ADAPTING FUNCTION POINTS TO REAL-TIME SOFTWARE

by Alain Abran, Marcela Maya, Jean-Marc Desharnais, and Denis St-Pierre

INTRODUCTION

Software measurement is essential for effective software management. Measures can be used to quantify the software product as well as the process by which it is developed. Once these measures are obtained, they can be used, for example, to build cost estimation models, productivity models, and cost-benefit models.

One important measure is the size of a software product. There are basically two kinds of size measures: technical measures and functional measures. Technical measures are used to measure software products and processes from a developer's point of view. They can be used in efficiency analysis to improve, for example, the performance of designs. Functional measures are used to measure 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.

Function points, first introduced by Allan Albrecht of IBM in 1979 [1], are an example of a functional size measure. FPs measure the size of software in terms of its delivered functionality, measuring such objects as inputs, outputs, and files. Since FPs were developed in an MIS environment, their concepts, rules, and guidelines are adapted to MIS-oriented software. As a result, FPs have gained a wide audience in this specific area of software applications and are now being used extensively to, among other things, analyze productivity and estimate project costs. Nevertheless, FPs have not achieved the same degree of acceptance in other software areas.

In this article, we present a measurement technique for extending FPs to take into account functional characteristics specific to real-time software. We call this new measure the Full Function Point (FFP). The FFP was designed with the objective of retaining the actual FP quality characteristics from a measurement technique perspective, including instrumentation, repetitiveness, and applicability. This extension takes into consideration current industry practices in the design of real-time software and what is currently documented regarding the user requirements from a functional perspective.

The FFP is based on the observation that real-time software has the following specific transactional and data characteristics:

★ Transactions. The number of subprocesses of a real-time process varies substantially. By contrast, processes of the same type in the MIS domain have a more stable number of subprocesses.

★ Data. In a real-time software system, there exists a large number of single control data; that is, data characterized by the fact that there is one and only one occurrence of each data point in the whole application. These data are used to control, directly or indirectly, the behavior of an application or a mechanical device.

New data and transactional function types were therefore introduced to take into account these characteristics: four control transactional function types (Entry, Exit, Read, and Write) and two control data function types (Updated Control Group and Read-Only Control Group).

To verify whether or not the FFP has met its stated objective of measuring the user functional requirements and the quality characteristics required for measurement techniques, field tests were conducted. The results of the field tests were positive: according to the real-time specialists present at the field tests, the FFP has met its stated objective of measuring the user's functional requirements adequately. These specialists believe that this has been done objectively, precisely, and in an auditable manner, and that someone else with the same set of rules would come up with the same results. They also believe that FFP counting procedures and rules are based on practices that are really being documented at the present time.

The FFP is now being promoted to IFPUG and to other national software metrics associations. The FFP was presented at the IFPUG 1997 Fall Conference [16], and three public reports are now available: an FFP definition and concepts [14], FFP counting procedures and rules [15] and a complete case study [11]. In the remainder of this article, we will present the key highlights of the new FFP measure.

LITERATURE REVIEW

For this project, we carried out an extensive literature review on function points. Most of the publications on FPs do not deal with their application to real-time software, and the data sets used in most publications originated from MIS software applications. Furthermore, several authors [3, 9, 13, 18] concur that FPs, with their current structure, are not adequate for measuring processor-intensive software with a high number of control functions and internal calculations.

Six previous attempts to adapt FP to real-time software have been identified: Feature Points [9], Mark II [17], Asset-R [13], 3D Function Points [18], Application Features [11], and IFPUG Case Study 4 [6]. These attempts can be classified into five types of solutions:

★ Addition of new function types (Feature Points, Asset-R, 3D Function Points)

★ Adjustment of the final function point count (Asset-R)

★ Approximation of the final function point count (Application Features)

★ Continuous adjustment tables (Mark II)

★ Orthodox approach (IFPUG Case Study 4)

It seems, however, that none of these approaches has succeeded in gaining market acceptance.

As the 3D Function Points proposal had often been referred to as a potential candidate for real-time software in the FP electronic discussion forum,[1] we decided to pretest it on some industrial applications in order to obtain empirical observations and, perhaps, use it as the starting point for the project.

We measured two different software applications from two different domains (telecommunications and power supply) and encountered the following problems:

★ The concepts and a few examples in [18] and the procedures and rules in [3] were not detailed enough to allow the identification, without ambiguity, of the proposed new function types (Transformations, States, and Transitions).

★ For the identification of the proposed States and Transitions function types, documentation for the finite state machines (FSM) was a prerequisite. However, at the industrial sites where this proposal was tested, this type of documentation was

[1]FPlist: function point listserv (function.point.list@crim.ca)

not in use. Based on feedback from many real-time designers at other organizations, we discovered that even though practitioners all know about this type of notation system, they did not document FSM. To our surprise, we found that FSM was not an industry practice and that it was felt that a measurement technique based on such a documentation requirement would not be practical. The feedback from these field tests and from other contacts gave us an indication that the industry might be very reluctant to endorse a measurement technique based on the documentation of FSM, as our contacts were not even doing it for the design of their own software.[2]

PROJECT OBJECTIVES AND PROJECT PLAN

The objective of the project was to propose an extension to FPs that, on the one hand, would take into account the specific functional characteristics of real-time software, and, on the other hand, would retain the quality characteristics of FP as a "de facto" standard for a measurement technique in the MIS domain.

Since "real-time software" is one of those generic expressions that often mean different things to different people (indeed, there is not even a consensus on a single definition among researchers in the field), here we use this expression to refer to software with the characteristics described in the following definitions:

- "A system in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process" [4].
- "Any system in which the time at which the output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness" [7].

Our focus was therefore on software that reacts to an external process or event subject to very tight time constraints.

The proposed FP extension would have to retain the key quality characteristics of the FP from a measurement perspective, such as:

- Relevance. Practitioners perceive that the measurement adequately measures the functional size of their applications.
- Measurement instrumentation. Instrumentation is an essential factor in achieving one of the quality attributes of a good measurement technique: repetitiveness. This means that different individuals, in different contexts, at different times, and following the same measurement procedures, will obtain measurement results that are relatively similar, that have been obtained with minimal judgment, and that can be audited. With FPs, the basis of the measurement instrumentation is the measurement standard published by the International Function Point Users Group,[3] a measurement technique in the public domain.
- Practicality. The measurement technique must be practical: that is, based on current software design practices and on the content of the software documentation.
- Transferability. Preferably, the measure would allow transferability to a standard-setting and monitoring body. (This has been a key concept in measurement and measurement instrumentation since the time of the pharaohs in ancient Egypt!)

We followed the following five major steps in this project:

1. Literature review. This step consisted in conducting a literature review on three aspects: utilization of FPs to measure real-time software, identification of extensions that had been proposed to adapt FP to this type of software, and analysis of these proposals.
2. Proposal of an extension to FPs for real-time software. From the analysis of previous

[2]It is important to note that in more recent publications of 3D function points [3, 19], the states are no longer counted, only the transitions. However, the problem of the unavailability of documentation to count transitions still exists.

[3]Counting Practices Manual: This document clarifies the measurement objectives and perspective and defines very precisely the measurement procedures.

attempts to extend FP, we developed a new approach to adapting FP for the measurement of real-time software.

3. **Field tests of the proposal.** The third step of the project consisted in measuring some real-time software applications. Since this project was supported and financed by three large North American organizations (in the power supply and telecommunications areas) and a Japanese organization (automotive), we measured at least one real-time application from each partner.
4. **Analysis of the counting results.** The fourth step of the project consisted in carrying out an analysis of the measurement results and of the measurement process [8], including:
   - ☆ Perception of the application specialists participating in the measurement sessions with respect to the coherence between the counting results and the size of the application measured
   - ☆ Time required to identify and measure the function types
   - ☆ Difficulty in learning and applying the basic concepts, as well as the counting procedures and rules of the proposal
   - ☆ Comparison between the measurement procedures and results obtained with the proposal and the standard version of FP [5]
5. **Public release.** The last step of the project consisted in preparing a public release of the proposed measurement technique, including a description of the extension proposed, the counting procedures and rules, an overview of the field tests, and comments on the counting results from a measurement instrumentation perspective.

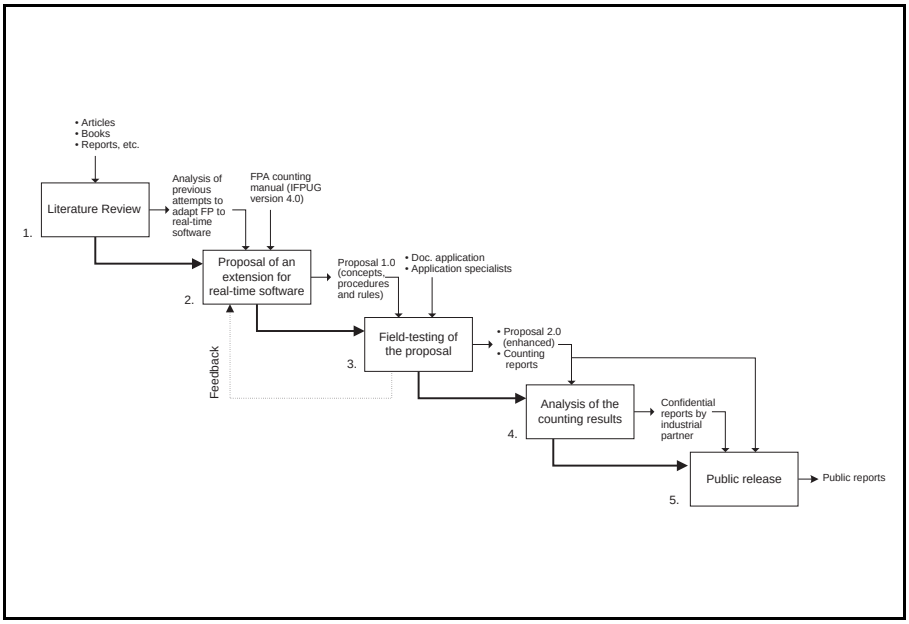

Figure 1: Project steps, inputs, and deliverables

Figure 1 illustrates the project steps, their inputs, and their deliverables. It is important to note that steps 2 and 3 are strongly related and interactive; that is, as the field tests progressed, the proposal was improved based on the feedback obtained from the participants at the counting sessions.

PROPOSAL OF A FUNCTION POINT EXTENSION FOR REAL-TIME SOFTWARE

Characteristics of Real-Time Software

To measure the functional size of software, FPs consider two function types: the transactional function type and the data function type. We identified the following transactional and data characteristics specific to real-time software that are difficult to capture with the current version of FPs:

Transactional Characteristics

According to FP rules, transactional function types are based on the concept of the elementary process.[4] However, FPs do not take into account the number and nature of the steps or subprocesses required to execute the elementary process. According to empirical

[4]Elementary process: the smallest unit of activity that is meaningful to the end user of the business [5]. This elementary process must be self-contained and leave the business of the application being counted in a consistent state.

observations, MIS processes of the same type have a relatively stable number of subprocesses. For example, a process that generates data sent to the user has, in general, four subprocesses:

1. Receive the request from the user
2. Read the information needed
3. Make some calculations
4. Send the information to the user

In the MIS environment, the number of subprocesses does not add any important information to the functional size of a given process.

In real-time software processes, by contrast, the number of subprocesses varies substantially. To illustrate this, consider the following two control processes:

Example 1: An engine temperature control process. The main purpose of this process is to turn on the engine's cooling system when necessary. A sensor enters the temperature in the application (subprocess 1). The temperature is compared to the overheating threshold temperature (subprocess 2). Finally, a turn-on message could be sent to the cooling system if needed (subprocess 3). The application is not in a consistent state until all subprocesses of the temperature control process are completed. The temperature control process has, therefore, three subprocesses (see Table 1). According to standard FP rules, only one transactional function would be identified, because there is only one elementary process.

Example 2: An engine diagnostic process. The main purpose of this process is to turn on an engine alarm when necessary. Fifteen different engine sensors send data to the diagnostic process (15 subprocesses, one unique subprocess for each kind of sensor). For each sensor, the set of external data received is compared to threshold values read from an internal file, with one unique file for each kind of sensor (15 other subprocesses, one unique subprocess for each kind of sensor). Depending on a number of conditions, an alarm on the dashboard may be turned on (one subprocess).

In this example, the engine diagnostic process consists of 31 subprocesses (see Table 2). The application is not in a consistent state until all subprocesses of the diagnostic process are completed. According to IFPUG rules, only a minimum of transactional points would be counted, because transactional function types are based on elementary processes rather than on subprocesses. Therefore, when the IFPUG rules are used, examples 1 and 2 would have approximately the same number of points related to transactions, even though the real-time community would strongly disagree that these two processes have similar functional sizes.

There is a strong case to be made that an adequate functional measure of real-time software should take into account the fact that some processes have only a few subprocesses, while others have a large number of subprocesses.

Process	Subprocess description	Number of subprocesses
Engine control	Temperature entry	1
	Read threshold for comparison	1
	Send turn-on message	1
	Total	3

Table 1: Subprocesses of Example 1

Process	Subprocess description	Number of subprocesses
Engine diagnostic	Sensor data entry	15
	Read thresholds for comparison	15
	Send alarm message	1
	Total	31

Table 2: Subprocesses of Example 2

Data Characteristics

The typical MIS logical file has the following data structure: multiple occurrences of a record, with each record having one or more fields. For example, an engine control application could have a group of data containing information on each cylinder (cylinder number, ignition timing, pressure, etc.). The cylinder record is repeated more than once. This kind of group of data, called a multiple occurrence group of data here, therefore has the same typical structure[5] as an Internal Logical File (ILF) or an External Interface File (EIF) in FP analysis.

[5]IFPUG CS Committee, Function Point Counting Practices, Case Study 1, International Function Point Users Group, Westerville, OH, May 1994.

Real-time software typically contains a large number of single-occurrence control data. These data are used to control, directly or indirectly, the behavior of an application or a mechanical device, and there is one and only one occurrence of each data point in the whole application. For example, the navigational system of an airplane periodically calculates the airplane's position according to data received from external signals. These data are then used to control the airplane's position. The airplane's position is a single control datum with only one occurrence in the whole application, assuming the system does not store old positions. The number of single control data in real-time software can be very important. However, these kinds of data are very difficult to group into an ILF or EIF. An extension of the ILF / EIF rules is therefore necessary to adequately measure single-occurrence control data.

FP Real-Time Extension: Full Function Points

To measure these functional characteristics of real-time software adequately, it is necessary to consider both the subprocesses executed by a control process as well as the single-occurrence control data. The extension we propose, called the Full Function Point (FFP), introduces new transactional function types (Entry, Exit, Read, and Write)[6] and data function types (Updated Control Groups and Read-only Control Groups) to measure these characteristics [14, 15] (see box, p. 38).

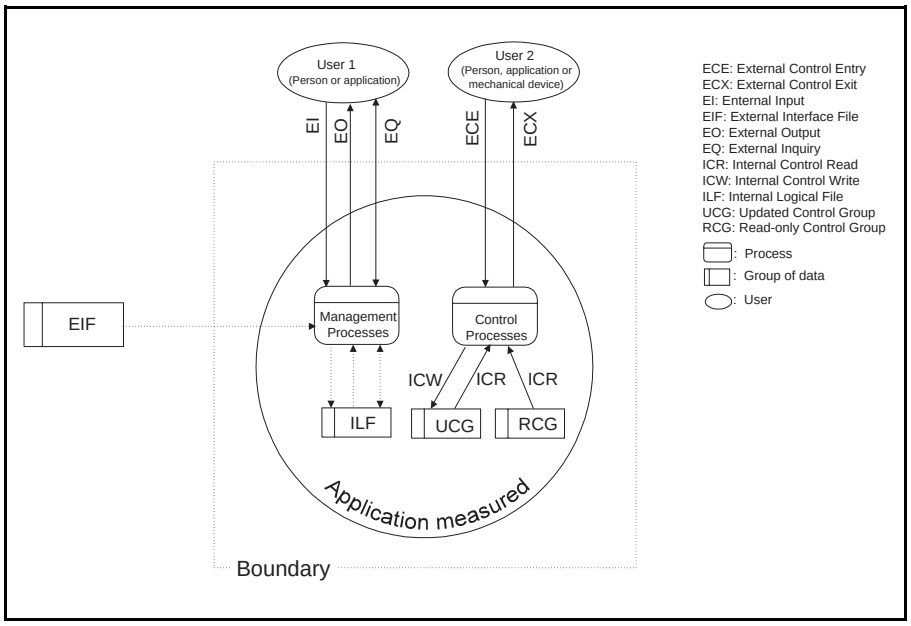

Figure 2: FFP framework

The new function types are only used to measure real-time control data and processes. The other types of data and processes, called management data and processes in the FFP context, are measured with the standard IFPUG rules, as illustrated in Figure 2.

Thus, the unadjusted count of an application using the proposed extension can be expressed as follows:

FFP = Management FP + Control FP
= (FP - Control information) + Control FP

The identification of the new control transactional function types of an application includes the following major steps [15]:

1. Look for the different processes executed by the application from a functional perspective.
2. Determine if the process is a management process or a control process.[7] If it is a management process, the FP transactional function types are identified using IFPUG counting procedures and rules. If it is a control process, the following steps are followed:
   a. Identify the different subprocesses, from a function perspective, executed by the process.

[6]The complete names of the new function types are the following: External Control Entry (ECE), External Control Exit (ECX), Internal Control Read (ICR), and Internal Control Write (ICW). The short name is used in the text for simplicity.

[7]Management process: a process the purpose of which is to support the user in managing information, particularly business and administrative information. Control process: a process that controls directly or indirectly the behavior of an application or a mechanical device.

FFP NEW FUNCTION TYPES

New Control Data Function Types

★ Updated Control Group (UCG): A UCG is a group of control data updated by the application being counted. It is identified from a functional perspective.[8] The control data live for more than one transaction.[9]

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

New Control Transactional Function Types

★ External Control Entry (ECE): An ECE is a unique subprocess identified from a functional perspective. 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 two ECEs. In Example 2, p. 36, 15 sensors send data to the application (control data cross the application boundary). Since there is a unique subprocess for each sensor, there are 15 ECEs.

★ External Control Exit (ECX): An ECX is a unique subprocess 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 two ECXs. In Example 2, the subprocess 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 subprocess 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 two ICRs. In Example 2, 15 unique subprocesses 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 subprocess 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 two ICWs.

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

[9]In Example 2, p. 36, 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.

b For each subprocess, determine whether to count it as an Entry, Exit, Read, or Write function type, according to their definitions and counting rules.

c Assign the corresponding points.

The complete set of FFP counting procedures and rules, as well as a counting example, can be found in [15].

Comparing FFP and FP Function Types

To illustrate the key differences between FFP and FP function types, let us compare two of them that seem similar: Entries from FFP and Inputs from FP. An Entry refers to a group of control data received by a process from a user outside the application boundary (see Figure 3a): that is, the action (or subprocess) of receiving a group of data from the user. An Input, in contrast, refers to a whole process that receives a group of data from the user and updates a logical file in some cases (depending on the type of data). An Input, as defined by IFPUG, can also read a logical file and send information to the user (see Figure 3b). In the FFP, the subprocesses of updating logical files, reading logical files, and sending information to the user are counted as separate function types.

We can say that the FFP takes into account a finer level of granularity, the subprocess level, while the FP considers only the process level. A finer level of granularity is important in real-time software, since its processes have a variable number of subprocesses

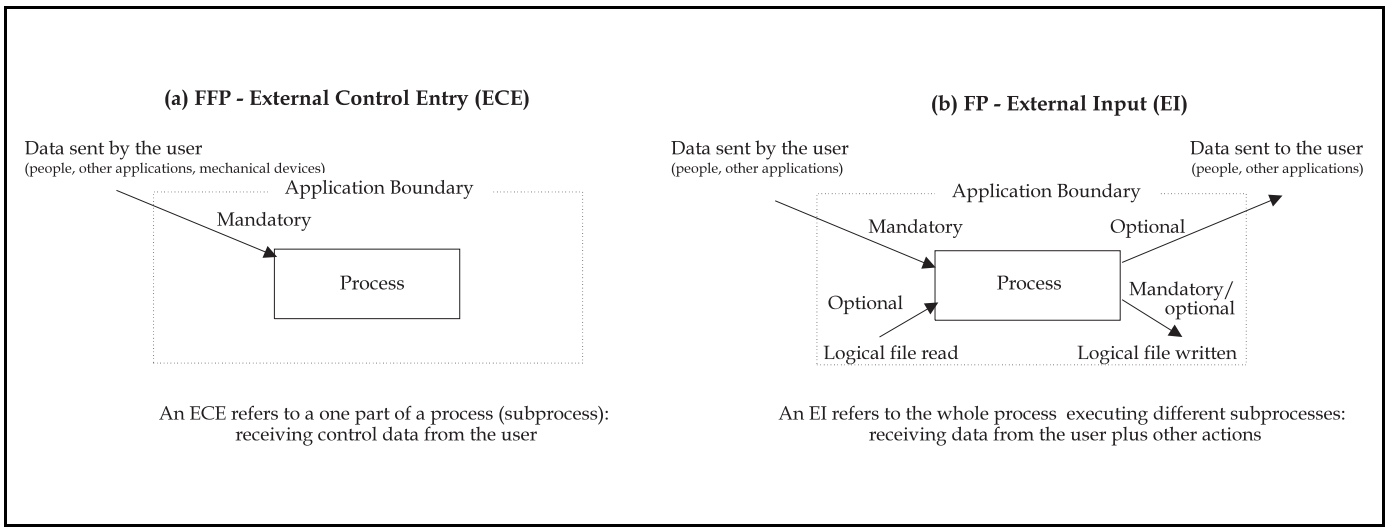

Figure 3: External control entry (FFP) vs. external input (FP)

as compared to MIS software. As we illustrated previously, this is an important element to consider in the measurement of the functional size of real-time software.

FIELD TESTS OF THE EXTENSION PROPOSAL

Field Test Context

The objective of the field tests was to measure some real-time software applications to verify in an empirical way the applicability of the concepts proposed, answering the following questions:

- ★ Do the users of the measurement technique agree that the proposed measure has met its stated objective of measuring the functional size of real-time software?
- ★ Is the proposed measure "good enough" from the users' perspective, in the sense that it is achieving the right balance of multiple simultaneous goals, even though any one goal might not have been achieved optimally [2]?
- ★ Do the users agree that the extension proposal has been designed objectively, precisely, and in an auditable manner, and that someone else with the same set of rules would come up with the same results (relatively, and again, "good enough")?
- ★ Do the users agree that the proposed measurement procedures are based on current practices of what is effectively documented at the present time?

To answer these questions, it was mandatory that an application specialist be present at the counting sessions. Due to the typical industrial constraints, such as the availability of application specialists, site measurement sessions were kept to two-day sessions. Therefore, the industrial partners had to select small applications or a self-contained portion (±25,000 LOC) of a medium or large application. The software applications were measured with the extension proposed as well as with the standard version of function points (IFPUG version 4.0) in order to be able to make some comparisons.

Three field tests were conducted between December 1996 and March 1997. For the measurement of these three applications, at least three people participated in the counting sessions: at least one application specialist and two certified function point specialists who were members of the FFP design team.

Analysis of the Counting Results

Table 3 presents the FP and FFP counting results for the transactional function types. We can see that in the presence of multiple subprocesses of a single process, FPs produce fewer points than FFPs. Indeed, in a real-time

	Application A		Application B		Application C	
Function Type	**Occurrences**	**Points**	**Occurrences**	**Points**	**Occurrences**	**Points**
Function Points (IFPUG 4.0)						
External Input (EI)	40	202	6	21	15	50
External Output (EO)	2	14	2	11	17	73
External Inquiry (EQ)	12	40	1	6	0	0
Total Points	**54**	**256**	**9**	**38**	32	**123**
Full Function Points (FFP)						
External Control Entry (ECE)	123	123	10	10	67	69
External Control Exit (ECX)	93	97	8	10	136	139
Internal Control Read (ICR)	395	403	14	18	100	103
Internal Control Write (ICW)	142	154	8	8	165	168
Total Points	**753**	**777**	**40**	**46**	**468**	**479**

Table 3: FFP and FP counting results for transactional function types

environment, FPs have been criticized for generating low FP counts [3, 9] that seem unrelated to the perceived functional size of the software measured [3]. Since FFPs take into account the subprocesses integrated within a single control process by identifying the different groups of data received, sent, read, and written, they will generate more points than the standard FP, as is the case here. In fact, real-time specialists strongly agree that a measure that does not take into account user-requested subprocesses (as the FP does not) cannot be a good enough functional size measure of their applications. Therefore, FFP results will represent for them a more relevant measure of the functional size of their applications.

The results of the field tests also confirmed that the number of subprocesses of a real-time process varies substantially. For example, there were processes with only three subprocesses, while others had over 50 subprocesses.

One of the project's industrial partners conducted a fourth field test without the assistance of the FP specialists. The feedback obtained from this industrial partner can be summarized as follows:

- ★ Concepts and counting 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.

- ★ FFPs counted 79 processes out of the 81 they expected were there to be counted, with an adequate functional size measure. At the end of the field test, they concluded that FFPs failed to count only 2 of the 81 processes, and this was because FFPs do not count processes containing only internal algorithms. The FFP count coverage rate was therefore 97 percent of the optimal coverage target.

Ease of Understanding

One criticism often made of FPs is that the definitions and procedures are complex and time consuming, and that it takes an FP expert to produce an accurate FP count. The new function types have been designed to be simpler to learn and master. This was confirmed during the field test counting sessions, where, once the application specialists understood the definitions of the new function types, they had no problem identifying them. Indeed, after a full day of FFP counting, they were able to count with limited assistance. According to application specialists participating in the counting sessions, this was mostly due to the fact that it is much easier to identify function types that refer to only one type of action (for example, receiving data) as in FFP counting, than to identify function types that potentially refer to more than one (some mandatory and some optional; multiple possible combinations, some of which qualify as EIs and others of which do not), as in FP counting. Furthermore, if the software is almost entirely real-time (with no MIS-type functions), the specialists do not even need to know the much more complex IFPUG FP rules.

In addition, application specialists believed that the definitions, counting procedures, and rules were clear and detailed enough that different people would be able to come up with

fairly similar results. They also believed that the proposed concepts and measurement procedures and rules were based on current practices of what is documented at the present time.

Counting Effort

Regarding the effort required to measure an application, the FFP and FP counting efforts are found to be similar. Even though more function types had to be counted with the FFP, they were easily identified, and therefore no greater effort was required. Indeed, the application specialists seemed to require less counting assistance from FP experts when counting with FFPs than with FPs, so the identification of more function types does not increase the counting effort.

Attribution of Points

The counting results showed that the total number of points assigned to a particular function type is almost the same as the number of occurrences of the given function type (see Table 3). This is explained by the fact that the majority of the function types are located in the first range of the table used to assign the points according to the number of fields [15]. One can therefore question the usefulness of the attribution of points. It is important to note that the weights assigned to a particular function were chosen with the purpose of aligning the sizes of the new function types as much as possible with the FP weight assignment structure. The weights may need to be recalibrated later by a standard-setting body.

Early FFP Counts

FPs are often used to build estimation models based on the functional size of the project. One can therefore inquire about the possibility of counting the new function types at an early stage of development (feasibility, for example), since they seem to require detailed specifications. In practice, the FFP levels of detail (Entry, Exit, Read, and Write) are similar to the FP ones (DET, FTR, RET) [5]. To use FPs at an early stage of development, the count must usually be approximated because not all the information is available. Over the years, a number of early count approximation techniques have been developed for the IFPUG FP. Similar approximation techniques can be developed for the FFP. Of course, FFP approximation techniques should not be expected to be as mature as FP ones.

CONCLUSION

The development of an FP extension for the measurement of real-time software was a challenge. It had to take into account not only the specific functional characteristics of real-time software in an adequate way, but also to retain the quality characteristics of the FP as a measurement technique and to consider current industry practices in the design and documentation of this type of software.

Feedback from the industry about the extension proposed, the Full Function Point, was positive. We are confident that this FFP measurement technique will expand the domain of applicability of function points and increase their relevance to industry.

ACKNOWLEDGMENTS

This project was carried out by the Software Engineering Management Research Laboratory at the Université du Québec à Montréal and by its industrial partner, the Software Engineering Laboratory in Applied Metrics (SELAM). We thank Nortel, Bell Canada, Hydro-Québec, and JECS System Research, a Japanese industrial partner, for providing project funds, industrial data, and valuable feedback from real-time software practitioners.

REFERENCES

1 Albrecht, A.J. "Measuring Application Development Productivity." In *Proceedings of the Joint IBM/SHARE/GUIDE Application Development Conference* (October 1979). Reprinted in Capers Jones, *Programming Productivity: Issues for the Eighties* (Los Alamitos, CA: IEEE Computer Society Press, 1986).

2 Bach, J. "Good Enough Quality: Beyond the Buzzword." *Computer*, Vol. 30, no. 8 (August 1997), pp. 96–98.

3 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.

4 IEEE. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries,* IEEE Std 610-1990. New York: Institute of Electrical and Electronics Engineers, 1990.

5 IFPUG. *Function Point Counting Practices Manual,* Release 4.0. Westerville, OH: International Function Point Users Group, 1994.

6 IFPUG. *IFPUG Case Study 4* (draft). Westerville, OH: International Function Point Users Group, 1992.

7 Illingworth, V., ed. *Dictionary of Computing,* 3rd edition. New York: Oxford University Press, 1991.

8 Jacquet, J.-P., and A. Abran. "From Software Metrics to Software Measurement Methods: A Process Model." In *Proceedings of the Third International Symposium and Forum on Software Engineering Standards.* Los Alamitos, CA: IEEE Computer Society Press, 1997.

9 Jones, C. *Applied Software Measurement: Assuring Productivity and Quality.* New York: McGraw-Hill, 1991.

10 Jones, C. *Applied Software Measurement: Assuring Productivity and Quality,* 2nd edition. New York: McGraw-Hill, 1997.

11 Maya, M., D. St-Pierre, A. Abran, and P. Bourque. *Full Function Points: Function Points Extension for Real-Time Software (Case Study),* Technical Report 1997-05. Montréal: Software Engineering Management Research Laboratory, Université du Québec à Montréal, September 1997.

12 Mukhopadhyay, T., and S. Kekre. "Software Effort Models for Early Estimation of Process Control Applications." *IEEE Transactions on Software Engineering,* Vol. 18, no. 10 (October 1992), pp. 915–924.

13 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.

14 St-Pierre, D., M. Maya, A. Abran, and J.M., Desharnais. *Full Function Points: Function Points Extension for Real-Time Software; Concepts and Definitions,* Technical Report 1997-03. Montréal: Software Engineering Management Research Laboratory, Université du Québec à Montréal, March 1997.

15 St-Pierre, D., M. Maya, A. Abran, J.M. Desharnais, and P. Bourque. *Full Function Points: Function Points Extension for Real-Time Software; Concepts, Definitions, and Procedures,* Technical Report 1997-04. Montréal: Software Engineering Management Research Laboratory, Université du Québec à Montréal, September 1997.

16 St-Pierre, D., A. Abran, M. Araki, and J.M. Desharnais. "Adapting Function Points to Real-Time Software." In *Proceedings of the IFPUG 1997 Fall Conference.* Westerville, OH: International Function Point Users Group, 1997.

17 Symons, Charles R. "Function Point Analysis: Difficulties and Improvements." *IEEE Transactions on Software Engineering,* Vol. 14, no. 1 (January 1988).

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

19 Whitmire, S.A. "An Introduction to 3D Function Points." *Software Development* (April 1995), pp. 43–53.

Alain Abran is the director of the Software Engineering Management Research Laboratory and a professor at Université du Québec à Montréal (Canada), where he has taught graduate courses in software engineering since 1993. Dr. Abran also has over 20 years of industry experience in information systems development and software engineering. The maintenance measurement program he developed and implemented at Montreal Trust has received one of the 1993 Best of the Best awards from the Quality Assurance Institute. Dr. Abran holds master's degrees in management sciences and electrical engineering from the University of Ottawa, and a Ph.D in software engineering from École Polytechnique de Montréal. His

research interests include software productivity and estimation models, risk management, functional size measurement models, and econometrics models of software reuse.

Dr. Abran can be reached at the Software Engineering Management Research Laboratory, Département d'informatique, Université du Québec à Montréal, P.O. Box 8888 (Centre-ville), Montréal, PQ H3C 3P8, Canada (+514 987 3000, ext. 8900; fax +514 987 8477; e-mail: abran.alain@uqam.ca).

Marcela Maya is currently a research assistant at the Université du Québec à Montréal. With over 10 years of experience in management of information systems, she joined the Software Engineering Management Research Laboratory after receiving her master's degree in software engineering at the same university. Her areas of interest are software metrics as applied to software maintenance processes and real-time systems, software reuse metrics, and productivity models. She has been a certified function points specialist since 1996.

Ms. Maya can be reached at the Software Engineering Management Research Laboratory, Département d'informatique, Université du Québec à Montréal, P.O. Box 8888 (Centre-ville), Montréal, PQ H3C 3P8, Canada (+514 987 3000, ext. 6477; fax +514 987 8477; e-mail: maya.marcela@uqam.ca).

Jean-Marc Desharnais is a specialist in software engineering metrics. He has carried out a number of software engineering research projects covering assessment, budgeting, and productivity evaluation. Mr. Desharnais has also evaluated productivity levels in several organizations and set up quantification programs to include the assessment, productivity, quality, and budgeting of software maintenance. Mr. Desharnais holds master's degrees in computer management and public administration. He is the owner of the Software Engineering Laboratory in Applied Metrics (SELAM). A certified function points specialist since 1993, he has participated in several committees of the International Function Point Users Group (IFPUG) and is a member of the IFPUG Education Committee.

Mr. Desharnais can be reached at the Software Engineering Laboratory in Applied Metrics, 7415 Beaubien East, Suite 509, Anjou, PQ H1M 3R5, Canada (+514 355 2872; fax +514 355 3600; e-mail: desharnais.jean-marc@uqam.ca).

Denis St-Pierre is an experienced senior software metrics consultant. Equipped with a master's degree in function points, he has consulted widely to American, Asian, and European firms. Recently, Mr. St-Pierre has developed an integrated corporate measurement framework as well as a function point extension for real-time software. He founded and moderates the Internet Function Point Listserv, comprising 700 members, and has published a significant number of industrial and scientific articles. A certified function points specialist since 1993, he has participated in developing case studies and guidelines to Software Measurement for the International Function Point Users Group (IFPUG). Mr. St-Pierre is co-author of the IFPUG Standard, and has been a member of the Counting Practices Committee since 1993.

Mr. St-Pierre can be reached at the Software Engineering Laboratory in Applied Metrics, 7415 Beaubien East, Suite 509, Anjou, PQ H1M 3R5, Canada (+514 355-2872; fax +514 355 3600; e-mail: denis.st-pierre@crim.ca). ★