Measuring the functional size of real-time software

Marcela Maya, Alain Abran, Serge Oligny, Denis St-Pierre, Jean-Marc Desharnais

Abstract

Function Point Analysis (FPA) is a technique designed to measure the functional size of software products. The technique measures product size from the user's point of view rather than from a technical perspective. FPA is now widely used in the MIS domain, where it has become the "de facto" standard. However, FPA has not enjoyed the same degree of acceptance in other domains, such as real-time software. This paper reports on work carried out to adapt FPA to the specific functional characteristics of real-time software. The extension proposed, called Full Function Points (FFP), is described and the results of field tests are discussed.

1. Introduction

Due to the important role played by software in today's organizations, the use of measures to control 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 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 [3], [6], [11], [15]. 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 [6], Mark II [14], Asset-R [11], 3D Function Points [15], Application Features [10] and IFPUG Case Study 4 - Draft version [5]. 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 - Draft version). 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 extend FPA to take into account the specific functional characteristics of real-time software. The extension proposed, *called Full Function Points (FFP)* [12], is based on the observation that real-time software has the following specific transactional and data characteristics:

- **Transactional characteristics:** The number of sub-processes of 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 new components to be measured in addition to the five already defined in FPA: four new components to take into account the transactional characteristics and two new components to take into account the data characteristics. FFP define detailed procedures and rules to identify and weight these new components.

Furthermore, FFP were designed to retain the actual FPA 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 [4]. 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 [8] [9] [13]: FFP results represent for them a more relevant measure of the functional size of their real-time software than do FPA results. They believe that 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 FPA, even though more function types have to be counted, the reason being that the new components are easily and quickly identified.

In this paper, the key concepts of FFP are described and the results of field tests conducted are discussed. Section 2 describes the limits of the current version of FPA for the functional size measurement of real-time software. Section 3 introduces the basic concepts of FFP.

Section 4 describes the field tests conducted and highlights the main results. The final section presents some observations and topics for further research.

2. FPA in a real-time environment

To measure the functional size of a software product, the current version of FPA considers two types of components or *function types* as described in [4]: the transactional function type and the data function type (readers unfamiliar with FPA are referred to Box A at the end of this paper for an overview of basic FPA concepts). However, the following transactional and data characteristics specific to real-time software were identified as not being well captured by the current FPA function types.

2.1. Transactional function type characteristics

Transactional function types in FPA are based on the concept of the *elementary process*[1]. For instance, a process that generates data to be sent to the user could have the following four steps: (1) receive the request from the user, (2) read the information needed, (3) make calculations, and (4) send the results back to the user. According to FPA rules, these four steps or sub-processes are considered a unique elementary process. This means that the four sub-processes are counted as a unique function type (Input, Output or Inquiry). In FPA, the number and nature of the sub-processes required to execute an elementary process are not taken into account.

According to empirical observations [8] [9], MIS processes of the same type have a relatively stable number of sub-processes. Therefore, in a typical MIS environment, the number of sub-processes does not add any important information to the functional size of a given process since it is relatively constant across all processes of the same type. By contrast, real-time software shows a varying number of sub-processes per elementary process. 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 cooling system of the engine when necessary. A sensor supplies the temperature to the application (sub-process 1). The temperature is compared to the overheating threshold temperature (sub-process 2). Finally, a turn-on message can be sent to the cooling system if needed (sub-process 3). The application is not in a consistent state until all three sub-processes are completed. The temperature control process therefore has *3 sub-processes*. According to FPA rules, only one transactional function type 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 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, with 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 may be turned on in the dashboard (1 sub-process). In this example, the engine diagnostic process consists of *31 sub-processes*. The application is not in a consistent state until all sub-processes of the diagnostic process are completed. According to FPA rules, only a minimum number of transactional points would be counted. Therefore, when FPA rules are used, examples 1 and 2 would have approximately the same number of

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

transactional points even though the real-time community would strongly disagree on the fact 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 sub-processes, while others have a large number of sub-processes.

2.2. Data function type characteristics

Typical MIS logical files have the following data structure: multiple occurrences of a record, each record having one or more fields. In real-time applications, this type of structure is also used. An engine control application, for instance, 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 grouped data, called a *multiple-occurrence group of data* here, has the same typical structure as that of an Internal Logical File (ILF) or an External Interface File (EIF) in FPA.

However, real-time software also contains a large number of *single-occurrence control data*. These data are characterized by the fact that there is *only one* occurrence of the data in the whole application. For example, the navigational system of an airplane calculates the airplane's position periodically 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 that the system does not store previous 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 FPA ILFs and EIFs. An extension of the ILF/EIF rules is therefore necessary to adequately measure single-occurrence control data.

3. FPA Real-time Extension: Full Function Points

To measure these specific functional characteristics of real-time software adequately, it is thus necessary to consider both the sub-processes performed by a control process and the single-occurrence control data. The proposed FPA extension, called Full Function Points (FFP), introduces new data and transactional function types accordingly:

- New data function types:
  - **Updated control group**: A group of control data updated by the application 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**: A group of control data used, but not updated, by the application being counted.
- New transactional function types:
  - **Entry**: A sub-process that receives control data coming from outside the application's boundary. In example 2, section 2.1, 15 different sensors send data to the application (control data cross the application boundary). Each sub-process receiving data coming from one sensor is considered an Entry.
  - **Exit**: A sub-process that sends control data outside the application boundary. In example 2 above, the sub-process that sends a message to the dashboard (control data sent outside the application boundary) is an Exit.
  - **Read**: A sub-process that reads a group of control data. In example 2, each of the 15 sub-processes that reads a threshold value is counted as a Read.
  - **Write**: A sub-process that writes a group of control data.

Unlike FPA transactional function types, the new FFP transactional function types are identified at the sub-process level instead of at the process level, as shown in Figure 1. It can 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.

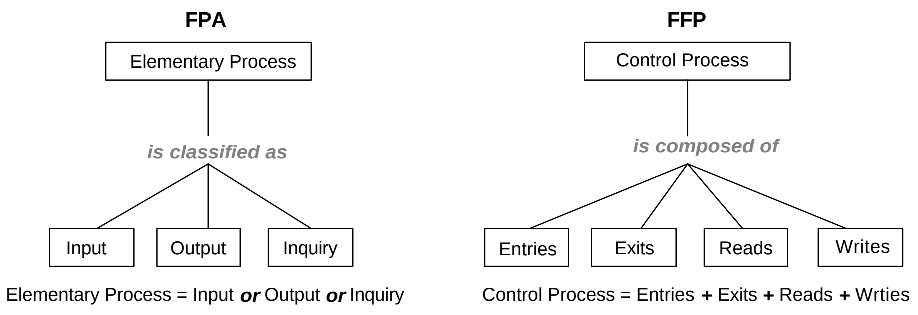

Figure 1: FPA transactional function types vs. FFP transactional function types

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 FFP, are measured using the standard FPA rules. Thus, the identification of the new transactional function types of an application includes the following major steps [12]:

1. Identify the different processes performed by an application from a functional perspective;
2. Determine if each process is a management process (used to support the user in managing information, particularly business and administrative information) or a control process (used to control, directly or indirectly, the behavior of an application or a mechanical device). If it is a management process, identify the traditional FPA transactional function types using FPA counting procedures and rules. If it is a control process, perform the following additional steps:
   a) From a functional perspective, identify the different sub-processes performed by the process;
   b) 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;
   c) Assign the corresponding points.

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 [12].

4. FFP field tests

Three real-time applications (telecommunications and power supply domains) were measured using FFP and FPA between December 1996 and March 1997. Due to industrial constraints, mainly the availability of the application specialists, participating organizations were asked to select one small application or a self-contained portion of a medium or large application (± 25.000 LOC). Each counting session lasted two full days. At least three people participated in each counting session: an application specialist and two certified function point specialists who participated in the design of FFP. A fourth field test was conducted by one of the project’s industrial partners without the assistance of the FFP specialists (automotive domain), using only the FFP documentation.

Table 1 presents the measurement results of the transactional function types for the three applications measured with the assistance of FFP experts [8] [9]. In order to compare FFP with FPA results, each application was measured twice: first using FFP (top part of the table) and then using FPA (bottom part of the table). The FFP and FPA results correspond to the measurement of the same set of processes. It is important to note that all three applications were real-time applications containing only control processes. As a result, no management processes were identified using FFP.

Table 1 - FPP and FPA measurement results for the transactional function types

	Application					
	A		B		C	
Function Type	**Occurrences**	**Points**	**Occurrences**	**Points**	**Occurrences**	**Points**
Full Function Points (FFP)						
- Entries	123	123	10	10	67	69
- Exits	93	97	8	10	136	139
- Reads	395	403	14	18	100	103
- Writes	142	154	8	8	165	168
Total:	**753**	**777**	**40**	**46**	**468**	**479**
Function Point Analysis (FPA)						
- Inputs	40	202	6	21	15	50
- Outputs	2	14	2	11	17	73
- Inquiries	12	40	1	6	0	0
Total:	**54**	**256**	**9**	**38**	**32**	**123**

Comparing the results of the two methods, it can be observed that, in the presence of multiple sub-processes of a single process, FFP generate larger counts than FPA. In fact, in a real-time environment, FPA have been criticized for generating low function point counts which do not seem to be related to the work product measured (Jones, 1991; Galea, 1995), that is, which do not correspond with the perception of the application specialists about the functional size of their applications (important elements regarding the functional size of real-time applications are not taken into consideration, like functions with no data crossing the boundary of the application, for instance). Since FFP takes into account the sub-processes integrated within a unique control process by identifying the different groups of data received, sent, read and written, they should generate more points than FPA, as is the case here. During the field tests, the real-time practitioners strongly agreed that a functional size measure that does not take into account user requested sub-processes (as FPA does not) cannot be a "good enough" functional size measure of their applications. Therefore, FFP measurement results represent, for them, a more adequate measure of the functional size of their applications.

It can also be seen in Table 1 that, for FFP, the number of occurrences of each function type is very similar to the total number of points of the given function type. This is explained by the fact that the majority of occurrences of the function types were located in the first range of the table used to assign points according to the number of fields used by the function type (fewer than 20 fields) [12].

The measurement results also confirmed the observation that the number of sub-processes of a real-time process varies a great deal. For instance, in application A, some processes embedded only 3 sub-processes and other processes embedded more than 50 sub-processes.

Results of the fourth field test conducted independently of the research team by one of the project's industrial partners can be summarized as follows [13]:

- Concepts and counting procedures in the FFP Counting Manual were deemed relatively clear and easy to understand. It was not difficult to count without the assistance of an FFP specialist.
- FFP counted 79 processes out of the 81 they expected to be counted with an adequate functional size measure. At the end of the field test, they concluded that FFP failed to take

into account 2 of the 81 processes because the current design of FFP does not take into account processes containing only internal algorithms. The FFP measurement coverage rate was therefore 97% of the optimal coverage target.

From the measurement sessions and the measurement process, the following observations were also made.

- **Ease of understanding**: One criticism often made about FPA is that the set of definitions and procedures is complex and time-consuming, and that it takes a FPA expert to produce an accurate FPA count [6] [7]. The new FFP function types were designed with the objective of making them simple to learn and master. This was confirmed during the field test counting sessions: once the application specialists understood the definition 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 little assistance. According to the application specialists participating in the counting sessions, this was mostly due to the fact that it is much easier to identify function types that only refer to one type of action (receiving data, for example) as in FPP, than to identify function types that potentially refer to more than one action as in FPA (for example, receiving and updating data in the Input function types). Furthermore, if the software is almost entirely real-time (with no MIS-type functions), the specialists do not even need to know about the more complex IFPUG FPA rules.

  In addition, application specialists reported that the definitions, counting procedures and rules were clear and detailed enough so that different individuals would be able to come up with relatively similar results. They were also of the opinion that the proposed concepts, measurement procedures and rules were based on current practices as to what is currently and effectively being documented.

- **Counting effort:** Counting sessions showed that FFP and FPA counting efforts are similar. Even though more function types have to be counted with FFP, it seems that they can be identified more quickly. Indeed, as pointed out earlier, application specialists seemed to require less counting assistance from function point experts when counting with FPP than with FPA, so the identification of more function types seems not to increase the counting effort.

- **Attribution of points**: The measurement 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. One can therefore question the usefulness of the attribution of points with three levels. It is important to note that the weights assigned to a particular function were chosen with the purpose of making the size of the new function types as aligned as possible with FPA. The weights may need to be calibrated, but, to do so, more empirical data are needed to verify the appropriateness of the weights.

5. Conclusion

This paper presented an approach to adapting FPA to the specific functional characteristics of real-time software, called Full Function Points (FFP). The development of this FPA extension 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 FPA as a measurement technique and to consider current industry practices in the design and documentation of this type of software.

Feedback from the industrial partners of the project indicates that the research purpose was achieved to a sufficient degree. They believe FFP has met its stated objective of measuring user-functional requirements and that it would meet their current, and foreseeable, needs. They believe that the measurement of the functional size of their applications 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 FFP counting procedures and rules are based on current practices as to what is currently and effectively being documented. We are confident that FFP will expand the domain of applicability of FPA and increase its relevance to industry.

This paper presented the first version of FFP, and the authors recognize that there is room for improvement. Among other things, more field-testing needs to be done, which would provide more valuable feedback to improve the proposed approach as well as the counting procedures and rules. This field-testing will also permit accumulation of enough empirical data to support the development of meaningful productivity and estimation models.

As mentioned in section 5, the weights assigned to the new function types may need to be calibrated. As in FP, these weights were derived from empirical observations. Empirical data collected with more field tests will help analyze the relevance of the weights and calibrate them in a formal way, using statistical analysis for instance.

The FFP Counting Practices Manual is a public document and can be found at the following Web Sites: http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html and http://www.lmagl.qc.ca. Other documents about FFP in English, French and Japanese are also available at these addresses.

6. Acknowledgements

We wish to thank Nortel, Bell Canada, Hydro-Québec and JECS System Research (Japan) for providing project funds, industrial data and valuable feedback from real-time software practitioners.

Box A.

FPA Overview

Function Point Analysis (FPA) measures the functional size of a software product in terms of its delivered functionality, measuring such objects as inputs, outputs and files. The first step in calculating FPA is to identify the counting boundary, that is, the border between the application being measured, and the external applications, or the user domain. A boundary establishes *what* functions are included in the function point count.

The next step consists in determining the Unadjusted Function Point (UFP) count, which reflects the specific countable functionality provided to the user by the application. Calculation of the UFP begins with the identification of five components or *function types* of the application. Two function types are related to the data used by the application and the other three are related to the transactions handled by the application:

- Data function types:
  - Files: Logical files (as the user might conceive of them, not physical files) updated by the application.
  - Interfaces: Logical files accessed by the application but not updated by it.
- Transactional function types:
  - Inputs: e.g. transactions to create, modify or delete records in logical files.
  - Outputs: e.g. report types.
  - Inquiries: e.g. on-line inquiries supported by the application.

Once the number of occurrences of each category of function types has been identified, their complexity is classified as low, average or high, according to a set of prescriptive standards. After making this classification for each occurrence of the five function types, the UFP is computed using predefined weights for each function type.

The last step involves assessing the environment and processing complexity of the application *as a whole*. This is carried out through a set of fourteen general systems characteristics (GSC). The impact of each of these fourteen general systems characteristics is rated on a scale of 0 to 5 in terms of their likely effect on the project or application. We thus obtain the value adjustment factor (VAF) 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, can be found in the FPA Counting Practices Manual [4].

7. 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] Albrecht, A. J. and Gaffney, J. E., “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”, IEEE Transactions on Software Engineering Vol. SE-9, No. 6, November 1983, pp. 639-648.

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

[5] IFPUG, “IFPUG Case Study 4 (Draft)”, International Function Point Users Group - IFPUG, Westerville, Ohio, 1997.

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

[7] Kemerer, C. F., “Reliability of Function Points Measurement”, Communications of the ACM, Vol 36, No. 2, February 1993, pp. 85-97.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

About the Authors

Marcela Maya, M.Sc.A., C.F.P.S.
Software Engineering Management Research Laboratory
Département d'informatique - Université du Québec a Montréal
C.P. 8888 succursale centre-ville
Montreal (Quebec), Canada - H3C 3P8
E-mail: maya.marcela@uqam.ca

Alain Abran Ph.D.
Software Engineering Management Research Laboratory
Département d'informatique - Université du Québec a Montréal
C.P. 8888 succursale centre-ville
Montreal (Quebec), Canada - H3C 3P8
E-mail: abran.alain@uqam.ca

Serge Oligny, M.Sc.
Software Engineering Management Research Laboratory
Département d'informatique - Université du Québec a Montréal
C.P. 8888 succursale centre-ville
Montreal (Quebec), Canada - H3C 3P8
E-mail: oligny.serge@uqam.ca

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
E-mail: Denis.St-Pierre@CRIM.CA

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
E-mail: desharnais.jean-marc@uqam.ca