COSMIC

# Early Software Sizing with COSMIC: Practitioners Guide

February 27, 2020

(minor update May 9, 2020)

# Table of Contents

| **Acknowledgements: Editor & reviewers 2020 (alphabetical order).** | | | |
|---|---|---|---|
| Alain Abran<br>École de Technologie Supérieure, Canada | Arlan Lesterhuis<br>COSMIC<br>the Netherlands | Bruce Reynolds<br>Tecolote Research<br>United States | Asma Sellami<br>University of Sfax<br>Tunisia |
| Hassan Soubra<br>German University of Cairo, Egypt | Sylvie Trudel<br>Université du Québec à Montréal, Canada | Francisco Valdés Souto<br>Spingere, Mexico | Frank Vogelezang *<br>METRI<br>The Netherlands |

* Editor

# 1. General principles of approximate sizing

The COSMIC method provides a standardized way of measuring a functional size of software.

In practice, it is at times sufficient or necessary to approximate a functional size[1]:

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

The intended audience for this document is for those who need to establish the size of a piece of software without time or details available to use the standard COSMIC method.

For those who are interested in a more in-depth view of the COSMIC approximation techniques, see the Experts Guide e[1] that discusses their pros and cons, and their recommended areas of application.

This document describes reasons why it may be necessary to approximate, how actual requirements are often expressed at varying levels of documentation and some general principles of how to recognise and apply approximation techniques.

The rest of the guide describes:

- Techniques for the Requirements Stage (sections 2-5).
- Techniques for the Feasibility Stage (section 6-8).
- A checklist with items to consider when using approximation.

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from the COSMIC worldwide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer.

Commercial organizations exist that can provide training and consultancy or tool support - see https://cosmic-sizing.org/organization/commercial-support/

## 1.1 When is approximate COSMIC sizing needed?

There are three main circumstances in which an approximate COSMIC functional size may be valuable:

1. when a size measurement is needed rapidly and an approximate size is acceptable if it can be measured much faster than with the standard method. This is known as 'rapid sizing';
2. early in the life of a project before the actual requirements have been specified in some detail but insufficient for a precise size measurement. This is known as 'early sizing';
3. 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.

[1] Instead of describing this subject as techniques to 'approximate sizing', it might be more accurate to describe it as techniques 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 refer to it as '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.

## 1.2 Issues in approximation of functional size.

### *1.2.1 Localization (calibration).*

Approximate sizing techniques are based on artefacts that are not standardized and may vary in their levels of functional details within organizations and across organizations. This implies that the scaling factors need to be calibrated locally. In this document, 'locally' implies that the environment in which the scaling factors for the approximation technique have been defined is representative of the environment the approximation technique is to be used in.

### *1.2.2 Approximate sizing by classification and scaling.*

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 requirement.

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.

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

### *1.2.3 Accuracy of approximate sizing.*

Any technique to approximate sizing is the result of a trade-off between ease and speed of measurement versus loss of accuracy. Therefore, the estimated amount of error of each technique should be established and reported. See the full Guideline [1] for more guidance.

## 1.3 Overview of the described techniques for the requirements stage.

| Technique | Strength | Weakness | Area of application |
|---|---|---|---|
| **Average Functional Process.** | Easy to use. | Domain dependent. | |
| | | Requires sampling. | Same as sample. |
| **Fixed Size Classification.*** | Easy to use. | Domain dependent. | |
| | Scaling factors are documented. | Assigning a class to an FP is subjective. | Size classification must fit the software. |
| **Equal Size Bands.** | Easy to use. | FPs need to be classified correctly. | Business and real-time embedded. |
| | More bands lead to a more accurate approximation. | Bands must be significantly far enough apart. | Skewed distribution of FPs. |
| | | Requires sample dataset. | |
| **Average size of Use Cases.*** | Easy to use when Use Cases are standardized. | Functionality assigned to a Use Case can vary. | Standardized UCs. |
| | | Scaling factor is a product of two factors that contain estimation. | |
| | | Requires sample dataset. | Same as sample dataset. |

**Table 1.1 – Overview of the approximation techniques for the requirements stage.**

* NOTE This technique works best with a symmetrical dataset and a standard deviation (σ-value) that is significantly smaller than the average functional process size.

## 1.4 Overview of the described techniques for the feasibility stage.

| Technique | Strength | Weakness | Area of application |
|---|---|---|---|
| Early & Quick COSMIC | Applicable when part of the requirements is not yet in detail. | Assigning a class to an FP is subjective. | Most suited when a part of the requirements is not detailed enough to identify FPs. |
| | Can handle different levels of documentation. | Not designed for enhancements. | |
| Software Iceberg Analogy | Can be mixed with standard measures. | Historical measurements needed for calibration. | Valid throughout the elicitation of the early requirements into real requirements. |
| | Can be scaled to different levels of documentation. | | |
| | Can be aligned with ISO-IEEE 29148 on Requirements Engineering. | | |
| EASY Function Point | Can be mixed with standard measures. | Calibration is time consuming. | Valid throughout the elicitation of the early requirements into real requirements. |
| | Can be scaled to different levels of documentation. | Mapping the requirements is subjective. | |
| | Can also be used to size enhancements. | | |

**Table 1.2 – Overview of the approximation techniques for the feasibility stage.**

# 2. AVERAGE SIZE OF FUNCTIONAL PROCESSES.

The technique of the average size of functional processes can be used when the actual requirements of a piece of software are known only to the level of functional processes but not to the level of data movements.

### A Determine the scaling factor.

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 technique.
5. Identify the standard deviation.

### B Approximation using the scaling factor.

1. Identify and count all the functional processes of the actual requirements of the software to be measured (e.g. = 40 functional processes).
2. *For a set of requirements:* The approximate functional size of the set if actual requirements of the software to be measured is estimated to be (number of functional processes x scaling factor) = 40 x 8 CFP = 320 CFP.

3. *For a specific requirement:* determine the approximate size range by using the average size +/- 1 standard deviation. From above: if the average is 8 CFP, and the standard deviation is 2 CFP, the range of a specific functional process is: [6,10 CFP].

## 3. Fixed Size Classification.

The technique depends on defining a typical size classification of the functional processes in the piece of software to be measured. A corresponding size, or scaling factor, is next assigned to each class, for all of its functional processes.

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.

Table 3.1 shows an example of a set of size classes that is in actual use in a specific business organization. The rows 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. For Medium or Large functional processes more data movements of each type are assumed. 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 [2].**

If the functional size of some actual requirements must be approximated 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 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.

## 4. Equal Size Bands.

In the 'Equal Size Bands' technique, 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.

**A   Determine the scaling factors.**

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. Sort the functional processes in ascending order and and present them graphically in ascending order together with their cumulative size.
5. Using the information from the cumulative distribution, split the sample into a number of bands that all have the same total size (and thus contain a different number of functional processes). 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.
6. Determine the average size, in CFP, of the functional processes in each band. These are the scaling factors for each band. These scaling factors are usually not integer numbers.

**B Approximation using the scaling factors.**

1. Identify for each of the functional processes to be approximated the size band in which it belongs.
2. Assign the scaling factor for that size band to the functional process.
3. Sum all the approximated functional processes to get the functional size approximation of the whole piece of software.

**Examples of reported use**

| 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 – Example of 4 Equal size bands from 37 business applications in [2].**

| 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 – Example of 4 Equal size bands from a major component of an avionics system in [2].**

Note the similarity between the *numbers* of functional processes in each of the four bands despite 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:

1. The functional processes of the new piece are identified.
2. They are classified as ‘Small’, ‘Medium’, ‘Large’ or ‘Very Large’.
3. The average sizes of each band (such as listed above but preferably calibrated locally) are 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.

## 5. Average Size of Use Cases.

The principle of the approximation is similar to the average functional process approximation from Section 2, but on a higher level of documentation, 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 Section 2). 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 will provide a basis for making a preliminary estimate of software size in units of CFP.

## 6. Early & Quick COSMIC Approximation

The Early & Quick COSMIC approximation technique 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 technique a Functional Process[2] (FP) is the smallest process. A Functional Process can be Small, Medium, Large or Extra Large, depending on its estimated number of data movements. This categorization is similar to the 'Fixed Size Classification approximation' technique discussed in Section 3.

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.

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.

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) 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

[2] The definition of a functional process in the E&Q technique differs from the COSMIC method standard definition. In a future version the COSMIC standard definition will be adopted. The functionality identified by the two definitions is intended to be exactly the same.

way not only leaves of the functionality tree may be directly quantified but also intermediate branches.

**Reference Table:**

The Early & Quick COSMIC approximation technique 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 6.1 then allows the measurer to assign a CFP value for that part of the actual requirements (this is applied for each identified level separately).

| 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)<br>CRUD + List (Small processes) | 15.6 | 20.4 | 27.6 |
| | Medium | CRUD (Medium/Large processes)<br>CRUD + List (Medium processes)<br>CRUD + List + Report (Small processes) | 27.6 | 32.3 | 42.0 |
| | Large | CRUD (Large processes)<br>CRUD + List (Medium/Large processes)<br>CRUD + List + Report (Medium pr.) | 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 – Estimation values for the functional categories of the Early & Quick technique** Error! Reference source not found.**.**

# 7. Software ICEBERG Approximation.

The software iceberg approximation [5] is based on the ISO/IEC/IEEE 29148:2018 [7] standard on Requirements Engineering and the iceberg analogy. In this analogy (Fig. 7.1), the visible part of the iceberg above the waterline (left) corresponds to the early description of the software requirements, which requirements are progressively described with more details, up to the final fully described details of these software requirements (the progressive visibility of the part of the iceberg that is under the waterline).

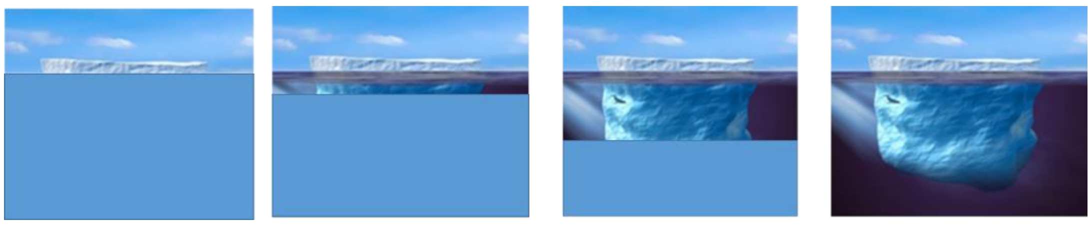

**Fig. 7.1. The Software-Iceberg analogy: from initially visible functions (left) to full view (right) [5].**

In physics there is a well-known ratio for the mass above to the mass under water for a floating iceberg, based on a number of empirical measurements and scientific observations. This ratio is a constant.

In software development there is no known constant ratio, but using empirical observations and measurements from COSMIC case studies, some approximation procedures can be worked out to come up with ratios for local contexts.

Across all software projects, functional visibility will vary across the development lifecycle, and hence software functional documentation across lifecycle phases will vary.

The ISO/IEC/IEEE 29148 standard on requirements engineering presents a number of concepts related to the sources, types and levels of detail of the requirements throughout the system and software life cycle.

The initial set of requirements originates from two sets of sources, the business stakeholders and other stakeholders, which leads to the ‘systems’ requirements. These sources provide the system contextual requirements, including the system purpose, system scope and system overview. From this contextual information, the following requirements are then identified:

- system functional requirements (some of which will be allocated to software),
- system non-functional quality requirements (some of which will be allocated to software).

ISO 29148 also notes that in addition to software functions explicitly identified, there may be interfaces identified, but not yet specified, as well as quality requirements, still at a high level.

In practice four levels of detail can be observed:

Level 1. **Business functions.** A list of ‘system’ functions, possibly including functionality relating to interfaces to other software applications.

Level 2. **Detailed business functions.** A list of ‘software’ functions: business functions allocated to software functional processes, including interfaces to other software applications.

Level 3. **Operational functions.** Implementation of the business requirements into designated software functional processes.

Level 4. **Detailed functions & quality functions.** Software functional processes that are fully detailed and include all quality requirements that the software must adhere to, including non-functional requirements allocated to software.

In [5] the requirements for the well-documented COSMIC Course Registration System case study have been broken down into the four levels of requirements. This led to the following scaling factors:

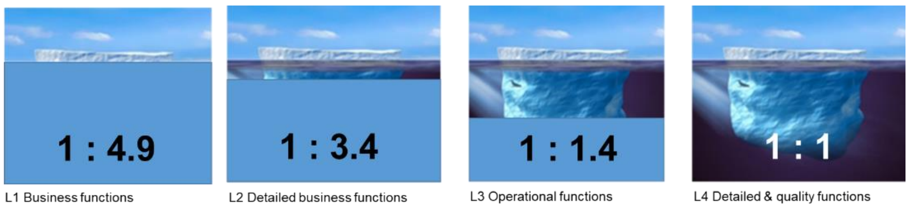

**Fig. 7.2. Scaling factors in the Iceberg approximation, derived from [5].**

Analysis of a second case study gave similar results. The iceberg-like ratios can then be used as scaling factors at various phases of the lifecycle and levels of documentation.

For instance, if COSMIC measurement is done very early on in the life cycle where only systems functions are identified and measurable with COSMIC, the scaling factor for level 1 functionality can be used to approximate the expected functional size at the end of the software development lifecycle.

Therefore, 100 CFP at the level of business functions would then reasonably be expected to grow to 490 CFP once the software is fully developed.

In the very earliest stages of new software development, such as at the system level in ISO-IEEEE 21948 before a project is formally defined, the requirements will probably be known only in the broadest outline. At this stage it may be possible to determine an approximate size using the Iceberg approximation with the known sizes of other existing software (preferably from your own domain), but it will be too early to apply any of the other approximate sizing techniques described in this Guideline.

# 8. EASY Function Point approximation.

The EASY (EArly & SpeedY) approximate technique provides most typical probability distributions for the measurer to pick from, and allows for approximate sizes and accurate sizes to be mixed.

Note: 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.

Example: "This report has '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 5x0.6 + 7x0.3 + 9x0.10 = 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' technique in previous Sections, where average or middle values are taken as being 'always' the most likely values.

Table 8.1 shows typical probability distributions of approximate values for common cases in the Business domain (FP stands for 'Functional Process') [4].

| 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 8.1 – Probability distributions of approximate values in the business domain [4] .**

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 technique.

# 9. Approximate sizing of changes of functionality & scope creep.

## 9.1 Approximate sizing of changes to functionality.

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 must 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 complement 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.

## 9.2 Approximate sizing & 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.

(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 techniques described in this Guide do not explicitly take into account scope creep. When using these approximation techniques 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 Guide. 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, can we then 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’.

## 9.3 Applying approximation in enhancement projects.

Often 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.

When approximating the size of a project at an early stage, the new and modifiefunctional processes must be identified, counted and multiplied by their respective average sizes.

# 10. Checklist.

Approximate sizing techniques need to be used with care. It is **strongly recommend** not to use any of the approximation techniques described in this Guide as simple ‘recipe books’. Always:

- ☐ Choose a technique that is optimal for the purpose of the measurement, given the time available for the measurement and the accuracy required of the approximate size and the availability of data for calibration. Other techniques can be found in the Experts Guide [1].
- ☐ Verify the approximation technique (by comparing precise and approximate measurements) using local requirements and measurements to ensure it produces reasonably accurate sizes in your local environment and calibrate the technique locally before using it for your own real measurements. See the Experts Guide for details [1].
- ☐ Try to obtain more detail than is given in the requirements from an expert in the software.
- ☐ Pay particular attention to identifying any large functional process and to determining good scaling factors for them: they can make a large contribution to the total size even though they are few in number.
- ☐ Whatever the purpose of using an approximation technique, whenever further information becomes available enabling a more accurate and/or precise sizing, refine and update the measurement. This is important when using the results as an input to estimation.
- ☐ Examine the requirements so that you understand the level(s) of documentation, the completeness and quality of the requirements before starting to use a technique. See the Experts Guide for detailed information on the level(s) of documentation [1].
- ☐ Consider whether a contingency should be made for ‘scope creep’ and for the contribution that incorporating Non-Functional Requirements may lead to.
- ☐ 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.
- ☐ Classifying the quality of the various parts of actual requirements by using the scheme of the ‘Guide for assuring the accuracy of measurements’ [6] can help determine the accuracy of an approximate size measurement.In contractual situations:
  - a) when enough measurement resources and calendar time are allocated for accurate measurement results, the COSMIC standard should be used.
  - b) when not enough measurement resources are available and when there are major time constraints, an approximate method form the Experts Guide could be used, provided that a range of measurement results (e.g. + or - %) must be supported and clearly identified and indicated in the measurement reporting’.

# 11. GLOSSARY OF TERMS

The terms in this Glossary are specific to this Guide. For other COSMIC terms, see the main Glossary in the COSMIC Measurement Manual.

**Accuracy.**
Closeness of agreement between a measured quantity value and a true quantity value of a measurand [ISO Guide 99] [8].

NOTE 1: The concept 'measurement accuracy' is not a quantity and is not given a numerical quantity value. A measurement is said to be more accurate when it offers a smaller measurement error.
NOTE 2: The term "measurement accuracy" should not be used for measurement trueness and the term "measurement precision" should not be used for 'measurement accuracy', which, however, is related to both these concepts.
NOTE 3: 'Measurement accuracy' is sometimes understood as closeness of agreement between measured quantity values that are being attributed to the measurand.

**Approximate Sizing.**
1. Approximate measurement of a size.
2. Measurement of a size by an approximate technique.

**Calibration**
Determining the scaling factors or classification values to be used in the local environment in which the approximation technique is used instead of the scaling factors or classification values published in reference documents like this Guide, aiming for the most accurate 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** (see also calibration)
Calibrating scaling factors or classification values to an environment that is representative of the environment the approximation technique 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 documentation of some actual requirements) to a size measured under another set of conditions (e.g. another level of documentation of the same requirements).

## 12. References.

[1] Vogelezang, COSMIC Group, “Early Software Sizing with COSMIC: Experts Guide’ https://cosmic-sizing.org/publications/guideline-for-early-or-rapid-cosmic-fsm/

[2] 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

[3] L. Santillo, "Early & Quick COSMIC-FFP Analysis Using the Analytic Hierarchy Process." 10th International Workshop on Software Measurement, *Lecture Notes in Computer Science 2006*, (pp. 147-160), Berlin (Germany), 2006.

[4] L. Santillo, "EASY Function Points – ‘SMART’ Approximation Technique for the IFPUG and COSMIC Methods", 22nd IWSM-MENSURA conference, Assisi (Italy), November 2012.

[5] A. Abran and S. Vedadi, “Development of COSMIC Scaling Factors using Classification of Functional Requirements”, 29th IWSM-MENSURA conference, Oct. 7-9, 2019, Haarlem, he Netherlands, CEUR Proceedings Vol-2476, 2019, pp. 31-46.

[6] COSMIC Group, Guide for assuring the accuracy of measurements.

[7] ISO 29148 Systems and Software Engineering -Life cycle processes- Requirements Engineering, International Organizations for Standardization (ISO), Geneva, 2011.

[8] ISO Guide 99: International vocabulary of basic and general terms in metrology (VIM), International Organization for Standardization – ISO, 2019.