FUNCTIONAL SIZE OF REAL TIME SOFTWARE: OVERVIEW OF FIELD TESTS

Serge Oligny[1], Alain Abran[1], Jean-Marc Desharnais[2], Pam Morris[3]

ABSTRACT

In most software cost estimation models, software size is the key cost driver. Such models use either a technical measure of software size, based on lines of code, or alternatively a functional size measure which can be known earlier in the software life cycle. However, even though Function Points is the most widely used functional size measure in the MIS domain, practitioners have often pointed out its limitations for measuring the size of real-time or embedded software; therefore, it is not currently considered as an adequate input parameter for estimating real-time software effort.

In 1997, a new extension to Function Points (referred to as Full Function Points – FFP) was introduced for measuring the functional size of real-time software in order to address the most obvious weaknesses of IFPUG's Function Points while retaining compatibility with traditional Function Points for MIS software. Full Function Points was also recently accepted as a new measurement standard for real-time software by the International Software Benchmarking Standards Group.

This paper reports on the key concepts of this extension, as well as on the preliminary results of the measurement field tests carried out in different organizations. The ability of FFP to adequately capture the functional size of real-time software is illustrated by FFP and FPA measurements taken on the same software products. Preliminary results, using additional collected data, to support exploratory analysis of the unit effort and schedule delivery date based on FFP are also presented.

1. CONTEXT

Measures can be used to quantify software products as well as the process by which they are developed. Once these measures are obtained, they can be used to build cost estimation models and productivity models [1], for instance. In this perspective, a key measure is the size of a software product. There are basically two kinds of size measures: technical measures and functional measures. On the one hand, technical size measures are used to quantify software products and processes from a developer's point of view. These can be used in efficiency analysis to improve the performance of the design, for instance. On the other hand, functional measures are used to quantify software products and services from a user's perspective. Being independent of technical development and implementation decisions, functional measures can therefore be used to compare the productivity of different techniques and technologies. In this context, major organizations frequently use functional size measurement methods to quantify the software products included in their outsourcing contracts.

Function Points Analysis (FPA), first introduced in 1979 by Allan Albrecht of IBM [2], is an

[1] Serge Oligny and Alain Abran are respectively Director – Technological innovations and Director of UQAM Software Engineering Management Research Laboratory in Montréal (http://www.info.uqam.ca/Labo_Recherche/lrgl.html) , they can be reached at oligny.serge@uqam.ca and abran.alain@uqam.ca

[2] Jean-Marc Desharnais is Managing Partner and Principal Consultant with Software Engineering Laboratory in Applied Metrics (http://www.lmagl.qc.ca/), he can be reached at desharnais.jean-marc@uqam.ca

[3] Pam Morris is Director Consulting and Training with Total Metrics (http://www.totalmetrics.com), she can be reached at Pam.Morris@Totalmetrics.com

example of a functional size measure. FPA measures the size of software in terms of the functionality it delivers to end-users by quantifying such objects as inputs, outputs and files. Since FPA was developed for an MIS[4] environment, its concepts, rules and guidelines are adapted to software typical of that environment. Consequently, FPA has gained a wide audience in this specific area of software applications and is now being used extensively to, among other things, analyze productivity and estimate project effort and costs [3, 1, 4, 5]. However, FPA has been criticized as not being universally applicable to all types of software [6, 7, 8, 9, 10, 11, 12, 13, 14]. Here is how D.C. Ince describes the FPA 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.*" [10, page 283)

When FPA is applied to such software, it does, of course, generate measurement results, but these do not constitute an adequate size measurement. Therefore, there is currently no FPA equivalent for real-time software that would allow meaningful productivity benchmarking or the development of estimation models based on an adequate functional size of real-time software.

2. REAL-TIME SOFTWARE LIMITATIONS OF FPA

2.1 *Data limitations*

Two kinds of control data structure are found in most types of software: multiple occurrence groups of data and single occurrence groups of data. Multiple occurrence groups of data display more than one instance of the same type of record[5]. Single occurrence groups of data display one and only one instance of a data type. Real-time software typically contains a large number of single occurrence data types that are difficult to group into IFPUG-defined logical groups. An extension rule is necessary to adequately measure single occurrence data types.

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 from one process to another. A real-time functional measurement method must take into account the fact that some processes have only a few sub-processes while others have a large number of them. The following two examples, taken from embedded automotive software, illustrate this point.

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

Suppose the main purpose of a process is to turn on an engine cooling system when required. A sensor provides the engine's temperature (sub-process 1). This temperature is compared to the overheating threshold temperature (sub-process 2). Finally a "switch-on" message could be sent to the cooling system if required (sub-process 3).

In this example, the temperature control process consists of 3 sub-processes (see Table 1).

[4] MIS: Management Information Systems.

[5] An engine control application, for instance, 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 data type for each cylinder). In other words, the cylinder data type is repeated more than once.

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

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

The main purpose of this second process is to turn on an inboard "engine" alarm when required. Fifteen engine sensors (all different) send data to a 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; there is a unique file for each kind of sensor (15 more sub-processes, 1 unique sub-process for each kind of sensor). Depending on a number of conditions, an "engine" alarm mounted in the dashboard may be switched on (1 sub-process).

In this example the engine diagnostic process consists of 31 sub-processes (see Table 2).

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

In IFPUG FPA measurement definitions, the basis for measuring transactions is its unique definition of an 'elementary process' which must leave the application in a consistent state. Therefore, both examples 1 and 2 constitute a single IFPUG-defined elementary process each [15].

Again, according to IFPUG FPA rules, approximately the same functional size would be assigned to both processes although, intuitively, most real-time practitioners would perceive them as being of different functional size.

3. FULL FUNCTION POINTS CORE CONCEPTS

Full Function Points was designed to measure the functional size of both MIS and real-time software. Since FFP is an extension of the IFPUG FPA measurement method, most IFPUG rules are included as a subset for the measurement of the MIS functions within a measured software product.

The key measurement concepts added in the Full Function Points extension to measure the functional size of the real-time or embedded type of software, are summarized below:

- FFP measures the functional size from a full functional perspective instead of from a narrower external user point of view. FFP measures the functionality required to be delivered by a process to the users (human or mechanical) of that process, not just the functionality experienced directly by its human users. Sub-processes that read and write the data to and from data groups are included in the measurement of functionality in addition to the sub-processes required to receive data (entry) and extract data (exit).

- The functional size of a process is the sum of the size of the individual sub-processes. There is no limit to the size assigned to one specific process, since each individual sub-process is measured.
- Changed functionality is measured at the level of the sub-processes. Only part of the process (identified by the modified sub-processes) is credited for the change.

These concepts are materialized by a fully documented method through detailed rules and procedures based on the identification of the following function types .

- External Control Entry (**ECE**): an ECE is a unique sub-process. It is identified from a functional perspective[6]. An ECE manipulates control data coming from outside the application's boundary. It is located at the lowest functional level of a process and acts on only one group of data. Consequently, if a process enters two groups of data, there are at least 2 ECEs. Referring to the engine diagnostic process (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, 15 ECEs are counted in this example.

- External Control Exit (**ECX**): an ECX is a unique sub-process. It is identified from a functional perspective. An ECX manipulates control data going outside of the application boundary. It is located at the lowest functional level of a process and acts on only one group of data. Consequently, if a process exits two groups of data, there are at least 2 ECXs. Referring to the engine diagnostic process (example 2), the sub-process that sends a message to the dashboard is such 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 located at the lowest functional level of a process and acts on only one group of data. Consequently, if a process reads two groups of data, there are at least two ICRs. Referring to the engine diagnostic process (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, this process embeds 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 located at the lowest functional level of a process and acts on only one group of data. Consequently, if a process writes on two groups of data, there are at least 2 ICWs. For the purposes of illustration, suppose the engine diagnostic process (example 2) 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, according to the FFP measurement technique.

- Multiple data type groups, which can be either read or updated by the real-time or embedded processes. These are similar to the Internal Logical files and External Interface files measured by the FPA technique.

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

- Single data type (single occurrence) groups. These data groups may be created or updated by the processes (Updated Control Group - **UCG**) or only read by the processes (Read-only Control Group - **RCG**). The single occurrence data groups contain all instances of single control values used by the processes. There may be only one instance of a UCG or RCG per measured application.

4. FEEDBACK FROM INITIAL FIELD TESTS

Initial field tests of the FFP measurement method were conducted prior to its initial release in 1997 [16]. One set of field tests was conducted by the research team that co-authored the FFP measurement method and another set of tests was conducted by an industrial partner without the assistance of the research team.

In the first field test, conducted by the research team, three real-time or embedded software products were measured using both FFP and IFPUG's FPA measurement methods. The purpose of the test was to compare the functional size obtained with both method. The software products measured were taken from the operational portfolio of organizations in the USA and in Canada. Results show that FFP provides a functional size that is close to FPA when there are few sub-processes within each process. Furthermore, for processes displaying a significantly larger functional size, there is a considerably larger number of embedded sub-processes and the functional size provided by the FFP measurement method is significantly larger than the functional size provided by the FPA measurement method.

In the second set of field tests, conducted by an industrial partner from Japan without the assistance of the research team, the FFP measurement method was used exclusively on real-time operational software product. The purpose of the tests was to evaluate the FFP measurement method for relevance and usability. Results obtained from this organization's practitioners indicate that:

- Concepts and measurement procedures in the FFP Counting Manual were relatively clear and easy to understand. It was not difficult to count without the assistance of an FFP specialist.
- In the larger of these independent tests, FFP measured 79 processes out of the 81 expected to be measured with an adequate functional size measurement method. At the end of this field test, the industrial partner concluded that FFP failed to take into account 2 of the 81 processes because the current design of FFP does not measure processes containing only internal algorithms. The FFP measurement coverage rate was therefore 97% of the overall functionality that they felt should be included in the measurement of the functional size of their real-time software.

Furthermore, all these tests indicated that the effort required to measure the functional size of an application using the FFP measurement method is similar to the effort required for measuring it using the FPA rules. Even though more function types have to be measured with FFP, these function types were more easily identified. Indeed, the application specialists seemed to require less assistance from function point experts when measuring with FPP than when using FPA, so the identification of more function types did not increase the measurement effort during those field tests.

5. INTERNATIONAL RECOGNITION FROM ISBSG AND NATIONAL ASSOCIATIONS

The International Software Benchmarking Standards Group (ISBSG) has recently recognized the FFP measurement method at the interim status, meaning that this organization will accept projects for its benchmarking repository for which functional size has been measured using FFP.

This recognition was based on the unanimous votes of the seven national organizations currently on the board of the ISBSG.

The interim status is recognized by the ISBSG based on the following criteria:

- A clear need has been demonstrated for the proposed functional size measurement method,
- The promotion of the method is not driven by commercial interests and the method does not contravene ISO requirements for a functional size measurement method (ISO-14143-1),
- Key concepts and all rules and procedures of the functional size measurement method must be documented. Examples must be provided for the rules. All documentation must be available, at least in English,
- A user group is in place, providing for the distribution of the documented standards and improvements to the standards, ensuring that only one current official version exists, implementing and controlling a certification process, maintaining a bibliography of relevant publications and promoting the use of the functional size measurement method in the industry.

6. ADDITIONAL FIELD TEST

The measurement results presented in this section were collected in field tests subsequent to the release of the FFP measurement method. Three of the industrial sites were in North America, and the fourth in Asia; seven of the software products measured were classified as telecommunications software, another as power utility software, and a fourth as military software. All measurement procedures were executed by the same measurement expert with twelve years of expertise in functional size measurement, thereby eliminating inconsistencies across measurement experts.

The data sets available allow two types of exploratory analysis:

- Exploring a software sizing comparison between FPA and FFP
- FFP: some preliminary economic results

6.1 Exploring a software sizing comparison between FPA and FFP

Table 3 presents measurement results of seven distinct software products. These products were measured using both the IFPUG 4.0 rules and the Full Function Points rules. They all come from the same organization. Out of these seven software products, four are typical real-time systems, two are typical MIS systems and one is mostly MIS but includes some real-time functionality. Distinction between real-time and MIS software products is based on the knowledge of functional experts and practitioners within the organization.

The results presented in Table 3 suggest the following observations:

- The size results are similar when both methods are applied to MIS-type software products, as demonstrated by measurements of products E and G,
- The software product “C” could not be measured at all using FPA because the functionality it delivered could not be reliably categorized into FPA function types, nor did it fit the definition of elementary process according to IFPUG 4.0 rules.
- The size difference is significant in all cases where the software product has complex real-time processes. Products A, B, C and D are examples of this. The FFP size is considerably greater than the FPA size.

Product	Product type	FPA size	FFP size	Size difference	Size difference (%)
A	Real-time	210	794	584	74%
B	Real-time	115	183	68	37%
C	Real-time	0	2604	2604	100%
D	Real-time	43	318	275	86%
E	Mostly MIS	764	791	27	3%
F	MIS (batch)	272	676	404	60%
G	MIS	878	896	18	2%

Table 3 – Software sizing comparison between FPA and FFP

6.2 FFP: some preliminary economic results

Table 4 presents size results for three real-time software products. Each of them comes from a different organization. In these instances, it was also possible to gather two key process measures: the actual effort (person-hours) expended to build and deliver each product and the duration of this process in elapsed months. From these data, two economic ratios are calculated: the process unit effort, expressed in person-hours/FFP and the schedule delivery rate, expressed in FFP/elapsed months.

Product	FFP size	Effort (person-hours)	Duration (elapse months)	Unit effort (phs/FFP)	Schedule delivery rate (FFP/month)
H	205,4	3 913	26	19,1	7,9
I	138,0	6 580	16	47,7	8,6
J	198,0	7 448	14	37,6	14,1

Table 4 – Key economic ratios derived from FFP size measurements

Unfortunately, this sample size is still too small for comparison purposes and for further analysis. However, further field tests are being planned to expand this data set.

7. CONCLUSION

Based on the shortcomings of the traditional Function Points measurement method when applied to real-time or embedded software, Full Function Points was proposed in 1997 as a functional size measurement method specifically designed for these types of software. The key concepts of this new measurement method have been presented.

Initial field test results have indicated: a) the adequacy of the FFP measurement method in terms of a high functional coverage ratio when applied to the software it is intended to measure, b) the nature of the difference observed in the measured functional size between the FFP and the FPA measurement methods when they are both applied to the same real-time or embedded software, and c) the recognition, by practitioners, of an adequate degree of applicability in typical industrial environments where these types of software are developed and maintained.

Furthermore, this new measurement method has been recognized by the ISBSG, an international software benchmarking organization, based on an extensive set of criteria.

Additional field test results have been presented which: a) provide further support for the observed difference between the FFP and the FPA measurement method when applied to real-time or embedded software, and b) provide an initial indication of the magnitude of key process ratios based on the FFP functional size measure, thus allowing exploratory research to be pursued by supporting the formulation of preliminary hypotheses.

8. ACKNOWLEDGEMENTS

Authors wish to thank Nortel, Bell Canada and Hydro-Québec as well as other industrial partners for providing project funds, industrial data and valuable feedback from real-time software practitioners.

UQAM's Software Engineering Management Research Laboratory co-sponsored the research leading to this publication. The Laboratory is supported through a partnership with Bell Canada. Additional funding for the Laboratory is provided by the Natural Sciences and Engineering Research Council of Canada.

9. BIBLIOGRAPHY

[1] 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.

[2] Albrecht, A.J. (1979), Measuring Application Development Productivity, Proceedings of Joint Share Guide and IBM Application Development Symposium, October, 1997, pp. 83-92.

[3] 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.

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

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

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

[7] 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.

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

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

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

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

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

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

[14] 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.

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

[16] Maya M., Abran A., Oligny S., St-Pierre D., Desharnais J. M., "Measuring the Functional Size of Real-Time Software", Proceeding of the 9th European Software Control and Metric Conference, Rome Italy, 1998