Sizing and Estimating for Real-time Software – the COSMIC-FFP method

Charles Symons[1]

If a reliable method existed that allowed managers of real-time software development or enhancement projects to:

a. estimate a software size from its requirements, early in the life of a project
b. enter that size together with other parameters into formulae to estimate project effort and duration (where the formulae had been calibrated on recently-completed projects in the same domain)
c. track the software size as requirements evolve (e.g. for contract control purposes) and compute project performance parameters (e.g. productivity, speed of delivery and defect density) when the project is delivered

…. then you would expect the method to be widely used.

Such a process is possible in the world of business application software. 'Function point' methods can be used to size requirements for steps (a) and (c) and both public and proprietary methods and tools exist to support step (b). But though possible, relatively few organizations use such a process nowadays, for reasons we will discuss. And in the world of real-time software, any such process has been pretty crude until recently.

In this paper we will describe how the COSMIC-FFP method can be used for sizing 'real-time' software requirements, and hence as the key ingredient for project estimating and project performance measurement. Under 'real-time' software we include process control, embedded, telecoms, infrastructure (i.e. operating system) software and such-like.

1. Function Point Analysis in the world of Business Application software

Allan Albrecht's original 'Function Point Analysis' method [1] is maintained and enhanced nowadays by the International Function Point User Group (IFPUG) [2]. Together with a number of variants, such sizing methods are collectively known as '1st Generation' Functional Size Measurement (FSM) methods.

My guess is that 1G FSM methods were most widely used around 1990 but as the years pass it has become increasingly difficult to use them, even in the world of business application software. There are two main reasons that are relevant to this paper.

- The concepts on which the methods are based are naturally those that were in use in the 1970's and 1980's. Consequently, it is difficult for a project

[1] Charles Symons is an independent consultant based in the UK. He leads COSMIC (the Common Software Measurement International Consortium) jointly with Professor Alain Abran of the Ècole de Technologie Supérieure, Université du Québec, Montréal, Canada.

manager whose requirements may be expressed in a language such as UML to apply the sizing method concepts

- Even in the world of business application software many projects nowadays (e.g. web-based software) involve developing or enhancing infrastructure software for which these 1G FSM methods are difficult to apply

As a consequence 1G FSM methods are still used for benchmarking and estimating (see Peter Hill article) for bespoke software developments in the business application domain and notably where it is imperative to control price/performance in, for example, large outsourcing contracts (see Terry Wright article). However, these methods are not used to anything like their real potential. Indeed, it is a surprise to find any software engineering method that is still in use almost 30 years after it was first conceived – which is both a remarkable tribute to Allan Albrecht's lateral thinking and a sad comment on the conservatism of the software metrics community.

The situation for real-time software is even more difficult. Since FSM methods have not so far worked for real-time software, step (a) of the above process has relied on estimating SLOC as the size measure. Many large real-time software developing organizations have therefore built up large databases including SLOC size measures for their projects. They perform the estimating step (b) either by formulae or by searching the database for analogues of the new project. However, estimating size in units of SLOC at the requirements stage is obviously not ideal. (The one great advantage of SLOC, namely that they can be counted automatically and accurately for step (c) on project completion is, however, significant.)

2. The background to COSMIC and its Aims

With this background in mind, a group of software metrics experts gathered in late 1998 to form 'COSMIC', the Common Software Measurement International Consortium. Most of the founding members (including this author) were members of an ISO Working Group (ISO/IEC JTC1 SC7 WG12) on Functional Size Measurement. WG12 had laboured for several years to try and agree some basic principles for FSM [3], but progress was slow. It became clear to some WG12 members that if ever we were to seriously improve on 1G FSM methods, this would have to be undertaken as a private initiative, outside the formal ISO processes.

An example of the sort of difficulty in this subject that WG12 faced is that there is still no definition of 'functional'.

'Functional size' is defined [3] as 'a size of the software derived by quantifying the Functional User Requirements', where the latter are defined as 'a sub-set of the users' requirements …. (they) represent the user practices and procedures that the software must perform to fulfil the users' needs. They exclude Quality Requirements and any Technical Requirements'. This definition, with hindsight, has a strong flavour of the business application software world.

For our purposes, think of 'functional size' as a size measure concerned purely with requirements for information processing, thus excluding any requirements concerned with quality, technical or implementation factors. The latter must be taken into account when estimating project effort, but do not contribute to functional size.

When COSMIC was established in late 1998, it set itself a clear initial aim, namely 'to develop, test, bring to market and seek acceptance of new software sizing methods to support estimating and performance measurement'. The methods should:

- measure a 'functional size' of software
- be applicable to business application and real-time software and hybrids of these
- be applicable for software in any layer of a multi-layer architecture
- be applicable at any time in a software project life-cycle from early requirements gathering through to post-implementation, and for maintenance and enhancement activities

Today, COSMIC is still a voluntary, 'open' organization, but now with representatives from 15 countries [4] on its International Advisory Council. The functional sizing method that has been developed, COSMIC-FFP[2], is stable; an International Standard for COSMIC-FFP [5] was published in 2003. And the method is rapidly gaining acceptance in the market place (see side panel for examples of two current users of the method in the real-time domain).

3. Overview of the COSMIC-FFP method

Space does not allow a full description of the method in this article, so here we will give only a summary of the key concepts with some elaboration of particularly important concepts for those working in the real-time software domain. For the full definition of the method, including definitions of all the main concepts, see the 'Measurement Manual'. This is freely available for download [6] in several languages.

The COSMIC-FFP Measurement Process consists of three main phases:

a) Setting the Measurement Strategy
b) Mapping the 'Functional User Requirements' (or 'FUR') of the software to be measured to the COSMIC-FFP concepts
c) Measuring the resulting COSMIC-FFP model of the FUR

Phase 1: Setting the Measurement Strategy

This phase consists of first establishing the Purpose and Scope of the measurement. The Purpose is important because it affects the accuracy that will be required of the measurement, the timing of the measurement in the software life-cycle, the software artefacts (statements of requirements, physical screens, etc) that must be used, and so on. The measurement Scope determines what functionality is included in any one measurement. It might be, for example, that if the Purpose is to support estimating for a distributed software system and if the various components are planned to be implemented on different types of technical platforms, then the FUR must be separated into a number of separate measurement Scopes, one for each platform.

The Purpose and Scope together determine the 'Measurement Viewpoint' that will be adopted. This is a vital, new concept for functional size measurement; understanding it will explain why we refer to COSMIC-FFP as a '2G' FSM method.

[2] 'FFP' stands for 'Full Function Points'

The Measurement Viewpoint defines the form of abstraction that is used for the FUR and which will be used for the measurement[3]. In this short article, it is best described by examples.

- In the domain of real-time (for example, embedded) software, the FUR are normally described in terms of the functionality that must be provided to the 'direct users' of the software. These 'direct users' are typically hardware devices, e.g. sensors, buttons, display panels, valves etc., and also other pieces of software that interact directly with the software to be measured. They are called 'direct' users because there is no software between the users and the software to be measured. We call this abstraction the 'Direct User' measurement viewpoint[4].

- In the domain of business application software, the FUR are normally described in terms of the functionality that must be provided to the human users of the software. All hardware (screens, keyboards, printers, etc) and software (operating system layers) between the human user and the application software to be measured is 'invisible' in this abstraction and is ignored. We call this the 'End User' measurement viewpoint

The COSMIC-FFP method can be applied to measure a functional size using either of these two standard measurement viewpoints, and the resulting sizes will obviously differ, since different functionality is revealed in each abstraction.

In the above examples, we used the word 'normally', because there is no absolute connection between the software domain and the choice of measurement viewpoint. It all depends on the Purpose of the measurement. If, for example, our Purpose it to support development effort estimation for some embedded avionics software, then we would almost certainly want to measure using the Direct User measurement viewpoint.

Alternatively, the total aircraft control system could be measured from the End User measurement viewpoint, where the end users might be the aircrew. Such a measurement could be relevant in the context of some aircraft simulator training software, but it would be useless for the developer of any embedded avionics software.

Introducing the concept of different standard measurement viewpoints has been a vital break-though for FSM. We now understand that 1G FSM methods were designed to work only in the End User measurement viewpoint – though this was not recognised when the methods were originally developed. It explains why 1G FSM methods cannot be easily applied to measure real-time software. For most practical purposes, the functionality of real-time software needs to be measured from the viewpoint of all of its 'direct' users.

---

[3] In ISO/IEC 19761:2003, there is a non-normative Note which states that 'The Purpose and Scope of a specific FSM when combined are referred to as the 'Viewpoint'. Subsequently, it was realized that 'Measurement Viewpoint' is a separate concept and that we should build upon the existing standard ISO/IEC 10746-2, which include the definitions of 'abstraction' and 'viewpoint'

[4] In the current v2.2 of the Measurement Manual [6], this is known as the 'Developer' measurement viewpoint. The replacement, 'Direct User' name is introduced in v2.3 of the Measurement Manual, due to be published in mid-2006

Phase 2: Mapping to the COSMIC-FFP conceptual models

The COSMIC-FFP method relies on a number of carefully defined and inter-related concepts (shown in inverted commas below, when first mentioned). The Measurer's task in this phase is, in effect, to extract a model of the software in terms of these COSMIC-FFP concepts from whatever software artefacts are available, in light of the chosen measurement viewpoint.

The principle COSMIC-FFP concepts are as follows.

- Software is structured in 'layers' (COSMIC-FFP provides rules for distinguishing layers for cases where the software to be measured does not exist in an established layered architecture; the Scope of any one measurement must not encompass software in more than one layer)
- Software is triggered to do something by 'events' in the world of its 'users' (which will be hardware devices and/or other items of software, in the direct user measurement viewpoint)
- Each unique event-type triggers a unique 'functional process' in the software that is complete when it has performed whatever is required to respond to the triggering event
- Any functional process comprises 'sub-processes' that either move or manipulate data (see Fig. 1)
- A 'data movement' is the movement of a single 'data group' (of one or more 'data attributes') about a single 'object of interest' (i.e. 'of interest' in the world of the users)
- There are four types of data movements. 'Entry' and 'Exit' data movements move data across a 'boundary', from or to a user respectively. 'Read' and 'Write' data movements move a single data group from or to 'persistent storage' respectively (where 'persistent' means that the data survives the functional process that stores it).

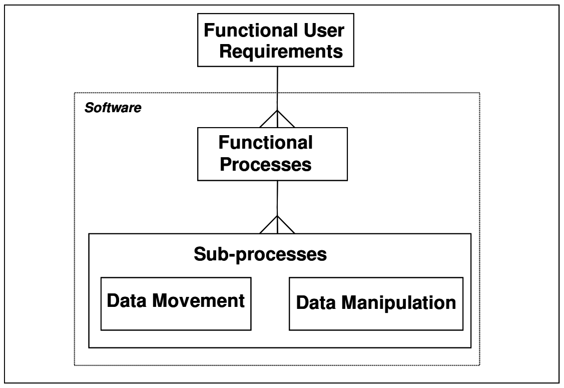

Figure 1. The COSMIC-FFP Generic Software Model

Phase 3: The Measurement Phase

In this phase we simply count the number of data movements identified in each functional process of the software to be measured, as the measure of its functional size. Each Entry, Exit, Read and Write is counted as one 'COSMIC functional size unit' or 'Cfsu'.

Any one functional process must have a minimum size of 2 Cfsu, since to be meaningful it must have a 'triggering Entry' plus a successful outcome, either one Exit or one Write. But there is no upper limit to the size of any one functional process (unlike the upper limit to the size of an IFPUG 'elementary process' which may not exceed 7 function points). In some individual military avionics functional processes, we have measured single functional processes with over 100 data movements (i.e. of size > 100 Cfsu).

The method can also be used to measure the size of software changes for maintenance or enhancement activities. The size of a changed functional process is defined as the sum of the number of added, modified and deleted data movements.

But what about data manipulation sub-processes? All software includes some data manipulation sub-processes; why not identify and count them?

The answer is simply that no-one agrees how to measure the functional size of an algorithm, so we cannot measure data manipulations on any scale that would be widely acceptable. We therefore make an approximation and assume that the functional size of an item of software is proportional to the number of its data movements, and assume each data movement includes certain associated data manipulation. For example, an Entry is defined to include all manipulations associated with input data validation that do not involve another data movement.

We can now see why the COSMIC-FFP method is claimed to be applicable only for business application and real-time software, that is, for 'movement-rich' software. We cannot measure 'algorithm-rich' software involving heavy mathematical manipulations such as used in weather forecasting, stress analysis and such-like. And by the way, it is arguable that this omission is not so important. It is much more important and difficult to estimate the mathematical or engineering effort involved in developing an algorithm in the first place, essentially a creative activity, than to estimate the effort to encode it in software.

But what if the software you wish to measure is mostly 'movement–rich' but includes some significant, localized mathematical algorithms? The COSMIC-FFP method allows for a 'local extension' whereby an organization can define its own local scale for sizing algorithms alongside the Cfsu scale for sizing data movements. Alternatively, if your Purpose is estimating project effort, the measurement Scope can be defined to exclude the algorithms. The sizing and estimating process can then be applied only to the 'movement-rich' functionality whilst estimating for the algorithms can be dealt with by another appropriate process.
.

4. A Simple Example

Applying the COSMIC-FFP method to size real-time software in the direct user measurement viewpoint is normally remarkably simple. Although today the measurement process is manual, there is reasonable hope for automation of the sizing if the FUR become available in a CASE tool [7].

Some simple, fully documented, 'real' case studies of real-time software are available for free download [8]. For the purposes of this article, I will illustrate the measurement process with a simple example from my house, which has a basic intruder alarm system. Until I started this article I had never considered how it might work, but having looked at it, I believe that a diagram showing the embedded software and its (direct) users must look something like the following.

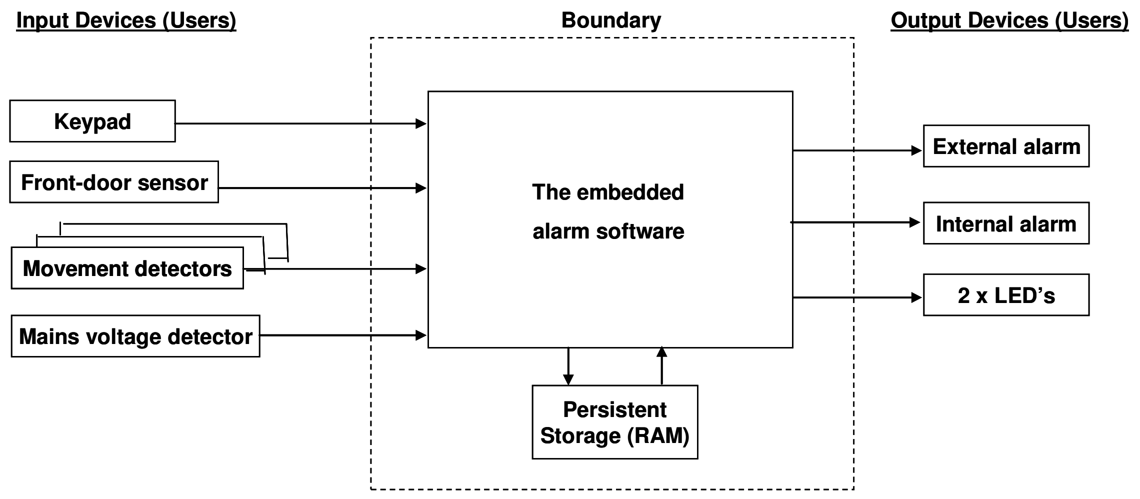

Figure 2. The simple intruder alarm system

The system has a conventional keypad and red/green LED's for the human interface. It is connected to a device that senses whether the main, front door is open or not, several internal movement detectors, and an internal and an external alarm. There is a battery to take over if the mains power supply fails, so there must be a mains voltage detector. Finally our PIN code is stored in the system and can be changed, so there must be some persistent storage hardware, which no doubt also contains various system parameters.

There must be an internal clock mechanism, since certain functions must be completed within pre-set times. For example having set the 'exit code' before leaving the house, the front door must be closed within a given time, or alarms sound. I assume the internal clock is started in the software each time it is needed, so this functionality is a form of data manipulation, which we can ignore.

I have no idea whether the software periodically polls its input devices or whether the devices send signals to the software whenever they detect something. But this difference has no effect on how we map the software functionality to the COSMIC-FFP Generic Software Model.

The following appear to me to be the unique types of events that the alarm can detect, each of which triggers a unique functional process. There are probably more event-types, known only to the maintenance engineer.

- A new or changed PIN code is entered via the keypad (2 events)
- The 'exit code' is entered via the keypad thereby activating the system (before leaving and closing the front door within a pre-set time)
- The front door sensor detects that it is open whilst the alarm system is activated
- The system is activated, or de-activated, whilst in the house (2 events)

- A movement detector signals a movement whilst the alarm system is activated
- The mains voltage detector signals mains electrical failure, or its restoration (2 events)

Now let's examine, as an example, the functional process that is triggered when I return to the house, with the alarm activated. When I unlock and open the front door, the internal alarm starts 'beeping', and I must then enter our PIN code within a pre-set time to de-activate the system and stop the internal alarm ringing. If I do not enter the PIN code before the pre-set time, or I enter the wrong code more than three times, the external alarm also starts ringing.

This functional process must have the following data movements.

Action	**Data Mvt.**	**Comment**
Sensor detects front door open	Entry	This is the 'triggering Entry' that starts the functional process
System retrieves the current PIN code from persistent storage	Read	The current PIN code must be held in a persistent data group in RAM
System starts its internal timer(s)	-	This is data manipulation. Ignore
System starts the internal alarm to 'beep'	Exit	
I enter my PIN code and the system compares it against the stored PIN	Entry	The validation of the PIN code is data manipulation, considered to be associated with the Entry
Suppose my first attempt has an error. I enter the PIN code again	-	This is a repeat occurrence of the previous Entry. We count data movement *types*, not *occurrences.* So ignore
System stops the internal alarm beeps on successful entry of PIN	Exit	
System switches LED from red to green on successful entry of PIN	Exit	(I don't know how the LED's work; it might require 2 Exits to switch one 'off' and the other 'on'). The functional process stops here on successful PIN entry and the system is de-activated
System starts internal and external alarms to 'wail' on unsuccessful entry of PIN	2 x Exits	
System stops external alarm after 20 minutes (a legal requirement)	Exit	The functional process stops here on unsuccessful PIN entry. (I don't know what happens then!)

The total functional size of this functional process is therefore 9 Cfsu.

Note that the analysis of this functional process is very simple. I do not need to understand the detailed logic of, or the precise sequence of steps of, the process. The size measured accounts for all the data movements arising from all possible paths through the process; the size does not depend on the input values to the process which may determine any particular actual path.

The example illustrates that any software engineer with a clear understanding of the COSMIC-FFP concepts should be able to apply the measurement process to any real-time software requirements, or to the functionality of an existing software system with which they are familiar.

But this example may now raise another objection to the method. Applying the COSMIC-FFP method as described above requires a quite detailed statement of requirements. In a real project, you may well need to estimate a size well before you have the requirements in such detail.

The Measurement Manual [6] therefore gives guidance on how to construct your own local, approximation-variant of the method that can be used when the requirements are known only at some 'higher level', that is, in less detail. See [14] for an example of an approach to early sizing.

5. COSMIC-FFP for estimating project effort and duration

As of today, if you have mastered step (a) as described above and can size your new or changed functional user requirements, there are three possible approaches to the estimating step (b).

The first approach is to simply re-calibrate your current estimating method by substituting sizes measured in Cfsu for the sizes currently used (most likely estimates of Source Lines of Code?). This will require measuring the functional sizes of several existing software products for which the corresponding effort and duration are known. There is really no way of avoiding this step, but measuring some existing software is essential anyway to gain experience and build confidence in the sizing method.

Several academic papers, e.g. [9] and [10], have been published showing results for this approach, including for maintenance and enhancement projects, using the COCOMO model and fuzzy logic. See [11] for an example of a tool to support this process.

A second approach, if you have no established estimating method, is to measure several existing software products (as above) from a domain where there is a high degree of project commonality (e.g. same technologies used, similar time/effort constraints, similar project team skill levels, similar risk, etc.) and to carry out your own statistical analyses so as to develop formulae to predict effort and duration from software size.

Finally, it is possible to measure the functional size of a new requirement and to use ISBSG benchmark data (see Peter Hill article) to produce a first, ball-park estimate of effort. As of early 2006, there are data on around 95 projects measured using COSMIC-FFP in the ISBSG database and a report is available [12] from ISBSG with an analysis of the data. Of course, not many of these projects are yet from the real-time defence software community, but the number of projects submitted to the ISBSG database is growing and a special effort is being made in 2006 to gather more such data. This approach will become much more valuable in time, especially to provide sanity checks on early estimates.

6. Where are we today with the COSMIC-FFP method?

Over the five years since the first detailed account of the COSMIC-FFP method was published, a lot of practical experience gained and a lot has been learned.

The most gratifying aspect of this experience is that the underlying software models have been found to be robust when applied to an enormous variety of types of software. I believe the greatest lesson we have learned has been to recognise the importance of defining the measurement viewpoint, or abstraction, to be used for a measurement. This breakthrough has enabled COSMIC-FFP to be applied successfully to measure a credible functional size of real-time software for the first time, whilst continuing to be applicable to business application software.

The method is stable and standardized. As we learn how to improve our explanation of the method, we continue to refine the Measurement Manual, but the underlying concepts are solid.

The method is 100% 'open'. The Measurement Manual is available in four languages and translations are scheduled or underway in another three.

The method has been very widely applied in both the business and real-time software domains. In the latter, I am personally aware of its successful application to size software in the fields of avionics, telecommunications, process control, paper copiers, digital cameras, vehicle engine management systems, and scientific instruments. Geographically, there is now substantial experience in Asia, Australasia, Europe and North America.

Various informal tests have been carried out on the repeatability of measurements by different measurers working from the same specification. The results are both encouraging and predictable. If the specifications are clear, the measured sizes are highly repeatable. If there is ambiguity in the specifications, the results will not be repeatable.

This leads on to one of our most intriguing and widespread findings, namely that applying the COSMIC-FFP sizing method is an excellent method for quality control of software requirements. If they can't be sized, the software cannot be built without the developers having to make some assumptions. Furthermore, if you have a mapping of the functional requirements to the COSMIC-FFP conceptual models you have, in the catalogue of functional processes, an excellent foundation for requirements traceability and test case design.

The necessary infrastructure to support world-wide use of the method is now becoming well-established. This includes benchmark data, organizations that can provide training and consulting services, automated tools to support data collection and estimating (see [4] for more details), certification examinations [13], etc.

There is now a steady stream of papers at software metrics conferences demonstrating the growing interest by academics in research using COSMIC-FFP based measurements.

And the future holds some interesting possibilities. Why not use the COSMIC-FFP method to size information processing functionality before the hardware/software allocation is decided? We can see no reason why not, but it needs testing. Why not size software functional requirements automatically if they are held in a CASE tool? – another topic already being explored [7]

7. Conclusion

The COSMIC-FFP method is a simple, proven, practical method of functional sizing for real-time software and is therefore a key ingredient for estimating methods and performance measurement.

Now for a plea. If your organisation decides to adopt the COSMIC-FFP sizing method, please submit as much of your data as possible to the ISBSG database. It's a real-chicken-and-egg challenge, which you can help us solve. The more benchmark data that is made available, the more valuable the method will become, the more data that will be generated, and so on. COSMIC has laid the foundations. Now the method's users can add the greatest value.

And finally, remember it is an open method. Our documentation and support can always be improved. If you have suggestions or contributions for improvement, e-mail me at mpc-chair@cosmicon.com.

Sample users of COSMIC-FFP for real-time software: (Text to be inserted in a side panel)

- Nokia first used the COSMIC-FFP method in the worldwide field trials in 2000. Since then its use has gradually been extended among different R&D units within Nokia Networks for managing the development of real-time and embedded software in various network elements. Some R&D units have implemented the method as standard practice for their effort estimation. It is also used as one vehicle for software development productivity measurement and monitoring. The results for improving estimation accuracy by means of COSMIC-FFP based size measurement are very encouraging, especially the larger an estimated item, the better the result that is achieved.

  (Nokia Networks is a business group of Nokia Corporation providing network infrastructure, communications and networks service platforms, as well as professional services to operators and service providers.)

- The software department of the Perkin-Elmer R&D Centre for Process Analytics and Control Technology in the UK has used COSMIC-FFP for around 2 years to provide early estimates of the size of new software projects

References

[1] A.J. Albrecht, 'Measuring application development productivity' in Proc. IBM Applications Development Symposium. GUIDE International and Share Inc., IBM Corp., Monterey, CA Oct 14-17, 1979, p83

[2] 'Function Point Counting Practices Manual, Release 4.2', the International Function Point Users Group, 2004

[3] ISO/IEC 14143:1998 'Software Engineering – Software Measurement – Functional Size Measurement, Part 1: Definition of Concepts, International Organization for Standardization, Geneva, Switzerland, 1998.

[4] See www.cosmicon.com

[5] ISO/IEC 19761:2003, 'Software Engineering- COSMIC-FFP – A functional size measurement method', International Organization for Standardization, Geneva, Switzerland, 2003.

'[6] The COSMIC Implementation Guide to ISO/IEC 19761:2003' (the 'Measurement Manual'), version 2.2, obtainable from www.gelog.etsmtl.ca/cosmic-ffp.

[7] Azzouz, S.; Abran, A., (2004), 'A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC-FFP', proceedings of Software Measurement European Forum - SMEF 2004, Rome, Italy.

[8] See www.gelog.etsmtl.ca/cosmic-ffp

[9] Ali Idri, Alain Abran, 'COCOMO cost model using fuzzy logic', 7th International conference on fuzzy logic theory and technology, Atlantic City, NJ, February 27th 2000

[10] Gu Xunmei, Song Guoxin, Xiao Lizhong, 'Estimating with COSMIC-FFP functional size measurement using fuzzy sets', 3rd SMEF 2006, Rome, Italy, May 10 – 12th, 2006

[11] See http://www.telmaco.com/pdmetco15e.pdf

[12] 'An analysis of software projects using COSMIC-FFP', the International Software Benchmarking Standards Group, January 2004

[13] 3rd Software Measurement European Forum 2006, Rome, May 10 -12th, 2006

[14] Conte,M.; Iorio,T.; Meli, R.; Santillo, L., (2004), E&Q: An Early & Quick Approach to Functional Size Measurement Methods, in Software Measurement European Forum (SMEF), Rome, Italy.