‘COSMIC’ – the COmmon Software Measurement International Consortium

COSMIC FFP – the new Software Functional Sizing Method

Supplementary Notes to ‘Introduction & Overview’ Slide Presentation

Slide 10 Terminology

A ***‘Software Item’*** is a separate piece of software at any time in its life from when it exists as a Statement of Requirements until it exists as executable code.

A ***‘Base Functional Component’*** (abbreviated as ‘BFC’) is a term defined in ISO/IEC 14343 Part 1 on ‘Functional Size Measurement’ as follows.

An elementary unit of functional user requirements defined by and used by an FSM Method for measurement purposes.

Note – Example, a Functional User Requirement could be “Maintain Customers” which may consist of the following BFC’s: “Add a new Customer”, “Report Customer Purchases” and “Change Customer Details”. Another example might be a collection of logically related business data maintained by the software under study such as “Customer Details”.

A ***‘BFC Type’*** is defined in ISO/IEC 14143 Part 1as follows.

A defined category of BFC’s

Note – Examples of BFC Types are ‘External Inputs’, ‘External Outputs’ and ‘Logical Transactions’ and data stores such as ‘Internal Logical Files’.

Slide 11

Interpretation of this slide.

“Software Item ‘X’” is simply any item of software

An overall set of requirements consists of ‘Functional User Requirements’ and other requirements, generally referred to as ‘Technical & Quality Requirements’

‘Functional User Requirements’, abbreviated as ‘FUR’, are defined in ISO/IEC 14143 Part 1 as follows.

A sub-set of the user requirements. The Functional User Requirements represent the user practices and procedures that the software must perform to fulfil the users’ needs. They exclude Technical and Quality Requirements.

The important conclusion from this slide is that some Technical and Quality Requirements of the principal Software Item ‘X’ may be satisfied by new or changed Functional User Requirements of *other* Software Items.

These other Software Items may be in other layers of the software, as we see from the next Slide 12.

Slide 12

The concept of ‘layering’ of software is very important, although there are no standard models of layers.

Existing Function Point methods, originally designed to measure the size of business application software, do not easily recognise the size of software in lower layers. But increasingly a given set of requirements can have implications for software in several layers.

Note that the model of layers shown on the Slide is 'conceptual', that is it shows how we often think of a Human User interacting directly with the application software. The latter calls on middleware and the operating system, etc, to complete its tasks. Actually, physically, a Human User interacts with the hardware which is served by a device driver, which passes the data through the upper layers to the application software.

The COSMIC FFP method aims to be able to measure the functional size of requirements of software in any layer. So, supposing we have a set of Functional User Requirements for a Software Item 'X' which also results in additions or changes to software in lower layers. If the COSMIC FFP method can measure

the size of the principal software item 'X' which has to be built
(plus) the size of these additions or changes to other software items,

then we have an improved functional size measure for all the requirements allocated to software.

The COSMIC FFP Measurement Manual will give guidance on how to analyse existing software which has never been 'architected' into layers of requirements, so that the method can be applied to obtain the functional size of existing software after it has been built.

Slide 13

The combined effect of the ideas in Slides 11 –12 is that the COSMIC FFP method will have less need for something like the 'Value Adjustment Factor' (or 'Technical Complexity Adjustment') of existing Function Point methods.

This VAF factor is now known to be a very unsatisfactory way of dealing with what it tries to measure. If we can eliminate the need for a VAF, or reduce its effect, that will be a big improvement over current methods.

It may not be possible to eliminate this type of factor completely for dealing with Technical and Quality requirements, but it should be a less significant factor in the future.

Furthermore, in the future, any such factor should only deal with the requirements of *software*. The existing VAF also deals with some factors which, arguably, are concerned with the wider *system*. Examples would be 'Installation Ease' and 'Multiple Sites'.

COSMIC FFP will give priority to designing the 'Functional Size' measure (of Functional User Requirements). The problem of accounting for the effect on size of remaining Technical and Quality Requirements will be dealt with as a second priority.

Slide 14

This is a most important slide in showing the basic model that Functional User Requirements have to be mapped to in the COSMIC FFP method, as the step to ensure consistency of interpretation, before measuring the size of the Requirements.

A 'Transaction-Type' is defined as "a sequence of data movement and data manipulation sub-process steps, triggered by an Event external to the software item. The sequence is complete when the data processed is consistent with respect to the external Event". In practice, it also ends when a self-induced wait state is reached.

The term 'Transaction-Type' is not commonly used in the real-time software community. 'Functional Process' is a possible alternative term. Another way of expressing this definition for the real-time community is that "a Transaction-Type or Functional Process corresponds to a functional state-transition of a thing external to the software that the software is required to respond to".

Examples would be:

- The Event of the recruitment of a new employee in the external world triggers a human user to enter basic personnel data about the new employee
- The Event of the arrival of a message in a message switching system triggers the software to process the message. (The state-transition is from 'incoming, un-switched' to 'outgoing, switched'.)
- The Event of the receipt of a time signal triggers process control software to go through one cycle of polling the sensors of a physical device to determine its state and to feed back signals to control the device if a correction is needed
- The Event of an application program needing to store a data record results in the application passing the data record to the data management software for storage (probably via lower layers of software)

A 'Data Movement Type' is "A sub-process of entering, exiting, reading or writing a logically related block of data"

A 'Data Manipulation Sub-Process' is "A sub-process of validating, changing e.g. re-formatting, or transforming data to create new data".

Data Movement Types can be further broken down into Data Attribute Types (also known as 'Data Element types' in existing function point methods).

Slide 15

This slide expands the FUR model. Any Software Item has a 'User' who may be a human, or other Software Item, or a physical device that sends/receives signals, or an animal, etc.

The 'User' interacts with the Software Item across a 'Boundary' via Entry and Exit Data Movement sub-processes. Any Transaction-Type must have at least one Entry and one Exit.

In addition there may be Write and Read Data Movement sub-processes which respectively store and retrieve persistent data for the User. As far as the User is concerned, these sub-processes take place within the Software Item with which the User is interacting.

Slide 16

In practice, the software architect or designer may decide to delegate certain tasks to another Software Item which may already exist or has to be created. In the example shown, the designer has delegated the task of Writes and Reads of persistent data to another Software Item, e.g. a disk driver, because of the complexity of these functions, and because the latter Software Item may be re-usable by other requirements.

The result is that Software Item 1 becomes the 'User' for Software Item 2 across another Boundary in a layer below Software Item 1. A Data Movement which is seen from the User of Software Item 1 as a Write, is interpreted by Software Item 1 as an Entry to Software Item 2 (since Software Item 1 is the User of Software Item 2). Similarly, a Data Movement which is seen by the User of Software Item 1 as a Read is interpreted by Software Item 1 as an Exit from Software Item 2. The User of Software Item 1 is unaware of the existence of Software Item 2. This process of partitioning requirements into layers can be repeated through successive layers, up or down.

Note that in this case the Write and Read will be interpreted differently by the software of the different layers. For example, if the upper layer is for business application software, it will probably regard a Write as a requirement to store a string of Data Attributes of an entity which is defined in terms meaningful to the (human) user of Software Item 1. But Software Item 2, the disk driver, will treat the same Data Movement as an Entry of a fixed-length block of data, of which the first part is the record key, to be passed to the physical disk.

The functionality of Software Item 1 is simple in the sense that it only has to deal with 'logical' Writes and Reads and does not have to worry about disk physical record management, error handling and such-like. This is all delegated to Software Item 2. If Software Item 2 had to be created as part of a new

development, its functionality could be missed by conventional function point methods which are designed to deal only with the 'application' layer.

Slide 17

This slide is of course a highly-simplified example. In practice there may be many types of Update and Read Transactions. The 'Create Customer' Transaction-Type may also have many more sub-processes.

Slide 18

The concept of 'Use Case' seems to have many possible interpretations and as a result it is very difficult to agree on the rules for measuring its size. The Transaction-Type, however, which is a special instance of a Use Case, is very well defined and can be unambiguously identified in practice. It is therefore a suitable concept for size measurement.

Slide 19

Why do we need to simplify the general model of Sub-Processes?

a) To define and size a 'Data Manipulation Sub-Process' is difficult. The answer depends on the designer and the architecture. But to define and assign sizes to 'Data Movement Types' is relatively easy.

b) For the COSMIC priorities of MIS and real-time software, we will assume that a fixed amount of simple data manipulation is included when allocating a size for every 'Data Movement Type'

c) Where there are very complex data manipulation processes, as in Scientific/Engineering software, the software sizing problem does not appear to be critical for people working in that Domain. The effort is in the creation of the algorithms, not in their implementation in software. So this sizing problem is not a high priority for COSMIC.

However, in the COSMIC FFP method we will allow for the possibility of a 'Local User Extension' for sizing a local domain-specific Data Manipulation component.

Slide 21

There are many possible sizes for a Software Item. The size can depend upon:

- Whether only the Functional User Requirements are taken into account (in which case it is known as a 'Functional Size') or whether all requirements, that is including Technical and Quality Requirements are accounted for in the size
- Whether the allocation of size units to the Base Functional Component Types is made by
  - expert judgement of 'functionality', or
  - a specific calibration process of determining the 'standard effort' to develop the component

The latter 'standard effort' can be calibrated by taking measurements either

- from projects developed using as wide a range as possible of different technologies and development environments, so that the average is independent of any one technology, etc; the result is then suitable for performance comparisons across measures from different technical environments, or
- or from projects developed with only one technology, so that the resulting standard hours are suitable for project estimating in that technology environment

The calibration process involves multiple regression analysis to correlate measurements of actual project development hours against the counts of components developed.

There are of course also many other possible size measures suitable for different purposes, such as measures of the amount of new information created by the software. But these are outside the scope of COSMIC.

Slide 22

We propose that the unit of functional size of the COSMIC FFP Method will be 'one' for a single Data Movement Type. The minimum size for a software requirement is therefore 'two', being the size of a Transaction-Type which has a single Entry and Exit.

There is the possibility that the Field Trials will show that some software requirements will have Data Movement Types which are consistently much bigger on average than for other software requirements, depending on the relative average number of their Data Attributes. For this reason we will, in the Field Trials, count the number of Data Attributes on each Data Movement, to see if some refinement of the size scale is needed.

Slide 23

This slide gives a 'meta-model' (an entity-relationship model) of all the concepts defined in the COSMIC FFP method and how the are related.

Note that normally each separate type of Event which the software is required to respond to results in a unique Transaction-Type. In some circumstance, however, a single Event such as the clock passing a certain time can trigger the processing of a string of different Transaction-Types, for example to generate several types of reports.

Note also that 'Inputs' and 'Outputs' are not concepts of the COSMIC FFP model. These terms are shown to illustrate the relationship of the COSMIC FFP model with normal language. (For example, an Input is the collection of all the Entries for a single Transaction-Type.)

Slides 26, 27

The COSMIC FFP Measurement Manual is now in its final stages of editing at UQAM (University of Quebec at Montreal, Canada), and will in effect become the first practical implementation of COSMIC, to be known as 'COSMIC FFP V2.0'. The COSMIC Team's target is to be ready to issue the COSMIC FFP V2.0 Measurement Manual very shortly after the IWSM'99 Workshop which was held in Canada in September 1999.

The COSMIC Team is now seeking organisations who are interested to provide project requirements data for Field Trials of the method, and who can contribute to the Field Trial costs. Field Trials will take place starting in the last quarter of 1999.

For the latter we propose a participation fee of US$ 15,000, plus any travel expenses for the COSMIC consultant. The target is to collect data from 50 – 100 projects from a wide variety of software domains, from around 20 organisations.

Each participant in the COSMIC FFP Field Trials should gain significant benefits from the technology transfer which will be involved and from the opportunity to influence the detailed rules for the method. Confidentiality of participants' own data will be assured.

COSMIC Team
17th September 1999