**FP COSMIC** (Common Software Measurement International Consortium)

Applicability of the method

The COSMIC method was designed to measure the Functional User Requirements (FUR) of business application (or 'management information system'), real-time and infrastructure software and some types of scientific/engineering software, in any layer of a software architecture, and at any level of decomposition.

The three phases of the COSMIC functional size measurement process

The COSMIC measurement process is shown in Figure 4.1. The three phases are explained in the next sections.

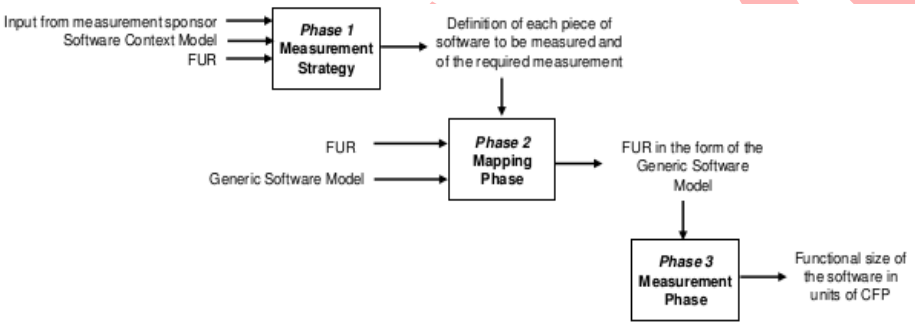


Figure 4.1 - The COSMIC measurement process

**Phase 1:** ***Measurement Strategy***

We must first define what will be measured. The size of a piece of software depends on the viewpoint of who or what we define as its functional users, i.e. the humans, hardware devices or other software that interact with the software. In order to measure the size of the piece of software, we must therefore first agree the purpose of the measurement, which leads to defining its scope (the extent of the software's FUR to be measured) and its functional users, and then usually some other parameters . .

It's essential to document the parameters of the measurement strategy so that the resulting measurement(s) will be correctly interpreted by all future users.

**Phase 2:** ***Mapping***

The task of the Mapping phase is to create the COSMIC model of the FUR, starting from whatever artefacts of the software are available, e.g. an outline or detailed statement of requirements, design models, the installed physical software, etc. To create the model, we apply the principles of the COSMIC Generic Software Model to the FUR to be measured.

This model of the FUR of software rests on four main principles:

1. Software functionality consists of functional processes. The task of each functional process is to respond to an event that has happened in the world of the software's functional users.
2. Functional processes consist of sub-processes. These do only two things: they move and they manipulate data. Data movement sub-processes that move data from functional users into a functional process and that move data out to them are called Entries and Exits respectively. Data movement sub-processes that move data to and from persistent storage are called Writes and Reads respectively. Figure 4.2 illustrates the four types of data movements.

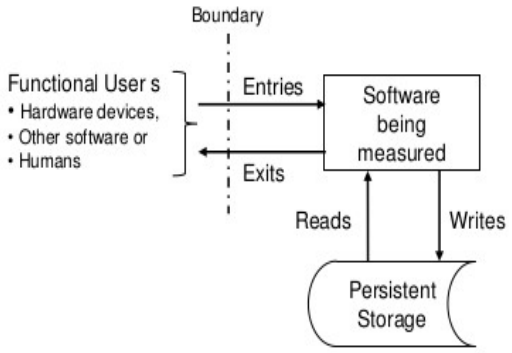


Figure 4.2 - The four types of data movements

3. Each data movement (Entry, Exit, Read or Write) moves a single data Group whose attributes describe a single 'thing' (an object of interest).
4. Data manipulation sub-processes are assumed to be accounted for by the data movement with which they are associated. Data manipulation is not measured separately.

A functional process finishes executing when it has done all that it is required to do to respond to the data it received about the event.

**Phase 3: *Measurement***

The COSMIC method measurement unit is the 'Cosmic Function Point' (CFP). Each data movement is measured as 1 CFP.

In the Measurement phase, we measure the size of a new piece of software by identifying all the data movements (Entries, Exits, Reads and Writes) of each functional process and sum these over all its functional processes.

A functional process must have at least two data movements (an Entry plus either an Exit or a Write) in order to provide a minimal but complete service. Hence the minimum size of a functional process is 2 CFP. There is no upper limit to the size of a functional process.

To measure an enhancement to existing software, we identify all the data movements to be added, changed and deleted, and sum these over all its functional processes. The minimum size of any modification to a functional process is 1 CFP.

EXAMPLE

We can now analyze some simple examples to map from outline statements of requirements to the COSMIC functional processes and data movements using the Generic Software Model.
BUSINESS EXAMPLE: A simple Personnel System
Outline statement of requirements. A system is required to enable personnel staff to hold and maintain data about employees, including their salary and the history of their salary progression over time. [The statement also describes the data (attributes) to be recorded about each employee and their validation criteria, but most of this detail need not concern the Measurer in this simple example.] A report is required each month listing all employees by name and their current salary, the total number of employees, and the total current salary cost.
Measurement strategy parameters
Measurement purpose: An accurate functional size measurement of the Personnel System for project effort estimating.
Measurement scope: The whole system as specified in the statement of requirements.
Functional users: Personnel staff.
Layer: Application layer.

Level of decomposition: ‘level 0’, i.e. no decomposition.
Mapping phase
Some assumptions:

The data structure of the Business Example in figure 6.4 applies.

Each employee will be allocated a unique ID by a member of personnel. The key of an ‘employee salary history’ record is [employee ID, salary start date].

The word ‘maintain’ in the outline statement of requirements normally implies that there must be Create, Read, Update and Delete functional processes (remember the ‘CRUD’ acronym?) for each object of interest. ‘Update’ will enable a change of any attribute except the key attribute(s) of any data group.

There are two objects of interest (‘employee’ and ‘employee salary history’) about which persistent data must be held. We will need the four ‘CRUD’ functional processes to maintain the employee base data.

We also assume that an employee salary history record must be created when an employee first starts work. Subsequently, an employee’s salary may be updated either when the employee base data must be updated, or separately from an employee base data update. So there is no need for separate ‘create’, or ‘delete’ functional processes for the employee salary history, but there is a need for an ‘employee salary update’ functional process and for a separate ‘read’ functional process for enquiries on employee salary history. Adding in the process to produce the monthly report means the requirements can be satisfied by 7 functional processes. (We show below the analysis of four of these.)

In practical situations there also may be FUR for a ‘read’ functional process to display the employee’s base data separately from the enquiry to display the employee’s salary history. Also, when an employee leaves, the ‘delete’ functional process may be required to archive the employee base and salary history data, rather than actually delete it. We ignore these possible requirements for simplicity.

Note also as a general rule when measuring on-line business applications, that ‘menus’ that only assist navigation and the selection of functional processes, and ‘blank’ data entry screens should be ignored. It is the movement of data by Entries, Exits, Reads and Writes that must be identified for measurement purposes.

**1. Analysis of functional process ‘Create employee’.**

The FUR is to enter data for a new employee.

(The examples show the data movements and the data group that each of them moves).

**Functional process ‘Create employee’. Triggering event: a new person is employed**

*Triggering Entry* : Employee base data

*Entry* : Initial salary and its start date

*Read* : Employee base data (to check that no employee already exists with the entered ID)

*Write* : Employee base data

*Write* : Employee salary history (a new record is created when the salary is first entered)

*Exit* : Error/confirmation messages (There must be error messages for various validation failures and also some form of confirmation on successful data entry. We include one Exit to account for all

such messages.)

**2. Analysis of 'Update employee data' process, including possible salary update**

We assume a user will first wish to retrieve and display the employee's base data, before entering a change to one or more attributes, including maybe a new salary. This procedure will require two functional processes. The first is triggered by the event of the user deciding to display the existing data; it is the 'read employee base data' functional process. The second is triggered by the event that one or more attribute(s) of the employee have changed in the real world; it is the 'update employee base data' functional process. The two functional processes are:

**Functional process 'Read employee data'. Triggering event: Decision to display existing data**

*Triggering Entry* : Employee ID
*Read* : Employee base data
*Read* : Employee salary history
*Exit* : Employee base data
*Exit* : Employee salary history
*Exit* : Error/confirmation messages (in case a non-existent ID was entered)

**Functional process 'Update employee data'. Triggering event: Employee data changed has changed in some way**

*Triggering Entry* : Updated employee base data (for the update of one or more attribute(s))
*Entry* : Updated salary and its start date
*Write* : Employee base data (the updated record)
*Write* : Employee salary history (a new record is created if the salary has been updated)
*Exit* : Error/confirmation messages (for entry of invalid data or the possible failure of the update)

**3. Analysis of the process to produce the monthly report for the payroll department**

**Functional process 'End-of-month employee' report. Triggering event: The end of the month**

*Triggering Entry* : End of month time signal (every functional process must have a triggering Entry, even though this one conveys no variable data)

*Read* : Employee base data (to get employee ID's and names)

*Read* : Employee salary history (to obtain the current salary)

*Exit* : Employee current salary (one line for each employee with their ID, name and salary

*Exit* : End-of-month employee totals (of numbers of employees and of their total salary)

NOTES: the final Exit moves a data group describing the object of interest 'All employees'. No data is stored about this object of interest, so the data group is transient; but the object of interest is a group of real people, i.e. a real 'thing' in the world of the functional user.
We have not counted an error message for this functional process as there does not seem to be any

reason for the application to have to generate such a message. (The operating system might generate an error message if the data cannot be found, but this is not part of the application.)

COSMIC METHOD - THE MEASUREMENT PHASE

By the end of the Mapping phase, the Measurer will have produced a COSMIC model of the FUR of
the piece of software to be measured (an instance of the Generic Software Model). We can then measure the functional size of the FUR of the software by applying the rules of the Measurement phase to this model.

7.1

The COSMIC measurement principle

The COSMIC measurement principle reflects the model shown in the right-hand part of Figure 6.2.

The COSMIC measurement principle

The functional size of a piece of software is equal to the number of its data movements

A functional size is measured in units of 'COSMIC Function Points', abbreviated as 'CFP'. 1 CFP is
defined by convention as the size of a single data movement (Entry, Exit, Read or Write).

7.2

Size aggregation

Sizes can be measured at various levels of aggregation.

The size of a functional process is equal to the number of its data movements

The size of a piece of software is equal to the sum of the sizes of its functional processes

The size of a piece of software can be derived from the size of its components provided the aggregation rules given in the Measurement Manual are followed

The following table shows a way of recording the results of the analysis of the four functional processes of the Personnel System analyzed in section 6.9, using the matrix given in Appendix A of the Measurement Manual

| | Data Group Names | | | | | | | Nos. of Data Movements | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Personnel System Functional Processess | Employee Base Data | Employee ID | Employee Salary History | Error/Confirmation Message | End of month 'clock tick' | Employee current salary | Employee totals | Entries | Exits | Reads | Writes | Total |
| Create Employee | E, R, W | | E, W | X | | | | 2 | 1 | 1 | 2 | 6 |
| Read Employee data | R, X | E | R, X | X | | | | 1 | 3 | 2 | | 6 |
| Update Employee data | E, W | | E, W | X | | | | 2 | 1 | | 2 | 5 |
| End of month report | R | | R | | E | X | X | 1 | 2 | 2 | | 5 |
| | | | | Totals for Personnel System: | | | | 6 | 7 | 5 | 4 | 22 |

**Size of required changes**

The size of some required changes to an existing piece of software, e.g. as handled by an 'enhancement' project, are measured as follows:

- The size of a required change to a data movement (i.e. that must be added, modified or deleted) is measured by convention as 1 CFP. ('Modified' could mean any changes to the data manipulation associated with the data movement and/or to any of the attributes of the data group moved.)
- The minimum size of a change to a functional process is therefore 1 CFP.
- The size of all the required changes to a piece of software is equal to the number of

datamovements that must be added, modified or deleted, summed over all functional processes.