INTERNATIONAL STANDARD

ISO/IEC 19761

Second edition
2011-03-15

Software engineering — COSMIC: a functional size measurement method

Ingénierie du logiciel — COSMIC: une méthode fonctionnelle de mesure de taille

Reference number
ISO/IEC 19761:2011(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19761 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

This second edition cancels and replaces the first edition (ISO/IEC 19761:2003), which has been technically revised.

Introduction

Software is a major component of many corporate budgets. Organizations recognize the importance of controlling software expenses and analysing the performance of the budgets allocated to software development and maintenance in order to benchmark against the best in the field. To do so, measures and models using these measures are needed.

Measures are needed for analysing both the quality and the productivity associated with developing and maintaining software. On the one hand, technical measures are needed to quantify the technical performance of products or services from a developer's viewpoint. Technical measures can be used for efficiency analysis; to improve the performance of designs, for instance.

On the other hand, functional measures are needed to quantify the performance of products or services from a user's or owner's perspective; for productivity analysis, for instance. Functional measures must be independent of technical development and implementation decisions. They can then be used to compare the productivity of different techniques and technologies.

The Full Function Points (FFP) method was proposed in 1997 with the aim of offering a functional size measure specifically adapted to real-time software. Field test results, coupled with the feedback received from organizations which used it, motivated the authors to improve the method. Many improvements were also inspired by the work of the Common Software Measurement International Consortium (COSMIC). The results of these efforts were published in May 2001 as version 2.1 of the COSMIC-FFP Functional Size Measurement Method (as it was first named), aiming to be applicable to business application, real-time and system software [4].

ISO/IEC 19761:2003 was based on this version 2.1 of the COSMIC-FFP Functional Size Measurement method.

Extensive experience of using the method convinced the Common Software Measurement International Consortium of the need for various clarifications and improvements to the description of the method. Version 3.0 of the method was therefore published in December 2007 [5]. These various changes have not altered the underlying model for the measurement of software functional size since it was first published in 2001.

The International Standard aims to meet the needs of

a) software suppliers facing the task of translating customer requirements into the functional size of software to be produced as a key activity in their project cost estimating,

b) customers who want to know the functional size of delivered software as an important component of measuring supplier performance.

With version 3.0 of the method, the name of the method was simplified from “COSMIC-FFP” to “COSMIC”. The name of the unit of measure has also been simplified from “Cfsu” (COSMIC functional size unit) to “CFP” (COSMIC Function Point).

Software engineering — COSMIC: a functional size measurement method

1 Scope

This International Standard specifies the set of definitions, conventions and activities of the COSMIC Functional Size Measurement Method. It is applicable to software from the following functional domains:

a) application software;

EXAMPLE Banking, insurance, accounting, personnel, purchasing, distribution or manufacturing.

b) real-time software;

EXAMPLE Software for telephone exchanges and message switching, software embedded in devices to control machines such as domestic appliances, lifts and car engines, for process control and automatic data acquisition, and within the operating system of computers.

c) hybrids of the above.

EXAMPLE Real-time reservation systems for airlines or hotels.

This International Standard has not been designed for measuring the functional size of a piece of software, or its parts, which

— is characterized by complex mathematical algorithms or other specialized and complex rules, such as can be found in expert systems, simulation software, self-learning software and weather forecasting systems, or

— processes continuous variables such as audio sounds or video images, such as can be found in computer game software, musical instruments and the like.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1
Base Functional Component
BFC
elementary unit of Functional User Requirements defined by and used by an FSM Method for measurement purposes

[ISO/IEC 14143-1:2007, definition 3.1]

NOTE The COSMIC Functional Size Measurement Method defines a data movement as a BFC.

2.2
Base Functional Component type
BFC type
defined category of Base Functional Component

[ISO/IEC 14143-1:2007, definition 3.2]

2.3
boundary
conceptual interface between the software being measured and its functional users

NOTE The COSMIC Functional Size Measurement Method uses the term “functional user”, which has a narrower definition than the term “user” as defined in ISO/IEC 14143-1:2007, definition 3.3. In consequence, this International Standard uses “functional user”, rather than “user”.

2.4
data attribute
smallest parcel of information, within an identified data group, carrying a meaning from the perspective of the software's Functional User Requirements

2.5
data group
data group type
distinct, non empty, non ordered and non redundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest (see 2.19)

2.6
data manipulation
any processing of the data other than a movement of the data into or out of a functional process, or between a functional process and persistent storage

2.7
data movement
data movement type
Base Functional Component which moves a single data group

NOTE 1 The COSMIC Functional Size Measurement Method has four types of data movements: Entry, Exit, Read and Write. These are the Method's four BFC types.

NOTE 2 For measurement purposes, each data movement is considered to account for certain associated data manipulation.

2.8
Entry
Entry type
data movement that moves a data group from a functional user across the boundary into the functional process where it is required

NOTE An Entry is considered to account for certain associated data manipulations (e.g. validation of the entered data).

2.9
Exit
Exit type
data movement that moves a data group from a functional process across the boundary to the functional user that requires it

NOTE An Exit is considered to account for certain associated data manipulations (e.g. formatting and routing associated with the data to be exited).

2.10
functional process
functional process type
elementary component of a set of Functional User Requirements, comprising a unique, cohesive and independently executable set of data movements

NOTE 1 It is triggered by a data movement (an Entry) from a functional user that informs the piece of software that the functional user has identified a triggering event, and is complete when it has executed all that is required to be done in response to the triggering event.

NOTE 2 In addition to informing the piece of software that the event has occurred, the Entry triggered by the event can include data about an object of interest associated with the event.

2.11
Functional Size Measurement
FSM
process of measuring Functional Size

[ISO/IEC 14143-1:2007, definition 3.7]

2.12
Functional Size Measurement Method
specific implementation of FSM defined by a set of rules, which conforms to the mandatory features of ISO/IEC 14143-1:2007

[ISO/IEC 14143-1:2007, definition 3.4]

2.13
functional user
user that is a sender and/or an intended recipient of data in the Functional User Requirements of a piece of software

2.14
Functional User Requirements
FUR
sub-set of the user requirements describing what the software does in terms of tasks and services

NOTE Functional User Requirements include, but are not limited to,

— data transfer (for example Input customer data, Send control signal),

— data transformation (for example Calculate bank interest, Derive average temperature),

— data storage (for example Store customer order, Record ambient temperature over time), and

— data retrieval (for example List current employees, Retrieve aircraft position).

User requirements that are not Functional User Requirements include, but are not limited to,

— quality constraints (for example usability, reliability, efficiency and portability),

— organizational constraints (for example locations for operation, target hardware and compliance to standards),

— environmental constraints (for example interoperability, security, privacy and safety), and

— implementation constraints (for example development language, delivery schedule).

[ISO/IEC 14143-1:2007, definition 3.8]

2.15
layer
partition resulting from the functional division of a software system, where

— layers are organized in a hierarchy,

— there is only one layer at each level in the hierarchy,

— there is a “superior/subordinate” hierarchical dependency between the functional services provided by software in any two layers in the software system that exchange data directly, and

— the software in any two layers in the software system that exchange data interpret only part of that data identically

2.16
measurement method
logical sequence of operations, described generically, used in the performance of measurements

[ISO Guide 99:1993]

2.17
measurement procedure
set of operations, described specifically, used in the performance of particular measurements according to a given method

[ISO Guide 99:1993]

2.18
measurement process
process of establishing, planning, performing and evaluating software measurement within an overall project or organizational measurement structure

NOTE Adapted from ISO/IEC 15939:2007, definition 2.24.

2.19
object of interest
object of interest type
any thing that is identified from the point of view of the Functional User Requirements about which the software is required to process and/or store data

NOTE 1 An object of interest can be any physical thing, as well as any conceptual object or part of a conceptual object in the world of the functional user.

NOTE 2 The term object of interest is used in order to avoid terms related to specific software engineering methods. The term does not imply objects in the sense used in Object Oriented methods. Similarly the word Entity is avoided because of its use in Data Modelling.

2.20
operating environment
operating environment software
set of software operating concurrently on a specified computer system

2.21
peer software
piece of software that resides in the same layer as, and exchanges data with, another piece of software

2.22
persistent storage
storage which enables a functional process to store data beyond the life of the functional process and/or which enables a functional process to retrieve data stored by another functional process, or stored by an earlier occurrence of the same functional process, or stored by some other process

NOTE 1 As persistent storage is on the software side of the boundary, it is not considered to be a functional user of the software being measured.

NOTE 2 An example of “some other process” would be in the manufacture of a read-only memory.

2.23
Read
Read type
data movement that moves a data group from persistent storage within reach of the functional process which requires it

NOTE A Read is considered to account for certain associated data manipulations necessary to achieve the Read.

2.24
scope
scope of the FSM
set of Functional User Requirements to be included in a specific functional size measurement instance

2.25
triggering event
triggering event type
event (something that happens) that causes a functional user of the piece of software to initiate ("trigger") one or more functional processes

NOTE In a set of Functional User Requirements, each event which causes a functional user to trigger a functional process

— cannot be sub-divided for that set of FUR,

— has either happened or it has not happened.

2.26
unit of measurement
particular quantity, defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitudes relative to that quantity

NOTE Units of measurement have conventionally assigned names and symbols.

[ISO Guide 99:1993]

2.27
user
any person or thing that communicates or interacts with the software at any time

NOTE Examples of "thing" include, but are not limited to, software applications, animals, sensors, or other hardware.

[ISO/IEC 14143-1:2007, definition 3.11]

2.28
Write
Write type
data movement that moves a data group lying inside the functional process to persistent storage

NOTE A Write is considered to account for certain associated data manipulations necessary to achieve the Write.

3 Abbreviated terms

BFC	Base Functional Component
CFP	COSMIC Function Point
FSM	Functional Size Measurement
FUR	Functional User Requirements

4 Unit of measurement

The COSMIC unit of measurement is denoted by the symbol CFP (Cosmic Function Point).

5 Measurement activities

5.1 General

The determination of the COSMIC functional size shall involve all of the activities described in Clause 5. Once the purpose of the FSM has been determined, the process of determining the scope(s) of the FSM, the functional users (as in 5.5), the layers (as in 5.4) and the boundaries (as in 5.6) may require some iteration.

5.2 Determination of the purpose and scope of the FSM

The purpose and the scope of the FSM shall be determined before commencing a particular measurement exercise.

NOTE It is outside the scope of this International Standard to describe or propose to the measurement analyst specific purposes for performing measurement. Based on the purpose of the measurement, it is up to the measurement analyst to determine the most appropriate artifacts to use to perform a specific measurement.

EXAMPLE 1 For one piece of software there might be a purpose to measure the functional size as seen by the (human) functional users of the software, and a separate purpose to measure the size of components which the software development team has to deliver. These two purposes will usually result in different functional sizes. The human functional user may see only the functionality supplied by the application layer. In contrast, the development team may be required to develop and/or amend software in lower layers of the software system in addition to developing the application layer, in order to meet the FUR and other (non-functional) requirements of the application layer.

EXAMPLE 2 If the purpose is to measure the functional size of software delivered by a particular project team, it will first be necessary to establish the separate scopes of the various components delivered. These might include software which was used once only to convert data from software which is being replaced. If then the purpose is changed to measure the size which the functional users have available once the new software is operational, the size measured would be smaller, as the software used for conversion would not be included in the scope of the measured size.

5.3 Identification of the FUR

The FUR identified to be within the scope of the FSM shall be used as the exclusive source from which the functional size of the software is to be measured.

NOTE Informative Annex A contains guidance about how to extract FUR from various sources.

5.4 Identification of the layers

5.4.1 The scope of the FSM and layers

Software may have components of its functionality that exist in different layers of the software's operating environment. If required for the purpose of the measurement exercise, each such layer shall be identified.

A single piece of software to be measured shall not have its scope defined to extend over more than one layer.

NOTE 1 FUR may state explicitly, may imply, or the measurement analyst may infer, that the FUR apply to software in different layers or to different peer items whose size must be measured separately. Alternatively, the measurement analyst may be faced with sizing existing software which appears to be in different layers or to consist of separate peer items. In both cases, guidance is needed (see 5.4.2) to help decide if the FUR of the software comprise one or more layers or peer items.

NOTE 2 Layer identification is an iterative activity. The exact identification of layers will be refined as the measurement activity progresses.

NOTE 3 Software within a layer may be developed and need to be sized as separate ‘pieces’.

NOTE 4 Data movements between separate pieces of software within the same layer are known as ‘peer-to-peer’ communication.

5.4.2 Characteristics of layers

Layers identified within the scope of the FSM shall have the following characteristics.

a) Software in each layer shall deliver functionality to its functional users.

b) Software in a subordinate layer shall provide functional services to software in a layer using its services.

c) Software that shares data with other software shall not be considered to be in different layers if they identically interpret the data attributes that they share.

NOTE 1 Software in a subordinate layer could perform without assistance from software in a layer using its services.

NOTE 2 Software in one layer might not perform properly if software in a subordinate layer on which it depends is not performing properly.

NOTE 3 Software in one layer does not necessarily use all the functionality supplied by software in a subordinate layer.

NOTE 4 In a hierarchy of layers, software in any one layer can be a subordinate to software in a higher layer for which it provides services.

NOTE 5 There are many software system models in use. The layered model is used here to provide a functional view of the software. Other models could be used if they provide a functional view of the software, fully or partly.

NOTE 6 The concept of layers presented in this International Standard is different from the concept of “layered architecture”. Although there are elements common to both concepts, the COSMIC layers are meant as a tool to help the practitioner in the identification of the scope and the boundaries. If a specific software system paradigm is used within an organization, then it may be necessary to establish an equivalence between specific software system elements in that paradigm and the concept of layers as defined in this International Standard, and the resulting layers should be used.

NOTE 7 Functional service packaged software, such as database management systems, operating systems or device drivers, are generally considered as residing in distinct layers.

5.5 Identification of the functional users

All functional users that trigger functional processes in the FUR of the software within the scope of the FSM shall be identified.

5.6 Identification of software boundaries

The requirements for identifying boundaries are as follows.

a) The boundary of each piece of software within each layer and in the scope of the FSM shall be identified.

b) Once the boundaries have been identified, each FUR within the scope of the FSM shall be allocated to a piece of software.

NOTE 1 When identifying a boundary, the following guidelines may be helpful:

— Start by identifying the functional users, then the triggering events that these functional users identify, then identify the functional processes triggered by those events. The boundary lies between the functional users and the functional processes;

— By definition, there is a boundary between each identified pair of layers where the software in one layer is the functional user of software in another, and the latter is to be measured. Similarly, there is a boundary between any two peer pieces of software in the same layer; in this case each piece of software is a functional user of its peer

NOTE 2 Boundary identification is an iterative activity. The exact boundary of any piece of software being measured will be refined as the measurement activity progresses.

5.7 Identification of functional processes

Each functional process identified in the scope of the FSM shall:

a) be derived from at least one identifiable FUR,

b) be initiated by an Entry data movement from a functional user informing the functional process that it has detected a triggering event,

c) comprise at least two data movements, namely always one Entry plus either an Exit or a Write,

NOTE 1 According to this characteristic and using 1 CFP as the unit of measurement, the smallest theoretical functional size for a functional process is 2 CFP.

d) belong to one, and only one layer,

e) be complete when a point of asynchronous timing is required to be reached according to its FUR.

NOTE 2 A point of asynchronous timing is reached when the final (terminating) data movement in a sequence of data movements is not synchronized with any other data movement.

5.8 Identification of data groups

Each data group identified in the scope of the FSM shall:

a) be unique and distinguishable through its unique collection of data attributes,

b) be directly related to one object of interest described in the software's FUR.

NOTE Constants or variables which are internal to the functional process, or intermediate results in a calculation, or data stored by a functional process resulting only from the implementation, rather than the FUR, are not data groups.

5.9 Identification of data movements

Each functional process identified in 5.7 shall be partitioned into its component data movements.

For any one functional process, a single Entry data movement shall be identified and counted for the entry of all data describing a single object of interest that the FUR require to be entered, unless the FUR explicitly require data describing the same single object of interest to be entered more than once in the same functional process.

Similarly, a single Exit, Read or Write data movement shall be identified and counted for the movement of all data describing a single object of interest that the FUR requires of that type (i.e. Exit, Read or Write, respectively), unless the FUR explicitly require data describing the same single object of interest to be moved more than once in the same functional process by a data movement of the same type (i.e. Exit, Read or Write, respectively).

If a data movement of a particular type (Entry, Exit, Read or Write) occurs multiple times with different data values when a functional process is executed, e.g. as in a loop, only one data movement of that type shall be identified and counted in that functional process.

5.10 Classification of data movements

5.10.1 Entry

An Entry shall:

a) receive data attributes from a single data group which originates from the functional user side of the boundary,

b) account for all required formatting and presentation manipulations along with all associated validations of the entered data attributes, to the extent that these manipulations do not involve another type of data movement,

NOTE An Entry accounts for all manipulations that might be required to validate some entered codes or to obtain some associated descriptions. However, if one or more Reads are required as part of the validation process, these are identified and counted as separate Read data movements.

c) include any ‘request to receive the Entry’ functionality, where it is unnecessary to specify what data should be entered.

5.10.2 Exit

An Exit shall:

a) send data attributes from a single data group to the functional user side of the boundary,

b) account for all required data formatting and presentation manipulations, including processing required to send the data attributes to the functional user, to the extent that these manipulations do not involve another type of data movement.

5.10.3 Read

A Read shall:

a) retrieve data attributes from a single data group from persistent storage,

b) account for all logical processing and/or mathematical computation needed to read the data, to the extent that these manipulations do not involve another type of data movement,

c) include any ‘request to Read’ functionality.

5.10.4 Write

A Write shall:

a) move data attributes from a single data group to persistent storage,

b) account for all logical processing and/or mathematical computation to create the data attributes to be written, to the extent that these manipulations do not involve another type of data movement.

A requirement to delete a data group from persistent storage shall be a single Write data movement.

5.11 Calculation of the functional size

5.11.1 Assignment of a unit size to a data movement

A unit of measurement, 1 CFP, shall be assigned to each data movement (Entry, Exit, Read or Write) identified in each functional process.

5.11.2 Aggregation of functional size

The results of 5.11.1, as applied to all identified data movements within the identified functional process, shall be aggregated into a single functional size value for that functional process by:

a) multiplying the number of data movements of each type by its unit size,

b) totalling the sizes from step a) for each of the data movement types in the functional process.

Thus, the Functional Size FS of a given functional process is calculated in CFP using the formula:

$$FS = (Ne \times Eus) + (Nx \times Xus) + (Nr \times Rus) + (Nw \times Wus)$$

where:

Ne	is the number of Entries for the functional process
Eus	is the unit size of an Entry (=1 CFP)
Nx	is the number of Exits for the functional process
Xus	is the unit size of an Exit (= 1CFP)
Nr	is the number of Reads for the functional process
Rus	is the unit size of a Read (= 1 CFP)
Nw	is the number of Writes for the functional process
Wus	is the unit size of a Write (= 1 CFP)

NOTE There is no upper limit to the functional size of a functional process.

5.11.3 Aggregation of functional size for the identified FUR for each piece of software to be measured

The size of each piece of software to be measured within a layer shall be obtained by aggregating the size of the functional processes within the identified FUR for each piece of software.

NOTE 1 Within each identified layer, the aggregation function is fully scalable. Therefore a sub-total can be generated for individual functional processes, individual software pieces or for the whole layer, depending on the purpose and scope of the FSM.

NOTE 2 Aggregating the measurement results by type of data movement might be useful for analysing the contribution of each type to the total size of a layer and might thus help characterize the functional nature of the measured layer.

NOTE 3 In a context where functional size is to be used as a variable in a model, for example to estimate effort, and the software to be sized has more than one layer, aggregation will typically be performed up to the layer level since layers are not necessarily implemented with the same productivity.

5.12 Calculation of functional size of changes to the FUR

Within each identified layer, the functional size of changes to the FUR within each piece of software within the scope of the FSM shall be calculated by aggregating the sizes of the corresponding impacted data movements according to the following formula:

$$FS_{changes} = FS_{added} + FS_{changed} + FS_{deleted}$$

where:

$FS_{changes}$	is the size of changes to a piece of software
FS_{added}	is the size of added data movements
$FS_{changed}$	is the size of changed data movements
$FS_{deleted}$	is the size of deleted data movements

summed over all functional processes for the piece of software.

NOTE A data movement is considered to be changed if any of the attributes of the data group are changed, or if any changes are needed to the data manipulation associated with the data movement.

EXAMPLE A requested change is: add one new functional process of size 6 CFP, and in another functional process add one data movement, make changes to three other data movements and delete two data movements. The total size of the requested change is 6 + 1 + 3 + 2 = 12 CFP.

6 Measurement reporting

6.1 Labelling

A COSMIC measurement result on the FUR for a piece of software that conforms to the mandatory rules of this International Standard shall be labelled according to the following convention.

CFP (ISO/IEC 19761:2011)

6.2 Documentation of the measurement results

The documentation of the COSMIC measurement results shall include the following information:

a) identification of each piece of software in the scope of the FSM (name, version identification or configuration identification);

b) a description of the measurement purpose and scope;

c) a description of the relationship of each piece of software within the scope of the FSM with its functional users and the position of the boundaries, both peer-to-peer and between layers;

d) the functional size of each piece of software within the scope of the FSM, calculated according to 5.11 or 5.12 and reported according to 6.1.

In addition, the documentation of the COSMIC measurement results should include the following information for each piece of software within the scope of the FSM:

e) a list of the functional processes identified and their associated data movement types;

f) a list of the data groups identified;

g) the total number of functional processes identified;

h) the total number of data groups identified;

i) the total functional size of the Entries;

j) the total functional size of the Exits;

k) the total functional size of the Reads;

l) the total functional size of the Writes.

NOTE Documentation is required for each piece of software measured within each layer.

Annex A
(informative)

Extraction of Functional User Requirements

There are many reasons to measure the functional size of software. In a particular context, it might be necessary to measure the functional size of software prior to its development. In a different context, it will be useful to measure the functional size of software *a posteriori*, that is, some time after it has been put into production.

Measuring the functional size of software prior to its development is based on the software “plans”; a collection of artifacts produced prior to development. The required dimensions (Base Functional Components) are extracted from the artifacts using appropriate conventions (from the perspective of the FUR, excluding technical and quality aspects) and the size is calculated according to a specific measurement function.

Likewise, measuring the functional size of software after it has been put into production entails a somewhat different measurement procedure when the required dimensions are extracted from the various artifacts. Although the nature of these artifacts differs, the dimensions, the unit of measurement and the measurement principles remain the same. There are many aspects of software. From the perspective of the COSMIC Functional Size Measurement Method, the aspect of interest is the functionality it delivers to its users. The functionality delivered by software to its users is described through the FUR. In practice FUR sometimes exist in the form of a specific document (requirements specifications, for instance), but often they have to be derived from other software engineering artifacts. As illustrated in Figure A.1, FUR can be derived from software engineering artifacts that are produced before the software exists (typically from software system and design artifacts). Thus, the functional size of software can be measured prior to its implementation on a computer system.

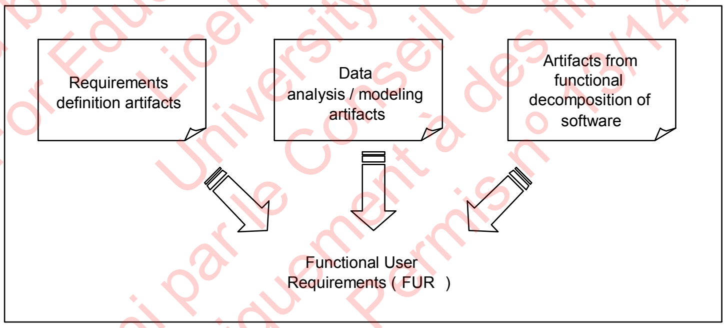
Requirements definition artifacts
Data analysis / modeling artifacts
Artifacts from functional decomposition of software
Functional User Requirements (FUR)

Figure A.1 — Extracting FUR at pre-implementation of software

In other circumstances, software might be used without there being any, or with only a few, software system or design artifacts available, and the FUR might not be documented (legacy software, for instance). In such circumstances, it is still possible to derive the software FUR from the artifacts installed on the computer system, as illustrated in Figure A.2.

The effort required to extract the FUR from different types of software engineering artifacts will obviously vary. The nature of the FUR remains constant, though, since no matter what type of software engineering artifacts are used to extract them, they always convey a description of the functionality delivered by the software to its users.

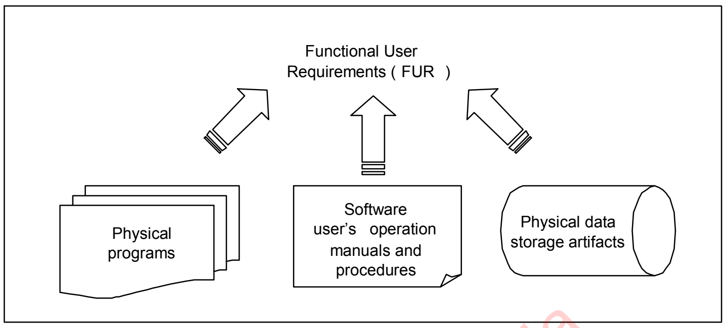
Functional User Requirements (FUR)
Physical programs
Software user's operation manuals and procedures
Physical data storage artifacts

Figure A.2 — Extracting FUR at post-implementation of software

Bibliography

[1] ISO Guide 99:1993, *International vocabulary of basic and general terms in metrology* (VIM)[1)]

[2] ISO/IEC 14143-1:2007, *Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts*

[3] ISO/IEC 15939:2007, *Systems and software engineering — Measurement process*

[4] Abran, A.; Desharnais, J.-M.; Oligny, S.; St-Pierre, D.; Symons, C., *COSMIC-FFP — Measurement Manual version 2.1*, Montreal (Canada), May 2001. www.cosmicon.com

[5] Lesterhuis, A.; Symons, C.R.; (Editors) — *The COSMIC Functional Size Measurement Method, Version 3.0, Measurement Manual*, December 2007. www.cosmicon.com

1) ISO Guide 99:1993 has been cancelled and replaced by ISO/IEC Guide 99:2007.

ICS 35.080

Price based on 14 pages