FULL FUNCTION POINTS: FUNCTION POINTS EXTENSION FOR REAL-TIME SOFTWARE, CONCEPTS AND DEFINITIONS

Software Engineering Management Research Laboratory
Software Engineering Laboratory in Applied Metrics (SELAM)

Technical Report 1997-03

In collaboration with

Nortel
Bell Canada
Hydro-Québec
Japanese Industrial Partner

Prepared by

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

March 1997

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 bests 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.

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 technique, 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 (Ince, 1991) 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."* page 283

A number of scientific papers have been published on the statistical accuracy of FPA (Abran, 1994; Desharnais, 1988; Kemerer, 1993). No published papers mention pure real-time software data. In fact, FPA does not capture the functional characteristics of processor-intensive applications well, such as the dynamics and the control functions of real-time software. When FPA is applied to such software, they do, 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 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 SELAM to adapt FPA to real-time software. In section 2, we present a short description of FPA as defined by the

International Function Point Users Group (IFPUG) version 4.0 (1994). Next, in section 3, definitions of real-time software are presented. In the fourth section, we present some real-time limitations of FPA. The fifth section presents the proposed extension. The practical aspects of counting applications with the extension are addressed in the sixth section. We conclude our report with comments on future standards and research directions.

2. FPA overview

Function Point Analysis, developed by Allan Albrecht of IBM, was first published in 1979, and, in 1984, the International Function Point Users Group was set up to clarify the rules, set standards and promote its use and evolution. Since then, four major releases of the rules and standards have been published. In this report, all references to FPA definitions and measurement rules will be based on the most recent publication, that of 1994.

The first step in calculating FPA is to identify the counting boundary, that is, the border between the application or project being measured and the external applications or the user domain. A boundary establishes which functions are included in the function point count. Figure 1 illustrates the boundary between an application and the user. The next step consists in determining the *unadjusted function point* (UFP) count, which reflects the specific countable functionality provided to the user by the project or application. Calculation of the UFP begins with the counting the five function types of a project or application: two data function types and three transactional function types. The five function types counted are (Figure 1):

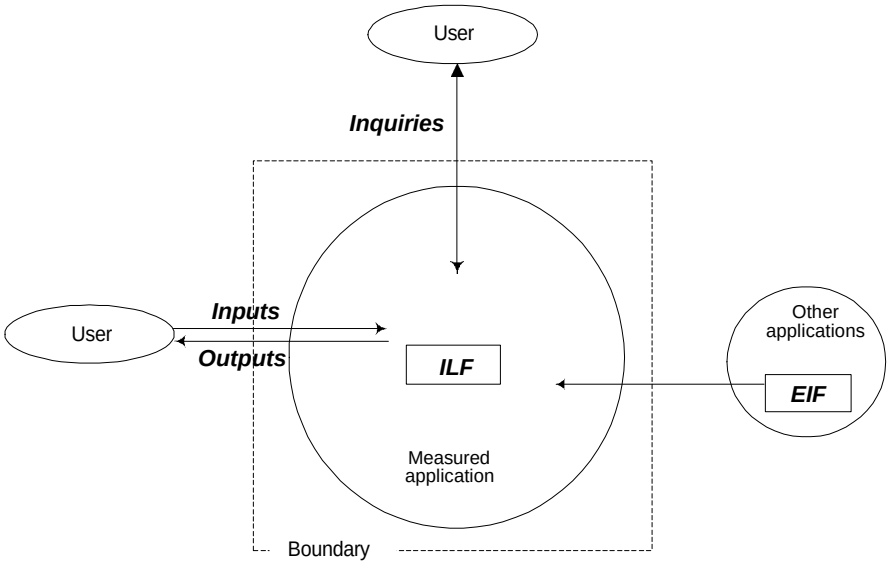

Figure 1 - FPA components

Data function types:

Internal Logical File (ILF): a user identifiable group of logically related data or control information maintained within the boundary of the application.

External Interface File (EIF): a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. This means that an EIF counted for an application must be an ILF in another application.

Transactional function types:

External Input (EI): An EI processes data or control information that comes from outside the application's boundary. The external input itself is an elementary process[1].

External Output (EO): An EO is an elementary process that generates data or control information sent outside the application's boundary.

External Inquiry (EQ): An EQ is an elementary process made up of an input-output combination that results in data retrieval. The output side contains no derived data[2]. No ILF is maintained during processing.

These five function types are then ranked according to their complexity: low, average or high, using a set of prescriptive standards. After classifying each of the five function types, the UFP is computed using pre-defined weights for each function type.

The last step involves assessing the environment and processing complexity of the project or application as a whole. The impact of fourteen general system characteristics is rated on a scale from 0 to 5 in terms of their likely effect on the project or application. We obtain the value adjustment factor that will adjust the UFP by a maximum of ±35% to produce the *Adjusted Function Points (AFP).*

A complete description of FPA, including definitions, procedures, counting rules and examples, may be found in the *Function Point Counting Practices Manual, Release 4.0*, published by IFPUG (IFPUG, 1994).

3. Real-time software

There are a number of definitions of "real-time software" in the literature:

Stankovic and Ramamritham (1988):

*'Real-time systems are defined as those systems in which the **correctness** of the system depends not only on the logical result of computation, but also on the **time** at which the results are produced.'*

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

[2] Derived data: data that requires processing other than direct retrieval and editing of information from ILFs and/or EIFs.

Cooling (1991) and Laplante (1993):

> *'Should computer responses exceed these time bounds then performance degradation and/or malfunction results.'*

IEEE (1990):

> *'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.'*

The Oxford Dictionary of Computing (Illingworth 1991):

> *'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.'*

In the last two definitions, two important aspects of real-time software are addressed: timing and interaction with external entities. The purpose of this interaction is to obtain information and/or to control external entities. This interaction takes place with tight constraints on the choice and on the timing of tasks. Real-time software is different because of the explicit constraints on timing. Such software incorporates dedicated components to manage these constraints; failure to achieve the timing constraints may result in a malfunction. Consequently, for real-time software, correctness of the result is linked to timing.

4. Real-time software limitations of FPA

4.1 Data limitations

There are two kinds of control data structure: *multiple occurrence logical group* and *single occurrence logical group*. Multiple Occurrence Logical Groups can have more than one instance of the same type of record[3]. Single Occurrence Logical Groups have one and only one instance of the record. Real-time software typically contains a large number of Single Occurrence Logical Groups which are difficult to group into ILFs or EIFs. An extension of the ILF/EIF rules is necessary to measure adequately Single Occurrence Logical Groups.

[3] 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.

4.2 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 measure 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 it 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. The application is not in a consistent state until all sub-processes of the temperature control process are completed. Therefore, there is only one 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 the engine temperature control process, example 1

According to standard FPA rules, only one transactional function would be identified, because there is only one *elementary process*.

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

The main purpose of this process it to turn on an alarm when necessary. Fifteen engine sensors (all different) send data to the diagnostic process (15 sub-processes, one 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, one 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. 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 the engine temperature control process, example 2

According to standard FPA 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.

5. Real-time extension

Full Function Points (FFP) is a functional measure based on the standard FPA technique. It was designed for both MIS and real-time software. Since FFP is an extension of the standard FPA technique, all IFPUG rules are included in this new measurement technique except for a few rules related to control. The control aspect of real-time software is addressed by new function types.

5.1 FFP Function Types

To measure the characteristics of a variable number of sub-processes adequately, it is necessary to consider not only processes as defined in FPA (elementary processes), but sub-processes as well. In addition, the particular structure of typical control data must be considered. Consequently, FFP introduces additional data and transactional function types.

Here are the two new Control Data Function Types. They are similar to the IFPUG Data Function Types.

Internal Control Logical Group (ICLG): An ICLG is a group of logically related control data updated by the application being counted. It is identified from a functional perspective[4]. The control data live for more than one transaction[5].

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

[5] In example 2 of section 4, 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.

External Control Logical Group (ECLG): An ECLG is a group of logically related 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[6].

Here are the four new Control Transactional Function Types. They 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[7]. 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. ECEs exclude updating data. The updating functionality is covered by another Control Function Type (Internal Control Write).

In the engine diagnostic example (example 2 section 4), 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 two ECXs. ECXs exclude reading data. The reading functionality is covered by another Control Function Type (Internal Control Read).

In the engine diagnostic example, 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 two ICRs.

In the engine diagnostic example, 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 two ICWs.

[6] These are different from the FPA rules, in order to allow FFP to measure the embedded data aspect of real-time software. FPA rules seem to be oriented toward typical MIS corporate data structures.

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

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 15 kinds of sensor control data updated at different times (15 unique sub-processes), there are 15 ICWs.

Here is how the Control Transactional Functions are related to control processes:

Figure 2 - Diagram of Control Transactional Functions

As with FPA, all new function types are based on the functional perspective of the application, rather than on the technical perspective. The difference between the standard FPA technique and the proposed extension (FFP) is therefore the additional function types (ICLG, ECLG, ECE, ECX, ICW, ICR). These new 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 counted with the standard FPA technique (see Figure 3 and Table 3).

FFP Management Function Types:	
Internal Logical File (ILF)	existing in FPA, unchanged in FFP
External Interface File (EIF)	existing in FPA, unchanged in FFP
External Input (EI)	existing in FPA, unchanged in FFP
External Output (EO)	existing in FPA, unchanged in FFP
External Inquiry (EQ)	existing in FPA, unchanged in FFP
FFP Control Function Types:	
Internal Control Logical Group	new function type, similar to ILF
External Control Logical 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 3 - List of FFP Function Types

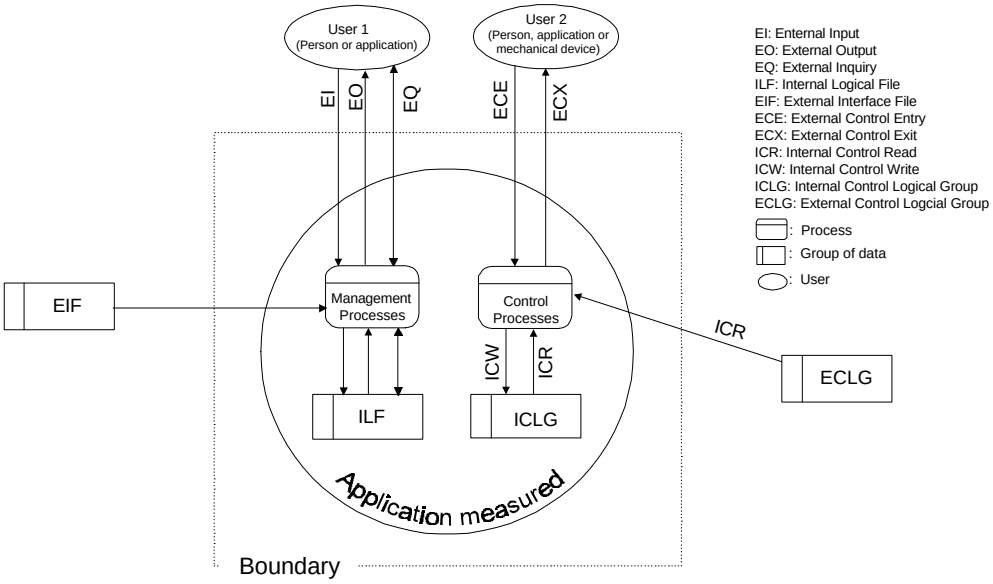

Figure 3 - Diagram of FFP Function Types

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

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

5.2 Point assignment

The previous section described how to identify function types. The next step is to assign points to each function identified in the measured application.

5.2.1 Point assignment for IFPUG Function Types

For the IFPUG Function Types (ILF, EIF, EI, EO and EQ: Table 3), the point assignment procedure is unchanged in FFP. Thus, the IFPUG (IFPUG, 1994) point assignment procedure is applicable as is.

5.2.2 Point assignment for Control Function Types

5.2.2.1 Point assignment for Control Data Function Types

5.2.2.1.1 Multiple Occurrence Logical Group

Multiple Occurrence Logical Groups (see Section 4.1) have the same structure as ILFs and EIFs in FPA. Consequently, they are counted in exactly the same way as these two FPA function types, that is, using the number of Data Element Types (DETs) and Record Elements Types (RETs) and the corresponding complexity matrix (see IFPUG 1994 for definitions and matrices).

5.2.2.1.2 Single Occurrence Logical Group

The number of points assigned to a Single Occurrence Logical Group (see Section 4.1) depends only on the number of DETs. Once it is determined, the number of points is calculated as the integer part of ((number of DETs / 5) + 5)[8] for ICLG. For ECLG, the formula is integer(number of DETs/5). These formulas are designed to keep the size of single occurrence groups as aligned as possible with the size of ILFs and EIFs of FPA. The following tables present examples of the formulas:

Single occurrence ICLG point assignment examples: Integer((number of DETs / 5) + 5):

Number of DETs:	10 DETs	35 DETs	50 DETs
Points:	7	12	15

Single occurrence ECLG point assignment examples: Integer(number of DETs/5):

Number of DETs:	10 DETs	35 DETs	50 DETs
Points:	2	7	10

5.2.2.2 Point assignment for Control Transactional Function Types

The number of points assigned to Control Transactional Functions Types (ECE, ECX, ICW and ICR) depends on the number of DETs[9]. Once the number of DETs is determined, the following table is used to translate DETs into points[10]:

---

[8] A Single Occurrence Logical Group comprises all single control values (from a functional perspective) of the application being measured. Since it contains all single values of the application, there can be only one of them in an application. In typical real-time applications, the number of such values varies from a few up to hundreds. That is why a formula is used rather than a 3-level table like the standard FPA technique. It allows FFP to consider a large range of Single Occurrence Logical Groups.

[9] DET: A unique user recognizable, nonrecursive field (IFPUG 94, glossary) that is either entered, exited, read or written depending the function type.

[10] In the engine diagnostic example, the sub-process that sends the alarm message to the dashboard has 2 DETs (number of the button, action code: turn-on), the number of points is therefore 1 (column "1 to 19 DETs" Table 4).

DETs:	1 to 19 DETs	20 to 50 DETs	51 + DETs
Points:	1	2	3

Table 4 - Control Transactional Function Type Translation Table

These range boundaries (1 to 19, 20 to 50, 51+) were chosen in order to bring the size of Control Transactional Function Types in as close alignment as possible with FPA.

6. FFP practice feedback

This report has presented FFP definitions and examples. We now discuss the practical aspects of counting FFP. A number of industrial real-time applications have already been counted. Feedback from these field tests is reported below.

6.1 Ease of understanding

All FFP counts were done with the support of real-time application experts (industry people familiar with the application being counted). Once the application experts had understood the definition of the Control Function Types (after a few hours), they had no problem in identifying them. In fact, after a full day of FFP counting, the application experts were able to count with little assistance from functional size experts familiar with FFP.

6.2 Counting effort

Based on the counting experiments carried out, the FFP counting 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[11], the new Control Transactional Functions are more quickly identified. Besides, application experts seem to require less counting assistance from functional size experts, again because Control Transactional Functions are simpler to deal with.

6.3 Importance of documentation

An adequate source of functional information must be available to count an application. This functional information is provided by application experts or by documentation of the application. Like FPA, FFP is dependent on the quality of the functional information. Based on the counting experiments carried out, the quality of the functional information typically available is adequate for counting FFP.

[11] 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.

6.4 Early FFP counts

It is possible to count control function types at an early stage of development (feasibility for example). Despite additional function types, the FFP level of detail is similar to the FPA one. To use FPA at an early stage of development, one must usually approximate the count because not all functional information is available. Over the years, a number of FPA early count approximation methods have been developed. Similar approximation methods can be applied to FFP. Of course, one should not expect, at this point, that FFP approximation methods will be as mature as FPA ones.

7. Summary

The development of a real-time software functional measure is an important research challenge. 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 current structure of Function Points in 1984 (Albrecht 1984), many people have been working with this measure. In fact, a number of project databases[12] 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 the first version of FFP, and the authors recognize that there is room for improvement. Among other things, the weights may need to be re-calibrated.

In addition, we are currently working on detailed counting rules and procedures based on the framework developed by IFPUG. Version 1.0 of the counting rules and procedures is now being tested at industrial sites.

We would like FFP to evolve in ways that are aligned as much as possible with the standard FPA technique (IFPUG, 1994). We are open to collaboration with corporations and measurement associations to produce the next version of FFP.

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

Glossary

The following definitions should be used for FFP, in addition to the IFPUG (IFPUG, 1994) glossary.

Control data: Data used by the application to control directly or indirectly the behavior of an application or a mechanical device.

Control event: An event that triggers processing from a functional perspective.

Control process: Process that controls directly or indirectly[13] the behavior of an application or a mechanical device. It comprises all sub-processes associated with a control event.

Data function type: The functionality provided to the user to meet internal and external data requirements. Data types are defined as ICLG, ECLG, ILF and EIF.

Functional perspective: Point of view is of the functionality delivered by the application; it excludes technical and implementation considerations.

Management data: Data used by the application to support a user in managing information, particularly business and administrative information**.**

Management process: Process the purpose of which is to support the user in managing information, particularly business and administrative information**.**

MIS applications: Applications produced to support information, business and administrative operations.

Process: In the context of this report, a process is all processing associated with a control event.

Sub-process: In the context of this report, it is the smallest processing step identifiable from a functional perspective as either an entry, exit, read or write.

Transaction: All processing associated with an occurrence of an external trigger. For example, in a transaction file, each record acts as trigger when the file is processed by an application.

Transactional function type: The functionality provided to the user by an application to process data. Transactional function types are defined as ECE, ECX, ICR, ICW, EI, EO and EQ.

Update: The ability to modify data.

User: In FFP, the IFPUG definition of "user" has been extended to include not only human beings, but also applications and mechanical devices.

[13] For example, a diagnostic process that reports to a human being is not directly controlling an application or mechanical device but it is a *control process*.

References

Abran, A., *Analysis of the measurement process of Function Points Analysis*, PhD. 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*, June 1, 1992.

For more information on FFP, here is how to contact the authors of the report:

Alain Abran Ph.D.

Software Engineering Management Research Laboratory
Departement d'informatique
Universite du Quebec a Montreal
C.p. 8888 succursale centre-ville
Montreal (Quebec) Canada H3C 3P8

Telephone: 514 - 987-3000 (8900)
Fax: 514 - 987-8477

E-mail: abran.alain@uqam.ca
Web site: http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

Jean-Marc Desharnais M.Sc.A., C.F.P.S.

Software Engineering Laboratory in Applied Metrics
7415 Beaubien East suite 509
Anjou (Quebec) Canada H1M 3R5

Telephone: 514 - 355-2872
Fax: 514 - 355-3600

E-mail: Desharnais.Jean-Marc@uqam.ca
Web site: http://www.lmagl.qc.ca

Marcela Maya M.Sc.A., C.F.P.S.

Software Engineering Management Research Laboratory
Departement d'informatique
Universite du Quebec a Montreal
C.p. 8888 succursale centre-ville
Montreal (Quebec) Canada H3C 3P8

Telephone: 514 - 987-3000 (6647)
Fax: 514 - 987-8477

E-mail: Maya.Marcela@uqam.ca
Web site: http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

Denis St-Pierre M.Sc., C.F.P.S.

Software Engineering Laboratory in Applied Metrics
7415 Beaubien East suite 509
Anjou (Quebec) Canada H1M 3R5

Telephone: 514 - 355-2872
Fax: 514 - 355-3600

E-mail: Denis.St-Pierre@CRIM.CA
Web site: http://www.lmagl.qc.ca