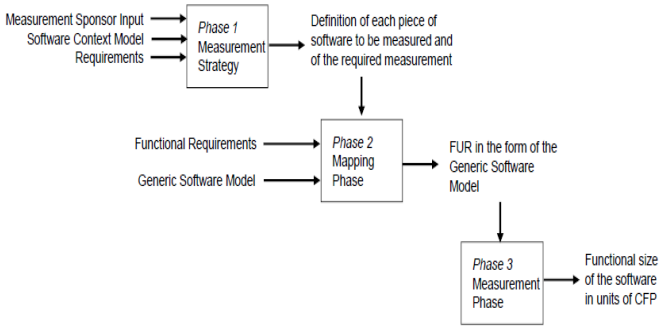


Functional User Requirements (FUR)
A sub-set of the user requirements. Requirements that describe what the software shall do, in terms of tasks and services.

Non-Functional Requirement.
Any requirement for the software part of a hardware/software system or software product, including how it should be developed and maintained, and how it should perform in operation, except a functional user requirement for software.

Functional Size.
A size of the software derived by quantifying the Functional User Requirements.

User
Any person or thing that communicates or interacts with the software at any time.

Functional Size Measurement (FSM)
The process of measuring functional size.

COSMIC unit of measurement
CFP (Cosmic Function Point), which is the size of one data movement.

Software Context Model
Enable a Measurer to define the software to be measured and the size measurement. These ensure that the results can be understood and interpreted consistently by future users.

Generic Software Model.
Define how the FUR of the software to be measured are modeled so that they can be measured.

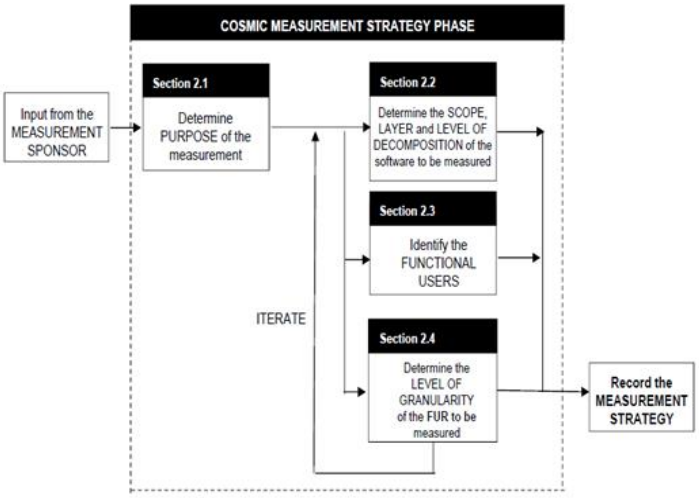


Figure 2.0 - The process of determining a Measurement Strategy

Purpose of a measurement
A statement that defines why a measurement is required, and what the result will be used for.

Scope of a measurement
The set of Functional User Requirements to be included in a specific functional size measurement exercise.

Layer
A functional partition of a software system architecture.

Peer pieces of software
Two pieces of software are peers of each other if they reside in the same layer.

Level of decomposition
Any level resulting from dividing a piece of software into components, then from dividing components into sub-components, then from dividing sub-components into sub-sub components, etc.

Functional user
A (type of) user that is identified in the Functional User Requirements of a piece of software being measured as a sender and/or an intended recipient of data processed by that software.

Boundary
A conceptual interface between the software being measured and its functional users.

Persistent storage
Storage which enables a functional process to store a data group beyond the life of the functional process and/or from which a functional process can retrieve a data group stored by another functional process, or stored by an earlier occurrence of the same functional process, or stored by some other process.

Level of granularity.
Any level of expansion of the description of any part of a single piece of software such that at each increased level of expansion, the description of the functionality of the piece of software is at an increased and uniform level of detail.

Measurement (Strategy) Pattern.
A standard template that may be applied when measuring a piece of software from a given software functional domain, that defines the types of functional user that may interact with the software, the level of decomposition of the software and the types of data movements that the software may handle.

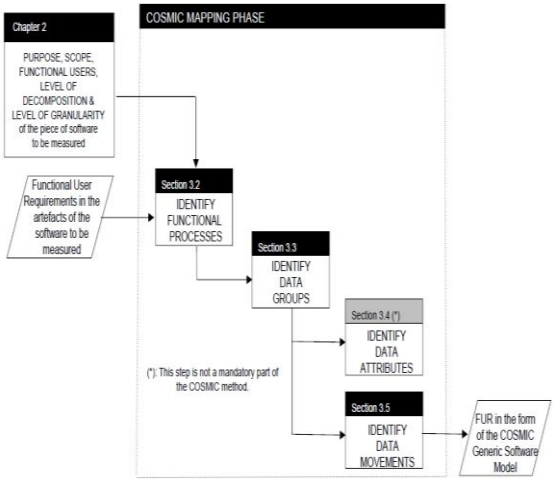


Figure 3.0 – General method of the COSMIC mapping process

Piece of software.
Any discrete item of software at any level of decomposition from the level of a whole software system down to and including the level of the smallest component of a software system.

Functional process type
A set of data movements, representing an elementary part of the Functional User Requirements for the software being measured, that is unique within these FUR and that can be defined independently of any other functional process in these FUR

Triggering event type.
An event, recognized in the Functional User Requirements of the software being measured, that causes one or more functional users of this software to each generate one or more data groups. The first data group generated by any one functional user will subsequently be moved by a triggering Entry. A triggering event cannot be sub-divided and has either happened or not happened.

Triggering Entry
The Entry data movement of a functional process that moves a data group generated by a functional user that the functional process needs to start processing.

Object of interest type.
Any 'thing' in the world of the functional user that is identified in the Functional User Requirements about which the software is required to move a data group in or out of the software, or to or from persistent storage. It may be any physical thing, as well as any conceptual object or part of a conceptual object.

Data Group
A data group consists of a unique set of data attributes that describe a single object of interest.

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

Base functional component (BFC).
An elementary unit of the Functional User Requirements defined by an FSM method for measurement purposes.

Data movement type
A base functional component which moves a single data group type.

Data manipulation
Anything that happens to data when it is processed by a functional process other than a movement of data into or out of a functional process, or between a functional process and persistent storage.

Control command
A command that enables human functional users to control their use of the software but which does not involve any movement of data about an object of interest of the FUR of the software being measured.

Error/confirmation message.
An Exit issued by a functional process to a human user that either confirms only that entered data has been accepted, or only that there is an error in the entered data.

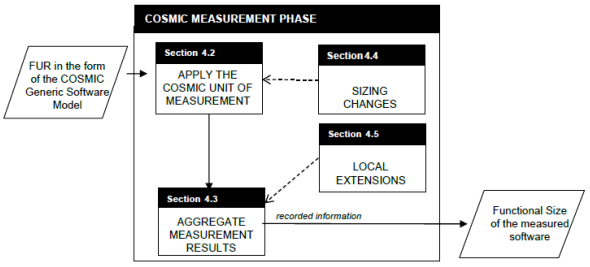


Figure 4.0 – General process for the COSMIC Measurement Phase

Entry type (E)

A data movement that moves a data group from a functional user across the boundary into the functional process where it is required.

Exit type (X)

A data movement that moves a data group from a functional process across the boundary to the functional user that requires it.

Read type (R)

A data movement that moves a data group from persistent storage into the functional process which requires it.

Write type (W)

A data movement that moves a data group from inside a functional process to persistent storage.

Modification (of the functionality of a data movement)

A data movement is considered to be functionally modified if at least one of the following applies: the data group moved is modified, the associated data manipulation is modified.

Size (functional process$_i$) =

Σ size(Entries) + Σ size(Exits) +

Σ size(Reads) + Σ size(Writes)

Size (Change(functional process$_i$)) =

Σ size (added data movements) +

Σ size (modified data movements) +

Σ size (deleted data movements)

COSMIC measurement labeling

A COSMIC measurement result shall be noted as 'x CFP (v) ', where: 'x' represents the numerical value of the functional size, 'v' represents the identification of the standard version of the COSMIC method used to obtain the numerical functional size value 'x'.

COSMIC local extensions labeling

A COSMIC measurement result using local extensions shall be noted as: ' x CFP (v) + z Local FP'

PRINCIPLES - THE COSMIC SOFTWARE CONTEXT MODEL

a) Software is bounded by hardware.
b) Software is typically structured into layers.
c) A layer may contain one or more separate 'peer' pieces of software.
d) Any piece of software to be measured, shall be defined by its measurement scope, which shall be confined wholly within a single layer.
e) The scope of a piece of software to be measured shall depend on the purpose of the measurement.
f) The functional users of a piece of software to be measured shall be identified from its Functional User Requirements (FUR) as the senders and/or intended recipients of data to/from the software respectively.
g) The functional requirements of software may be expressed at different levels of granularity.
h) A precise COSMIC size measurement of a piece of software requires that its FUR are known at the levels of granularity at which its functional processes and sub-processes can be identified.
i) An approximate COSMIC size measurement of a piece of software is possible if its functional requirements are measured at a high level of granularity by an approximation approach and scaled to the levels of granularity of the functional processes and sub-processes.

RULES DATA MOVEMENT (E, X)

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'. 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'.
b) Clock-ticks that are triggering events shall always be external to the software being measured. Therefore, for example, a clock-tick event occurring every 3 seconds shall be associated with an Entry moving a data group of one data attribute. Note that it makes no difference whether the triggering event is generated periodically by hardware or by another piece of software outside of the boundary of the software being measured.
c) Unless a specific functional process is necessary, obtaining the date and/or time from the system's clock shall not be considered to cause an Entry or any other data movement.
d) If an occurrence of a specific event causes the Entry of a data group comprising up to 'n' data attributes of a particular object of interest and the FUR allows that other occurrences of the same event can cause an Entry of a data group which has values for attributes of only a sub-set of the 'n' attributes of the object of interest, then one Entry shall be identified, moving a data group comprising all 'n' data attributes.
e) When identifying Entries in a screen that enables human functional users to input data into functional processes, analyze only screens that are filled with data. Ignore any screen that is formatted but otherwise 'blank' except for possible default values, and ignore all field and other headings that enable human users to understand the input data required.

a) An enquiry which outputs fixed text, (where 'fixed' means the message contains no variable data values e.g. the result of pressing a button for 'Terms & Conditions' on a shopping web-site), shall be modeled as having one Exit for the fixed text output.
b) If an Exit of a functional process moves a data group comprising up to 'n' data attributes of a particular object of interest and the FUR allows that the functional process may have an occurrence of an Exit that moves a data group which has values for attributes of only a sub-set of the 'n' attributes of the object of interest, then one Exit shall be identified, moving a data group comprising all 'n' data attributes.
c) When identifying Exits, ignore all field and other headings that enable human users to understand the output data.

PRINCIPLES – THE GENERIC SOFTWARE MODEL

a) A piece of software interacts with its functional users across a boundary, and with persistent storage within this boundary.
b) Functional user requirements of a piece of software to be measured can be mapped into unique functional processes.
c) Each functional process consists of sub-processes.
d) A sub-process may be either a data movement or a data manipulation.
e) A data movement moves a single data group.
f) There are four data movement types, Entry, Exit, Write and Read.
g) A data group consists of a unique set of data attributes that describe a single object of interest.
h) Each functional process is started by its triggering Entry data movement. The data group moved by the triggering Entry is generated by a functional user in response to a triggering event.
i) The size of a functional process is equal to the total count of its data movements
j) A functional process shall include at least the triggering Entry data movement and either a Write or an Exit data movement, i.e. it shall include a minimum of two data movements. There is no upper limit to the number of data movements in a functional process
k) As an approximation for measurement purposes, data manipulation sub-processes are not separately measured; the functionality of any data manipulation is assumed to be accounted for by the data movement with which it is associated

NOTE: In general, the 'type' of a thing is an abstract class of all the things that share some common characteristic in a given set of FUR, so that data about each occurrence of the 'thing-type' must be processed by the same functionality

RULES DATA MOVEMENT (R, W)

a) Identify a Read when, according to the FUR, the software being measured must retrieve a data group from persistent storage.
b) 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.

a) Identify a Write when, according to the FUR, the software being measured must move a data group to persistent storage.
b) 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.

RULES FUNCTIONAL PROCESS

a) A functional process shall belong entirely to the measurement scope of one piece of software in one, and only one, layer. Cualquiera Entrada desencadenante de una pieza de software que se está midiendo puede iniciar sólo un proceso funcional en ese software.
b) A functional process shall comprise a minimum of two data movements, namely the triggering Entry plus either an Exit or a Write, giving a minimum size of 2 CFP. There is no upper limit to the number of data movements in a functional process and hence no upper limit to its size.
c) An executing functional process shall be considered terminated when it has satisfied its FUR for all the possible responses to its triggering Entry. A pause during the processing for technical reasons shall not be considered as termination of the functional process.

PRINCIPLES DATA MOVEMENT (E, X, R, W)

a) An Entry shall move a single data group describing a single object of interest from a functional user across the boundary and into the functional process of which the Entry forms part. If the input to a functional process comprises more than one data group, each describing a different object of interest, identify one Entry for each unique data group in the input.
b) An Entry shall not exit data across the boundary, or read or write data from/to persistent storage.

a) An Exit shall move a single data group describing a single object of interest from the functional process of which the Exit forms part across the boundary to a functional user. If the output of a functional process comprises more than one data group, identify one Exit for each unique data group in the output.
b) An Exit shall not enter data across the boundary, or read or write data from/to persistent storage.

a) A Read shall move a single data group describing a single object of interest from persistent storage to a functional process of which the Read forms part. If the functional process must retrieve more than one data group from persistent storage, identify one Read for each unique data group that is retrieved.
b) A Read shall not receive or exit data across the boundary or write data to persistent storage.
c) During a functional process, movement or manipulation of constants or variables which are internal to the functional process and that can be changed only by a programmer, or computation of intermediate results in a calculation, or of data stored by a functional process resulting only from the implementation, rather than from the FUR, shall not be considered as Read data movements.
d) A Read data movement always includes any 'request to Read' functionality (so a separate data movement shall never be counted for any 'request to Read' functionality).

a) A Write shall move a single data group describing a single object of interest from the functional process of which the Write forms part to persistent storage. If the functional process must move more than one data group to persistent storage, identify one Write for each unique data group that is moved to persistent storage.
b) A Write shall not receive or exit data across the boundary, or read data from persistent storage.
c) A requirement to delete a data group from persistent storage shall be measured as a single Write data movement.
d) The following shall not be considered as Write data movements:
- The movement or manipulation of any data that did not exist at the start of a functional process and that has not been made persistent when the functional process is complete;
- Creation or update of variables or intermediate results that are internal to the functional process;
- Storage of data by a functional process resulting only from the implementation, rather than from the FUR. (An example would be the use of storage to store data temporarily during a large sort process in a batch-processed job.)

RULES 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 shall be 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 shall not be 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 shall not be identified as a changed CFP.

RULES ERROR/CONFIRMATION MESSAGES AND OTHER INDICATIONS OF ERROR CONDITIONS

a) One Exit shall be 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, e.g. successes or failures of: validation of entered data or for a call to retrieve data or to make data persistent, or for the response from a service requested of another piece of software.
NOTE: If the FUR of the functional process do not require any type of error/confirmation message to be issued, do not identify any corresponding Exit.
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 should be identified as a data group moved by an Exit in the normal way, in addition to the error/confirmation Exit.
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 shall be identified for any error indication received as a result of a Read or Write of persistent data.
e) No Entry or Exit shall be 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.

RULES DATA MOVEMENT UNIQUENESS AND POSSIBLE EXCEPTIONS

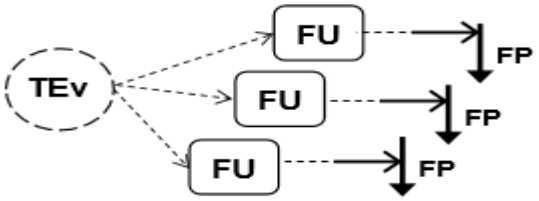
a) Unless the Functional User Requirements are as given in rules b) or c), all data describing any one object of interest that is required to be entered into one functional process shall be identified as one data group moved by one Entry.
The same equivalent rule applies to any Read, Write or Exit data movement in any one functional process.
NOTE: A functional process may, of course, have multiple Entries, each moving data describing a different object of interest.
b) If Functional User Requirements 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 shall be identified for each of these different data groups.
The same equivalent rule applies for Exits of data to different functional users from any one functional process.
NOTE: Any one functional process shall have only one triggering Entry.
c) If Functional User Requirements 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 shall be identified for each of these different data groups.
The same equivalent rule applies for Writes in any given functional process.
NOTE: This rule 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.
d) Repeated occurrences of any data movement type when it is being executed shall not be counted.
This applies even if multiple occurrences of the data movement type differ in their execution because different values of the data attributes of the data group moved result in different processing paths being followed through the functional process type.

CARDINALITY OF TRIGGERING EVENTS, FUNCTIONAL USERS AND FUNCTIONAL PROCESSES

All the relationships along the triggering event / functional user / triggering Entry / functional process chain may be many-to-many in principle, with one exception. The table below shows examples of possible relationships. Note that the cases may not be exhaustive. The table uses the following abbreviations and symbols.

TEv	Triggering event	FU	Functional User
----→	Data group (dotted part) moved by a triggering Entry (solid arrow)	FP	Functional Process

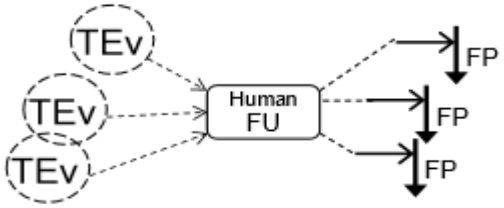
1. A single triggering event may cause multiple FUs to each initiate a triggering Entry in the same or in different software systems. Each triggering Entry starts its own FP



REAL TIME EXAMPLE: The triggering event of an earthquake may be detected by multiple independent FU sensors. Each FU initiates a triggering Entry that starts its FP in the same or in different systems.

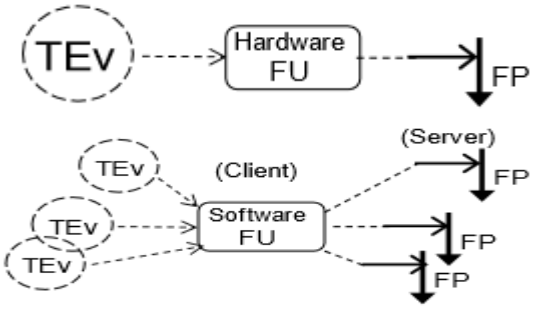
BUSINESS EXAMPLE: The triggering event of a new employee starting work causes one human FU to enter basic employee data to a Personnel system and another human FU to enter salary data to a Payroll system

2. Each triggering event causes a human FU to initiate a different triggering Entry. Each triggering Entry starts its FP in the same or in different software systems



BUSINESS EXAMPLE: In a police emergency telephone call-handling system, many types of triggering events may be reported causing a human FU to decide to initiate different triggering Entries. Each of these starts its FP to record the event. Additionally, the human user may initiate different enquiry triggering Entries. Each of these starts its FP in the same call-handling system or in other systems.

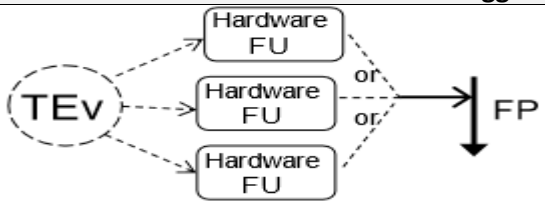
3. A hardware or software FU may be designed to sense (or 'generate') one or more specific type(s) of events. Each of these causes the FU to initiate a triggering Entry. Each of these starts its FP in the same software system



REAL-TIME EXAMPLE: When the temperature of a liquid reaches a pre-set level (the triggering event), a thermocouple FU initiates a triggering Entry to start its FP in one specific software system.

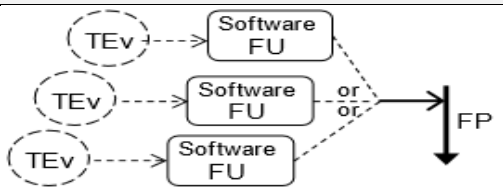
BUSINESS EXAMPLE: In a distributed software application, the client component is a FU of the server component. Different needs for information (the triggering events) of the client component cause it to initiate different triggering Entries, each to start its FP of the server component, for each different type of service it needs

4. Two or more hardware FUs of the same software may sense the same triggering event. Each FU may initiate the triggering Entry that starts the same one FP



REAL-TIME EXAMPLE: The triggering event of an abnormal situation in a real-time process control system, may be sensed by one or more hardware FUs. Each FU may initiate the one emergency shut-down FP. (NOTE: Any one occurrence of this FP will be initiated by the first FU to sense the triggering event).

5. Two or more software FUs may each initiate a triggering Entry that starts the same one FP



INFRASTRUCTURE EXAMPLE: Several software FUs may each 'call', i.e. initiate, the same FP in the same re-usable software component. (In this case the software FU 'generates' the event when it calls the component.) (NOTE: Any one occurrence of this FP can be initiated by only one of its possible software FUs at any one time.)

6. On sensing one triggering event, a FU may initiate two or more triggering Entries. Each triggering Entry starts its FP

REAL-TIME EXAMPLE: In a duplex safety-critical control system, one triggering event may cause a FU (usually hardware) to initiate two triggering Entries, each starting its FP. The two FPs could, for example, have the same FUR but be developed by separate groups as a result of a diversity strategy