Functional Size Measurement Methods - COSMIC-FFP: Design and Field Trials

Abran A., Oligny, S., Symons C., St-Pierre D., Desharnais J.M.
(on behalf of the COSMIC Core Team *)

Abstract

Measuring the functional size of software was proposed by Albrecht, more than 20 years ago, as a solution to the limitations of lines of code (SLOC) when quantifying the output of the software engineering process. While this approach has been, and still is, successful when applied to MIS type of software, it has not enjoyed the same success for measuring the size of non MIS software, as demonstrated in publications from a number of authors in the past 15 years. Three types of approach have been proposed in the literature to apply Albrecht's concepts to non MIS types of software. Although some of these approaches offered interesting insights, none has gained sufficiently wide usage to be recognized as a de facto standard.

Building on the strengths of previous work in this field, the Common Software Measurement International Consortium (COSMIC) proposed a set of principles in 1998 onto which a new generation of functional size measurement methods could be built. The COSMIC group then published version 2.0 of COSMIC-FFP, in 1999, an example of a functional size measurement method built on those principles. Key concepts of its design and of the structure of its measurement process are presented.

1. CONTEXT

With the advent of large scale software development in the mid-sixties, the software engineering community recognized the need to better control the economic aspects of software production and maintenance. One key aspect of this endeavor was to figure out a way of quantifying software as the output of a complex process, in order to be able to better understand and manage the multiple facets of its production. Source lines of code was the first generally accepted measure for this purpose and was used extensively, as demonstrated by the many estimation models that included this measure as a key parameter [1, 2].

As a measure of software size though, source code measure carries some inherent limitations, and this was soon recognized by software engineering practitioners and researchers [3]. Among the practitioners, Alan Albrecht was the first to propose, over 20 years ago, a new way of quantifying software size, based on the users view of the software [4]. Albrecht's method is still used today and provides useful results in many organizations, but it also bears some limitations, which have been documented over the past 15 years.

The purpose of this paper is to summarize the evolution of the state of the art in the field of software functional size measurement and to present an example of a new generation of software functional size measure: the COSMIC-FFP measurement method.

* Alain Abran, Moritsugu Araki ,Jean-Marc Desharnais, Reiner Dumke, Peter Fagg, Roberto Meli, Pam Morris, Serge Oligny, Jolijn Onvlee, Risto Nevalainen, Grant Rule, Denis St Pierre, Charles Symons.

2. ORIGIN OF SOFTWARE FUNCTIONAL SIZE MEASUREMENT

Measuring the functional size of software was originally proposed in 1979 by Albrecht in a communication [4] describing the results of an effort started in the mid-seventies in one IBM business unit. The overall goal of the work described in Albrecht's original paper was to measure the productivity of software development, as viewed from an economic perspective. The functional size of software was proposed as a generic measure of the "output" of the development process which allows a comparison of projects in which software was developed using different programming languages.

The lasting merit of Albrecht's proposal was to offer a new approach for quantifying the size of software. At a time when "software size" meant "lines of code" for the majority of software engineering practitioners, many of whom wrestle with problems linked to variation in size due to the use of different programming languages, Albrecht proposed a method of sizing software which did not depend on the programming language used.

The overall approach described in Albrecht's paper to achieve that goal was to select a specific subset of 22 software projects, mostly MIS software, completed within this specific organization. The measurement of the functional size of the software delivered by these projects consisted of a weighted sum of "inputs", "outputs", "inquiries" and "master files". The weights assigned to these items "were determined by debate and trial" [4]. Some extra adjustments (+/- 25%) were provided for "extra complicated" [4] items. In this communication [4], Albrecht also offered a set of forms and rules to aid in calculating "function points". Many of the rules proposed originally were based on some aspects of the physical implementation of software.

In 1984, the International Function Point Users Group (IFPUG) was formed to foster and promote the evolution of the function point method. The group used Albrecht's revised version of the original method, using a fifth function type ("interface files") and a set of weight tables. Much work went into the subsequent releases of the method to include rules allowing an interpretation of functionality increasingly independent of the particular physical implementation of software. The contribution of IFPUG to the field of functional size measurement has been the documentation of the measurement procedure, which enabled a certain level of uniformity in the application of the method. The basic concepts and implicit model of software, though, remained unchanged from what was proposed by Albrecht in 1984.

The approach proposed by Albrecht can also be characterized by its empirical nature. The scope of the problem he tackled was, in his own words, defined by the clients of one service at IBM, and the software he selected was governed by some specific criteria (described in [4]).

Although this approach proved quite useful for Albrecht at the time when he had to solve this problem, it can hardly be postulated that this sample of software, developed between 1974 and early 1979, is representative of all software developed in the '80s and '90s. More specifically, the weights assigned to software function types, those determined by "debate and trial" along with the very nature of those function types, have been the object of much debate during the past 20 years.

* Alain Abran, Moritsugu Araki ,Jean-Marc Desharnais, Reiner Dumke, Peter Fagg, Roberto Meli, Pam Morris, Serge Oligny, Jolijn Onvlee, Risto Nevalainen, Grant Rule, Denis St Pierre, Charles Symons.

3. ATTEMPTS AT IMPLEMENTING FUNCTION POINTS OUTSIDE MIS

Albrecht's approach was criticized by other practitioners when they tried to apply it for measuring the functional size of other types of software, notably real-time software[3, 5, 6, 7]. This led some software engineering practitioners to try to improve this functional size method so that it could be applied to non MIS software. In doing so, an issue must be considered:

a) does the method adequately capture the functional size of the type of software to which it is applied ?

Given software to measure and a method to measure it, a measurement method will produce a numerical figure. But this numerical figure has limited practical meaning without investigating, at the same time, the answer to question a) above, because the meaning and usefulness of this numerical figure lies in the answer provided to this question.

A review of the literature [8] shows that three different approaches have been used in trying to use Albrecht's work to measure non MIS software: a) status quo, b) attempts to approximate the size, and c) addition of new function types. Examples of each are presented next.

3.1. Status quo

IFPUG: This approach assumes that Albrecht's function point method is indeed applicable "as is" to non MIS software. The assumption is that, when applied "as is" to non MIS software, Albrecht's method produces a numerical figure and that this figure adequately represents the functional size of the non MIS software. It is the approach adopted by IFPUG; however many practitioners [3, 5, 6, 7, 9, 10, 11, 12, 13, 14] have indicated that, in regard to question a) above, "status quo" method applied "as is" to non MIS software does not adequately capture the functional size of these software.

3.2. Approximating size

Application Features (1992): Mukhopadhyay and Kerke [7] proposed the concept of Applications Features. They were not directly interested in redefining function point as a measurement method, but were rather looking for a way to estimate functional size, in function points, earlier in the development life cycle. Their goal was to apply the technique to process control software in the manufacturing domain (a category of real-time software). This is interesting because it identifies specific characteristics of a certain category of real-time software which contribute to functional size and which are known very early in the development life cycle. However, these characteristics (physical positioning and movement) are specific to a specialized application domain and cannot be generalized across all types of real-time software. Furthermore, since the authors have concentrated specifically on the presentation and analysis of a functional size estimation model in their paper, the measurement process itself is not covered. Detailed procedures and rules, to ensure precision and consistency across time in the measurement of the three application features identified, are not provided.

3.3. Addition of new function types

Feature Points (1986): Feature Points were developed by Capers Jones in 1986 [3]. According to Jones, real-time software is high in algorithmic complexity but sparse in inputs and outputs, and, when the function point method is applied to such software, the results generated appear to be misleading. However, it was pointed out [5] that the definition of algorithms, and the rules proposed in [3], are not sufficient for measurement purposes: "No guidance is provided to identify algorithms at the desired level of abstraction." [5].

Asset-R (1990): According to Reifer [6], Albrecht's method failed to successfully handle four important characteristics associated with real-time and scientific software: parallelism, synchronization, concurrency and intensive mathematical calculation. New function types were therefore introduced, according to the type of software measured. The weighting factors and the adjustment factors proposed in Albrecht's method were eliminated. However, there is a lack of detailed definitions, measurement rules and examples for the new function types (modes, stimulus/response relationships, rendezvous) and it is not clear how to measure it.

3D Function Points (1992): Whitmire [5] developed *"3D Function Points"* as a result of a study investigating the use of function points in the measurement of scientific and real-time software. According to this study, function point does not accurately measure this type of software because it does not factor in all the problems and characteristics that contribute to the size of the software. This criticism is based on the premise that all software has three dimensions: data (stored data and user interfaces), functions (internal processing) and control (dynamic behavior). According to this scheme, function point measures only one dimension: the data dimension. Whitmire therefore proposed a set of function types for each of the dimensions he identified and a mechanism to combine the points assigned into a single "3D FP index".

Full Function Points, version 1.0 (1997): Based on a review of the literature [8, 15] and using a formal research framework [16], Full Function Point was proposed in 1997 [17]. Version 1.0 of Full Function Points introduced six new function types and combined them with the five already proposed by traditional function points to measure the functional size of MIS software, real-time software or a combination of both types of software. These new function types were defined based on an analysis of the strengths and weaknesses of previous attempts at enhancing traditional function points [8].

Industrial usage of Full Function Points (version 1.0) revealed that the six new function types alone were sufficient, in the opinion of software engineering practitioners, to adequately measure the functional size of both MIS and real-time software [18]. The method was used in various industrial contexts (telecomunication, automobile industry embedded software, system software) in a number of organizations [8, 15, 18, 19, 20, 21, 22].

In 1998 version 1.0 of Full Function Points was recognized as a standard for measuring the functional size of software by the International Software Benchmarking Standards Group (ISBSG) and software projects measured with this method were therefore accepted and included in the international benchmarking repository. The detailed rules of version 1.0 of Full Function Points were documented and have been made available in the public domain [17].

4. A NEW GENERATION OF FUNCTIONAL SIZE MEASURE

A group of experienced software measurers gathered, in 1998, to form the Common Software Measurement International Consortium (COSMIC). This group aimed at designing and bringing to market a new generation of software measurement methods. With the support of industrial partners and tapping on the strengths of IFPUG, MarkII [23], NESMA [24] and version 1.0 of Full Function Point methods, the group proposed some basic principles on which a new generation of software functional size measurement method could be based [25, 26, 27]. These principles are summarized in section 4.1. In November of 1999, the group published version 2.0 of COSMIC-FFP, a measurement method implementing these principles. Overall, close to 40 people from 8 countries participated in the design of this measurement method. Key aspects of the COSMIC-FFP measurement method are highlighted in sections 4.2 to 4.6

4.1. COSMIC principles

4.1.1. Allocation of functional user requirements

From the perspective proposed by COSMIC, software is part of a product or service designed to satisfy functional user requirements. From this high-level perspective, functional user requirements can be allocated to hardware, to software or to a combination of both.

The functional user requirements allocated to software are not necessarily allocated to a single unit of software. Often these requirements are allocated to pieces of software operating at different levels of abstraction and cooperating to supply the required functionality to the product or service in which they are included. This is illustrated in Figure 1.

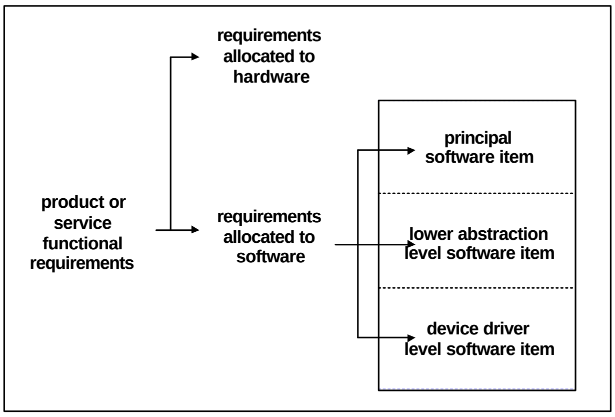
requirements allocated to hardware
product or service functional requirements
requirements allocated to software
principal software item
lower abstraction level software item
device driver level software item

Figure 1 – Allocation of functional user requirements, adapted from [25]

In the context of the COSMIC-FFP measurement method, which is aimed at measuring the functional size of software, only those functional user requirements allocated to software are considered. For instance, as illustrated in Figure 2, the functional user requirements in this example are allocated to three distinct pieces, each exchanging data with another through a specific organization: one piece of the software lies at the application level and exchanges data with the software’s users and with a second piece lying at the operating system level. In turn, this second piece of the software exchanges data with a third piece lying at the device driver level. This last piece then exchanges data directly with the hardware. The COSMIC-FFP

* Alain Abran, Moritsugu Araki ,Jean-Marc Desharnais, Reiner Dumke, Peter Fagg, Roberto Meli, Pam Morris, Serge Oligny, Jolijn Onvlee, Risto Nevalainen, Grant Rule, Denis St Pierre, Charles Symons.

measurement method associates each level with a specific layer. Each layer possesses an intrinsic boundary for which specific users are identified. The functional size of the software described through the functional user requirements is thus broken down into three pieces, each piece receiving parts of the functional user requirements.

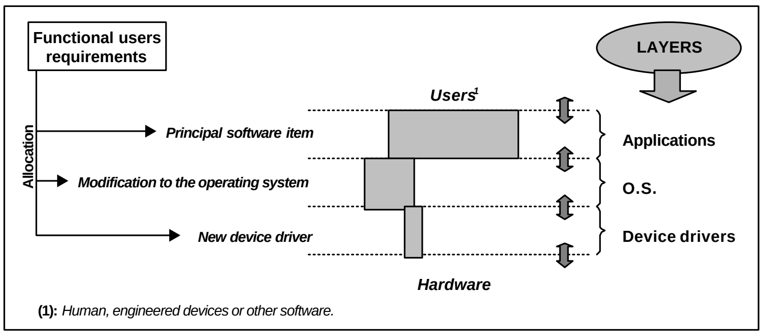
Functional users requirements
LAYERS
Users[1]
Allocation
Principal software item
Modification to the operating system
New device driver
Applications
O.S.
Device drivers
Hardware
(1): Human, engineered devices or other software.

Figure 2 – Example of functional user requirement allocation to different layers [28]

4.1.2. Representation of functional user requirements in software

The functional user requirements in the subset allocated to one or more software pieces are represented by functional processes. Each functional user requirement is thus represented, within the piece of software to which it has been allocated, by one or more functional processes. In turn, each functional process is represented by sub-processes. A sub-process can either be a data movement type or a data transform type. Version 2.0 of the COSMIC-FFP measurement method recognizes only data movement type sub-processes. Further research is deemed necessary to incorporate data transform sub-process types in the measurement method. This approach is illustrated in Figure 3.

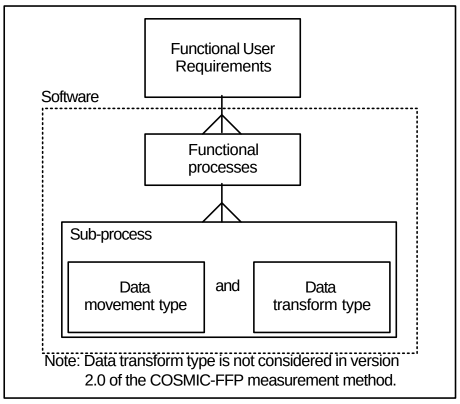
Functional User Requirements
Software
Functional processes
Sub-process
Data movement type
and
Data transform type
Note: Data transform type is not considered in version 2.0 of the COSMIC-FFP measurement method.

Figure 3 – COSMIC representation of functional user requirements within a piece of software [25]

4.2. COSMIC-FFP software model

A key aspect of the COSMIC-FFP measurement method is the explicit definition of the attributes of software relevant to the measurement of its functional size and their arrangement into a coherent, generic software model.

The COSMIC-FFP measurement method defines an explicit model of software functionality, derived from the functional user requirements. Based on this explicit model of functionality, relevant functional attributes of software are identified. Their extent and limits are defined and their generic interactions are described. Taken as a whole, these functional attributes form a generic model of software offering a universal basis for measuring its functional size. Furthermore, the COSMIC-FFP software model was designed to be compliant with the ISO-14143 standard [29] for software functional size measurement. The model is illustrated in Figure 4.

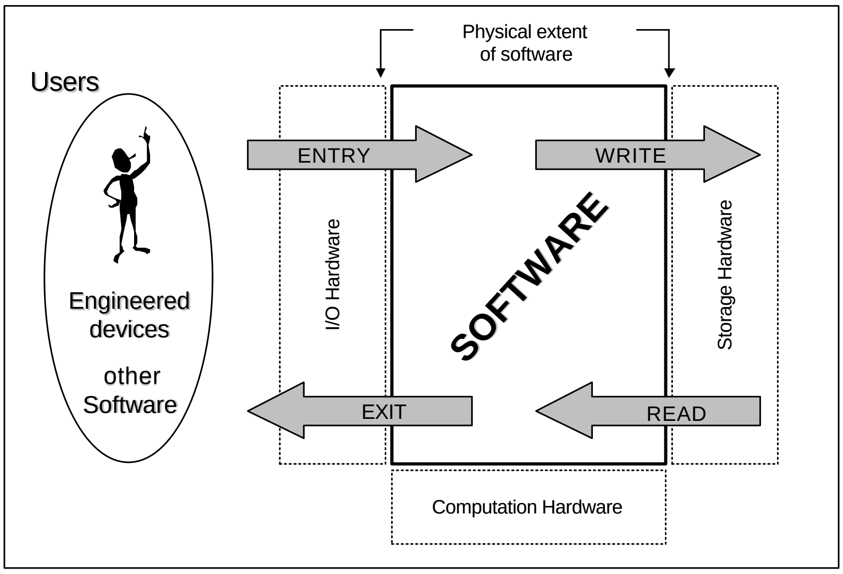

Figure 4 – COSMIC-FFP software model

Four types of data movement are defined within this model, and these form the basis for defining the standard unit of functional size. The four types of data movement are defined in Table 1 below.

4.3. COSMIC-FFP measurement system

Another key aspect of the COSMIC-FFP measurement method is the explicit definition of a measurement system, including a measurement principle, base functional components, a standard unit of measure and an aggregation function.

Measurement principle: based on the COSMIC-FFP software model, the method, through a defined set of rules and procedures, produces a numerical figure the magnitude of which is directly proportional to the functional size of this model. These rules are based on the principle that the functional size of a piece of software is directly proportional to the number of its data-moving sub-processes. By convention, this numerical figure is then extended to represent the functional size of the software itself.

Two elements characterize the measurement rules and procedures: the base functional components that constitute the arguments of the measurement function and the standard unit of measurement, which is the yardstick defining one unit of functional size (one COSMIC functional size unit or *CFSU*).

Data Movement Type	Definition
ENTRY	An ENTRY (E) is a movement of the data attributes found in one data group from the user side of the software boundary to the inside of the software boundary. An ENTRY (E) does not update the data it moves. Functionally, an ENTRY sub-process brings data lying on the user's side of the software boundary within reach of the functional process to which it belongs. Note also that in COSMIC FFP an entry is considered to include certain associated data manipulation (validation) sub-processes.
EXIT	An EXIT (X) is a movement of the data attributes found in one data group from inside the software boundary to the user side of the software boundary. An EXIT (X) does not read the data it moves. Functionally, an EXIT sub-process sends data lying inside the functional process to which it belongs (implicitly inside the software boundary) within reach of the user side of the boundary. Note also that in COSMIC FFP an exit is considered to include certain associated data manipulation sub-processes.
READ	A READ (R) refers to data attributes found in one data group. Functionally, a READ sub-process brings data from storage, within reach of the functional process to which it belongs. Note also that in COSMIC FFP a READ is considered to include certain associated data manipulation sub-processes.
WRITE	A WRITE (W) refers to data attributes found in one data group. Functionally, a WRITE sub-process sends data lying inside the functional process to which it belongs to storage. Note also that in COSMIC FFP a WRITE is considered to include certain associated data manipulation sub-processes.

Table 1 – Definition of COSMIC-FFP data movements [30]

Base functional components: version 2.0 of the COSMIC-FFP measurement method uses only four base functional components: entry, exit, read and write. Data manipulation sub-processes are not used as base functional components. The method assumes, as an acceptable approximation for many types of software, that the functionality of this type of sub-process is represented among the four types of sub-process defined earlier.

Standard unit of measurement: the standard unit of measurement, that is, 1 *CFSU*, is defined by convention as equivalent to one single data movement at the sub-process level. Another alternative, still being considered by the COSMIC group, is to define the standard unit of measurement based on the number of data elements moved by a data movement type sub-process. Such an alternative is included in the current work in progress.

Aggregation function: using the standard unit of measurement, base functional components are thus assigned size units. The functional size of the base functional components can then be combined to obtain the size of higher-level functional structures like functional processes or layers. This is performed, using an aggregation function, by arithmetically adding together the functional sizes of the constituent functional structures.

The measurement system proposed by the COSMIC-FFP measurement method offers a scalable result, which means that the functional size figure can be constructed at the desired level of abstraction.

4.4. COSMIC-FFP measurement process

Another key aspect of the COSMIC-FFP measurement method is the definition of a generic, two-phase, measurement process using the functional user requirements of the software to be measured in input.

According to the ISO-14143 standard [29], a functional size measurement method is based on the perspective provided by the functional user requirements of the software to be measured. In practice though, “functional user requirements” do not often exist in a “pure” form, as a stand-alone document. Therefore, very often, “functional user requirements” constitute a relatively abstract view of the software which needs to be extracted from other documents generated by the software engineering process.

Essentially, functional user requirements can be extracted from software engineering documents which are produced before the software exists (typically from architecture and design artifacts) or after the software has been created (typically from user documentation, physical programs and storage structure layouts). This is illustrated in Figure 5.

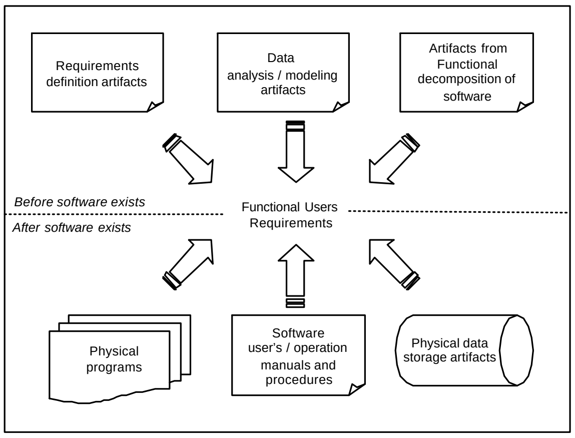
Requirements definition artifacts
Data analysis / modeling artifacts
Artifacts from Functional decomposition of software
Before software exists
Functional Users Requirements
After software exists
Physical programs
Software user's / operation manuals and procedures
Physical data storage artifacts

Figure 5 – Extracting functional user requirements for input to the COSMIC-FFP measurement process [30]

Thus, the functional size of software can be measured prior to its creation or after its creation. The effort to extract the functional user requirements will obviously vary depending on the quality of the documents used but, as long as the focus is placed on extracting the functional user requirements, the functional size of software can be measured.

Furthermore, experience with other functional size measurement method have shown that there are two distinct activities involved in measuring the functional size of software: a) identify some base functional components within the software and b) assign size units to each of these components. The COSMIC-FFP measurement method recognize this practice explicitly by proposing a measurement process where the software to be measured is a) first mapped to the generic software model (section 4.2) and b) a measurement function is subsequently applied to this software model in order to generate a numerical figure representing the functional size. This measurement process is illustrated in Figure 6.

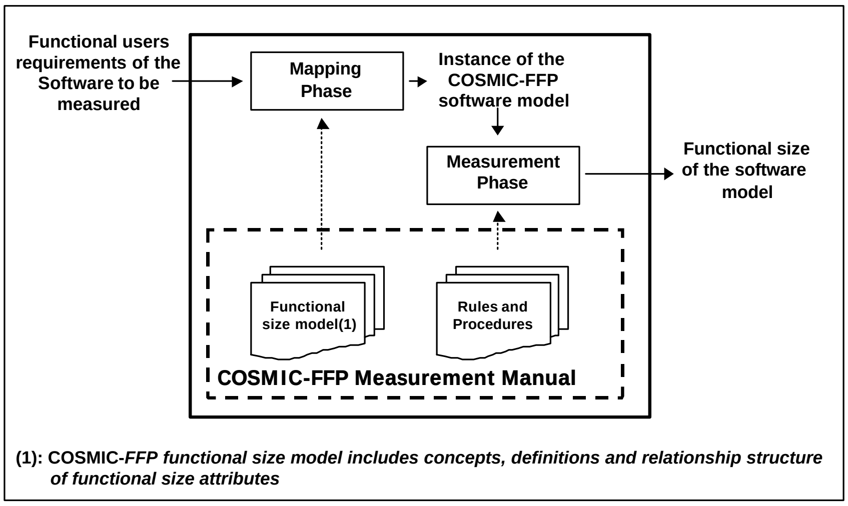
Functional users requirements of the Software to be measured
Mapping Phase
Instance of the COSMIC-FFP software model
Measurement Phase
Functional size of the software model
Functional size model(1)
Rules and Procedures
COSMIC-FFP Measurement Manual
(1): COSMIC-FFP functional size model includes concepts, definitions and relationship structure of functional size attributes

Figure 6 – COSMIC-FFP measurement process [30]

From the perspective of the measurement practitioners, the benefit of this approach lies in the flexibility of organizing a measurement exercise where tasks can easily and clearly be assigned to distinct individuals while retaining a common "interface" (the software model) that guarantees the uniformity of the interpretation while allowing each individual involved to focus their effort on a specific aspect of the process.

4.5. COSMIC-FFP Method vs. Procedure

Another key aspect of the COSMIC-FFP measurement method is the distinction made between a measurement method and a measurement procedure. As defined by ISO [31], in relation to metrology, a measurement method describes the concept and generic principles pertaining to a measure while a measurement procedure describe how to apply the method in specific instances.

This can be illustrated using temperature. A method for measuring temperature would define what "temperature" is, what is the unit of measure and what principles are governing the reading of temperatures. Based on this method, one can design a procedure to measure the temperature of the human body for medical purpose and someone else can design a procedure to measure the temperature of the lava inside a volcano. Obviously both procedures will be quite different but both procedures will measure the same concept, namely temperature.

Similarly, in the COSMIC-FFP measurement method, the concepts, the generic principles and a unit to measure the functional size of software are described in detail in a measurement manual which contains the measurement standards adopted by the COSMIC group. The method is designed to be applicable to a wide range of software types, including MIS and real-time software. A generic process, applicable to this range of software types is also described, but the specifics of measuring one particular type of software are left to a specific measurement procedure.

From the perspective of the measurement practitioners, the benefit of this approach lies in the combination of the relative universality of the functional size unit and the specificity of a tool (the procedure) adapted to the particular type of software to be measured. In addition, the standards have been put in the public domain and are available at http://www.lrgl.uqam.ca/ffp.html.

4.6. COSMIC-FFP industrial field trials

Another key aspect of the COSMIC-FFP measurement method is the conduct of a formal field trial period designed to demonstrate that it can withstand being scaled up to industrial software from multiple and varied contexts.

Started at the end of 1999, a 12 months period has been allocated for conducting formal and organized field trials of the COSMIC-FFP measurement method in a significant number of organizations around the world. During this period, each participating organization receives formal training on the application of the method. Furthermore, multiple software are selected from each organization's portfolio and their functional size is measured. These results, along with some key data on effort and schedule involved in delivering each software is registered and centralized for analysis. Once analyzed, a specific report is prepared for each participating organization, offering: a) guidelines for applying the method based on this organization's software engineering standards, and b) some preliminary benchmarking information allowing the organization to leverage its investment in the new data and put it to use immediately.

From the perspective of the participants in the field trials, the benefit of this approach lies in the availability, at the end of the field trial period, of a database of historical data useful for jumpstarting the implementation of the measurement method within their own organizations while respecting the confidentiality of the sources,.

5. CONCLUSION

Albrecht proposed the function points method, more than 20 years ago, as a new way to measure the size of software. His method was picked up by IFPUG in 1984 and promoted by this organization without fundamental changes in the concepts. In the past 15 years, many practitioners have found that Albrecht's measurement method cannot be applied satisfactorily to non MIS software. This led to the development of three types of approaches for measuring the functional size of non MIS software: status quo, approximation of functional size and addition of new function types.

In 1998, building on the strengths of previous methods, the COSMIC group identified the principles on which the next generation of functional size measurement methods were to be built. A year later, the group published COSMIC-FFP, a functional size measurement method implementing these principles. Key aspects of this method were presented here and industrial field trials are underway to demonstrate that it can withstand being scaled up to industrial software environments in multiple and varied contexts.

6. ACKNOWLEDGMENTS

The authors of this paper wish to acknowledge the specific contributions of Pam Morris, Grant Rule and Peter Fagg in the elaboration of the COSMIC-FFP measurement method and the thoughtful and generous comments from all the reviewers of the COSMIC-FFP Measurement Manual [30]; their names are listed in the preliminary pages of this document.

7. BIBLIOGRAPHY

[1] B.W. Boehm, R.W. Wolverton, “Software cost modelling: some lessons learned”, Journal of System and Software, 1:195-201, 1980.

[2] V. Côté, P. Bourque, S. Oligny, N. Rivard, “Software metrics: an overview of recent results”, Journal of System and Software, 8:121-131, 1988.

[3] C. Jones, Applied software measurement - Assuring productivity and quality, 2nd Edition. New York, NY: McGraw-Hill Inc., 1996.

[4] A. J. Albrecht, “Measuring Application Development Productivity,” presented at IBM Applications Development Symposium, Monterey, CA, 1979.

[5] S. A. Whitmire, “3D Function Points: Scientific and Real-Time Extensions to Function Points,” presented at Pacific Northwest Software Quality Conference, 1992.

[6] D. J. Reifer, “Asset-R: A Function Point Sizing Tool for Scientific and Real-Time Systems,” *Journal of Systems Software*, vol. 11, pp. 159-171, 1990.

[7] T. Mukhopadhyay and S. Kekre, “Software effort models for early estimation of process control applications,” *IEEE Transactions on Software Engineering*, vol. 18, pp. 915-24, 1992.

[8] A. Abran, J.M. Desharnais, M. Maya, D. St-Pierre, P. Bourque, “Design of a functional size measurement for real-time software”, Research report no. 13, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, November 1998.

[9] S. D. Conte, H. E. Dunsmore, V. Y. Shen, *Software engineering metrics and models*. Menlo Park: The Benjamin/Cummings Publishing Company, Inc., 1986.

[10] R. B. Grady, *Practical software metrics for project management and process improvement*. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1992.

[11] B. Hetzel, *Making Software Measurement Work - Building an Effective Measurement Program*. Boston: QED Software Evaluation Series, 1993.

[12] S. H. Kan, *Metrics and models in software quality engineering*. Readings, Massachusetts: Addison-Wesley Publishing Company, 1995.

[13] D. C. Ince, “History and industrial application”, in N.E. Fenton, *Software metrics: a rigorous approach*, Chapman & Hall, UK, 337 pages, 1991.

[14] S. Galea, “The Boeing Company: 3D function point extensions, v. 2.0, release 1.0”, Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.

[15] A. Abran, M. Maya, J.M. Desharnais, D. St-Pierre, “Adapting Function Points to real-time software”, American Programmer, Vol. 10, No. 11, pp. 32-43, November 1997.

[16] V.R. Basili, R.W. Shelby, D. H. Hutchens, “Experimentation in Software Engineering”, IEEE Transactions on Software Engineering [ISO], vol. SE-12, pp. 733-743, 1986.

[17] D. St-Pierre, M. Maya, A. Abran, J.M. Desharnais, P. Bourque, “Full Function Points: Function Points Extension for Real-Time Software - Counting Practices Manual”, Technical Report no. 1997-04, Software Engineering Management Research Laboratory, Université du Québec à

* Alain Abran, Moritsugu Araki ,Jean-Marc Desharnais, Reiner Dumke, Peter Fagg, Roberto Meli, Pam Morris, Serge Oligny, Jolijn Onvlee, Risto Nevalainen, Grant Rule, Denis St Pierre, Charles Symons.

Montréal, Montreal, Canada, September 1997. Downloadable at http://www.lrgl.uqam.ca/ffp.html.

[18] S. Oligny, A. Abran, J.M. Desharnais, P. Morris, "Functional Size of Real-Time Software: Overview of Field Tests", Proceedings of the 13th International Forum on COCOMO and Software Cost Modeling, Los Angeles, USA, October 1998.

[19] N. Kececi, M. Li, C. Smidts, "Function Point Analysis: An application to a nuclear reactor protection system", Proceedings of the Probabilistic Safety Assessment - PSA' 99 conference, Washington DC, USA, August 1999.

[20] F. Bootsma, "Applying Full Function Points To Drive Strategic Business Improvement Within the Real-Time Software Environment", Proceedings of the 1999 Annual IFPUG Conference, New Orleans, USA, October 1999.

[21] A. Schmietendorf, R. Dumke, E. Foltin, "Applicability of Full Function Points for Siemens AT", Proceedings of IWSM '99, September 1999. Proceedings are downloadable at http://www.lrgl.uqam.ca/iwsm99/index2.html.

[22] G. Büren, I. Koll, "Process improvement by introduction of an effort estimation process", Proceedings of the 12th International Conference on Software and System Engineering and their Applications (ICSSEA), Paris, France, December 1999.

[23] C. R. Symons, *Software sizing and estimating – MkII FPA (function point analysis)*, John Wiley & sons, Chichester, UK, 1991.

[24] The Netherlands Software Metrics Users Association (NESMA), "Definitions and counting guidelines for the application of function point analysis, version 2.0", 1997.

[25] C. R. Symons, P. G. Rule, "One size fits all – COSMIC aims, design principles and progress", Proceedings of ESCOM '99, pp. 197-207, April 1999.

[26] A. Abran, "FFP Release 2.0: An Implementation of COSMIC Functional Size Measurement Concepts", Proceedings of FESMA '99, Amsterdam, Oct. 1999.

[27] C.R. Symons, " COSMIC aims, design principles and progress", Proceedings of IWSM '99, pp. 161-172, September 1999. Proceedings are downloadable at http://www.lrgl.uqam.ca/iwsm99/index2.html.

[28] S. Oligny, A. Abran, D. St-Pierre, "Improving Software Functional Size Measurement", Proceedings of 14th International Forum on COCOMO and Software Cost Modeling, Los Angeles, USA, October 1999.

[29] International Organization for Standardization (ISO), "ISO/IEC 14143-1:1997 – Information technology – Software measurement – Functional size measurement – Definition of concepts", October 1997.

[30] A. Abran, J.M. Desharnais, S. Oligny, D. St-Pierre, C. Symons, "COSMIC-FFP Measurement Manual, version 2.0", Ed. S. Oligny, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Montreal, Canada, Oct. 1999. Downloadable at http://www.lrgl.uqam.ca/ffp.html

[31] International Organization for Standardization (ISO), "International Vocabulary of Basic and General Terms in Metrology", Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.