COSMIC Measurement Manual for ISO 19761

Part 2: Guidelines

Version 5.0

March 31 2020

Foreword.

The COSMIC Measurement Manual for ISO/IEC 19761:2011 consists of three Parts:

Part 1: Principles, Definitions & Rules.

Part 2: Guidelines.

Part 3: Examples of COSMIC concepts and measurements.

This Part 2 of the COSMIC Measurement Manual presents the set of Guidelines developed by the COSMIC Group to facilitate accuracy, repeatability and reproducibility of COSMIC measurement results.

This COSMIC measurement manual is based on ISO/IEC standard 19761:2011, reconfirmed in 2019. This version 5.0 of the COSMIC Measurement Manual replaces version 4.0.2: it shortens version 4.0.2 while not modifying its substance in terms of measurement definitions, rules and guidance.

A public domain version of the COSMIC Measurement Manual and other technical reports, including translations into other languages can be found in the Knowledge base of www.cosmic-sizing.org.

Editors:

Alain Abran, Ecole de technologie supérieure – University of Quebec (Canada),
Peter Fagg, Pentad (UK),
Arlan Lestherhuis (The Netherlands).

Other members of COSMIC Measurement Practices Committee:

Diana Baklizky, (Brazil),
Jean-Marc Desharnais, Ecole de technologie supérieure – University of Quebec (Canada),
Cigdem Gencel, Free University of Bozen-Bolzano (Italy),
Dylan Ren, Measures Technology LLC (China),
Bruce Reynolds, Telecote Research (USA),
Hassan Soubra, German University in Cairo (Egypt),
Sylvie Trudel, Université du Québec à Montréal - UQAM (Canada),
Frank Vogelezang, Metri (The Netherlands).

Table of Contents

1. INTRODUCTION.

1.1 Purpose of this document.

The purpose of this document is to provide guidance to the practitioners for the application of the COSMIC ISO 19761 Function Points standard presented in Part 1.

This Part 2 is complementary to Part 1 and is supplemented by Part 3 which presents a large number of examples.

1.2. The COSMIC measurement process.

The COSMIC measurement process presented in Part 1 is reproduced in Figure 1.1

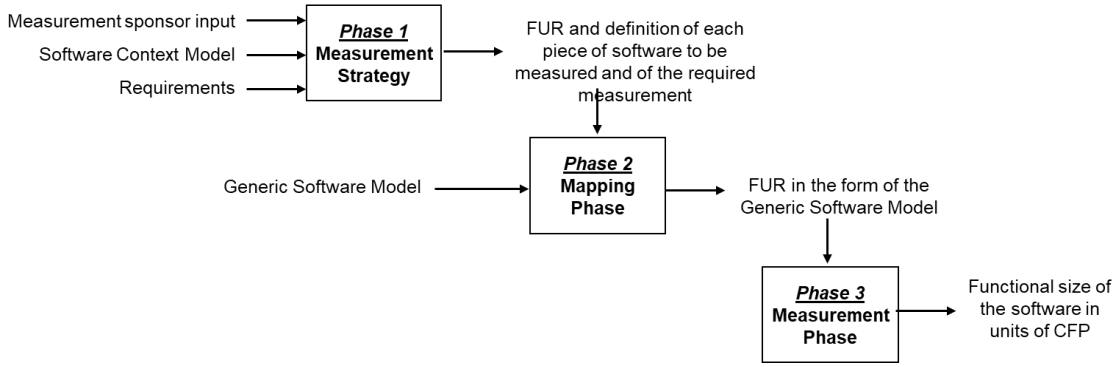


Figure 1.1 – The COSMIC method measurement process.

2. THE MEASUREMENT STRATEGY PHASE.

2.1 Overview of the measurement strategy phase.

Figure 2.1 presents graphically the Measurement Strategy Phase.

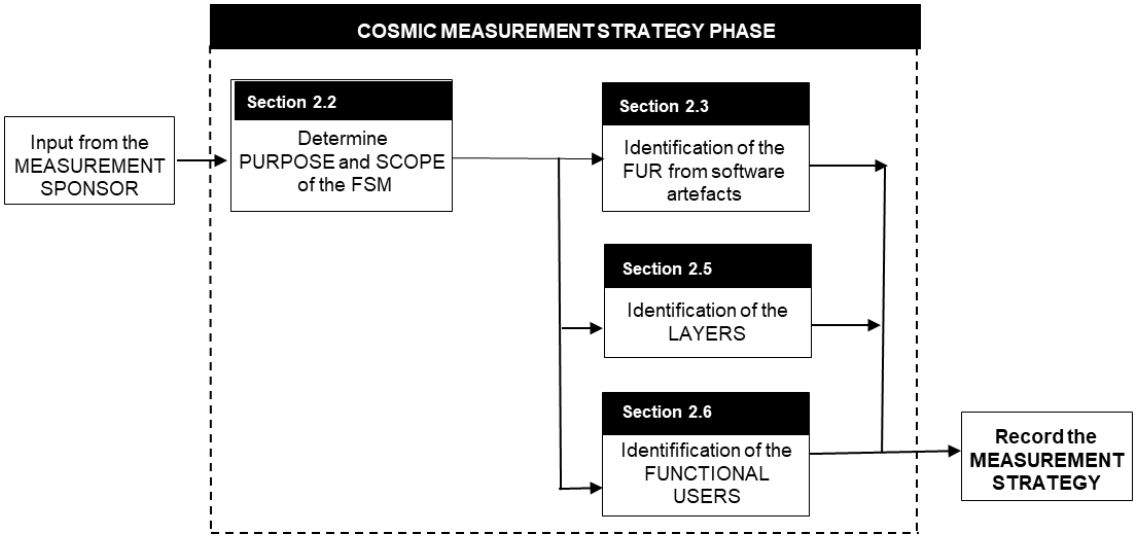


Figure 2.1 - The process of determining a Measurement Strategy.

2.2 Determine purpose and scope of the FSM.

The term ‘purpose’ is used in its normal English meaning. The purpose helps the measurer to determine:

- The scope of the measurement and hence the artefacts which will be needed for the measurement.
- The functional users.
- The functional changes.
- The point in time in the project life-cycle when the measurement will take place.
- The required accuracy of the measurement, and hence whether the standard COSMIC method should be used, or whether an approximation technique should be used (e.g. early in a project’s life-cycle, before the FUR are fully elaborated).

As an aid to determining a measurement strategy, the Guideline for 'Measurement Strategy Patterns' describes, for each of several different types of software, a standard set of parameters for measuring software sizes, called a ‘measurement strategy pattern’ (abbreviated to ‘measurement pattern’. Consistent use of the same measurement patterns should help measurers to ensure that measurements made for the same purpose are made in a consistent way, may be safely compared with other measurements made using the same pattern and will be correctly interpreted for all future uses. A side benefit of using a standard pattern is that the effort to determine the Measurement Strategy parameters is much reduced.

The COSMIC Group recommends that measurers study and master the COSMIC method, especially the Measurement Strategy parameters, before using the standard patterns.

2.3 Identification of the FUR from software artefacts.

Usually the measurer will have to extract the FUR from the available artefacts of the software, before mapping them to the concepts of the COSMIC ‘models of software’.

As illustrated in Figure 2.2, FUR can be derived from software engineering artefacts that are produced before the software exists. Thus, the functional size of software can be measured prior to its implementation in a computer system.

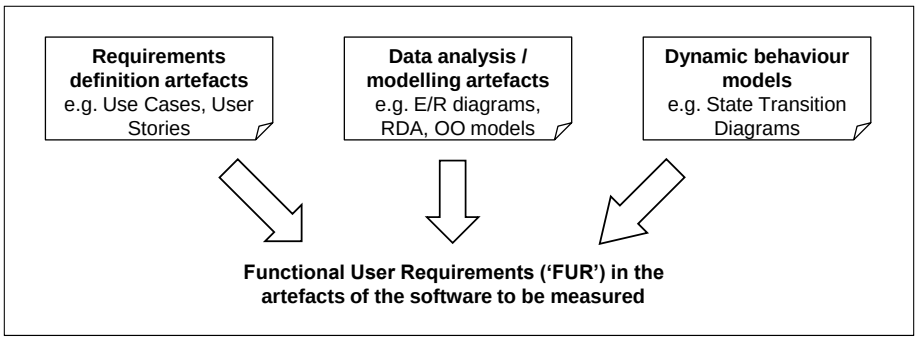


Figure 2.2 – Pre-implementation sources of Functional User Requirements.

NOTE: Some existing software may need to be measured without there being any, or with only a few, architecture or design artefacts available, and the functional requirements might not be documented (e.g. for legacy software). In such circumstances, it is still possible to derive the FUR from the artefacts of the computer system even after it has been implemented, as illustrated in Figure 2.3.

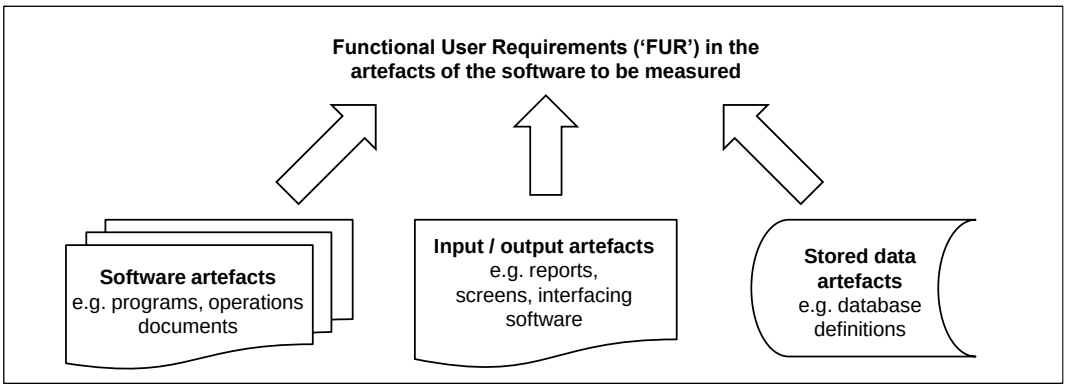


Figure 2.3 – Post-implementation sources of Functional User Requirements (FUR).

The process to be used and hence the effort required to extract the FUR from different types of software engineering artefacts or to derive them from installed software will obviously vary; these processes cannot be dealt with in the Measurement Manual. The COSMIC method assumes that the FUR of the software to be measured either exist or that they can be extracted or derived from its artefacts, in light of the purpose of the measurement.

If the measurer understands these two models, it will always be possible to derive the FUR of a piece of software to be measured from its available artefacts, though the measurer may have to make some assumptions due to missing or unclear information.

2.4 Non-Functional Requirements.

System NFR can be very significant for a software project. In extreme cases, a statement of requirements for a software-intensive system can require as much documentation for the NFR as for the functional requirements. The COSMIC method can be used to measure some requirements that may be first expressed as non-functional. Several studies have shown that some requirements that initially appear as *system* NFR evolve as a project progresses into a mixture of requirements that can be implemented in software functions, and other requirements or constraints that are truly 'non-functional'. See Figure 2.4.

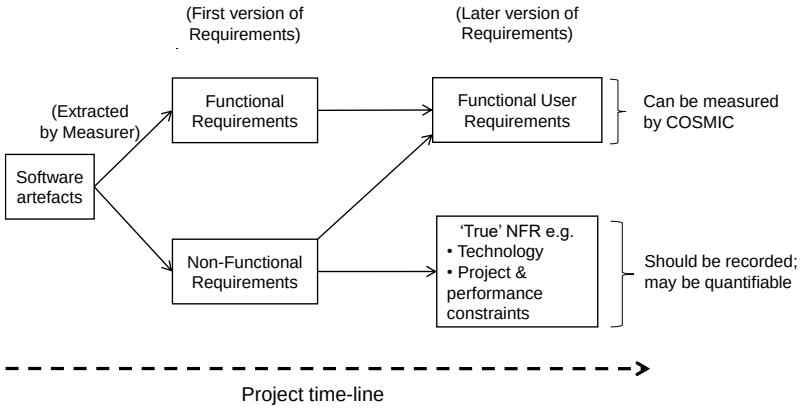


Figure 2.4 - Many requirements initially appearing as NFR evolve into FUR as a project progresses.

This is true for many system quality constraints, such as response time, ease of use, maintainability, etc. Once identified, these software functions that have been 'hidden' in NFR at the beginning of a project can be sized using the COSMIC method just as any other software functions. Not recognizing this 'hidden' functional size is one reason why software sizes can appear to grow as a project progresses.

More in-depth discussions on system and software Non-Functional Requirements (NFR) are presented in: 'Guideline on Non-Functional & Project Requirements'.

2.5 Identification of the layers.

Software architectures can be very complex, however COSMIC employs a very simplified view of the software architecture that is sufficient for the purpose of measurement. It is the task of the measurer to perform that simplification. This section provides guidance to the measurer.

Since the scope of a piece of software to be measured must be confined to a single software layer, the process of defining the scope(s) of FSM may require that the measurer first has to decide what are the layers of the software's architecture. This sub-section discusses 'layers' of software as these terms are used in the COSMIC method because:

- the measurer may be faced with measuring some software in a 'legacy' environment of software that evolved over many years without ever having been designed according to an underlying architecture The measurer may therefore need guidance on how to distinguish layers according to the COSMIC terminology;
- the expressions 'layer' and 'layered architecture' are not used consistently in the software industry. When the measurer must measure some software that is described as being in a 'layered architecture', it is advisable to check that 'layers' in this architecture are defined in a way that is compatible with the COSMIC method. To do this, the Measurer should establish the equivalence between specific architectural objects in the 'layered architecture' paradigm and the concept of layers as defined in this manual.

In a defined software architecture, each layer may have the following characteristics:

a) Software in one layer provides a set of services that is cohesive according to some defined criterion, and that software in other layers can utilize without knowing how those services are implemented.

b) The relationship between software in any two layers is defined by a 'correspondence rule' which may be either:

   - 'hierarchical', i.e. software in layer A is allowed to use the services provided by software in layer B but not vice versa (commonly referred to as Client-Server);
   - 'bi-directional', i.e. software in layer A is allowed to use software in layer B, and vice versa (commonly referred to as peer-to-peer (P2P).

c) Software in one layer exchanges data groups with software in another layer via their respective functional processes.

d) Software in one layer does not necessarily use all the functional services supplied by software in another layer.

e) Software in one layer of a defined software architecture may be partitioned into other layers according to a different defined software architecture.

Guidance on Rule 4: Identification of Layers

If the overall scope of the FSM extends over multiple layers, the measurer should proceed as follows:

- If the software to be measured exists within an established architecture of layers that can be mapped to the COSMIC layering characteristics as defined above, then that architecture should be used to identify the layers for measurement purposes.
- If however, the purpose requires that some software is measured that is not structured according to the COSMIC layering characteristics, the measurer should try to partition the software into layers by applying the principles defined above.

- Conventionally, infrastructure software packages such as database management systems, operating systems or device drivers, that provide services that can be used by other software in other layers, are each located in separate layers.

Normally in software architectures, the 'top' layer, i.e. the layer that is not a subordinate to any other layer in a hierarchy of layers, is referred to as the 'application' layer. Software in this application layer relies on the services of software in all the other layers for it to perform properly. Software in this 'top' layer may itself be layered, e.g. as in a 'three-layer architecture' of User Interface, Business Rules and Data Services components.

Once identified, each layer can be registered in the Measurement report, with the corresponding label.

2.6 Identification of the functional users.

GUIDANCE on Rule 7: Identification of the functional users.

The identification of functional user (or users) is determined by the Functional 'User' Requirements that must be measured and by the purpose of the measurement.

2.7 Levels of decomposition.

Size measurements of the components of a piece of software are only directly comparable for components at the same level of decomposition. This is important because sizes of pieces of software at different levels of decomposition cannot be simply added up without taking into account the aggregation rules of chapter 5. Further, as a consequence, the performance (e.g. the productivity = size/effort) of projects to develop different pieces of software can only be compared if all the pieces of software are at the same level of decomposition.

Different levels of decomposition of a piece of software may correspond to different 'views' of the software's layers, e.g. as in Figure 2.4 in Part 3. However, software may be decomposed into 'levels' regardless of whether or not it is designed using a layered-architecture model.

2.8 Context diagrams.

It can be helpful when defining a scope of FSM and the functional users to draw a 'context diagram' for the software being measured. Context diagrams show the flows of data between the piece of software and its functional users (humans, hardware devices or other software) and also show the flows of data between the piece of software and persistent storage.

A context diagram is an instance of a measurement pattern applied to the software being measured. Symbols used in context diagrams are presented in Figure 2.5.

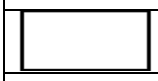
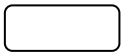
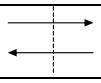
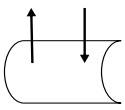
Symbol	Interpretation
	The piece of software to be measured (box with thick outline) i.e. the definition of a measurement scope.
	Any functional user of the software being measured.
	The arrows represent <u>all</u> the movements of data crossing a boundary (the dotted line) between a functional user and the software being measured.
	The arrows represent <u>all</u> the movements of data between the software being measured and 'persistent storage'. (The flowchart symbol for 'data storage' emphasizes that persistent storage is an abstract concept. This symbol indicates that the software does not interact directly with physical hardware storage.)

Figure 2.5 - Symbols of context diagrams.

2.9 Identification of the level of granularity.

COSMIC requires the FUR to be expressed at a level of detail sufficient to create the COSMIC measurement models: this is called the level of granularity.

To derive a functional size using the COSMIC FSM using the RULES of ISO 19761, the necessary level of granularity is that at which individual functional processes and their data movements can be identified and defined. When functional details are missing at other levels of granularity of the requirements, measurements should be made using one of the size approximation techniques described in the related 'Early Software Sizing with COSMIC: Practitioners Guide'.

NOTE 1 In the initial stages of a software development project, actual requirements are specified 'at a high level', that is, in outline, or in little detail. As the project progresses, the actual requirements are refined, (e.g. through versions 1, 2, 3 etc.), revealing more and more detail 'at lower levels'. These different degrees of detail of the actual requirements are known as different 'levels of granularity'.

NOTE 2: Measurers should be aware that when requirements are evolving early in the life of a software project, at any moment different parts of the required software functionality will typically have been documented at different levels of granularity.

For an example of measuring at varying levels of granularity and of decomposition, see the telecoms system example in the 'Guideline for early or rapid COSMIC functional size measurement using approximation approaches'

3. THE MAPPING PHASE.

Functional processes are composed of sub-processes that move data (*'data movements'*) and optionally, may manipulate data (*'data manipulation').*

3.1 Mapping the FUR to the Generic Software Model.

Figure 3.1 shows the steps for mapping the FUR in the available software artefacts to the form required by the COSMIC Generic Software Model.

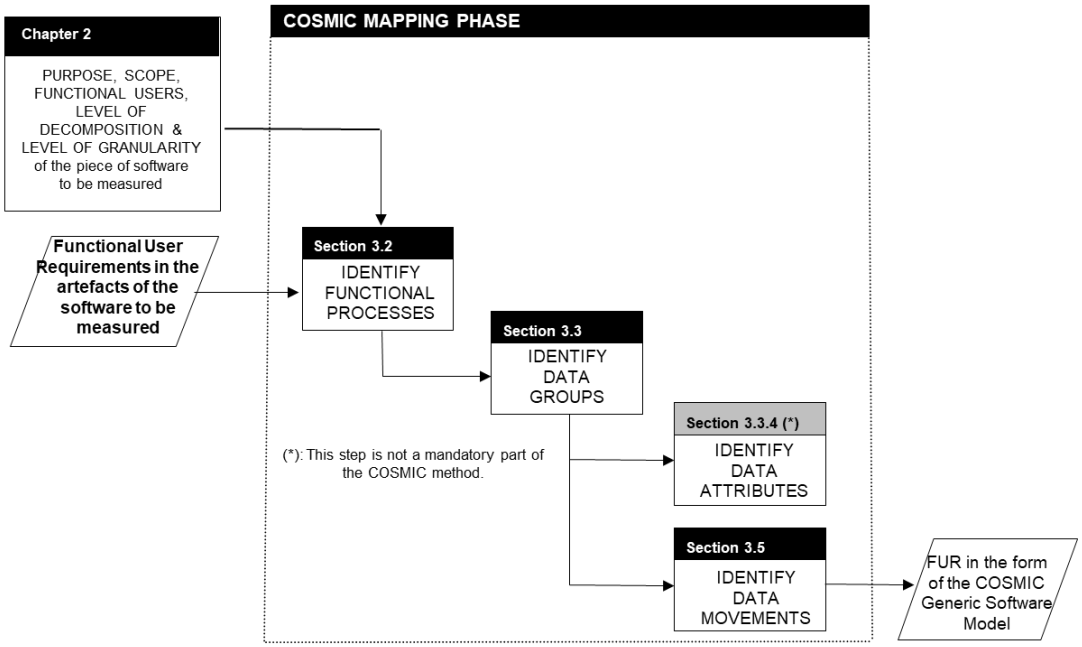


Figure 3.1 – General method of the COSMIC mapping process.

Several guidelines are available that describe how to map from various data-analysis and requirements-determination methods, used in different domains to the concepts of the COSMIC method:

- Guideline for Sizing Business Application Software,
- Guideline for Sizing Data Warehouse Application Software,
- Guideline for Sizing Service-Oriented Architecture Software, and
- Guideline for Sizing Real-time Software.

For the business and real-time domains there are also Quick Reference Guides available that give an overview of the process in a few pages.

3.2 Identifying functional processes.

The first step of the Measurement Phase is to identify the set of functional processes of the piece of software to be measured, from its FUR.

The relationships between a triggering event, the functional user and the Entry data movement that triggers a functional process being measured are presented in Figure 3.2 where: *a triggering event causes a functional user to generate a data group that is moved by the triggering Entry of a functional process to start the functional process.*

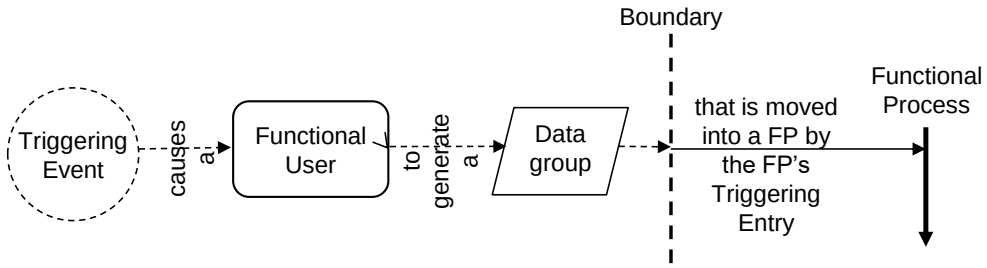


Figure 3.2 – Relationships between a triggering event, a functional user & a functional process.

NOTE 1: For ease of reading, the reference to the data group is omitted when stating that a functional user *initiates* a triggering Entry that starts a functional process, or even more simply that a functional user *initiates* a functional process.

NOTE 2: All the relationships between the concepts in Figure 3.2 (the triggering event / functional user / data group / triggering Entry / functional process) may be one-to-many, many-to-one, or many-to-many, with one exception. The exception is that the data group moved by any one triggering Entry may initiate only the functional process it is part of.

Guidance on Rule 10: Identification of Functional Processes

The process of identifying functional processes, after the functional users have been identified, given the FUR for the software being measured follows the chain of Figure 3.2:

1. Identify the separate events in the world of the functional users that the software being measured must respond to – the 'triggering events'

   NOTE: Triggering events can be identified in state diagrams and in entity life-cycle diagrams, since some state transitions and entity life-cycle transitions correspond to triggering events to which the software must react.

2. Identify which functional user(s) of the software may respond to each triggering event;
3. Identify the data group(s) (i.e. the triggering Entry or Entries) that each functional user may initiate in response to the event;
4. Identify the functional process started by each triggering Entry.

Use the following checks to ensure that candidate functional processes (FP) have been properly identified:

1. Do all the identified FPs of the piece of software measured reside in the same layer?

2. Are all the identified FPs comprised of an Entry and at least 1 Write or Exit data movement?

3.3 Identification of data groups.

3.3.1 Identification of data groups.

Having identified the functional processes, the next step is to identify their data movements. The following guidance assists in the identification of data groups and hence objects of interest particularly in the output of functional processes.

GUIDANCE on Rule 11: Identifying different data groups moved in the same one functional process.

For all the data attributes appearing in an Entry-Exit-Read-Write of a functional process:

a) sets of data attributes that have different frequencies of occurrence describe different objects of interest;

c) sets of data attributes that have the same frequency of occurrence but different identifying key attribute(s) describe different objects of interest;all the data attributes in a set resulting from applying parts a) and b) of this guidance belong to the same one data group, unless the FUR specify that there may be more than one data group describing the same object of interest (see the Guidance on Rules **13 and 14 – Data Movements Uniqueness**, cases b) and c)).

NOTE 1: A functional user of the software being measured may be the object of interest of a data group sent or received by the functional user.

NOTE 2: In theory, a data group might contain only one data attribute if this is all that is required, from the perspective of the FURs, to describe the object of interest. In practice, such cases occur commonly in real-time application software (e.g. the data group entered to convey the tick of a real-time clock or the entry of the state of a sensor); they are less common in business application software.

The origin of a data group can be of many forms, e.g.:

a) A physical record structure on a hardware storage device (file, database table, ROM memory, etc.).

b) A physical structure within the computer's volatile memory (data structure allocated dynamically or through a pre-allocated block of memory space).

c) A clustered presentation of functionally-related data attributes on an input/output device (display screen, printed report, control panel display, etc.).

d) A message in transmission between a device and a computer, or over a network, etc.

3.3.2 About the identification of objects of interest and data groups.

The definition and principles of objects of interest and of data groups are intentionally broad in order to be applicable to the widest possible range of software: this sometimes results in it being difficult to apply the definition and principles when measuring a specific piece of software. See Part 3 for examples to assist in the application of the principles to specific cases

When faced with a need to analyze a group of data attributes that is moved in or out of a functional process or is moved by a functional process to or from persistent storage, *it is*

critically important to decide if the attributes all convey data about a single 'object of interest', since it is the latter that determine the number of separate 'data groups' as defined by the COSMIC method that will be moved by data movements.

For instance, if the data attributes to be input to a functional process are attributes of three separate objects of interest, then three separate 'Entry' data movements must be identified. Deciding on the number of data groups can be difficult when analyzing the output of a functional process of a business application which may include:

- multiple data groups, each describing a different object of interest, e.g. a report showing totals at various levels of aggregation;
- the results of enquiries where the output will vary depending on the input;
- data groups that may even be unrelated to each other, e.g. an invoice which includes an advertisement for an unrelated service.

When analyzing complex output, e.g. reports with data describing several objects of interest, consider each separate candidate data group as if it were output by one separate functional process. Each of the data group types identified this way must also be distinguished and counted when measuring the complex report.

3.3.3 Data or groups of data that are not candidates for data groups.

Any data appearing on input or output screens or reports that are not related to an object of interest to a functional user should not be identified as indicating a data group.

The COSMIC Generic Software Model assumes that all manipulation of data within a functional process is associated with the four data movement types. Hence no data groups may be identified arising from data manipulation within a functional process in addition to the data groups moved by the Entries, Exits, Reads and Writes of the functional process.

3.3.4 Identification of data attributes (optional).

In the COSMIC method, it is not mandatory to identify the data attributes. However, understanding the concept of a 'data attribute' is necessary to understand how to measure changes'. A requirement to change a data attribute can result in the data movement to which the attribute belongs being indicated as 'changed'.

Also, it may be helpful to analyze and identify data attributes in the process of distinguishing data groups and objects of interest.

3.4 Identification of data movements.

This step consists in identifying the data movements (Entry, Exit, Read and Write) of each functional process.

GUIDANCE on Rule 12: Data Movements

In general, the 'type' of a thing is an abstract class of all the things that share some common characteristic, so that the things are subject to the same FUR. (Synonyms of 'type': 'category', 'kind').

An 'occurrence' of a thing is when it becomes real by assigning values to its attributes. Known in the Object-Oriented world as 'instantiation' – the creation of an instance.

The following guidance helps to confirm the status of a candidate Entry data movement:

GUIDANCE on Rule 16: Entry (E).

a) The data group of a triggering Entry may consist of only one data attribute which simply informs the software that 'an event Y has occurred'.

b) For clock-ticks that are triggering events identify an Entry from a functional user, in this case the Clock.

c) Unless a specific functional process is necessary, obtaining the date and/or time from the system's clock is not considered an Entry or any other COSMIC data movement.

NOTE: Very often, especially in business application software, the data group of the triggering Entry has several data attributes which inform the software that 'an event Y has occurred and here is the data about that particular event'.

GUIDANCE on Rule 17: Exit (X).

a) For an enquiry which outputs fixed text, (where 'fixed' means the message contains no variable data values), identify an Exist for the fixed text output.

b) When identifying Exits, ignore all fields and other headings that enable human users to understand the output data.

GUIDANCE on Rule 18: Read (R).

Do not identify a Read when the FUR of the software being measured specify any software or hardware functional user as the source of a data group, or as the means of retrieving, a persistently-stored data group.

GUIDANCE on Rule 19: Write (W).

Do not identify a Write when the FUR of the software being measured specify any software or hardware functional user as the destination of the data group or as the means of storing the data group.

NOTE: When the FUR require data to be stored or to be retrieved from storage, the measurer must investigate whether the data can be stored or retrieved within its own boundary, i.e. to/from 'persistent storage', or whether data is required to be stored/retrieved with help of a functional user of the software being measured (i.e. via some other piece of software, or directly to or from a hardware device).

GUIDANCE on Rules 16 to 19 – Data manipulation associated with data movements.

The data manipulation associated with any of these data movements does not include any data manipulation that is needed after the data movement has been successfully completed, nor does it include any data manipulation associated with any other data movement.

The following guidance cover the most common situation (guidance a)) and other possible valid cases (guidance b) and c)):

- In a) the occurrences of the data group are subject to the same FUR: so one data group and one data movement is identified.
- In b) and c) the same applies to *each different data group separately:* so one data group and one data movement per different data group is identified.

GUIDANCE on Rules 13 and 14 – Data Movements Uniqueness.

a) Unless the FUR are as given in guidance b) or c), all data describing any one object of interest that is required to be entered into one functional process is identified as one data group moved by one Entry.

NOTE 1: A functional process may, of course, have multiple Entries, each moving data describing a different object of interest.
NOTE 2: The same equivalent guidance applies to any Read, Write or Exit data movement in any one functional process.

b) If FUR specify that different data groups must be entered into one functional process, each from a different functional user, where each data group describes the same object of interest, then one Entry is identified for each of these different data groups.

NOTE 1: The same equivalent guidance applies for Exits of data to different functional users from any one functional process.
NOTE 2: Any one functional process has only one triggering Entry.

c) If FUR specify that different data groups must be moved from persistent storage into one functional process, each describing the same object of interest, then one Read is identified for each of these different data groups.

NOTE 1: The same equivalent guidance applies for Writes in any given functional process.
NOTE 2: This guidance is analogous to rule b). In the case of the FUR to read different data groups describing the same object of interest, they will likely have originated from different functional users. In the case of the FUR to write different data groups, they will likely be made available to be read by different functional users.

GUIDANCE on Rules 16-17: Functional process requiring data from a functional user.

a) When the functional process does not need to tell the functional user what data to send, a single Entry is sufficient (per object of interest).

b) When the functional process needs to tell the functional user what data to send, an Exit followed by an Entry are necessary.

GUIDANCE on Rules 16-19: Control commands in applications with a human interface.

In an application with a human interface 'control commands' are ignored as they do not involve any movement of data about an object of interest.

Error and confirmation messages are specific forms of an Exit and the Rules governing identification apply.

GUIDANCE on Rules 16-19: Error/confirmation messages & other indications of error conditions.

a) One Exit is identified to account for all types of error/confirmation messages issued by any one functional process of the software being measured from all possible causes according to its FUR.

b) If a message to a human functional user provides data in addition to confirming that entered data has been accepted, or that entered data is in error, then this additional data is identified as a separate data group moved by an Exit in the normal way.

c) All other data, issued or received by the software being measured, to/from its hardware or software functional users should be analyzed according to the FUR as Exits or Entries respectively, according to the normal COSMIC rules, regardless of whether or not the data values indicate an error condition.

d) Reads and Writes are considered to account for any associated reporting of error conditions. Therefore no Entry to the functional process being measured is identified for any error indication received as a result of a Read or Write of persistent data.

e) No Entry or Exit is identified for any message indicating an error condition that might be issued whilst using the software being measured but which is not required to be processed in any way by the FUR of that software, e.g. an error message issued by the operating system.

3.5 Measuring the components of a distributed software system.

When the purpose of a measurement is to measure separately the size of each component of a distributed software system, a separate scope of FSM must be defined for each component. In such a case the sizing of the functional processes of each component follows all the rules as already described.

From the process for each measurement (… define the scope, then the functional users and boundary, etc. …) it follows that if a piece of software consists of two or more components, there cannot be any overlap between the scope of FSM of each component. The scope of FSM for each component must define a set of complete functional processes. For example, there cannot be a functional process with part in one scope and part in another. Likewise, the functional processes within the measurement scope for one component do not have any information about the functional processes within the scope of another component, even though the two components exchange messages.

The functional user(s) of each component is/are determined by examining where the events occur that trigger functional processes in the component being examined. (Triggering events can only occur in the world of a functional user.)

3.6 Re-use of software.

Any two or more functional processes in the same software being measured may have some functionality that is identical or very similar in each process and is described separately elsewhere in the requirements. This phenomenon is referred to as 'functional commonality', or functional 'similarity'.

However, each functional process is defined, modelled and measured independently of, i.e. without reference to any other FUR in the same software being measured.

Therefore, if the FUR for a given functional process makes a reference to FUR elsewhere in the requirements then the size of that referenced functionality is to be included in the size of the functional process being measured.

3.7 Measurement of the size of changes to software.

A 'functional change' to existing software is interpreted in the COSMIC method as 'any combination of additions of new data movements or of modifications or deletions of existing data movements, including to the associated data manipulation'. The terms 'enhancement' and 'maintenance'[1] are often used for what we here call a 'functional change'.

The need for a change to software may arise from either:

- a new FUR (i.e. only additions to the existing functionality), or
- from a change to the FUR (perhaps involving additions, modifications and deletions) or
- from a 'maintenance' need to correct a defect.

The rules for sizing any of these changes are the same but the measurer is alerted to distinguish the various circumstances when making performance measurements and estimates.

A data movement is considered to be functionally modified as follows.

GUIDANCE on Rules 21 to 24: – Modifying a data movement.

a) If a data movement must be modified due to a change of the data manipulation associated with the data movement and/or due to a change in the number or type of the attributes in the data group moved, one changed CFP is measured, regardless of the actual number of modifications in the one data movement.

b) If a data group must be modified, data movements moving the modified data group whose functionality is not affected by the modification to the data group is not identified as changed data movements.

NOTE: A modification to any data appearing on input or output screens that are not related to an object of interest to a functional user is not identified as a changed CFP.

c) A normal measurement convention is that the functional size of a piece of software does not change if the software must be changed to correct a defect so as to bring the software in line with its FUR. The functional size of the software does change if the change is to correct a defect in the FUR.

Modified data movements have no influence on the size of the piece of software as they exist both before and after the modifications have been made.

When a piece of software is completely replaced, for instance by re-writing it, with or without extending and/or omitting functionality, the functional size of this change is the size of the replacement software, measured according to the normal rules for sizing new software.

NOTE Usually, the size of a functional change (discussed here) and the change in the functional size of the software differ.

4. OTHER TOPICS.

4.1 Extending the COSMIC measurement method – Local extension.

The COSMIC method of measuring a functional size does not presume to measure all possible aspects of software 'size'. Thus the method is currently not designed to measure separately and explicitly the size of the FUR of data manipulation sub-processes. The influence on size of data manipulation sub-processes is taken into account via a simplifying assumption that is valid for a wide range of software domains.

Nevertheless, the COSMIC size measure is considered to be a good approximation for the method's stated purpose and domains of applicability. Yet, it may be that within the local environment of an organization using the COSMIC measurement method, it is desired to account for such functionality in a way which is meaningful as a local standard. When such local extensions are used, the measurement results must be reported according to the special convention presented in section 6.

4.2 COSMIC in Agile.

The COSMIC method is being used successfully to measure the size of User Stories in Agile software developments, which may have very few data movements. This practice is well-established, with many reports of the CFP sizes of Agile iterations (or 'sprints'), aggregated from the sizes of individual User Stories, that correlate very well with the effort to develop the iteration (and much better than the correlation with effort of sizes measured using Story Points). See (https://cosmic-sizing.org/publications/guideline-for-sizing-agile-projects-with-cosmic/) .

5. MEASUREMENT PHASE.

GUIDANCE on Rule 23: Aggregation of functional sizes.

a) Sizes of pieces of software or of changes to pieces of software may be added together only if measured at the same functional process level of granularity of their FUR.

b) Sizes of pieces of software and/or changes in the sizes of pieces of software within any one layer or from different layers are added together only if it makes sense to do so, for the purpose of the measurement.

c) The size of a piece of software is obtained by adding up the sizes of its components (regardless of how the piece is decomposed) and eliminating the size contributions of inter-component data movements.

Within each identified layer, the aggregation function is fully scalable. A sub-total can be generated for individual functional processes or for all the functional process of the software, depending on the purpose and scope of the measurement exercise.

6. MEASUREMENT REPORTING.

The result must be reported and data about the measurement recorded so as to ensure that the result is always unambiguously interpretable. COSMIC measurement results are to be reported and archived according to the following conventions.

GUIDANCE on Rule 25: COSMIC measurement labeling.

A COSMIC measurement result is noted as '***x*** CFP (v) ', where:

- '**x**' represents the numerical value of the functional size,
- 'v' represents the identification of the version of the standard COSMIC method used to obtain the numerical functional size value 'x'.

NOTE: If a local approximation method was used to obtain the measurement, but otherwise the measurement was made using the conventions of a standard COSMIC version, the above labeling convention is used, but use of the approximation method should be noted elsewhere.

GUIDANCE on Rule 25: COSMIC local extensions labeling.

A COSMIC measurement result using local extensions is noted as:

' ***x*** CFP (v.) + z Local FP', where:

- '**x**' represents the numerical value obtained by aggregating all individual measurement results according to the standard COSMIC method, version v,
- 'v' represents the identification of the version of the standard COSMIC method used to obtain the numerical functional size value 'x'.
- 'z' represents the numerical value obtained by aggregating all individual measurement results obtained from local extensions to the COSMIC method.