A METHOD FOR MEASURING THE FUNCTIONAL SIZE OF EMBEDDED SOFTWARE

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

[1]: UQAM's Software Engineering Management Research Laboratory (http://www.lrgl.uqam.ca/), Montréal, oligny.serge@uqam.ca and abran.alain@uqam.ca.
[2]: SELAM (http://www.lmagl.qc.ca/), Montréal, desharnais.jean-marc@uqam.ca.

Abstract

Software has become a key component of most automated process control devices. It offers a high degree of flexibility in adjusting the behavior of those devices. Proper management of the development and maintenance of process control software is therefore a key issue be it for reasons related to internal organization performance or for benchmarking against the best in the industry.

Measures are essential for quantitative management; they are needed to analyze both the quality and the productivity of the software processes. For instance, technical measures are useful to quantify the performances of a product's design through efficiency analysis. On the other hand, functional measures are needed for quantifying products from a user perspective and are well suited for productivity analysis. For instance process control projects that exhibit cost or schedule difficulties originating from the work related to the software components could benefit from functional measures to alleviate such difficulties. Functional measures are independent of technical and implementation decisions, they can be used to compare the productivity of different techniques and technologies.

This paper presents a measure of a fundamental dimension of software: its size. Although software functional size measures are not new, the most popular one, called function points, have often been described as ill-suited for the quantification of real-time or embedded software for a number of reasons. The measure presented in this paper, called Full Function Point (FFP), has been specifically designed for real-time or embedded type of software.

The paper explains the criteria for designing an adequate software functional size measure. The characteristics of FFP and the associated measurement method is then presented along with references to relevant documentation. Results are introduced; these results demonstrate that, from a practitioner perspective, Full Function Point is a functional measure that adequately captures the perceived functional size of real-time or embedded software. The paper conclude on the evolution perspectives for this size measure.

1. Introduction

Due to the important role played by software in today's process control devices, the use of measures to manage the software development processes and products is recognized as an essential element in effective software management. One important measure is the size of the software product. There are basically two kinds of software size measures: technical measures and functional measures. Technical measures, like the number of lines of code, size software products from the developer's point of view. They are useful for conducting efficiency analysis, for instance. Functional measures size software products from the user's point of view. Being independent of technical development and implementation decisions, they are useful for performing productivity analysis and for building estimation models.

Function Point Analysis (FPA), first introduced by Allan Albrecht in 1979 [1], is an example of a functional size measure. FPA measures the size of a software product in terms of the functionality it delivers to the users, taking into account objects such as inputs, outputs and files. FPA is now widely used in the MIS domain, where it is becoming the de facto standard. FPA is being used extensively, among other things, to analyze productivity and to estimate software costs. However, FPA has not enjoyed the same degree of acceptance in the domain of embedded real-time software. In fact, a literature review has shown that the data sets used in most publications originate from MIS software applications. Several authors concur that when FPA is applied to real-time software, the results do not constitute an adequate size measurement [2], [5], [9], [13]. Currently, there is no FPA-equivalent technique for the real-time domain.

Six previous attempts to adapt FPA to real-time software have been identified in the literature: Feature Points [5], Mark II [12], Asset-R [9], 3D Function Points [13], Application Features [8] and IFPUG Case Study 4 [4]. These attempts can be classified into five types of solutions: introducing new components to be measured in addition to those already proposed by FPA (Feature Points, Asset-R, 3D Function Points); adjusting the final function point count (Asset-R); estimating the final function point count (Application Features); continuous adjustment of the matrices used to assign points to the different components (Mark-II) and the orthodox approach (IFPUG Case Study 4). However, it seems that none of these approaches has succeeded in gaining

market acceptance, even though some of them were proposed a decade ago.

This paper presents the results of a research project carried out at the Software Engineering Management Research Laboratory at the Université du Québec à Montréal in cooperation with the Software Engineering Laboratory in Applied Metrics (SELAM) to propose a functional size measure specifically adapted to the functional characteristics of real-time software. The proposed measurement method, called Full Function Points (FFP) [10], is based on the observation that real-time software has the following specific transactional and data characteristics:

- Transactional characteristics: The number of sub-processes found in a real-time process varies substantially. By contrast, processes in the MIS domain display a more stable number of sub-processes.
- Data characteristics: There are usually a large number of single-occurrence control variables in a real-time software product. These variables are characterized by the fact that there is only one occurrence of them in the whole application (for example, the status of a physical device).

To take into account these characteristics, FFP introduce six components to be measured: four components to take into account the transactional characteristics and two components to take into account the data characteristics. FFP define detailed procedures and rules to identify and weight these components.

Furthermore, FFP were designed to offer quality characteristics from a measurement perspective, including:

- **Relevance**: Practitioners perceive that the measurement technique adequately measures the functional size of their applications.
- **Instrumentation**: Instrumentation means the transformation of the preliminary specifications of a measurement technique into a set of well-documented procedures. These procedures will ensure the application of the measurement technique in a consistent manner across contexts, culture and time, and independently of the designers of the technique. An example of instrumentation is the FPA Counting Practices Manual [3]. Instrumentation is an essential factor in achieving repetitiveness, which means that different individuals, in different contexts and at different times and following the same measurement procedures, will obtain measurement results that are relatively similar, have been obtained with minimal judgment and can be audited.
- **Practicality and applicability**: The measurement technique is based on current software design practices, as observed in the industry, and on the content of the documentation of user requirements from a functional perspective.
- **Transferability**: The measure should allow transferability to a standards-setting and monitoring body.

FFP were field-tested in different organizations. The feedback obtained from these organizations was positive [6] [7] [11]: for these organizations, FFP results represent a more relevant measure of the functional size of their real-time software than traditional Function Points did. These organization's practitioners believe that, using Full Function Points, the functional size was measured objectively, precisely and in an auditable manner. Furthermore, they believe that someone else with the same set of rules would come up with the same results. The concepts, counting rules and procedures in the FFP counting manual were deemed relatively clear and easy to understand. The effort required to measure an application with FFP was judged to be similar to that required using traditional Function Points.

In this paper, the key concepts of FFP are described and the results of field tests conducted are discussed. Section 2 introduces the basic concepts of FFP. Section 3 describes the field tests conducted and highlights the main results. The final section presents some observations and topics for further research.

2. Full Function Points Basic Concepts

To measure the specific functional characteristics of real-time software adequately, it is necessary to consider both the sub-processes performed by each control process and the single-occurrence control data. Full Function Points (FFP) introduces data and transactional function types accordingly. Furthermore, the concept of boundary allows to distinguish between the software components that are included in the measured product from the ones which can be found in its surroundings.

- **Boundary:** The boundary of a piece of software is the functional frontier allowing the segragation of the components that are included in the measured product (inside the boundary) and the components that are part of the environment of the measured product (outside the boundary).
- **Data function types**:
  Updated control group (UCG): A group of control data updated by the software being measured. Control data means data used by the application to control, directly or indirectly, the behavior of an application or of a mechanical device.
  Read-only control group (RCG): A group of control data used, but not updated, by the software being measured.
- **Transactional function types**:
  Within the boundary of the software being measured, all processes are identified. Within each identified processes the following four transactional function types are identified.
  External Control Entry (ECE): A sub-process that receives control data coming from outside the application's boundary. For instance, each sub-

process receiving data coming from one sensor is considered an ECE.

External Control Exit (ECX): A sub-process that sends control data outside the application boundary. For instance, each sub-process sending a signal to a device's control panel display is considered an ECX.

Internal Control Read (ICR): A sub-process that reads a group of control data. For instance, each sub-process that obtain a threshold value from a group of control data is considered an ICR.

Internal Control Write (ICW): A sub-process that writes to a group of control data. For instance each sub-process updating current value of key variables inside the measured application is considered an ICW.

The FFP transactional function types are identified at the sub-process level (Figure 1), instead of the process level as it is done with traditional FPA. It can thus be said that FFP take into account a finer level of granularity, the sub-process level, while FPA only considers the process level. A finer level of granularity is important in real-time software since, unlike MIS processes, real-time processes display a variable number of sub-processes.

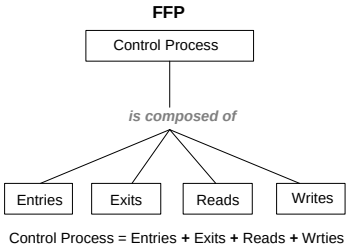

Figure 1: *FFP transactional function types*

The identification of the transactional function types of an application includes the following major steps [10]:

1. Identify the boundary of the software application to be measured;
2. Identify and assign points to each data control groups;
3. Identify the different processes performed by an application from a functional perspective;
4. From a functional perspective, identify the different sub-processes performed by each process;
5. For each sub-process, determine whether to count it as an Entry, Exit, Read or Write function type, according to their definitions and counting rules;
6. Assign the corresponding points to each sub process.
7. Sum up sub process points for each process, then for the entire application.
8. Functional size is the sum of processes points (step 7) and points allocated to data (step 2).

The complete set of FFP concepts, definitions, counting procedures and rules, as well as a counting example, can be found in the FFP Counting Practices Manual [10].

3. FFP Field Tests

3.1 Initial Field Tests

Initial field tests of the FFP measurement method were conducted prior to its initial release in 1997 [14]. 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 (processes that do not refer at all to any data groups). 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.

3.2 Additional Field Tests

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 the Pacific region; 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 experience in functional size measurement, thereby eliminating inconsistencies across measurement experts.
The data sets available allow two types of exploratory analysis:

- Further comparison between FPA and FFP
- FFP: some preliminary economic results

3.2.1 Exploring a software sizing comparison between FPA and FFP

Table 1 presents measurement results of seven distinct software products. These products were measured using both the IFPUG 4.0 rules and the Full Function Points measurement method. 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 1 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	**Type**	**Size(FPA)**	**Size(FFP)**
A	Real-time	210	794
B	Real-time	115	183
C	Real-time	N/A	2604
D	Real-time	43	318
E	Mostly MIS	764	791
F	MIS(batch)	272	676
G	MIS	878	896

***Table 1** –Sizing comparison between FPA and FFP*

3.2.2 FFP: some preliminary economic results

Table 2 presents size results for five software products. Products H, I, J and K are real-time software and product L is MIS software. Projects data comes from four different organizations. 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	Size[1]	Effort[2]	Duration[3]	U.E.[4]	S.D.R.[5]
H	205	3 913	26,0	19,1	7,9
I	138	6 580	16,0	47,7	8,6
J	198	7 448	14,0	37,6	14,1
K	837	2 500	19,0	3,0	44,1
L	608	438	2,7	0,7	225,2

Notes:

1: Software product size in FFP
2: Development effort in person-hour
3: Development project duration in calendar months
4: Unit effort in person-hour per FFP
5: Schedule delivery rate in FFP per months

***Table 2** – Key economic ratios derived from FFP size measurements*

This sample size is still too small for comparison purposes or for statistical analysis. Further field tests are being conducted to expand this data set.

4. 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 (http://www.isbsg.org.au/index.html), 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 and range 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.

5. Acknowledgments

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.

6. References

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

[2] Galea, S., "The Boeing Company: 3D Function Point Extensions, V2.0, Release 1.0", Seattle, WA: Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.

[3] IFPUG, "Function Point Counting Practices Manual, Release 4.0", International Function Point Users Group - IFPUG, Westerville, Ohio, 1994.

[4] IFPUG, "IFPUG Case Study 4", International Function Point Users Group - IFPUG, Westerville, Ohio, 1997.

[5] Jones, C., "Applied Software Measurement - Assuring Productivity and Quality, McGraw-Hill, New York, 1991, 493 pages.

[6] Maya, M., St-Pierre, D., Abran, A. and Desharnais, J-M, "Full Function Points: Function Points Extension for Real-Time Software - Counting Experiments at Nortel", Confidential Reports, Université du Québec a Montréal, March and May, 1997.

[7] Maya, M., St-Pierre, D., Abran, A., and Desharnais, J-M, "Mesure de la taille fonctionnelle des logiciels temps réel - Comptage chez Hydro Quebec", Confidential Report, Université du Québec à Montréal, Avril, 1997.

[8] Mukhopadhyay, T. and Kekre, S., "Software Effort Models for Early Estimation of Process Control Applications", IEEE Transactions on Software Engineering, Vol. 18, No. 10, October 1992, pp. 915-924.

[9] Reifer, D. J., "Asset-R: A Function Point Sizing Tool for Scientific and Real-Time Systems", Journal of Systems and Software, Vol. 11, No. 3, March 1990, pp. 159-171.

[10] St-Pierre, D., Maya, M., Abran, A. et Desharnais, J-M, "Full Function Points - Counting Practices Manual", Technical Report 1997-04, Software Engineering Management Research Laboratory, Université du Québec à Montréal (UQAM), September 1997, 49 pages.

[11] St-Pierre, D., Abran, A., Araki, M., and Desharnais, J-M, Adapting Function Points to Real-Time Software", presented at IFPUG 1997 Fall Conference, International Function Point Users Group, Scottsdale, Arizona, September, 1997.

[12] Symons, C.R., "Function Point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, Vol. 14, No. 1, January 1988.

[13] Whitmire, S. A., "3-D Function Points: Scientific and Real-Time Extensions to Function Points", Proceedings of the 1992 Pacific Northwest Software Quality Conference, Portland, OR, 1992.

[14] 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