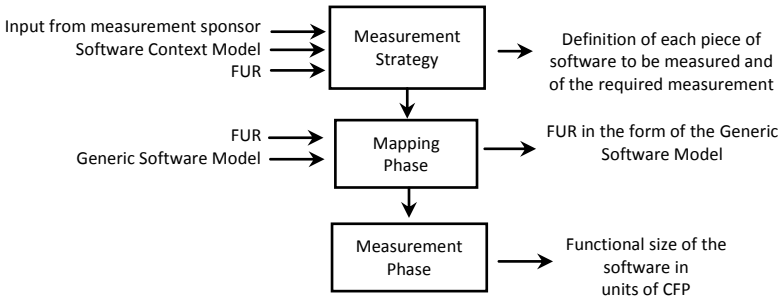


The COSMIC method measurement process

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 (NFR): 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 FUR for software.

Functional Size: A size of the software derived by quantifying the FUR.

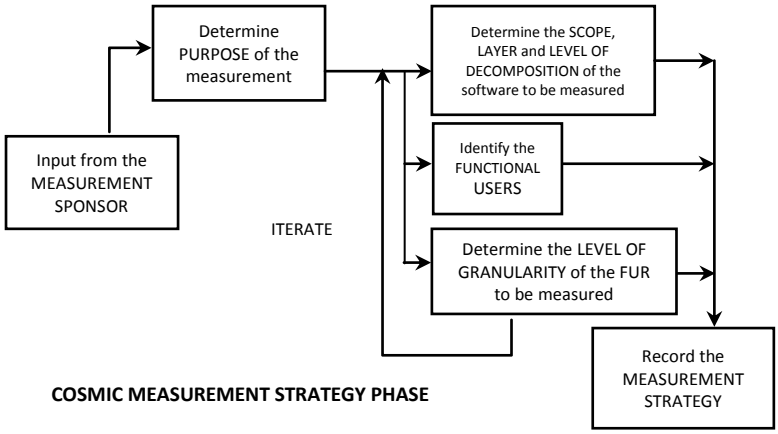
User: Any person or thing that communicates or interacts with the software at any time.

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

COSMIC unit of measurement: 1 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.



COSMIC MEASUREMENT STRATEGY PHASE

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 FUR to be included in a specific FSM 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.

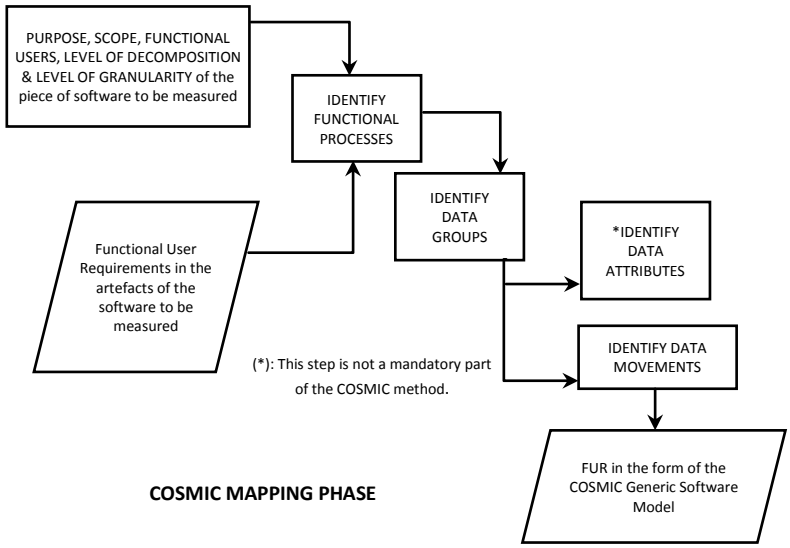
Level of decomposition: Any level resulting from dividing a piece of software into components then from dividing components into sub-components, and so on.

Functional user: A (type of) user that is a sender and/or an intended recipient of data in the FUR of a piece of software.

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

Level of granularity: Any level of expansion of the description 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.

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.



COSMIC MAPPING PHASE

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: A set of data movements, representing an elementary part of the FUR 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.

Object of interest (OOI): Any 'thing' in the world of the functional user that is identified in the FUR, about which the software is required to process and/or store data. It may be any physical thing, as well as any conceptual object or part of a conceptual object.

Data group: A distinct, non-empty and non-ordered set of data attributes where each included data attribute describes a complementary aspect of the same one object of interest.

Data attribute: The smallest parcel of information, within an identified data group, carrying a meaning from the perspective of the software's FUR.

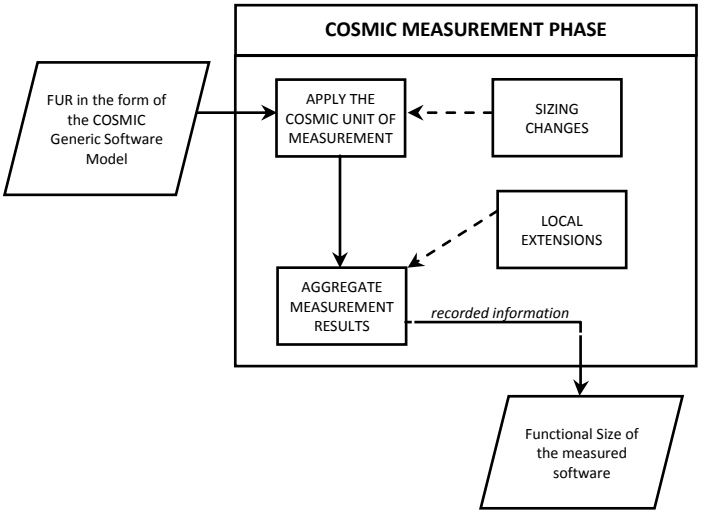
Base Functional Component (BFC): An elementary unit of the FUR defined by an FSM method for measurement purposes.

Data movement: A BFC which moves a single data group type.

Data manipulation: Anything that happens to data other than movement of the data into or out of a functional process, or between a functional process and persistent storage.

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 data.

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 OOI of the FUR of the software being measured.



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

Read (R): A data movement that moves a data group from persistent storage into the functional process which requires it.

Size (functional process$_i$) = **Σ size(Entries$_i$) + Σ size(Exits$_i$) + Σ size(Reads$_i$) + Σ size(Writes$_i$)**

Size (Change(functional process$_i$)) = **Σ size (added data movements$_i$) + Σ size (modified data movements$_i$) + Σ size (deleted data movements$_i$)**

COSMIC measurement labeling: a measurement result shall be noted as '**x** CFP (v.**y**) ', where: 'x' represents the numerical value of the functional size, and 'v.**y**' represents the id of the standard version of the method used to obtain the value 'x'.

Exit (X): A data movement that moves a data group from a functional process across the boundary to the functional user that requires it.

Write (W): A data movement that moves a data group lying inside a functional process to persistent storage.

Modification a data movement: At least one of the following applies:
- the data group moved is modified (when one or more attributes are added, removed or modified, e.g. in meaning or format, but not in their values)
- the associated data manipulation is modified.

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 FUR as the senders and/or intended recipients of data to/from the software respectively.
g) The FUR 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 a level of granularity at which its **functional processes** and sub-processes may be identified.
i) An approximate COSMIC size measurement of a piece of software is possible if its FUR are measured at a high level of granularity by an approximation approach and scaled to the level of granularity of the functional processes and sub-processes.

DATA MOVEMENTS - RULES DESCRIPTION

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'. 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 time from the system's clock shall not be considered to cause an Entry.
d) If an occurrence of a specific event triggers the Entry of a data group comprising up to 'n' data attributes of a particular OOI and the FUR allows that other occurrences of the same event can trigger an Entry of a data group which has values for attributes of only a sub-set of the 'n' attributes of the OOI, then one Entry shall be identified, 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.

Exit (X)

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 OOI 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 OOI, then one Exit shall be identified, comprising all 'n' data attributes.
c) When identifying Exits, ignore all field and other headings that enable human users to understand the output data.

Read (R)

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 stored data group. (For this case see the principles and rules for Entries and Exits.)

Write (W)

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. (For this case see the principles and rules for Entries and Exits.)

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.
b) Any one triggering Entry of a piece of software being measured may initiate only one functional process in that software.
c) A functional process shall comprise at least two data movements, an Entry plus either an Exit or a Write. There is no upper limit to the number of data movements in a functional process.
d) An executing functional process shall be considered terminated when it has satisfied its FUR for the response to its triggering Entry. A pause during the processing for technical reasons shall not be considered as termination of the functional process.

PRINCIPLES – The COSMIC Generic Software Model

a) A piece of software interacts with its functional users across a boundary, and with **persistent storage** within this boundary.
b) FUR 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 **OOI**.
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) A functional process shall include at least one 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.
j) 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.

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 OOI 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 OOI.
b) A Functional User Requirement may specify different data groups to be entered into one functional process from functional users that must be separately identified by that functional process, where each data group describes the same object of interest. 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) A FUR may specify different data groups to be moved from persistent storage into one functional process, each describing the same OOI. One Read shall be identified for each of these different data groups. The same equivalent rule applies for Writes in any given functional process.
d) Repeated occurrences of any data movement type when it is being executed shall not be counted. Numbers of occurrences when software executes are irrelevant to FSM. 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.

PRINCIPLES – DATA MOVEMENTS

Entry (E): a) An Entry shall move a single data group describing a single OOI 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 OOI, identify one Entry for each unique data group in the input.
b) It shall not exit data across the boundary, or read or write data from/to persistent storage.

Exit (X): a) An Exit shall move a single data group describing a single OOI 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) It shall not enter data across the boundary, or read or write data from/to persistent storage.

Read (R): a) A Read shall move a single data group describing a single OOI 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) It 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).

Write (W): a) A Write shall move a single data group describing a single OOI 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) It 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 storage of data temporarily during a large sort process in a batch-processed job.)