Full Function Points for Embedded and Real-Time Software

UKSMA Fall Conference
London (UK) Oct. 30-31, 1998

Software Engineering Management Research Laboratory –
Université du Québec à Montréal
&
Software Engineering Laboratory in Applied Metrics (SELAM)

Alain Abran
Denis St-Pierre
Marcela Maya
Jean-Marc Desharnais

TABLE OF CONTENTS

1. Introduction

Given its increasing importance in products and services, software has become a major component of corporate budgets. Like any other budget component, it is important to control these expenses, to analyze the performance of the amounts allocated to software development and to benchmark against the best in the field. To do so, measures and models using these measures are needed. Measures are needed for analyzing both the quality and the productivity of development and maintenance. Technical measures are needed to measure the technical performance of products or services, from a developer's view point. Technical measures will be used for efficiency analysis to, for example, improve the performance of the designs.

On the other hand, functional measures are needed to measure the performance of products or services from a user's perspective, and they are needed for productivity analysis. Functional measures must be independent of technical development and implementation decisions. They can then be used to compare the productivity of different techniques and technologies.

Such a functional measurement method, Function Point Analysis (FPA), is available for the MIS domain where it has been used extensively in productivity analysis and estimation (Abran, 1996; Desharnais, 1988, Jones, 1996; Kitchenham, 1991). It captures the specific functional characteristics of MIS software well.

However, FPA has been criticized as not being universally applicable to all types of software (Conte, 1986; Galea, 1995; Grady, 1992; Hetzel, 1993; Ince, 1991; Jones, 1988; Jones, 1991; Kan, 1993; Whitmire, 1992). Here is how D.C. Ince describes the Function Point (FP) scope issue:

> *"A problem with the function point approach is that it assumes a limited band of application types: typically, large file-based systems produced by agencies such as banks, building societies and retail organizations, and is unable to cope with hybrid systems such as the stock control system with a heavy communication component."* (Ince, 1991, page 283)

A number of academic papers have been published on the statistical accuracy of FPA (Abran, 1994; Desharnais, 1988; Kemerer, 1993) based mostly on historical databases from the MIS domain. However, FPA does not capture all functional characteristics of real-time software. When FPA is applied to such software, it does, of course, generate counts, but the counts do not constitute an adequate size measurement. Therefore, there is currently no FPA equivalent for real-time software that would allow meaningful productivity benchmarking and development of estimation models based on the functional full size of real-time software.

This report describes work done by the Software Engineering Management Research Laboratory at the Université du Québec à Montréal and its industrial research partner SELAM to adapt FPA to real-time software. Section 2 identifies some real-time limitations of FPA. Next, in section 3, the proposed extension is presented. The practical aspects of measuring applications with the extension are addressed in section 4. An integrated approach is then presented in section 5 for the organizations with historical FPA databases.

2 Real-time software limitations of FPA

2.1 Data limitations

There are two kinds of control data structure: multiple occurrence groups of data and single occurrence groups of data. Multiple occurrence groups of data can have more than one instance of the same type of record[1]. Single occurrence groups of data have one and only one instance of the record. Real-time software typically contains a large number of single occurrence groups of data that are difficult to group into IFPUG-Internal Logical Files or IFPUG-External Interface Files. An extension is therefore necessary to adequately measure single occurrence groups of data.

2.2 Transaction limitations

Real-time software processes have a specific transactional characteristic in common: the number of their sub-processes varies a great deal. A real-time functional measurement technique has to take into account that some processes have only a few sub-processes, while others have a large number of sub-processes. To illustrate this phenomenon, consider the following two examples:

Example 1 - An engine temperature control process (process with a few sub-processes)

The main purpose of this process is to turn on a cooling system when necessary. A sensor enters the temperature in the application (sub-process 1). The temperature is compared to the overheating threshold temperature (sub-process 2). Finally, a turn-on message could be sent to the cooling system if needed (sub-process 3).

In this example, the temperature control process has 3 sub-processes (see Table 1). This process is not an application; it is only one of the many processes of an engine control application. According to IFPUG rules, the application is not in a consistent

[1] For example, an engine control application could have a group of control data containing information on each cylinder (cylinder number, ignition timing, pressure, etc.). Such a group of data has a multiple occurrence structure (one record for each cylinder). In other words, the cylinder record is repeated more than once.

state until all sub-processes of the temperature control process are completed. Therefore, there is only 1 IFPUG-elementary process (IFPUG, 1994).

Process	Sub-process description	# of sub-processes
Engine control	Temperature entry	1
	Read threshold for comparison	1
	Send turn-on message	1
	Total	**3**

Table 1 - Sub-processes of example 1

According to IFPUG-standard rules, only one transactional function would be identified, because there is only 1 elementary process.

Example 2 - An engine diagnostic process (process with many sub-processes)

The main purpose of this process is to turn on an alarm when necessary. Fifteen engine sensors (all different) send data to the diagnostic process (15 sub-processes, 1 unique sub-process for each kind of sensor). For each sensor, the set of external data received is compared to threshold values read from an internal file, one unique file for each kind of sensor (15 other sub-processes, 1 unique sub-process for each kind of sensor). Depending on a number of conditions, an alarm on the dashboard may be turned on (1 sub-process).

In this example, the engine diagnostic process consists of 31 sub-processes (see Table 2). This process is not an application; it is only one of the many processes of an engine control application. According to IFPUG rules, the application is not in a consistent state until all sub-processes of the diagnostic process are completed.

Process	Sub-process description	# of sub-processes
Engine diagnostic	Sensor data entry	15
	Read thresholds for comparison	15
	Send alarm message	1
	Total	**31**

Table 2 - Sub-processes of example 2

According to IFPUG standard rules, only a few transactional points would be counted because transactional functions are based on elementary processes rather than on sub-processes. Therefore, examples 1 and 2 would have approximately the same number of points related to transactions with current IFPUG measurement procedures.

3. Measurement of Real-time Functional Size

Full Function Points (FFP) is a measure of the functional size of software, from a user perspective and not taking into account the technical characteristics of the software. It was designed for both MIS and real-time software.

3.1 FFP Function Types

It was illustrated in the two previous example that to measure the characteristics of a variable number of sub-processes adequately, the IFPUG-defined 'elementary process' concept is inadequate since the functionality at the sub-process level should also be taken into account in the measurement of the functional size from a user perspective. Therefore, a new measurement basis was introduced in FFP, together with corresponding data and transactional function types closer to the control data characteristics.

The two new Control Data Function Types have a structure similar to that of the IFPUG Data Function Types:

Updated Control Group (UCG): A UCG is a group of control data updated by the application being measured. It is identified from a functional perspective[2]. The control data live for more than one transaction[3].

Read-only Control Group (RCG): An RCG is a group of control data used, but not updated, by the application being measured. It is identified from a functional perspective. The control data live for more than one transaction[4].

The four new Control Transactional Function Types address the sub-processes of real-time software:

External Control Entry (ECE): An ECE is a unique sub-process. It is identified from a functional perspective[5]. An ECE processes control data coming from outside the application's boundary. It is the lowest level of decomposition of a process acting on one group of data. Consequently, if a process enters two groups of data, there are at least 2 ECEs. ECEs exclude the updating of data.

---

[2] This means that the group of data appears in the requirements of the application, assuming they are complete.

[3] In example 2 of section 2, the sensor data entered live only for one transaction, since after the diagnostic process, the system doesn't remember them. In contrast, the thresholds are reused for each new entry, and consequently live for more than one transaction.

[4] UCG and RCG rules are different from the IFPUG-ILF and EIF rules, in order to allow FFP to measure the data aspect of real-time software. ILF and EIF rules seem to be oriented toward typical MIS corporate data structures.

[5] This means that the sub-process is referenced in the requirements of the application, assuming they are complete.

The updating functionality is covered by another Control Function Type (Internal Control Write).

In the engine diagnostic (example 2), 15 sensors send data to the application (control data cross the application boundary). Since there is a unique sub-process for each sensor, there are 15 ECEs.

External Control Exit (ECX): An ECX is a unique sub-process. It is identified from a functional perspective. An ECX processes control data going outside the application boundary. It is the lowest level of decomposition of a process acting on one group of data. Consequently, if a process exits two groups of data, there are at least 2 ECXs. ECXs exclude the reading of data. The reading functionality is covered by another Control Function Type (Internal Control Read).

In the engine diagnostic (example 2), the sub-process that sends a message to the dashboard (control data sent outside the application boundary) is an ECX.

Internal Control Read (ICR): An ICR is a unique sub-process. It is identified from a functional perspective. An ICR reads control data. It is the lowest level of decomposition of a process acting on one group of data. Consequently, if a process reads two groups of data, there are at least 2 ICRs.

In the engine diagnostic (example 2), the sub-processes that read the threshold values are ICRs. In this example, 15 unique sub-processes read different kinds of threshold values at different times for comparison purposes. Therefore, there are 15 ICRs.

Internal Control Write (ICW): An ICW is a unique sub-process. It is identified from a functional perspective. An ICW writes control data. It is the lowest level of decomposition of a process acting on one group of data. Consequently, if a process writes on two groups of data, there are at least 2 ICWs.

As a writing sub-process example, the engine diagnostic is extended with the following functionality: "The 15 sets of control data are stored. They are all stored separately at different times in different files (15 different sub-processes)." Since there are fifteen kinds of sensor control data updated at different times (15 unique sub-processes), there are 15 ICWs with the new FFP measurement technique.

Here is how the Control Transactional Functions are related to control processes (Figure 1):

Figure 1 - Diagram of Control Transactional Functions

As with IFPUG, all new function types are based on the functional perspective of the application, rather than on the technical perspective.

FFP Function Types:

Updated Control Group	new, similar to ILF
Read-only Control Group	new, similar to EIF
External Control Entry	new, similar to a simpler subset of EI
External Control Exit	new, similar to a simpler subset of EO/EQ
Internal Control Read	new, similar to a simpler subset of EI/EO/EQ
Internal Control Write	new, similar to a simpler subset of EI

Table 3: List of FFP Function Types

4. FFP practice feedback

An overview of FFP concepts and definitions was presented in the previous sections, together with two examples. We now discuss some practical aspects of measuring with FFP. A number of industrial real-time applications have already been measured with FFP. Feedback from these field tests is reported here.

4.1 Ease of understanding

All FFP measures were done with the support of real-time application experts (industry people familiar with the application being measured). Once the application experts had understood the definitions of the new Function Types, they had no problem in identifying them. In fact, after a full day of measuring with FFP, the application experts were able to measure with little assistance from functional size experts familiar with FFP.

4.2 Measurement effort

Based on the field tests carried out, the FFP measurement effort is similar to the FPA one. On the one hand, more functions have to be identified with FFP; on the other hand, being

simpler[6], the new FFP Transactional Functions are more quickly identified. Besides, application experts seem to require less measurement assistance from functional size experts, again because FFP Transactional Functions are simpler to deal with.

4.3 Importance of documentation

An adequate source of functional information must be available to measure an application. This functional information is provided by application experts or by application documentation. Like IFPUG method, FFP is dependent on the quality of the functional information. Based on the field tests carried out, the functional information typically available in industry is adequate for measuring with FFP.

4.4 Early FFP measures

It is possible to measure FFP function types at an early stage of development provided that the functional requirements are documented. Despite additional function types, the FFP level of detail is similar to the IFPUG one. To use IFPUG at an early stage of development, one must usually approximate the measures because not all functional information is available. Over the years, a number of IFPUG early measurement approximation methods have been developed; similar approximation methods could be developed for FFP[7].

5. IFPUG transitioning

For the organizations with a sizable portfolio of size measurement with IFPUG method, it is important to stay relevant and protect its past investment. Indeed, since FFP is an extension of the IFPUG measurement method, all IFPUG rules can be included in an application of this new measurement technique. However, the small number of subsets of IFPUG rules dealing with control concepts need then to be replaced by the new FFP function types. Thus, the unadjusted count of an application using the proposed extension (FFP) can be expressed as follows:

$$\begin{aligned} FFP &= \text{Management FP} + \text{Control FP} \\ &= (\text{FPA} - \text{Control information}) + \text{Control FP} \end{aligned}$$

The difference between the IFPUG method and the proposed extension (FFP) is therefore the FFP function types (UCG, RCG, ECE, ECX, ICW and ICR).

---

[6] Control Transactional Function Types are simpler because they are associated with only one sub-process. Management Transactional Function Types may comprise many sub-processes, and grouping them into elementary processes may take time.

[7] Of course, one should not expect, at this point, that FFP approximation methods be as mature as FPA ones.

The new FFP function types are only used to measure control data and control processes. The other types of data and processes, called management data and processes here, are measured with the IFPUG method (see Figure 2 and Table 4). It is of course important to document adequately which measurement strategy was selected when reporting results and using them for benchmarking purposes.

FFP Management Function Types:
- Internal Logical File (ILF)
- External Interface File (EIF)
- External Input (EI)
- External Output (EO)
- External Inquiry (EQ)

FFP Control Function Types - New:

Updated Control Group	new function type, similar to ILF
Read-only Control Group	new function type, similar to EIF
External Control Entry	new function type, similar to a subset of EI
External Control Exit	new function type, similar to a subset of EO/EQ
Internal Control Read	new function type, similar to a subset of EI/EO/EQ
Internal Control Write	new function type, similar to a subset of EI

Table 4: List of integrated Function Types

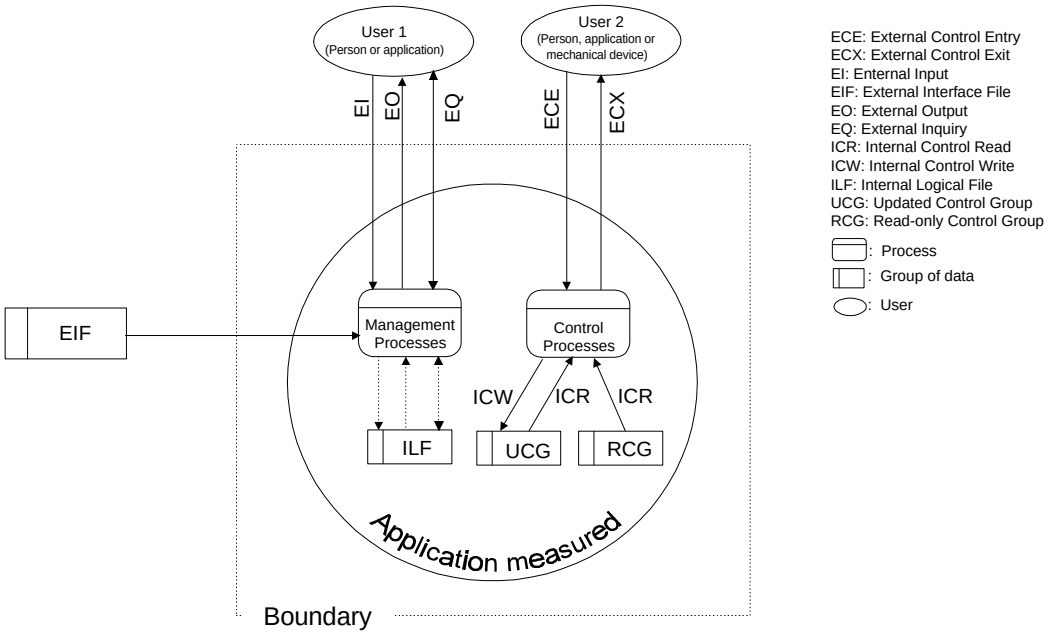

Figure 2 - Diagram of FPA + FFP Function Types

5. Summary

The development of a real-time software functional measure is an important challenge to meet. Such a measure should allow meaningful productivity benchmarking and the development of estimation models based on functional size. One of the first steps in achieving real-time software benchmarking and estimation models is to have measurement specialists working with the same set of rules. To some extent, this would allow measurement specialists to build on what has been achieved by others, rather than to start from scratch with their own rules or interpretations of existing rules. In a way, this is what happened with Function Points for MIS applications. Since Albrecht published the initial structure of Function Points in 1984 (Albrecht 1984), many people have worked with this measure. Indeed, a number of project databases[8] are based on this set of rules which has been significantly refined by IFPUG over the past 10 years. Hopefully, once the industry agrees on a set of rules to measure the functionality of real-time software, it will become possible to create better productivity and estimation models for this type of software.

This report has presented a summary of the first public release of FFP, and the authors recognize that there is room for improvement.

A public report on FFP detailed measurement concepts, rules and procedures is available at the following addresses:
http://www.lmagl.qc.ca/rtreport.pdf
or
http://saturne.info.uqam.ca/Labo_Recherche/Lrgl/publi/treports/LRGL-1997-015.pdf

We are positive that this FFP measurement method will expand the domain of applicability of Function Points and increase its relevance in industry. We are open to collaboration with corporations and measurement associations to produce the next release of FFP.

[8] For example: Gartner Group, International Software Benchmarking Standards Group, Howard Rubin Associates, Software Productivity Research.

References

Abran, A., *Analysis of the measurement process of Function Points Analysis*, Ph.D. Thesis, Département de génie électrique et de génie informatique, École Polytechnique de Montréal, 1994, 405 pages.

Abran, A., and Robillard, P. N., *Function Point Analysis, An Empirical Study of its Measurement Processes* IEEE Transactions on Software Engineering, vol. 22, no. 12, pp. 895-909, Dec. 1996.

Albrecht, A.J., *AD/M Productivity Measurement and Estimate Validation*, IBM Corporate Information Systems, IBM Corp., Purchase, N.Y., May 1984.

Conte, S.D., Shen, V.Y., and Dunsmore, H.E., *Software Engineering Metrics and Models*, Benjamin Cummins Publishing, 1986, 396 pages.

Cooling, J. E., *Software Design for Real-Time Systems*, Chapman and Hall, 1991.

Desharnais, J. M., *Statistical Analysis on the Productivity of Data Processing with Development Projects using the Function Point Technique.* Université du Québec à Montréal. 1988.

Galea, S., *The Boeing Company: 3D Function Point Extensions, V2.0, Release 1.0*, Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.

Grady, R. B., *Practical software metrics for project management and process improvement* Prentice Hall, New Jersey, 1992, 270 pages.

Hetzel, B., *Making Software Measurement Work*, QEB Publishing Group, 1993, 290 pages.

IEEE, *IEEE Standard Computer Dictionary: A compilation of IEEE Standard Computer Glossaries*, IEEE Std 610-1990, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1990.

IFPUG (1994). *Function Point Counting Practices Manual, Release 4.0*, International Function Point Users Group - IFPUG, Westerville, Ohio, 1994.

Illingworth, V., (1991) (editor), *Dictionary of Computing*, Oxford University Press, 3rd edition, 1991, 510 pages.

Ince, D. C., *History and industrial applications*, in Fenton, N.E., *Software Metrics: A Rigorous Approach*, Chapman & Hall, UK, 1991, 337 pages.

Jones, C., *A Short History of Function Points and Feature Points*, Software Productivity Research, Inc., Cambridge, Mass, 1988.

Jones, C., *Applied Software Measurement - Assuring Productivity and Quality*, McGraw-Hill, 1991, 493 pages.

Jones, C., *Applied Software Measurement - Assuring Productivity and Quality*, McGraw-Hill, 1996, 618 pages.

Kan, S. H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1993, 344 pages.

Kemerer, C. F., *Reliability of function point measurement: a field experiment*, Communications of the ACM, vol. 36, pp. 85-97, 1993.

Kitchenham, B., *Making Process Predictions*, in Fenton, N.E., *Software Metrics: A Rigorous Approach*, Chapman & Hall, UK, 1991, 337 pages.

Laplante, P., *Real-Time Systems Design and Analysis: An Engineer's Handbook*, The Institute of Electrical and Electronics Engineers Inc., New York, NY, 1993, 339 pages.

Stankovic, J. A. and Ramamritham, K., *Tutorial Hard Real-Time Systems*, IEEE Computer Society Press, Washington D.C., 1988, 618 pages.

Whitmire, S. A., *3-D Function Points: Scientific and Real-Time Extensions to Function Points, Proceedings of the 1992 Pacific Northwest Software Quality Conference*, 1992.

For more information on FFP:

Denis St-Pierre M.Sc., C.F.P.S. Denis.St-Pierre@CRIM.CA
Marcela Maya M.Sc., C.F.P.S. Maya.Marcela@uqam.ca
Alain Abran Ph.D. abran.alain@uqam.ca
Jean-Marc Desharnais M.Sc., C.F.P.S. Desharnais.Jean-Marc@uqam.ca

