of the IFPUG method all have an upper size limit, whereas there is no upper limit to the size of a MkII or COSMIC functional process. Hence a piece of software whose COSMIC functional processes have large numbers of data movements (say, well above the maximum size of an IFPUG elementary process, namely 7 FP) will probably have a size measured in CFP that is much larger than the size measured in IFPUG FP. Similarly, a piece of software that has a high number of data attributes per functional process will probably have a size measured by the MkII method that is much larger relative to the sizes measured by both the IFPUG and COSMIC methods.

---

[8] 'Code tables' mostly have two attributes, a code and a description that identify some 'thing', e.g. 'customer type', 'sex' (e,g, of an employee), 'currency', etc.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

EXAMPLE 5: Sizes of software measured with the COSMIC method take into account the concept of software 'layers'. 1st generation FSM methods such as IFPUG and MkII FPA were designed to measure only business application software, so the meaning of size measurements obtained by using these methods to measure software in other layers is questionable.

Suppose, then that we have a software development project requiring a new business application and also changes to multiple layers of the infrastructure software. The IFPUG or MkII FP methods may well give a size for the business application component that can be converted reasonably accurately to a COSMIC size using a statistically-based conversion formula. But as neither of the 1st generation methods was designed to measure software from other layers, it is unlikely that these methods could be used to measure any sizes of the changes to the software in the infrastructure layers that would sensibly correlate with sizes measured by the COSMIC method.

EXAMPLE 6: As noted in the Measurement Manual, sizes of changes to software are measured in quite different ways. The COSMIC method (and the MkII FPA method) measures the size of the required changes to the software that must be changed. The IFPUG method measures the total size of the parts of the software that are required to be changed. It is highly unlikely, therefore that any useful statistical correlation will be found for sizes measured by the IFPUG and COSMIC methods on 'change' projects (or for 'enhancement' projects as they are often called).

3.1.2 Theoretical conversion within an empirical range

Cuadrado et al[9] have published a method based on a mapping of the IFPUG and COSMIC BFC's that gives an empirically-defined upper and lower bound for the COSMIC size corresponding to a given IFPUG size. The method requires knowledge only of the number of file-type references made in each of the elementary process types (EI, EO and EQ) of the IFPUG measurement.

The process for determining the upper and lower bounds of a COSMIC functional process size, for a piece of software for which the IFPUG size is known is as follows.

- The COSMIC upper bound size for each IFPUG elementary process must be the maximum of (2, twice the number of its file references plus 1)
- The COSMIC lower bound size for each IFPUG elementary process must be the minimum of (2, the number of its file references plus 1)

The maximum and minimum COSMIC sizes for the piece of software are then obtained by summing the above bounds over all elementary processes (EI's, EO's and EQ's).

The method was tested on measurements of 33 software applications measured with the IFPUG v4.1 and COSMIC v2.2 methods.

This method would still be subject to the types of uncertainty discussed in Examples 2, 4, 5 and 6 in section 3.1.1.

3.1.3 Manual conversion

'Manual' conversion of a functional size measured by a 1st generation method to a COSMIC size is possible when the basic raw data of the 1st generation size are available, AND there is some expertise available on the software which can help make good judgments or intelligent guesses on the equivalence between the BFC's on the two methods.

Manual conversion will be illustrated by an outline of the process for conversion of IFPUG sizes to COSMIC sizes. A very similar process should be used for conversion from MkII to COSMIC sizes.

---

[9] Cuadrado, Rodriguez, Machado and Abran, 'Convertibility between IFPUG and COSMIC Functional Size Measurements', 8th International conference on product-focussed software process improvement, Riga, Latvia, July 2007. Paper available from www.gelog.etsmtl.ca/cosmic-ffp

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

The 'raw data' needed for the IFPUG sizes would ideally be the list of file types and their numbers of record types plus, for each of the types of elementary processes, the numbers of 'DET's ('data attributes' in COSMIC terminology) and file references.

The manual conversion process would then entail the following steps. Some knowledge of the business functionality of the software will be necessary for each step, if the conversion is to be reasonably accurate. Detailed knowledge of both measurement methods is essential.

- Use the list of files and record types to draw up the corresponding list of objects of interest, showing the mapping
- Examine each IFPUG elementary process to determine the equivalent COSMIC functional process(es). They will mostly be equivalent, but not always, e.g. consider Example 2 in section 3.1.1
- Examine the DET's of each elementary process and, using the list of the objects of interest, determine the number of Entries and Exits for each functional process
- Examine the file references of each elementary process and, using the list of the objects of interest, determine the number of Reads and Writes for each functional process
- The total of Entries, Exits, Reads and Writes over all functional processes will then be the size in CFP

This process may not be reliable as described (i.e. it will need further refinement locally) in the various circumstances such as listed in the Examples of section 3.1.1. As a check on the accuracy of the result, an analogous manual conversion process could be run in reverse, starting with the COSMIC size components and list of objects of interest, to generate the equivalent IFPUG size

3.1.4 Statistically-based conversion formulae

Statistically-based conversion formulae face two difficulties. First, to be reasonably accurate they need a reasonable number (say 10 and ideally many more) of a range of functional sizes to be measured on both the '1st generation' method and on the COSMIC method. Such repeat measurements require a lot of effort, though several studies have now been published (typical specific results are mentioned in 3.3). Second, due to the non-matching of the BFC's of any of the 1st generation method with the BFC's of the COSMIC method, the statistical correlation may be valid for only a limited dataset and/or whilst the average correlation of sizes may be good, there may be significant outliers.

Nevertheless the statistical correlation studies published so far have shown reasonable correlations in various circumstances. So an organization wishing to convert its base of measurements made with a 1st generation method to COSMIC sizes using a statistical approach is recommended to make sufficient measurements to develop its own statistically-based conversion formulae, using regression analysis, subject to the guidance given below.

3.2 A process for developing a statistically-based conversion formula to COSMIC sizes

If an organization has an historical database of measurements with a '1st generation' method and wishes to adopt the COSMIC method, then the following process should be used to establish a statistically-based formula to convert the existing measurements.

- Re-measure with the COSMIC method the FUR of a sample of the pieces of software which have been measured with the 1st generation method
- Seek correlations between the two sets of measurements so as to develop a local statistically-based 'average conversion' formula (or more than one formula if the data indicate there may be sub-sets of the measurements with different characteristics)
- Apply these formulae derived from the sample to convert the bulk of the measurements to the new scale

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

- Examine the components of the original measurements (before conversion) for any reasons as to why the result converted with the 'average conversion' formulae should be inaccurate.

The last point is important. Statistically-based 'average conversion' formulae may be 'on average' accurate enough, but may not be accurate enough for all individual cases. We need therefore to understand the factors such as listed in the Examples of section 3.1.1 above that may, for a given 1st generation method, cause measurements for certain types of software to be inaccurately converted with this statistical approach. (There is also the other important consideration of whether the sample used to derive the conversion formulae is statistically representative of the full set of measurements to be converted.)

In the following description of some measurements of convertibility and of the factors leading to outliers, we use expressions such as 'relatively high proportion', or 'large numbers of', etc. These expressions are relative to 'average' or 'typical' software, for the parameter concerned, for the software of the organization carrying out the conversion.

3.3 Statistically-based conversion from IFPUG Function Points, v 4.1 [10]

The table below lists the results of five studies where measurements have been made of sizes on the same pieces of software by both the IFPUG and COSMIC methods. Apart from the last dataset of van Heeringen, all the measurements were made on data collected almost entirely within a single organization.

(In the table below R is the correlation coefficient and R^2 is the determination coefficient. The latter expresses the proportion of variation in the COSMIC size (in CFP) that is explained by a change in the FP size (either IFPUG or NESMA)).

Author	No. of data points	Range of Sizes (FP)	Conversion Formula obtained by regression analysis	R^2
Fetke (1999)	4	40 - 77	CFP = 1.1 x FP (IFPUG) – 7.6	0.97
Vogelezang & Lesterhuis (2003)	11	39 - 1424	CFP = 1.2 x FP (NESMA) – 87	0.99
Abran, Desharnais, Azziz (2005)	6	103 - 1146	CFP = 0.84 x FP (IFPUG) + 18	0.91
Desharnais & Abran (2006)	14	111 - 647	CFP = 1.0 x FP (IFPUG) - 3	0.93
Van Heeringen (2007)	26	61 - 1422	CFP = 1.22 x FP (NESMA) - 64	0.97

The first point to note from these results is the significant variation in the conversion formulae, dependent on the dataset. This emphasizes that an organization wishing to convert using a statistical approach would be best advised to establish its own conversion formula based on data from its own software

Van Heeringen's results[11] from measurements on 26 new development projects in a variety of organizations are shown in the graph below[12].

[10] Results quoted here were measured following the rules of the IFPUG method v4.1 or the NESMA equivalent. The results should be equally valid for IFPUG versions 4.2 and 4.2.1

[11] 'Changing from FPA to COSMIC. A Transition Framework' H. van Heeringen, presesented at the Software Metrics European Forum 2007, Rome, Italy (obtainable from http://metrieken.sogeti.nl/Home/SEC/Publicaties.jsp).

[12] Note that the vertical scale is shown in units of 'Cfsu', which was the abbreviation of the name of the COSMIC size unit for version 2.2 of the method. The abbreviation with version 3.0 is now 'CFP'.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

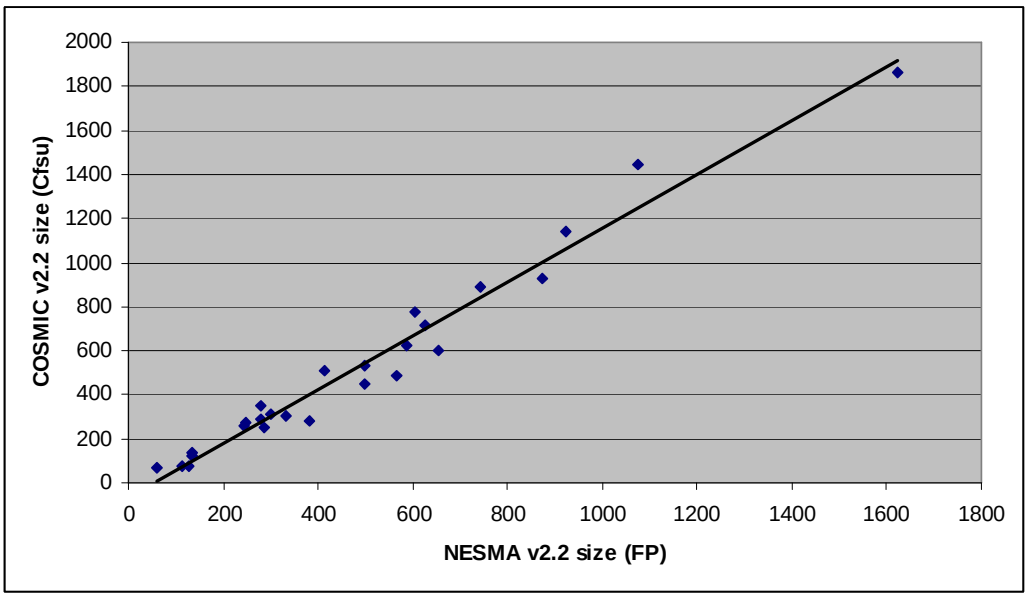

Figure 3.3.1 – NESMA (IFPUG) sizes versus COSMIC-FFP v2.2 sizes for 26 projects (van Heeringen, 2007)

Some observations about these measurements are as follows:

- All authors have (presumably for simplicity) fitted a straight line by least squares to the IFPUG and COSMIC size measurement pairs. With one exception13, all results show the straight line having a slope of 1.0 or more and passing through a point below zero on the IFPUG scale for a size of zero on the COSMIC scale. This result is to be expected, since the true 'average relationship' between the IFPUG and COSMIC size scales should be a curve that (a) has its origin at (3 FP, 2 CFP) – the smallest possible sizes on each scale respectively and (b) that shows COSMIC sizes on average rising faster than IFPUG sizes. The latter effect is expected due to the cut-off in the maximum possible size of IFPUG elementary processes (see section 1.3.1, Example 4 above).
- In general it is important to pay attention to the outliers of any simple statistical correlation. For example, there are several significant outliers on the graph of van Heeringen's data, though it must be remembered that his data originate from several different organizations, so would not be expected to be homogeneous. For his 26 projects, the regression line predicts the COSMIC size from the IFPUG size to within plus or minus 10% accuracy in 11 cases (42%). Eight (8) COSMIC sizes predicted by the regression line are more than plus or minus 20% different from the actual size.
- Van Heeringen provides a detailed explanation for the outliers. In particular, he notes that three of the 8 outliers are for small projects, that is, less than 150 FP. Some authors, notably Albrecht, have cautioned that special care is needed when applying FSM methods to smaller pieces of software, e.g. of size smaller than 100 FP, since the assumptions of FSM models may be less valid, the smaller the software. So poorer conversion accuracy for smaller sizes is to be expected. It is also worth noting that four of the conversion formulae given above have a negative constant, implying negative COSMIC sizes for low IFPUG sizes, which is impossible. These formulae should therefore not be used for conversion of sizes below around 150 – 200 IFPUG function points.

These three observations are further supported by fitting regression lines to van Heeringen's data after splitting the data into two groups, less than and greater than 200 NESMA FP. The regression lines are:

[13] The slope of the straight line fitted to the 6 data points in the paper by Abran et al is strongly influenced by one data point (934 CFP, 1146 IFPUG FP). In general, this dataset is not very homogeneous, leading the authors to note that it led to 'non-straightforward convertibility'.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

For NESMA FP < 200, (5 points) CFP = 0.68 x FP (NESMA) + 16 (R^2 = 0.45)

For NESMA FP > 200 (21 points) CFP = 1.24 x FP (NESMA) – 80 (R^2 = 0.96)

Abran, Desharnais and Azziz (2005) found precisely the same effects when grouping Vogelezang and Lesterhuis' 11 data points into two sets below and above 200 NESMA FP.

Summarizing the observations from statistically-based size conversion studies

- The true 'average relationship' between the IFPUG and COSMIC size scales should be a curve that (a) has its origin at (3 FP, 2 CFP) and (b) that starts 'flatter' (i.e. IFPUG sizes greater than COSMIC sizes), but above about 200 FP shows COSMIC sizes on average rising faster than IFPUG sizes;
- A statistically-determined average relationship between IFPUG and COSMIC sizes should not be relied upon for conversion of sizes much less than 200 IFPUG FP; the actual data points are too scattered; Manual conversion or re-measurement of sizes using the COSMIC method is recommended for small software sizes;
- Above 200 IFPUG FP, measurers should be aware of the factors (see the examples in section 3.1.1) that can give rise to outliers from any statistically-derived average relationship and should be prepared to manually correct sizes obtained by a conversion formula for these factors.

3.4 Statistically-based conversion from Mk II FP's, v. 1.3.1

The MkII FP method regards FUR as a set of 'logical transactions' which are identical in intent to the functional processes of COSMIC. Each logical transaction is broken down into input, processing and output components.

The size of a processing component of a logical transaction in the MkII FP method is defined to be proportional to the number of entity-types referenced during the processing. Since an entity-reference is generally equivalent to a Read or Write, the contribution to size of the processing component should be roughly equivalent on both scales.

However, the size of the input and output components of a MkII FP logical transaction is defined to be proportional to the number of data element types (or DET's) on the components respectively. Hence software with an exceptionally low proportion of DET's per logical transaction, or with an exceptionally high proportion will, if its size is converted by an 'average conversion' formula, be under-sized or over-sized respectively.

The smallest size of a MkII FP logical transaction is 2.5 MkII function points and there is no upper limit to its size (compare 2 CFP and no upper limit on the COSMIC method, respectively). In general, therefore, one would expect a good, average linear correlation of MkII FP and COSMIC sizes, with fluctuations about the average caused by the relative numbers of DET's on the inputs and outputs of the logical transactions.

3.5 Conclusions on COSMIC size convertibility

For an organization wishing to move from a 1st generation FSM method to the COSMIC method, the issue of 'can we convert our existing data?' (and then use the converted data to re-calibrate e.g. our estimating method) may be economically important.

Whilst exact conversion of the sizes of 'whole' business applications is impossible for a variety of reasons, the conversion methods outlined in this chapter should, with care, provide results of sufficient accuracy for most practical purposes. The following is recommended

- Organizations wishing to convert sizes measured with a 1st generation method to COSMIC sizes are recommended to use one or (ideally) more of the three methods described in this chapter

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

- It is probably best to start by accurately measuring some pieces of software with both the 'old' and 'new' methods, in order to seek a local statistical correlation of sizes, since there is evidence that local characteristics of the software can influence the nature of the correlation
- When using the statistically-based formula to make the conversion, each measurement pair ('old' and 'new') should be examined using the ideas of the 'manual conversion' method and/or using the upper and lower bounds of the 'empirical bounds method' to test the plausibility of the 'new' size, and to make a correction based on best judgment, if needed. Statistically-based conversion formulae should not be relied upon for converting sizes of much less than 200 IFPUG FP to COSMIC CFP

Finally, it should be remembered from Example 6 of section 3.1.1 that conversion of sizes of *changes* (or of 'enhancements') to software according to the IFPUG method to the equivalent COSMIC sizes is practically impossible. Re-measurement of the sizes using the COSMIC method, if needed, is probably the only option.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

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@cosmicon.com

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.cosmicon.com site.

The decision of the COSMIC MPC on the outcome of a CR review and, if accepted, on which version of the business application Guideline the CR will be applied to, is final.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)

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.cosmicon.com web-site for further detail.

All rights reserved. The Common Software Measurement International Consortium (COSMIC)