Using COSMIC-FFP to Quantify Functional Reuse in Software Development

Vinh T. Ho, Alain Abran, Serge Oligny
Dept. of Computer Science, Université du Québec à Montréal, Canada
vho@lrgl.uqam.ca, abran.alain@uqam.ca, oligny.serge@uqam.ca

Abstract

One of the means organisations use to adequately measure the performance of their software engineering process, is to try to identify how much reuse has actually occurred. In this paper, the COSMIC-FFP (COSMIC-Full Function Points) measurement method is proposed as a method for quantifying reuse from a functional perspective rather than from a technical perspective.

The COSMIC-FFP method has been developed to improve the measurement of the functional size of various software types: real-time, technical, system and MIS software. By using functional user requirements as input, the method makes it possible to measure the size of software from the user's viewpoint. When other functional perspectives are taken into account in the measurement process, the other results may be used as complementary information related to the measured software. The value of this new information includes the ability to quantify reuse from a functional perspective, and as such it would be worth considering taking it into account in the software productivity model. Some practical results on industrial software are presented, along with the concepts involved.

1. Introduction

Software reuse is generally recognised as providing a great opportunity for reducing costs, both in software development and in maintenance. Even though software reuse can occur at any point during the software life cycle, the literature has reported mostly on software reuse performed at the source code level [4][5][6][9], which saves effort only late in the life cycle. Of interest to organisations as a means to improve the performance of their software engineering process is reuse applied much earlier in the life cycle, such as at the requirement analysis or software design stage. Software measurement can play an important role in quantifying such potential for reuse at the functional level, when major development costs have not yet been incurred. Unfortunately, there are few measures of reuse at that level.

An approach to measure reuse based on functional size measurement has been proposed in [2]. This paper illustrates the concept of using functional size to quantify functional reuse by identifying avoided functions, those that did not need to be redeveloped. The approach allows estimation of an alternative functional size which takes into account a subset of functions: those reused and not redeveloped. This alternative functional size can then be used to quantify the benefits of major enhancement work in terms of a lower unit cost when non redeveloped functions are taken into account. This initial work [2] has been initially performed using the data function types of the traditional Function Points method, with MIS software only, without the investigation of functional reuse at the transaction level.

This paper extends the concept presented in [2] by using the more recent COSMIC-FFP measurement method developed to improve the measurement of the functional size of various software types: real-time, technical, system and MIS software [1][8]. By using the functional user requirements as input, the method allows measurement of the functional size of software from the user's viewpoint. When other functional perspectives are taken into account to feed the measurement process, the other results may be used as complementary information related

to the measured software. The value of this new information includes the ability to quantify the functional reuse, thereby opening the door to assessing the economic impact of reuse.

This paper thus presents and illustrates the applicability of COSMIC-FFP for measuring software functional reuse. Section 2 presents the COSMIC-FFP measurement method and section 3 presents the concept for the measurement of functional reuse with this measurement method, while results from empirical case studies are presented in section 4 and observations in section 5.

2. COSMIC-FFP Measurement Method

In this section are presented some of the key concepts of the COSMIC-FFP method [3]. The method consists in applying a set of rules and procedures to a given piece of software in order to measure the functional size of this software. Two distinct and related phases are necessary for measuring the functional size of software: mapping the functional user requirements embedded in the artifacts of the software to be measured onto the COSMIC-FFP software model and then measuring the specific elements of this software model, as illustrated in Figure 1.

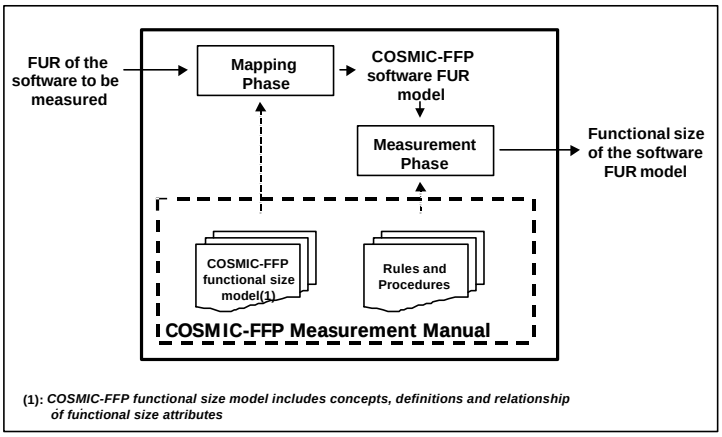

***Figure 1**. COSMIC-FFP measurement process model [3]*

Prior to applying the measurement rules and procedures, the software to be measured must be mapped onto a generic model (the COSMIC-FFP software model – Figure 2) which captures the concepts, definitions and relationships (functional structure) required for a functional size measurement exercise. According to this model, software functional requirements are implemented by a set of functional processes. Each of these functional processes is an ordered set of sub-processes and each sub-process performs a data movement.

The COSMIC-FFP generic software model distinguishes four types of data movement: entry, exit, read and write. *Entries* move the data from outside the software boundary to the inside; *exits* move it from inside the software toward the outside of the software boundary; reads and writes move data across the storage side. These relationships are illustrated in Figure 2.

The COSMIC-FFP measurement rules and procedures are applied to the software model in order to produce a numerical figure representing the functional size of the software. The standard unit of measurement is defined as 1 data movement. One COSMIC Functional Size Unit, e.g. 1_{CFSU}., is therefore attributed to each identified data movement.

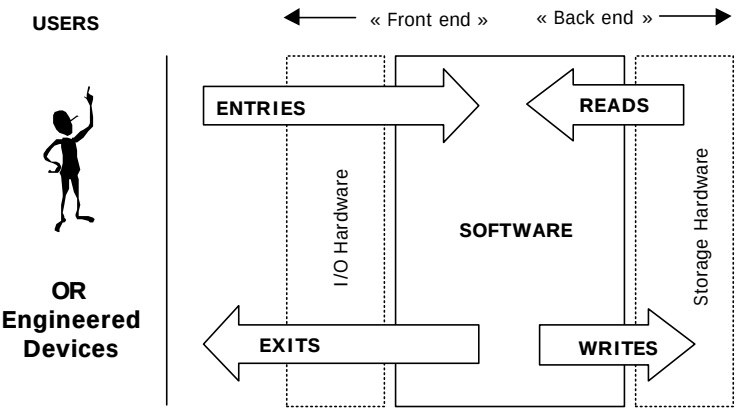

Figure 2*. COSMIC-FFP software model and sub-process types [3]*

3. Concept for Measuring Functional Reuse using COSMIC-FFP

In software engineering, there are two kinds of reuse, determined according to the perspective of the reuser: reuse without modification, referred to as black-box reuse and reuse with modification, referred to as white-box reuse. This paper investigates black-box reuse at the functional process level. The COSMIC-FFP method is proposed as a method for identifying and measuring functional reuse.

The COSMIC-FFP measurement method measures the size of software based on identifiable functional user requirements. Depending on how these requirements are allocated, the resulting software might be implemented in a number of pieces. While all the pieces exchange data, they will not necessarily operate at the same level of abstraction. The COSMIC-FFP method introduces the concept of the software layer to help differentiate Functional User Requirements (FURs) allocated at different levels of functional abstraction. Each level is designated as a distinct layer and each layer encapsulates functionality useful to other layers using its services. The functionality of a layer may be composed of a number of functional processes and sub-processes. An illustration of layer configurations is presented in Figure 3.

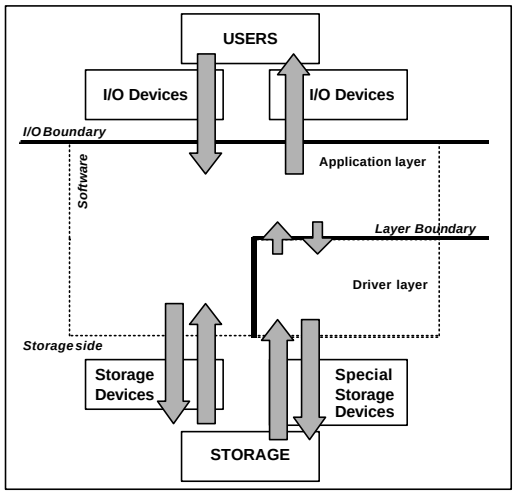

Figure 3: Example of a configuration of functional layers [3]

Practice has shown that identifying functional layers within the software to be measured has made it possible to identify some of the functionality that was being reused in the software system, based on the functional relationships between layers.

The approach proposed in this paper quantifies the functional reuse based on the size of the processes referenced in the functional relationships between layers.

When a functional process is identified as involving reuse, the amount of reuse associated with it is determined by its own functional size multiplied by the number of processes using its services:

$$Amount\ of\ reuse\ associated\ with\ process = Size\ of\ process\ reused \times Number\ of\ processes\ using\ its\ services \quad (1)$$

For the entire software being measured, the amount of reuse is defined as the total of the amounts of reuse from all reused processes:

$$Overall\ amount\ of\ reuse = \sum Amount\ of\ reuse\ associated\ with\ process \quad (2)$$

The percentage of reuse in the software is derived as follows:

$$Percentage\ of\ reuse = \frac{Overall\ amount\ of\ reuse}{Size\ of\ software\ measured\ without\ reuse} \times 100 \quad (3)$$

4. Illustrative Examples

This section presents some results from an empirical case study illustrating the application of the proposed Functional Reuse concept for quantifying the functional reuse in some industrial software. Discussions on the impact of reporting the measurement results for productivity analysis and estimation are also presented.

4.1. ISDN tester software

The ISDN tester software was designed and manufactured for an Australian organisation. The ISDN tester is a device used to test the integrity of four wires ISDN circuits, cross-connected at a remote 'Point of Presence' location, from either end of the ISDN service. Test officers at a local exchange are able to dial up the ISDN tester installed at a remote location and use DTMF (Dual-Tone Multi-Frequency touchtone dialing) to instruct the ISDN software either to open circuits on both sides of a transmit and receive circuit or to provide loops via the transmit and receive wires back to the local exchange. When the test officer is connected to the ISDN device, he can perform maintenance programming functions as well as testing functions.

For the purpose of measuring functional size, the boundary between the ISDN software and its environment is identified such as illustrated in Figure 4. Since the user must be connected to the ISDN device via the Access line before performing any remote programming functions, the software can be modelled as having two layers (Figure 4):

- *Access layer*: contains the Access functional process allowing the user to connect to the ISDN device;
- *Application layer*: contains a set of functional processes implementing the maintenance and testing functions of the ISDN software.

Thirty-eight functional processes were identified within the ISDN software: 1 functional process in the Access layer (the Access process), and 37 processes in the Application layer. Since they need to receive remote commands from the user before operating, thirty-two processes in the Application layer use services provided by the Access process. From a functional perspective, these processes make up a functional reuse of the Access process. In other words, the Access process is functionally reused in the modelling of the ISDN software according to the COSMIC-FFP software model.

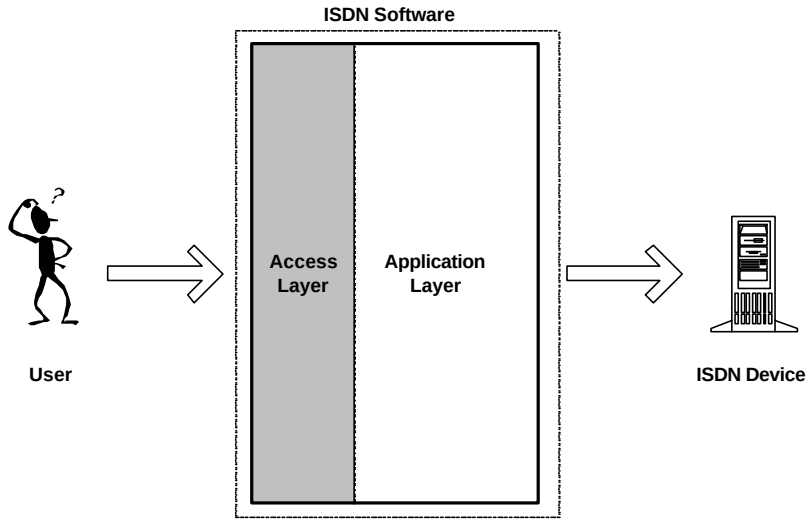

***Figure 4**. The COSMIC-FFP software boundary of the ISDN software*

For illustrative purposes, the measurement results of the Access process are presented in Table 1, including its 10 data movement sub-processes. The functional size of the Access process is 10_{CFSU}.

Table 1. Measurement results for the Access process

Data movement sub-process	Type	COSMIC-FFP size
Receive dialing event	E	1
Read LRD Code	R	1
Issue message reading LRD Code	X	1
Enter PIN	E	1
Verify PIN	R	1
Write Invalid Password Counter	W	1
Read Invalid Password Counter	R	1
Write Log on Counter	W	1
Receive canceling signals	E	1
Issue error message	X	1
Total:		**10_{CFSU}**

By applying equation (1), the amount of reuse derived from functionally reusing the Access process becomes:

Amount of reuse = $10_{CFSU} \times 32$ (number of processes using its services) = 320_{CFSU}.

The size of the ISDN software measured with the identification of the functional reuse of this Access layer is 136_{CFSU} (this means that the size of the Access process is accounted only once). If its size had been measured without taking into account functional reuse (meaning that the size of the Access process is added to that of every process using its services, i.e. 32 times), the software size would have been 436_{CFSU}.

It is important to mention that the measurement of the ISDN software was performed based solely on the Operator and Installation manual of the ISDN system. The measurer had no knowledge of the actual implementation of the software. It is therefore worth noting that the functional reuse results presented here were identified and measured from the user's point of view, using the specific measurement tool of the software layer concept of the COSMIC-FFP method.

4.2. Other examples

The results of measuring reuse in three other system software are presented in this section. The first two are control software and the third is a surveillance sub-system software. For the sake of simplicity, the details are not presented, only the final measurement results.

All identified functional processes within the two control software receive data from external sensors and, by convention, the data acquisition functionality was included inside the software's boundary. However, instead of including the functionality repetitively in each process, the control functionality was identified as a distinct layer referred to by other processes in a black-box reuse style. This layer was therefore measured, and assigned its own functional size, even though its functionality was used in multiple processes. This is a clear case where functionality, structured inside a layer, is an instantiation of actual reuse. In Table 2, column (2) identifies the number of units assigned to the control layer, and column (3), the number of times this layer is invoked by other processes. The next three columns present the impact on the measurement results:

Column (4) = (2) x (3) = explicit amount of reuse associated with a process (in the data acquisition layer, for example, for Control Systems 1 and 2); this would be equivalent to the sum of the functional size of all duplicate functions if there had been no reuse.

Column (5) = (6) + (4) = size of software when including the multiple duplicates of the same functionality (without implementation of functional reuse).

Column (6) = size of software measured when taking into account that a subset of functions has been put into a layer (the data acquisition layer), each function to be reused wherever needed. This means that the size of the reused layer is included only once in the measurement results. This of course generates a smaller size.

Column (7) = the ratio of the two measurements, with and without the inclusion of reuse from the functions in the data acquisition layer. It is obvious that, for these three empirical case studies, having included the duplicates of functions would have increased the functional size by a factor of 2.26 and 2.98. Of course, it is to be expected that other case studies would have their own ratios, depending on their own functional relationships.

Table 2. Measurement results in CFSU units

Software **(1)**	**Size of reused process** **(2)**	**Number of processes using reused process's services** **(3)**	**Quantity of duplicates** **(4)**	**Size of software measured *without* implementation of reuse** **(5)**	**Size of software measured *with* reuse implemented** **(6)**	**Ratio (5) / (6)** **(7)**
Control System 1	50	9	450	807	357	2.26
Control System 2	25	8	200	359	159	2.26
Surveillance Sub-System 3	29	3	87	131	44	2.98

For all cases presented in this paper, the size of the software measured taking into account functional reuse is obviously much smaller than if it had been measured with the duplicates of functions.

It is interesting to note that the measurer of the two control systems had no knowledge on the actual implementation of the software, while in the surveillance sub-system the measurer knew about its actual implementation. For the two control systems, the software team agreed that the measurement performed at the abstract level, using the specific measurement tool of the

software layer concept of the COSMIC-FFP method, with the reused functionality segregated in a distinct layer, was much more representative of the team's perceived functional size of the software. The software practitioners commented that the alternative of measuring multiple duplicates would have represented a significant distortion of the functional size of the software.

This observation illustrates the importance of documenting the basis of the measurement procedure in the presence of functional reuse, whether implemented or not.

The proper recording of this information then becomes critical in the development of productivity models, and, later on, for their use in an estimation process. For example, if we were to study Control System 1, we would find different unit costs using either column (5) or column (6) in the denominator of the unit cost formula (unit cost = Effort / Size = Hours / CFSU): when taking reuse into account before measuring, the unit cost would appear to be significantly higher (since the duplicates would not have been taken into account), while the project would appear much cheaper unit-wise if column (5) were used in the formula. This underlines the importance of proper documentation of the measurement procedures, and proper recording of measurement results. Such an interpretation context would ensure the proper interpretation of results across multiple projects, both for productivity analysis and for estimation purposes.

Readers interested in investigating how to take functional reuse into account in software productivity analysis should refer to [2] for further detail on the concept of identification and measurement of avoided duplicate functions. Integrating the more sophisticated concepts from the economics models presented in the literature on the measurement of reuse at the code level could help improve productivity models (for example, taking into consideration the normally higher costs of building and testing more generic reused functions).

5. Discussion and Conclusion

This paper has presented and illustrated the applicability of the COSMIC-FFP method for identifying and quantifying software functional reuse. The software layer concept introduced in the measurement process of the COSMIC-FFP method was used as a means to identify potential sources of functional reuse in the software to be measured. The results from the four empirical case studies illustrated that there can be important variations between the size of the software with and without functional reuse. On the one hand, this underlines the fact that such information on functional reuse must be taken into account when analysing software productivity and cost. On the other hand, the observation of such alternative functional size could be useful for analysing the quality of a design (or architecture). It has also been recommended, in [7], that the presence of reuse must be recorded in the data collection process and in the database used for benchmarking purposes. Research is ongoing to gain further insight into the identification of reuse, its measurement, recording and use in more sophisticated productivity and estimation models.

It would be also interesting to perform more detailed analyses of the software measured here with a view to finding opportunities for additional reuse, that is, the identification and measurement of potential for reuse (i.e. the reusability) at the functional level. With the COSMIC-FFP method, the process of modelling software to be measured allows for characterisation of the software at the level of data movements. While looking at the identified functional processes, software engineers can figure out whether some functional processes might be conceptually similar even though they occur in different instances from the user's point of view. From their experience and knowledge about the software's design principles, they would be able to identify functional processes which could result in a similar implementation. The functional reuse potential is therefore identified based on the conceptual (implementation) closeness acknowledged by the software engineer between the identified

functional processes. Note that still do not exist a well-defined notation for the concept of conceptual closeness. The COSMIC-FFP method could be used to quantify the potential amount of reuse based on data movement sub-processes identified within the processes to be reused. Future research is planned to investigate this approach in various industrial software projects.

Acknowledgements

This work has been financially supported by Bell Canada and the Natural Sciences and Engineering Research Council of Canada.

6. Reference

[1] Abran, A., "FFP Release 2.0: An Implementation of COSMIC Functional Size Measurement Concepts", Presented at FESMA 99, Amsterdam, Oct. 4-7, 1999.

[2] Abran, A., Desharnais, J-M., "Measurement of Functional Reuse in Maintenance", Software Maintenance: Research and Practice, 1995, pp. 263-277.

[3] Abran, Desharnais, Oligny, St-Pierre, Symons, "COSMIC-FFP Measurement Manual, version 2.0", Ed. S. Oligny, Software Engineering Management Research Lab., Université du Québec à Montréal (Canada), Oct., 1999.

[4] Burd, E., Munro, M., "A Method for the Identification of Reusable Units through the Reengineering of Legacy Code", Journal of Systems and Software, 1999, pp. 121-134.

[5] Gaffney, J.E., Cruickhank, R.D., "A General Economics Model of Software Reuse", ACM, 1992, pp. 327-337.

[6] Leach, R.J., "Method of Measuring Software Reuse for the Prediction of Maintenance Effort", Software Maintenance: Research and Practice, 1996, pp. 309-320.

[7] Meli, R., "Software Reuse as a Potential Factor of Database Contamination for Benchmarking in Function Points", Proceedings of ISBSG Workshop, Rome, Italy, Feb. 1998.

[8] Oligny, S., Abran, A., St-Pierre, D., "Improving Software Functional Size Measurement, Proceedings of COCOMO and Software Cost Modeling International Forum 14, USA, Oct., 1999.

[9] Poulin, J.S., "Measuring Software Reusability", Proceedings of the Third International Conference on Software Reuse: Advances in Software Reusability, 1994, pp. 126-138.