DESIGN OF A FUNCTIONAL SIZE MEASUREMENT FOR REAL-TIME SOFTWARE

Alain Abran[1], Jean-Marc Desharnais[2], Marcela Maya[1], Denis St-Pierre[2], Pierre Bourque[1]

[1]Université du Québec à Montréal
Software Engineering Management Research Laboratory
P.O. Box 8888 (Centre-Ville)
Montréal (Québec), Canada, H3C 3P8
Tel: +1 514 987-3000 ext. 8900

[2]SELAM
Software Engineering Laboratory in Applied Metrics
7415, rue Beaubien Est, Suite 509
Anjou (Québec), Canada, H1M 3R5
Tel: +1 514 355-2872

Abstract

A requirement for software productivity analysis and estimation is the ability to measure the size of a software product from the user's viewpoint, that is, from a functional perspective rather than from a technical perspective. One example of such a measurement method is Function Point Analysis (FPA). FPA is now widely used in the Management Information Systems (MIS) domain, where it has become a "de facto" standard. However, FPA has not achieved the same degree of acceptance in other domains of software engineering, such as real-time software. The general opinion is that when FPA is applied to such software the results do not constitute an adequate size measurement of this type of software. This paper reports on a research project carried out to adapt FPA to the specific functional characteristics of real-time software. The proposed extension, called Full Function Points (FFP), is described and the results of field-testing are discussed. This research was conducted using an adaptation of Basili's framework for empirical research in software engineering and it is described accordingly.

1. Introduction

Given the importance of software in an increasing range of products and services, software has become a major component of many corporate budgets. As with any other budget component, it is vital for these organizations to control these expenses, analyze the performance of the amounts allocated to software development and maintenance, and benchmark their software against the best in the field. To do so, measures and measurement models are required.

On the one hand, technical measures are needed to assess the technical performance of products and services from a developer's point of view. Such technical measures can be used, for instance, in efficiency analysis to improve the performance of design. On the other hand, functional measures are needed to assess the performance of products or services from a user's perspective. Such measures are also needed for productivity analysis. Since functional measures are independent of technical development and implementation decisions, they can be used to compare, for instance, the productivity of different techniques or technologies.

Function Point Analysis (FPA) is a functional measurement technique. It is used extensively in the MIS domain to support productivity analysis and estimation models ([1]; [2, 3]; [4]; [5], 1996; [6]; Kitchenham, 1991). It seems to capture well the specific functional characteristics of MIS software. However, FPA has been criticized as not being applicable to all types of software ([7]; Galea, 1995; [8]; [9]; Ince, 1991; [5]; [10]; [11]). Ince (1991) for instance, describes the Function Point (FP) scope issue as follows: "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." In the opinion of these authors, even though FPA does, of course, generate measures when applied to real-time software, these measures do not adequately represent the functional size of their object. Therefore, there is currently no FPA equivalent for real-time software that would allow meaningful productivity benchmarking and the development of estimation models based on the functional size of real-time software.

This paper describes work carried out at the Software Engineering Management Research Laboratory of 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)*, is based on the premise that real-time software has the following specific transactional and data characteristics:

- **Transactional characteristics**: The number of subprocesses of a real-time process varies substantially from one process to another. By contrast, similar processes in the MIS domain display a more stable number of subprocesses.
- **Data characteristics**: There is a large number of single-occurrence control data in a real-time software. Single-occurrence data occur only once in a whole application. This data is usually used to control, directly or indirectly, the behavior of another application or of a mechanical device.

To take into account these characteristics, FFP introduces six new types of components: four types related to the transactional characteristics and two types related to the data characteristics of an application.

In this paper, the key concepts of FFP are described and the results of industry field tests are discussed. Section 2 describes the framework, used for exploratory research in software engineering, sustaining this project. The remainder of the paper is structured according to this framework. Thus, section 3 presents the project definition and defines the motivation, the object and the purpose of the research project. Candidate users of the results of this research are also identified. Section 4 presents the planning phase, which includes the definition of the project steps, their inputs and corresponding deliverables. The actual execution of the project steps and the development of the project deliverables are presented in section 5. The project results are then interpreted and discussed in section 6. This paper concludes with some closing remarks and suggestions for further research directions.

2. Research Framework

To address the research issue with an appropriate research protocol, an adaptation of Basili's experimental framework ([12]) for empirical research in software engineering was followed throughout the project. Basili *et al.* proposed this framework to "help structure the experimental process and to provide a classification scheme for understanding and evaluating experimental studies." The experimental framework is articulated around four phases: definition, planning and execution and the interpretation of results.

Successful application of this framework has been reported and illustrated in Bourque and Côté ([13]), Côté *et al.* ([14]) and Bourque *et al.* ([15]). In the author's experience, this framework is an excellent tool for avoiding some of the all-too-common experimental pitfalls like: ill-defined project purpose and objectives; jumping into data collection without properly defining what should be measured; gathering data for a period of time only to realize that the right measures have not been collected; results not interpreted to their fullest and their generalization possibilities not clearly stated. We believe this empirical research framework contributes to the improvement of research rigor in the software engineering field.

Basili's framework was initially designed for projects in research areas where the body of knowledge is already fairly advanced and structured and where extensive data collection is possible. However, within the context of exploratory research, where the concepts are not yet well defined and the body of knowledge is not yet well structured ([16]), some adaptation of the framework is required.

Exploratory research is performed to introduce researchers to a new field of research, to enable them to observe this field and to prepare them for further research by generating hypotheses that are plausible but still require empirical validation (De Ketele, 1991). Exploratory research is not governed by rigorous constraints on sampling or measurement issues. However, repeatability still remains an essential requirement so that other researchers can reproduce the research process. By providing a simple and easy-to-use research protocol for conducting and documenting exploratory research projects, repeatability of these projects is greatly improved. This protocol does not compromise creativity, which is obviously a core requirement for exploratory research.

The four-phase structure of the protocol remains untouched. The following adaptations were implemented at the sub-phase level. Since there is no formal data collection in the statistical sense included in a framework for exploratory research, substeps regarding sample design through statistical data analysis were removed from the framework. These are replaced by activities such as definition of project steps, project inputs and project deliverables in the planning phase, and actual development of project deliverables in the operation phase. Figure 1 shows Basili's framework adapted for exploratory research. The framework is explained in the following paragraphs.

The definition phase has four components: motivation, object, purpose and users of the research results. *Motivation* is the reason for tackling the project; it identifies what high-level or general issue we are trying to address. *Object* defines the principal entity to be studied. The *purpose* is the precise objective of the project; it can also be seen as the explicit problem that we wish to resolve. *Users of the research results* identifies right up-front who could put these results into practice.

I. Definition			
Motivation	Object	Purpose	Users of research results

II. Planning		
Project steps	Project inputs	Project deliverables

III. Operation		
Development of project deliverables	Field-testing	Analysis of field-test data

IV. Interpretation		
Interpretation context	Extrapolation	Impact

Figure 1 - Framework for exploratory research in software engineering (adapted from Basili *et al.*, 1986)

A detailed plan of the exploratory research project is developed in the second phase of the framework. A project plan is developed through the identification of project steps. The project inputs are identified next and the project deliverables are precisely defined so that all project participants and collaborators clearly agree on the expected deliverables.

Execution of the project steps identified in the planning phase is actually carried out during the operation phase where the proposed deliverables are developed. Field-testing of the project deliverables should then be conducted, preferably in an industrial environment, and field-test results analyzed.

In the interpretation phase of the framework, the interpretation of the project results is undertaken in two stages. First, all project results are interpreted in the context of the stated purpose and then compared to other research results in similar area. Field test representativeness is the determining factor in extrapolating or generalizing the results to different environments and contexts. As previously mentioned, exploratory research projects are aimed at investigating an area that is not yet well understood. It produces results, often models, which should be considered as hypotheses. These results are considered as hypotheses because they require further experimentation and rigorous validation. Lastly, issues requiring further investigation and offering new research opportunities are identified.

3. Project Definition

This section presents the motivation, object, purpose and users of the research results. Table 1, at the end of the section, summarizes the key elements of the project definition.

3.1 Motivation

A requirement for software productivity analysis and estimation is the ability to measure the size of a software product from the user’s viewpoint, that is, from a functional perspective rather than from a technical perspective. Function Point Analysis (FPA), first introduced by Allan Albrecht of IBM in 1979 ([2]), is an example of such a functional measurement technique. Since FPA was developed in a management information systems (MIS) context, its concepts, rules and guidelines have been adapted to the types of software most frequently used in such a context. FPA has gained a wide audience in this specific area of software application. This measurement method is being used extensively to, among other things, analyze productivity and estimate project effort.

Nevertheless, FPA has not achieved the same degree of market penetration in other areas of software application. Indeed, several researchers agree that, with its current structure, FPA seems inadequate for measuring embedded and real-time software, for instance, because the results do not reflect the perceived functional size of the software measured ([17]; Jones, 1996; [11]; Galea 1995). As Withmire ([18]) says: “Function points, as defined by the International Function Point Users Group (IFPUG), do not measure all the problems and characteristics that contribute to the size of a software application.”

Some modifications to the current structure of FPA have been proposed to take into account the specific functional characteristics of real-time and scientific software (Jones, 1986; [17]; [11]; [19]; Galea, 1995). However, it seems that none of these approaches has gained significant recognition from the practitioners in the field, even though some of these modifications have been presented over a period of more than a decade now.

The motivation for this research project, expressed according to Basili's framework, is therefore to understand the limitations of the FPA technique when applied in embedded and real-time environments and to improve this measurement technique for usage in a real-time environment.

3.2 Object

The object, or the main entity, being addressed by the project is the functional size of real-time software, as measured by the FPA technique. A brief explanation of elements of this statement is presented in the following paragraphs.

Functional Size

Up until fairly recently, there has not been an officially recognized definition of the functional size of a software product. However, this situation changed in 1998 with the adoption of the following definitions by ISO (ISO/IEC, 1998):

- *Functional Size*: A size of the software derived by quantifying the Functional User Requirements.
- *Functional Size Measurement*: The process of measuring Functional Size.
- *Functional User Requirements*: A subset of the user requirements. The Functional User Requirements represent the user practices and procedures that the software must perform to fulfil the user’s needs. They exclude Quality Requirements and Technical Requirements.

The important concept to retain for the purpose of this paper is that a functional size measure must quantify the functions provided to the user by the software in a way that is independent of the techniques and tools used to implement the measured software.

Real-time software

“Real-time software” is one of those general terms that often mean different things to different people. In fact, a literature review has shown that there is not even a consensus on a definition of “real time” among researchers in the field. In this research project, real-time software refers to software displaying the characteristics outlined by 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." (IEEE, 1990).
>
> "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." (Illingworth, 1991).

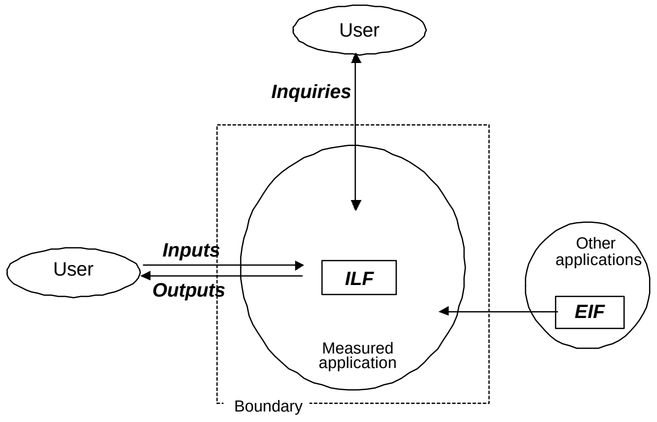

Figure 3 - FPA components

The project was therefore aimed at focusing on the types of software that react to external events and are subject to very tight timing constraints.

FPA method

FPA measures the functional size of software in terms of its delivered functionality, measuring such objects as inputs, outputs and files[1]. The first step in calculating FPA is to identify the measurement boundary. This boundary distinguishes the application being measured from other external applications in the user domain. A boundary establishes *which* functions are to be included in the function point measurement process. Figure 3 illustrates the boundary between an application, its user and other applications.

The next step consists in determining the Unadjusted Function Point (UFP) count, an intermediate measurement result reflecting the specific countable functionality provided to the user by the application. Calculation of the UFP begins with the identification of five *function types* of an application, two of them related to the data used by the application and the other three related to the transactions handled by the application:

- Data function types:
  - Internal Logical File (ILF): Logical files (as the user might conceive them, not physical files) updated by the application.
  - External Interface File (EIF): Logical files accessed by the application but not maintained, that is, not updated by it.
- Transactional function types:
  - Input: e.g. transactions to create, modify or delete a record in a internal logical file.
  - Output: e.g. report types.
  - Inquiry: e.g. types of on-line inquiries supported by the application.

Once the occurrences of function types have been identified, they are classified as low, average or high, using a set of prescriptive standards. The UFP is then computed using predefined weights for each function type.

The last step involves the functional assessment of the development and processing environment of the application *as a whole*. This is carried out through a set of fourteen general systems characteristics (GSC).

[1] In the FPA technique, these terms (inputs, outputs and inquiries) are not employed based on their generic definitions, but on constraints specific to the FPA measurement model.

The impact of these characteristics is rated on a scale of from 0 to 5 in terms of their likely effect on the global functional size of the project or application. A value adjustment factor (VAF) is then obtained and applied to the UFP, modifying its value by a maximum of ±35% to produce the Adjusted Function Points (AFP).

A complete description of FPA, including definitions, procedures, counting rules and examples, is found in the *Function Point Counting Practices Manual, Release 4.0*, published by the FPA standards-setting body, the International Function Point Users Group - IFPUG ([20]).

3.3 Purpose

The purpose of this project is to *design* and *test* an extension of the FPA technique that takes into account the specific functional characteristics of real-time software. The requirements established for the design and the tests are explained in the following paragraphs.

Design of an extension to FPA

Design is the first step of the measurement process as modeled by Jacquet and Abran ([21]), the other three steps being the application of the measurement technique, the analysis of the measurement results and their exploitation. Jacquet and Abran have also identified the substeps required within each step. For the step we are interested in, the design of a measurement technique, the substeps are:

- *Definition of the objectives*: What do we want to measure? What will the measurement technique point of view be? What are the intended uses of the measurement method?
- *Characterization of the concepts to be measured*: Once a concept or an attribute has been chosen, an operational definition of this attribute must be supplied. This can easily be done for a tangible attribute, such as the size (attribute) for a person, for example. For more abstract attributes, such as the functional size of a piece of software, the process is expected to be more complex. In such cases, the characterization of the concept to be measured can be achieved by decomposing the attribute into subattributes. From a mathematical point of view, to define an attribute in this way is to define an empirical relational set.
- *Design or selection of a metamodel*: The set of characteristics selected as representative of the software to be measured, and the set of their relationships, constitute the metamodel. The metamodel must describe the entity types that will be used to describe the software and the rules that allow the identification of the entity types.
- *Definition of the numerical assignment rules*: The rules governing the assignment of a numerical value to the measured attribute are defined.

This research project was structured to follow these substeps in the design of an extension to FPA for embedded and real-time software.

Test of the proposed extension

If we want the proposed extension to be practical, an important element to consider in its design is the opinion of, and the feedback from, its users. The second objective of the research project is therefore to conduct field tests using the extension proposed. In exploratory research, the purpose of these field tests is not to carry out large-scale experimentation, but rather to measure some real-time software applications to verify in an empirical way the applicability of the concepts proposed.

3.4 Users

The following groups of users will benefit from the results of this research project:

- Managers interested in using the functional size of real-time applications for conducting productivity analysis and project estimation.

- Software engineers in charge of software measurement and software measurement programs.
- Consultants in the field of software measurement.
- Professionals and industrial associations which define, maintain and promote the use of functional measurement techniques, such as the International Function Point Users Group (IFPUG), the ISO/IEC working group on functional size measurement method standards (ISO/IEC JTC1/SC7 WG12) and the International Software Benchmarking Standards Group (ISBSG).

Project Definition			
Motivation	**Object**	**Purpose**	**Users of Research Results**
Understand and adapt the study object	Function Point Analysis (FPA)	Adapt FPA to take into account the specific functional characteristics of real-time software	Software measurement program managers, designers, consultants and researchers

Table 1 – Exploratory Research Framework - Project Definition Phase

4. Project Planning

In the second phase of this project, the planning phase (ref. Table 2), six major steps were identified and planned in order to fulfill the project objectives:

1. *Literature review*. This step of the project consisted in reviewing the literature on three topics: utilization of FPA to measure real-time software, extensions that had been proposed to adapt FPA to this type of software and industrial usage of these proposals.
2. *Exploratory test*. This step consisted in selecting what would look like the most promising of the solutions proposed in the literature and test it at industrial sites on real-time software. The purpose of these tests would be threefold: to look at the actual performance of FPA on real-time software (strengths and weaknesses), to look at the performance of the proposal at addressing the problems and to define criteria which would have to be met by an improved measurement method.
3. *Design of a prototype of an extension to FPA for real-time software*. From the analysis of the exploratory test results and from the author's own experience, an extension to FPA for the measurement of real-time software was to be proposed. This extension proposal would take the form of a prototype design which would describe the basic concepts of the approach as well as a draft set of measurement procedures and rules to apply the measurement concepts.
4. *Field-testing of the prototype*. The fourth step of the project consisted in measuring actual real-time software applications using the prototyped proposed extension. As this research project was supported and financed by three large Canadian organizations and a Japanese organization, the plan was to measure at least one real-time application from the Canadian partners. Due to industrial constraints, mainly the low availability of the application specialists, the experimental industrial measurement sessions were limited in scope. It was agreed that the industrial partners would select one small application or a self-contained portion of a medium or large application (± 25,000 LOC). The applications were to be measured with the proposed extension prototype as well as with the FPA version (IFPUG, version 4.0) for comparison purposes. At this point in the project, several internal reports would have been produced from each industrial partner, containing their own confidential corporate data and measurement results.
5. *Analysis of the measurement process and measurement results*. This step consisted in carrying out the analysis of the measurement results and the measurement process in order to observe how the following measurement quality criteria were being met, through researchers' observations, and from the feedback and perceptions of the application specialists who participated in the field tests:

- Learnability: How difficult was it to learn, master and apply the basic concepts of the proposed extension prototype, as well as the measurement procedures and rules?;
- Effort required to measure (identify and weight) the new function types;
- Coverage: Were there important elements regarding the functional size of the application that were not being considered by the proposed extension prototype?;
- Coherence between the measurement results and the functional size of the application as perceived by the application's experts (the empirical set);
- Practicality: Were measurement procedures and rules based on current design and documentation practices?;
- Repeatability: Was it possible for different individuals, using the same set of rules, to come up with the same (or relatively similar) results?;
- Convertibility: Comparison between the results obtained with the proposed extension prototype and the FPA version (IFPUG, version 4.0).

6. *Production of a measurement instrument*. The sixth and last step of the project consisted in consolidating the preliminary design and the lessons learned from the field tests in order to build the instrumentation for the proposed design. In essence, the aim was to transform the preliminary specifications of the measurement method into a set of well-documented procedures that would ensure its application in a consistent manner with independence from its designers, from organizational contexts, from cultures and time. This of last step was to produce an equivalent to the IFPUG measurement method standard, most often referred to as the Counting Practices [20]).

According to the research agreement with the industrial partners, the research results in terms of a design of the measurement method was to be placed in the public domain so that it could eventually be adopted as a *de facto* or a *de jure* standard. The industrial partners recognized the benefits that could be derived from its use by the industry for benchmarking purposes. Similarly, it would also be available to the research community, either for further refinements from a measurement technique viewpoint or as a size parameter in productivity studies or in the construction of estimation models.

The project steps identified in this planning phase are listed in Table 2 (left-hand column), together with their inputs (middle column) and their expected deliverables (right-hand column). It is to be noted that steps 3, 4 and 5 are strongly related and interactive. As the field-testing progressed, the proposal was improved, according to the observations and feedback obtained from the participants in the field tests.

Project Planning		
Project Steps	**Project Inputs**	**Project Deliverables**
1. Literature review on: • Utilization of FPA to measure real-time software • Other attempts to adapt FPA to real-time software • Utilization of these proposals	• Articles • Books • Reports	• Literature review report
2. Exploratory tests of FPA and other proposal(s)	• Literature review report • FPA Counting Practices Manual	• Exploratory test report
3. Design of a prototype of an extension to FPA for real-time software	• Literature review report • Exploratory test report • Team expertise with FPA	Prototype: • Measurement method • Description of concepts • Draft of measurement procedures and rules
4. Field tests of the prototype	• Measurement method • Prototype • Documentation of applications measured • Knowledge of application specialist(s)	• Individual field test reports
5. Analysis of measurement process and measurement results	• Individual field test reports	• Report on the results of the analysis
6. Production of a standard measurement instrument for the measurement process	• Measurement method • Prototype • Report on the results of the analysis	• Detailed measurement rules and procedures (Counting Practices Manual)

Table 2 – Exploratory Research Framework - Project Planning Phase

5. Project Operation

In the third phase of this project, the plan outlined in the previous phase is carried out. This section is divided into 6 subsections, one corresponding to each step of the project. Table 5, at the end of the section, summarizes the project's operation phase.

5.1 Literature review

A literature review was carried out and three distinct approaches were identified:

- Addition of new function types: Feature Points ([5]), Asset-R ([17]) and 3D Function Points (Whitmire, 1992);
- Approximation of the final function point results: Application Features ([19]);
- Status quo approach: IFPUG ([22], [23] and IFPUG, 1998).

The results of the literature review can be summarized as follows:

- There are few publications on the usage of FPA to measure real-time software;
- There is no documented structured analysis or empirical study on the strengths and weaknesses of FPA in a real-time environment. Most publications report on the researcher's observations or opinions (Jones, 1986; [17]; [11]; Galea, 1995);
- Despite the common belief that FPA is not adequate for measuring the functional size of real-time software, only three different approaches for adapting FPA to real-time software have been proposed (Jones, 1986; [17]; [11]). None has gained significant recognition from practitioners outside the immediate sphere of influence of their respective designers;
- The status quo proposal is based on an assertion of FPA relevance to real-time software and is not supported by discussion or descriptive argumentation ([22], [23]).

An analysis of these proposals is now presented. The strengths and weaknesses identified in these proposals influenced the design proposed in this research project.

Feature Points (1986)

Feature Points were developed by Capers Jones in 1986 ([5]). According to Jones, real-time software is high in algorithmic complexity but sparse in inputs and outputs, and, when FPA is applied to such software, the results generated appear to be misleading.

Jones introduces a new function type, the *algorithm*, in addition to the five standard FPA function types. An algorithm is defined as the set of rules that must be completely expressed in order to solve a significant computational problem. As examples, he gives a square-root extraction routine and a Julian date-conversion routine. He also gives a set of rules to help determine which algorithms are measurable and significant. He then proposes an approach to weighting algorithms according to the number of their calculation steps and the number of data elements they manipulate. Jones eliminated the classification of the FPA function types as low, average and high, using only the average weights (except for the internal files where the low weight is used).

However, Whitmire ([11]) pointed out that the definition of algorithms, and the rules proposed, are not sufficient for measurement purposes: "No guidance is provided to identify algorithms at the desired level of abstraction." Measurement of algorithms is an issue that has not been solved in mathematics over the centuries, and it is felt that further work is still required to go beyond generalized definitions and rules and reach ones that are applicable.

Furthermore, according to the definition of real-time software discussed previously (section 3.2), algorithms do not necessarily address the main characteristics of this type of software. Real-time software is characterized by interaction with external processes or events (mainly to control them) and the tight time constraints in which the software has to react to changes in these external processes. In real-time software, there is a great deal of internal processing, but the algorithms executed are not necessarily very complex due to the timing constraints. Algorithmic complexity may be an important characteristic of other types of software, like scientific software for instance, but it does not seem to be an important factor influencing the functional size of real-time software.

Asset-R (1990)

According to Reifer ([17]), FPA failed to successfully handle four important characteristics associated with real-time and scientific software: parallelism, synchronization, concurrency and intensive mathematical calculation.

Reifer therefore introduced new function types, according to the type of software measured:

- Data processing software (MIS) = FPA
- Scientific software = number of inputs + number of outputs + number of master files + *number of modes* + number of inquiries + number of interfaces
- Real-time software = number of inputs + number of outputs + *number of stimulus/response relationships* + *number of rendezvous* + *number of modes* + number of inquiries + number of master files + number of interfaces.

In addition to adding the new function types, the weighting factors and the adjustment factor have been eliminated.

However, there is a lack of detailed definitions, measurement rules and examples for the new function types (modes, stimulus/response relationships, rendezvous). It is not clear what has to be measured and how to measure it. It is therefore challenging to apply Asset-R in a consistent manner.

In addition, Reifer proposes different ways of classifying an FP according to the type of the software measured (MIS, scientific or real-time software). He does not define these types of software and no guidance is provided for classifying a given piece software into one of these categories.

3D Function Points (1992)

Whitmire ([11]) developed *"3D Function Points"* as a result of a study investigating the use of FPA in the measurement of the productivity of scientific and real-time software. According to this study, FPA does not accurately measure this type of software because it does not factor in all the problems and characteristics that contribute to the size of the software. This criticism is based on the premise that all software has three dimensions ([24]): data (stored data and user interfaces), function (internal processing) and control (dynamic behavior). According to this scheme, FPA measures only one dimension: the data dimension.

For each of these three dimensions, 3D Function Points (3D FP) identify a set of characteristics or function types that can be quantified and transformed into a single 3D FP index. Once a function type is identified, it is ranked according to its complexity. The various function types are further weighted according to their influence on the overall software. Two characteristics are identified for the data dimension: the retained data and the external interfaces. According to Whitmire, these characteristics are measured in an adequate manner by FPA using the Inputs, Outputs, Inquiries, ILFs and EIFs. The function dimension is characterized by the number and complexity of the internal operations required to transform input data into output data. These operations are called *Transformations*. For the control dimension, two characteristics are selected among several others proposed as possible contributors: *States and Transitions*. A definition and some rules and examples are proposed for the three new function types introduced by 3D FP (transformations, states and transitions).

The concepts introduced by Whitmire are indeed very interesting, and they appear to address key functional characteristics of a wide variety of software.

Application Features

Mukhopadhyay and Kerke ([19]) proposed the concept of *Applications Features*. They were not directly interested in redefining FPA as a measurement technique, but were rather looking for a way to estimate functional size, in FPA units, earlier in the development life cycle. Their goal was to apply the technique to process control software in the manufacturing domain (a category of real-time software).

The authors identified three main Application Features of process control software in the manufacturing domain:

- Communication Features (CF): Several control processes requiring real-time coordination with an upstream/downstream process or with a master process.
- Position Features (PF): Features used to accurately position a component for machining, into a subassembly, or onto a moving conveyor or automated guided vehicle.
- Motion Features (MF): A feature used in applications requiring synchronized movements of components and tools in processes such as complex cutting, gluing or painting of parts. This feature ensures that the part velocity and acceleration requirements are within the specifications.

The study of Mukhopadhyay and Kerke is interesting because it identified specific characteristics of a certain category of real-time software that contribute to functional size and which are known very early in the development life cycle. However, these characteristics (*positioning* and *movement*) are specific to a specialized application domain and cannot be generalized across all types of real-time software. Furthermore, since the authors concentrated specifically on the presentation and analysis of a functional size estimation model in their paper, the measurement process itself is not well covered. They did not provide detailed procedures and rules to ensure precision and consistency across time in the measurement of the three application features identified.

Application Features were included in this literature review because the characteristics identified for real-time software could be useful in the design of the FPA extension.

5.2 Exploratory Tests on Real-time Software

Out of the above proposals, identified in the literature review, the 3D approach was the one most often quoted in the electronic forum of Function Points experts. It was therefore selected as a basis for the exploratory tests aimed at measuring some real-time applications in order to obtain empirical data. Two software applications covering distinct business areas (telecommunications and power supply) were selected by the industrial partners for the exploratory tests. Both applications were measured using both the 3D approach, and the IFPUG FPA status quo approach in order to gain an understanding of how each method worked when attempting to measure the functional size of industrial real-time software.

5.2.1 3D tests results

The following issues were identified during the 3D exploratory tests:

- While the 3D concepts are described well ([11]), the measurement rules are not detailed enough to allow the identification, without ambiguity, of the new function types. What is the difference between an Output in FPA and a Transformation in 3D, for instance? An output, as defined by FPA, contains derived data, which implies a transformation of data. Therefore, with the available 3D procedures and rules, there is room for interpretation. There is a probability that different people can come up with fairly distinct measurement results.
- Finite-State Machines (FSMs) are required to identify States and Transitions. However, in the authors' experience, this type of documentation is generally not available in industrial settings. Feedback from real-time practitioners indicates that, although they all know about and use FSM, they do not document them; even when they do, the documentation is not kept up to date! Therefore, the documentation of FSMs was not observed as an industry practice at the sponsors' sites. A measurement proposal based on such a documentation requirement would not have been a practical proposal to these practitioners.
- Another lesson learned from the exploratory tests was that, if the industry is not ready to spend the resources to document FSM while designing software, indeed a critical step in software development, it is not realistic to expect them to do so for the sole purpose of measurement.

5.2.2 FPA tests results

There are two reasons for the use of FPA during these exploratory tests: a) to learn how it behaved when measuring real-time software, and b) to get first-hand knowledge from these trials to confirm or to infer the opinions expressed by the authors mentioned in the literature review.

This subsection identifies the characteristics of real-time software that were found from the tests not to be adequately tackled with FPA.

The following data and transactional characteristics, specific to real-time software, are difficult to capture with the current version of FPA.

Transactional characteristics

According to the FPA measurement rules, transactional function types rest on the concept of the elementary process, that is, "the smallest unit of activity that is meaningful to the end-user of the business … this elementary process must be self-contained and leave the business of the application being counted in a consistent state" ([20]). However, the number and nature of the steps or *subprocesses* required to perform an elementary process are not taken into account by FPA.

In a MIS environment, processes supplying similar types of services generally display a relatively constant number of subprocesses from one to another. A process that generates data which are then sent to the user usually display four subprocesses, for instance. It receives the request from the user, reads the required information, performs some calculations and sends resulting information back to the user. In MIS software, the number of subprocesses does not seem to have a significant impact on the perceived functional size of any given process. Therefore, even though the FPA measurement technique does not take it into account, FPA is still perceived as an adequate measure of functional size for MIS-type software. The technique provides enough discriminant power in such cases.

Real-time software processes, on the other hand, have a specific transactional characteristic in common: the number of their subprocesses tends to vary a great deal. The following examples illustrate this.

- *Example 1 - An engine temperature control process*. The main purpose of this process is to turn on the cooling system of an engine when necessary. According to FPA, all subprocesses must be completed by the process if it is to be in a consistent state and considered an elementary IFPUG process. A sensor supplies the temperature to the process (subprocess 1). This temperature is then compared to the overheating threshold temperature (subprocess 2). Finally, a turn-on message could be sent to the cooling system if required (subprocess 3). The temperature control process therefore has *3 subprocesses*.
- *Example 2 - An engine diagnostic process.* The main purpose of this process is to turn on an engine alarm when required. Fifteen engine sensors (all different) 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, one unique file for each kind of sensor (15 other subprocesses, one unique sub-process for each kind of sensor). Depending on a number of conditions, an alarm signal may be turned on in the car's dashboard (1 subprocess). In this example, the engine diagnostic process consists of *31 subprocesses*.

If they were to be measured with the FPA technique, the two examples above would be assigned approximately the same number of transactional function points since FPA transactional function types are based on the concept of elementary processes rather than on subprocesses.

However, practitioners at test sites are of the opinion that a real-time functional size measure should take into account the fact that some processes have only a few subprocesses, while others have a large number of subprocesses; they therefore perceived the process in the second example as being larger than the process in the first example.

Data characteristics

Real-time software typically contains a large number of *single-occurrence control data*. Such data are used to control, directly or indirectly, the behavior of the application or a mechanical device. There is *one and only one* occurrence of such data in the whole application. The navigation system of a ship, for instance, periodically calculates the ship's position according to data received from external signals. The ship's position is therefore a single-occurrence control datum because there is only one occurrence of it in the whole application. The number of single-occurrence control data in real-time software can be very large.

The typical MIS logical file usually displays the following data structure: multiple occurrences of a record, each record having one or more fields. A Human Resources application, for instance, could access a group of data containing information on each employee (employee number, name, address, etc.). The employee record is repeated more than once. These groups of data are *multiple-occurrence groups of data*. According to the FPA technique, they are measured as logical groups of data, either as an Internal Logical File (ILF) or an External Logical File (EIF).

Single-occurrence control data are very difficult to gather into logical groups as required by the FPA measurement rules. An extension of the FPA rules is therefore necessary to adequately measure those single-occurrence control data.

5.3 Design of a prototype measurement extension to FPA for Real-time Software

The literature review and the exploratory tests pointed out that an adequate measure of the functional size of real-time software would require that such real-time characteristics as single-occurrence control data and a variable number of subprocesses executed by a single process be taken into account for an adequate measurement of the functional size of such software. From the lessons learned, a prototype of an extension to FPA was designed. This subsection presents the structure of the extension proposed for the measurement of real-time software and describes the new function types introduced in the measurement model.

5.3.1 FPA Real-time Extension: Full Function Points

The new design for measuring real-time software, called *Full Function Points (FFP)*, introduces additional data and transactional function types to measure such software characteristics (St-Pierre *et al.*, 1997a). The new function types are used to measure *control data* and *control processes*[2] (Figure 5, right-hand side), while the other types of data and processes, designated as management data and processes in FFP[3], are measured using the standard FPA technique (Figure 5, left-hand side).

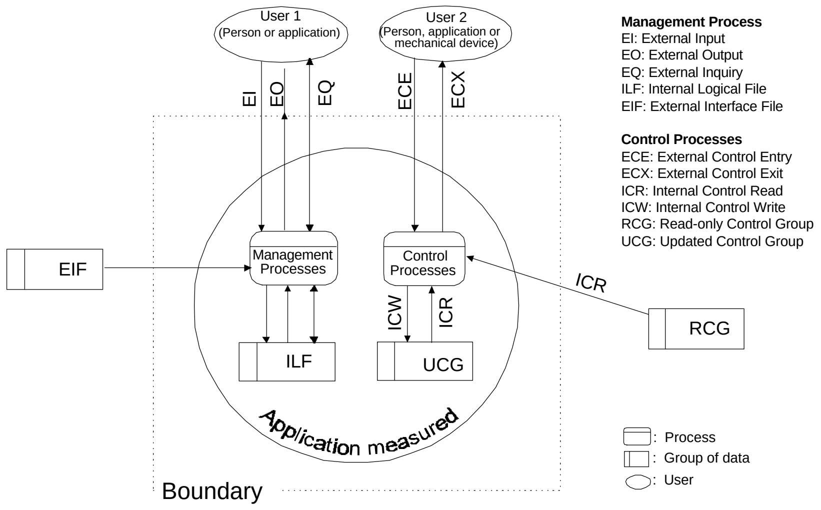
User 1
(Person or application)
User 2
(Person, application or mechanical device)
EI
EO
EQ
ECE
ECX
EIF
Management Processes
Control Processes
ICW
ICR
ICR
ILF
UCG
RCG
Application measured
Boundary
Management Process
EI: External Input
EO: External Output
EQ: External Inquiry
ILF: Internal Logical File
EIF: External Interface File
Control Processes
ECE: External Control Entry
ECX: External Control Exit
ICR: Internal Control Read
ICW: Internal Control Write
RCG: Read-only Control Group
UCG: Updated Control Group
: Process
: Group of data
: User

Figure 5 - FFP components

5.3.2 New control data function types

FFP introduces two new *Control* Data Function Types for the control portion of an application:

- Updated Control Group (UCG): A UCG is a group of control data updated by the application being counted.
- Read-Only Control Group (RCG): An RCG is a group of control data used, but not updated, by the application being counted.

Each of these two groups of control data can be of two types: *multiple-occurrence* or *single-occurrence*. A multiple-occurrence control group displays a typical MIS file structure (see section 5.2.2) and is treated accordingly using FPA rules. A single-occurrence control group, according to FFP rules, is defined as a group of data including *all* single control variables, read or updated, from a functional perspective, inside

[2] Control data: Data used by the application to control directly or indirectly the behavior of another application or a mechanical device.
Control process: Process that controls directly or indirectly the behavior of another application or a mechanical device.
[3] Management process: Process whose purpose is to support the user in managing information, particularly business and administrative information.

the boundary of the application being measured. This also implies that, by FFP rules, the measurement process will take into account only *two* single-occurrence groups: one single-occurrence UCG for all the single-occurrence data *updated* by the application and another single-occurrence RCG for all the single control data *read but not updated* by the application, thereby eliminating the FPA difficulty of constructing logical groupings of such data based on their semantics and relationships.

5.3.3 New control transactional function types

FFP introduces four new *Control* Transactional Function Types:

- External Control Entry (ECE): a subprocess receiving control data coming from outside the application's boundary. In the engine diagnostic process (example 2, section 5.2.2), 15 sensors send data to the application (control data crossing the application boundary). Since there is a unique subprocess for each sensor, there are 15 ECEs within this specific example.
- External Control Exit (ECX): a subprocess sending control data outside the application boundary. In the engine diagnostic example above, the subprocess that sends a message to the dashboard (control data sent outside the application boundary) is an ECX.
- Internal Control Read (ICR): a subprocess reading control data. In the engine diagnostic example above, the subprocesses that read the threshold values are ICRs. In this example, 15 unique subprocesses read different kinds of threshold values at different times for comparison purposes; therefore, there are 15 ICRs in this example.
- Internal Control Write (ICW): a subprocess writing control data.

As opposed to the FPA transactional function types, the new FFP transactional function types are identified at the subprocess level instead of being identified at the IFPUG-defined "elementary process" level. Thus, FFP takes into account a finer level of granularity, the subprocess level, while FPA considers only the process level. A finer level of granularity is important in real-time software since, as explained before, real-time processes display a variable number of subprocesses compared to MIS software processes, and is therefore an important consideration in the measurement of the functional size of real-time software.

The identification of the new transactional function types of an application includes the following major steps ([25]):

1. Look for the different processes performed by the application from a functional perspective;
2. Determine if each of these processes is a management process or a control process. If the process is a management process, the FPA transactional function types are identified using IFPUG procedures and rules. If the process is a control process, the following steps are followed;
   a) Identify the different subprocesses, from a function perspective, executed by the process;
   b) For each subprocess, determine its type (ECE, ECX, ICR or ICW), according to the definitions and the measurement rules;
   c) Assign the corresponding weights (points) in FPA terminology.

5.3.4 Assigning points to the new function types

Control Data Function Types

The number of points assigned to control data function types depends on the updated or read-only types (UCG/RCG). Since multiple-occurrence control data groups have the same structure as 'internal logical files' (ILFs) and 'external logical files' (EIFs) in FPA, they retain the same assignment method of weights, or points, as these two FPA function types, that is, using both the number of fields and the number of record type, and pre-defined intervals and corresponding weights per transactional function type ([20]).

However, for the assignment of points to a single-occurrence control data group, the number of assigned points depends only on the number of fields, or Data Element Types (DETs), in the group. Once the number of DETs is determined following the appropriate counting rules ([25]), the number of points assigned is calculated using the following formulas:

For UCG: ((number of DETs / 5) + 5);
For RCG: (number of DETs/5).

These formulas are designed to keep the size of a single-occurrence group of data as aligned as possible with the size of data function types in FPA.

A look at the FPA normative matrices and the translation tables used to assign points to ILFs and EIFs reveals that updated data groups are assigned more points than read-only data groups. Updated data groups can vary from 7 points (low) to 10 points (average) or 15 points (high). Read-only data groups can obtain 5 points (low), 7 points (average) or 10 points (high). The maximum point difference across data group types is 5 points. In order to maintain this difference between single-occurrence updated and read-only data control groups, the formula for the UCG includes the "+5" constant. Therefore, to obtain a number of points in the same range as the range offered by FPA the number of DETs is divided by 5.

For instance, according to FPA rules, an updated data group with 1 to 19 DETs can obtain 7 or 10 points depending on the number of record element types (RETs). Since single-occurrence updated control groups (UCGs) do not have record element types, the number of DETs is divided by 5 to obtain a number of points that is close to the corresponding FPA range. As a result, a UCG with 1 DET will obtain 5.2 points and a UCG with 19 DETs will obtain 8.8 points. However, contrary to the FPA rules, there is no constraint on the upper limit. According to FPA, updated data groups displaying more than 50 DETs (51...100...1000, etc.) will obtain 10 or 15 points depending on the number of RETs. According to FFP, a UCG with 100 DETs will obtain more points than a UCG with 51 DETs, and a UCG with 1000 DETs will obtain more points than a UCG with 100 DETs. With such an assignment method, the size value assigned to a single-occurrence control data group is not restricted to only three values. It can vary considerably from fairly small to relatively large values without any arbitrary restriction on the upper limit. This approach removes a limitation found within FPA under which very large data groups are not adequately accounted for.

Control Transactional Function Types

The number of points assigned to control transactional function types depends on the number of DETs according to the table below:

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

Table 3 - Control Transactional Function Types: Point Assignment Table

The same set of FPA DET intervals was retained in this case. The two-dimensional structure of the FPA table, with file types referenced as the second dimension, was not retained, however.

According to FPA, transactional function types are based on the concept of an FPA-elementary process. One FPA-elementary process can be an FPA-Input, an FPA-Output or an FPA-Inquiry. One FPA-elementary process (equivalent to one transactional function type) can thus obtain 3, 4 or 6 points (FPA-Input or Inquiry) or 4, 5 or 7 points (FPA-Output).

In contrast, FFP control transactional function types are based on the concept of the control process. One control process is subdivided in subprocesses. Each of the subprocesses is then recognized as an FFP control transaction (ECE, ECX, ICR or ICW). The functional size assigned to one control process (a group of subprocesses or control transactional function types) therefore depends on the number of control transactional function types found within the control process and, mathematically, on the sum of the points assigned to each subprocess. Empirical observations of MIS software measurements have shown that MIS processes usually display no more than 4 or 5 subprocesses. Since with FPA, points are assigned at a higher level (referred to as an IFPUG-elementary process) with a correspondingly in larger number of points, with FFP the assignment is performed at a more granular level and for a greater number of smaller subprocesses, the number of points assigned to a control transactional function type being defined as 1, 2 or 3. Therefore, if a control process has 2 or 3 subprocesses, for instance, it will obtain a number of points within a range similar to FPA's range for its elementary process level. For a control process with more subprocesses, however, more points will be assigned than allowed by FPA rules, thereby lifting another constraint of FPA pertaining to processes with a greater functional size.

The complete set of FFP definitions, measurement procedures and rules, as well as an example, can be found in St-Pierre *et al.* ([25]).

5.4 Field tests of the measurement design prototype

5.4.1 Field-test strategy

To test the FFP measurement design prototype and to verify whether or not the proposed measurement method met the quality criteria expected from a good measurement method, a strategy of industry field tests was selected: a first set of industry field tests under the direction of the research team, and another set of independent field tests conducted by a fourth industrial partner without the assistance of the research team.

5.4.2 Field tests by the research team

For the field tests under the direction of the research team, three real-time software applications from organizations in Canada and the USA were used and measured by at least three people: an application specialist and two IFPUG-certified function point experts who contributed to the design of FFP.

In order to compare FFP and FPA results, each of the three applications was measured twice by the research team: first using FFP (Table 4, upper part) and then using FPA (Table 4, lower part). The results shown correspond to the measurement of the same set of processes. It is important to note that all three applications were real-time applications with control processes only, and therefore did not include any management processes. Table 4 only shows the transactional function type results. For each application, the number of subprocesses measured (or elementary processes for FPA) is indicated, along with the corresponding point assignment.

Examining the measurement results for applications B and C, it can be observed that FFP provides for a functional size that is close to FPA when there are few subprocesses imbedded within each process (Application B: 9 FPA-elementary processes but 40 FFP-subprocesses, for a difference in size of 8 points). For processes displaying a significantly larger functional size, there is a considerably larger number of embedded subprocesses (Application C: 32 FPA-elementary processes, but 468 FFP-subprocesses, for a difference in size of 356 points).

Measurement Method and Transactional Function Types	**Software Applications**					
	A		**B**		**C**	
	Subprocesses	**Points**	**Subprocesses**	**Points**	**Subprocesses**	**Points**
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 size with FFP	**753**	**777**	**40**	**46**	**468**	**479**
	Elementary Processes	**Points**	**Elementary Processes**	**Points**	**Elementary Processes**	**Points**
Function Point Analysis (FPA)						
- 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 size with FPA	**54**	**256**	**9**	**38**	**32**	**123**
Delta: FFP – FPA	**699**	**521**	**31**	**8**	**436**	**356**

Table 4 - FPP and FPA measurement results for the transactional function types

5.4.3 Field tests without the research team's assistance

Another set of three field tests was conducted by one of the project's industrial partners without the assistance of the FFP specialists, using only FFP documentation. Similar results were reported by the industrial partner who independently conducted another set of three parallel field tests ([26]). The detailed measurement results were then submitted to the research team for verification. Only a few minor inconsistencies were detected. These were caused mainly by some initial ambiguous wording in the draft version of the measurement guide. This led to a minor revision of this guide to increase and facilitate the consistency of the measurement process.

As for this second set of field tests conducted independently of the research team by one of the project's industrial partners, the results obtained can be summarized as follows([26]):

- 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.
- In the larger of the independent tests, FFP measured 79 processes out of the 81 expected to be measured with an adequate functional size measure. At the end of the field test, the industrial partner concluded that FFP failed to take into account 2 of the 81 processes because the current design of FFP does not measure processes containing only internal algorithms. The FFP measurement coverage rate was therefore 97% of the overall functionality that they felt should be included in the measurement of the functional size of their real-time software.

5.5 Analysis of the measurement process and measurement results

The following qualitative observations can be made based on the industrial field tests measurement results, and comments collected during the measurement sessions are included.

Impact of real-time software characteristics on measurement results

Comparing the results of the two methods (FPP and FPA), it can be observed that, in the presence of multiple subprocesses, FFP generates a larger functional size than FPA. Indeed, in a real-time environment, FPA has been criticized for generating low size ([5]; Galea, 1995), one which does not seem to be related to the work product measured (Galea, 1995). Since FFP takes into account the subprocesses embedded 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 was observed in the field tests results presented above. During those field tests, the real-time practitioners strongly agreed that a functional size measure which did not take into account user requested subprocesses, like FPA, cannot be a "good enough" functional size measure of their software applications. Therefore, FFP measurement results did represent, for them, a more adequate measure of the functional size of their applications.

It can also be seen in Table 4 that, for FFP, the number of occurrences of each function type is almost the same as the total number of points of the given function types. This is explained by the fact that the majority of the function types were located in the first range of the table used to assign the points according to the number of DETs (see section 5.3.4). This means that the majority of the function types manipulate fewer than 20 DETs.

The measurement results also confirmed the observation that the number of subprocesses of a real-time process varies a great deal from one control process to another. In application A, for instance, there were processes with 3 subprocesses and processes with more than 50 subprocesses.

Ease of understanding

A criticism often made about FPA is that even though the concepts appear to be simple the detailed rules and procedures are challenging to apply correctly, and that FPA-certified measurement experts are required to obtain traceable and replicable measurement results. Within the FFP rules, the complex notion of elementary FPA process has been eliminated and replaced by a simpler concept of control transaction types. During the conduct of the field-tests, once the application specialists had understood the definition of the new function types, they had no problem identifying them. In fact, after a full day of practical FFP coaching, these practitioners were able to measure FFP with little assistance from the full function point experts familiar with the new function types. It is much easier to identify function types that only refer to one type of action (for example, receiving data) as proposed by the FPP rules, than to identify function types that potentially refer to more than one, as proposed by the FPA rules (receiving and updating data as in FPA-defined Input transactions). Furthermore, if the measured software is almost entirely real-time (containing 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, measurement procedures and rules were clear and detailed enough. They felt 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 of what is effectively being documented in the industry.

Measurement effort

Regarding the effort required to measure an application using FFP rules, it has shown to be similar to the effort required for measuring by the FPA rules. Even though more function types had to be measured with FFP, these function types were more easily identified in this research. Indeed, the application specialists seemed to require less assistance from function points experts when measuring with FPP than when using FPA, so the identification of more function types did not increase the measurement effort during the field tests.

Assignment 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 only three intervals and levels. It is important to note that the points 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 point weighting may need to be calibrated, but to do so would require more empirical data.

Early FFP measures

FPA is often used to build estimation models based on the functional size of software applications. However, to use FPA at an early stage of development, one must usually approximate the inputs to the measurement process because not all information is available. The end result is an approximation rather than an exact measure based on detailed rules. Over the years, a number of early approximation FPA methods have been developed. Similar approximation methods could be applied to FFP. Of course, at this point in time, one should not expect FFP approximation methods to be as mature as the ones developed for FPA since FFP is a recently introduced measurement method.

5.6 Production of a Standard for the Measurement Process

With the exception of the FPA and Mark II measurement methods, none of the other proposed alternatives identified in the literature review were supported by a detailed set of procedures to ensure the consistency of the measurement process. Such a feature is deemed to be a key quality characteristic of a good measurement method.

It was therefore seen as imperative for a new measurement method of functional size to retain quality characteristics of FPA from an instrumentation perspective, such as:

- A document that clarifies the measurement objectives and perspective and defines the measurement procedures at a highly detailed level. This is known as measurement instrumentation. Instrumentation is essential for achieving one of the quality attributes of a measurement technique: replicatability. Replicatability entails that different individuals, in different contexts and at different times, who follow these procedures will obtain measurement results that are sufficiently similar, with minimal judgment, and that are auditable.
- The measurement technique is practical, that is, it is based on current software design practices and on the content of the software documentation generally being produced by the practitioners.
- The measurement technique has detailed and documented procedures.
- The official measurement rules are approved and maintained by a standards-setting and monitoring body: this is a key concept in measurement and measurement instrumentation.

At this point in the project, the final version of a measurement instrument referred to as the Counting Practices Manual in IFPUG terminology, was produced. The document produced has the following structure: a brief overview of FPA; a definition of real-time software; a description of the limitations of FPA in a real-time environment; a description of FFP; the detailed FFP measurement procedures and rules; a measurement example and some observations about the use of FFP in the industry. The industrial partners have agreed to place this document in the public domain; it can be found on the following Web sites: http://www.lrgl.uqam.ca and http://www.lmagl.qc.ca. Other reference documents about FFP in English, French and Japanese are also available at these addresses. Draft versions of this measurement instrument were tested during the fields test and improved as required to ensure applicability, repeatability and replicatability.

Project Operation					
1. Literature review	**2. Exploratory test**	**3. Design of a prototype**	**4. Field tests of the prototype**	**5. Analysis of results**	**6. Production of standard**
Utilization of FPA to measure real-time software: • Few publications • Limited data, mostly observations. **Other attempts to adapt FPA to real-time software** • Feature Points (Jones, 1991) • Asset-R (Reifer, 1990) • 3D FP (Whitmire, 1992) • Application Features (Mukhopadhyay/Kerke 1992), **Utilization of these proposals** • Limited	**3D-FP tests:** • Most interesting proposal • 2 applications measured. Problems: – Procedures and rules not detailed enough for measurement purposes – Documentation required not available at the industrial sites **FPA tests:** Characteristics of real-time that are not adequately tackled by FPA: – Variable number of subprocesses of a single processes – Many single-occurrence data Key strengths of FPA from an instrumentation perspective to be kept: replicatability, auditability, practicality, standards	The approach proposed, called Full Function Points (FFP), introduces 6 function types: • Data function types: – Updated Control Group – Read-Only Control Group • Transactional function types: – External Control Entry – External Control Exit – Internal Control Read – Internal Control Write	Research team: field tests: Tests by the research team: - Three real-time applications were measured with FFP and FPA - Three types of professionals were present at the measurement sessions: – Measured application specialists – FFP experts – Observers • Independent field tests: conducted by an industrial partner using the FFP counting manual only and without the assistance of the FFP specialists	**Measurement results** • FFP produces more points than FPA, confirming the observation that FPA generates smaller size. FFP results represent for the application specialists a more relevant measure. • The observation that real-time processes have a variable number of sub-processes was also confirmed. There were processes with only 3 subprocesses, while others had 50. **Measurement process** • FFP seems easier to understand than FPA • FFP concepts and procedures are well described in the FFP manual. An industrial partner was able to measure using the documentation only. • FFP measurement effort is similar to that of FPA. More function types have to be measured, but this is easier to do.	A Measurement Manual was produced. It has the following structure: • Brief overview of FPA • Definition of real-time software • Limitations of FPA in a real-time environment • Detailed description of FFP • Detailed procedures and rules • Example • Observations about the use of FFP in industry. This public document can be found at the following Web sites: http://www.lrgl.uqam.ca and http://www.lmagl.qc.ca. Other documents in English, French and Japanese are also available at these addresses.

Table 5 – Exploratory Research Framework - Project Operation Phase

6. Interpretation of the results

6.1 Interpretation context

The approach selected in this research project for designing the FFP extensions is based on the quality characteristics of a measurement technique such as: fitness for purpose, applicability and replicatability through measurement instrumentation. This approach encouraged the researchers to take into consideration current practices in the design of real-time software, that is, what is currently and effectively being documented regarding user requirements from a functional perspective. Field tests were conducted in order to verify whether or not the proposed extension met its stated objective of measuring the functional characteristics of real-time software. Since the nature of the project is exploratory, the objective of field-testing was not to carry out large-scale experimentation, but rather measure some real-time software applications to verify the applicability of the proposed concepts with practitioners in an empirical way.

From the research purpose viewpoint, feedback from the industrial partners indicates that the research purpose, as described in the project definition, was achieved to a sufficient degree. These practitioners believe that the proposed extension would meet their current and foreseeable needs.

With respect to the field of research, this is an important contribution since there is currently a need in the real-time domain for a meaningful functional size measure that can be used as a reliable independent variable in productivity studies and benchmarking as well as for the development of estimation models based on the projected functional size of the software.

6.2 Extrapolation of results

The project focused on software that reacts to external events and is subject to very tight time constraints (see section 3.2). Field-testing results demonstrate that FFP seems to be adequate for measuring the functional size of this particular type of software, as stated by practitioners at the industrial partners.

According to the research methodology used, results and findings cannot be generalized to different contexts and environments. However, neither is there any *a priori* reason to believe that the new function types of FFP cannot be applied to different contexts and to different types of software.

After project completion, many of the research team members measured additional real-time software at the industrial sites. In these circumstances, additional uses of FFP were identified and implemented, in the context of outsourcing, for instance (Morris *et al.*, 1998a) and in the measurement of multi-layered software (Morris *et al.*, 1998b). Since the reporting of these research results at industrial conferences, further feedback was received from organizations in many countries. Their qualitative comments, based on their actual testing of the FFP measurement method, concur with the research findings presented in this paper. Negative feedback based on actual usage has yet to be reported to the team members.

As well, the research scope was deliberately limited by the industrial partners to the design of a functional size measurement method for real-time software in order to ensure timely delivery of research results within the research mandate and timeframe for such projects in the domain of software engineering measurement methods.

Both the research partners and the research team had agreed up front that the next steps required in the domain of software engineering for the construction of productivity analyses and estimation models would take years of data collection. It was further agreed that such a goal could not be attempted without a solid foundation for the key independent variable of such models, that is, through an adequate functional size measure for real-time software.

Data collection is currently being carried out to investigate these issues. Rapid progress will depend significantly on the willingness of additional industry partners in sharing data.

6.3 Further work

This paper presented the first version of FFP. The authors recognize that further research is required on both the issues investigated and the issues not tackled. The issues investigated through this exploratory research method dealt with the first three steps of the high-level process model of measurement methods ([21]) illustrated in Figure 7, that is, the design of the measurement method, its application on industrial software and the analysis of the measurement results per se. As indicated in the previous subsection, step 4 in the process model was out of scope, that is, FFP uses and contribution in the analysis of other phenomena, such as quality, productivity and estimation.

Figure 8 presents the detailed process model for designing measurement methods including the major substeps. This detailed model is useful both for positioning the work carried out to date and for identifying areas for further research.

Within substep 1 for instance, that is, the definition of the objectives, the exploratory research was purposely limited to real-time software; additional research needs to be carried out to investigate the functional size measurement of scientific and multimedia software, for instance.

Use of more formal experimental research methodologies will be required for further improvements to the substeps previously investigated with exploratory research methods. In some areas such as the quantitative investigation of the replicatability of the measurement method, research methodologies have already been designed. They were applied in investigating this issue using the FPA method as their research object ([27];

Rudolph, 1983, 1989; Low, 1993), ease of use ([28]). However, it must be noted that much of this research work on FPA as a measurement method was carried out 5 to 15 years after the first publication on FPA. Hopefully, needs for such research result information will be identified more quickly and sponsored accordingly by industry leaders in the field of software measurement. Another aspect that could be explored in the future is the applicability of FFP to other domains, like, for example, scientific and multimedia software.

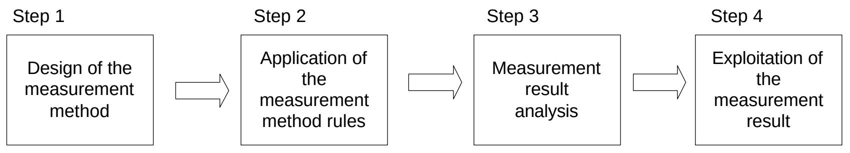
Step 1
Design of the measurement method
Step 2
Application of the measurement method rules
Step 3
Measurement result analysis
Step 4
Exploitation of the measurement result

Figure 7: Measurement Process - High-level Model (Jacquet and Abran 1997)

Research methods for the substeps of designing and improving the characterization of the concept to be measured, as well as the substep for the designing or selecting a metamodel, are not common knowledge in the software engineering community.

Similarly, for the subset of assigning numerical rules, as mentioned in section 5.5, the method for assigning numerical values to new function types needs to be revisited. In this exploratory research, the basis for the assignment of numerical values to the function types was aligned on the current FPA assignment process with some adjustments from empirical observations obtained during the industrial field tests. Some of the FPA assignment rule weaknesses were corrected to increase granularity, simplify assignment rules and eliminate arbitrary upper limits. Data collected through further field-testing will help analyze the relevance of the assignment rules and, therefore, help in adapting them in a formal way. While many have criticized the FPA measurement method on the basis of the numerical value assignment rules, most of them have not provided specific recommendations for remedial action, nowithstanding the notable exception of the work-in-progress by ([29]).

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 measurement procedures and rules, and to obtain objective and quantitative measures of the replicatability of the FFP measurement technique.

Similarly, automation of what is currently an almost entirely manual measurement process needs to be tackled. After close to 20 years of existence of the FPA measurement method there has not yet been a significant breakthrough in this area. The design of the FFP metamodel was purposely made simpler to create a more favorable environment for breakthroughs on this issue, and do so in a more timely fashion.

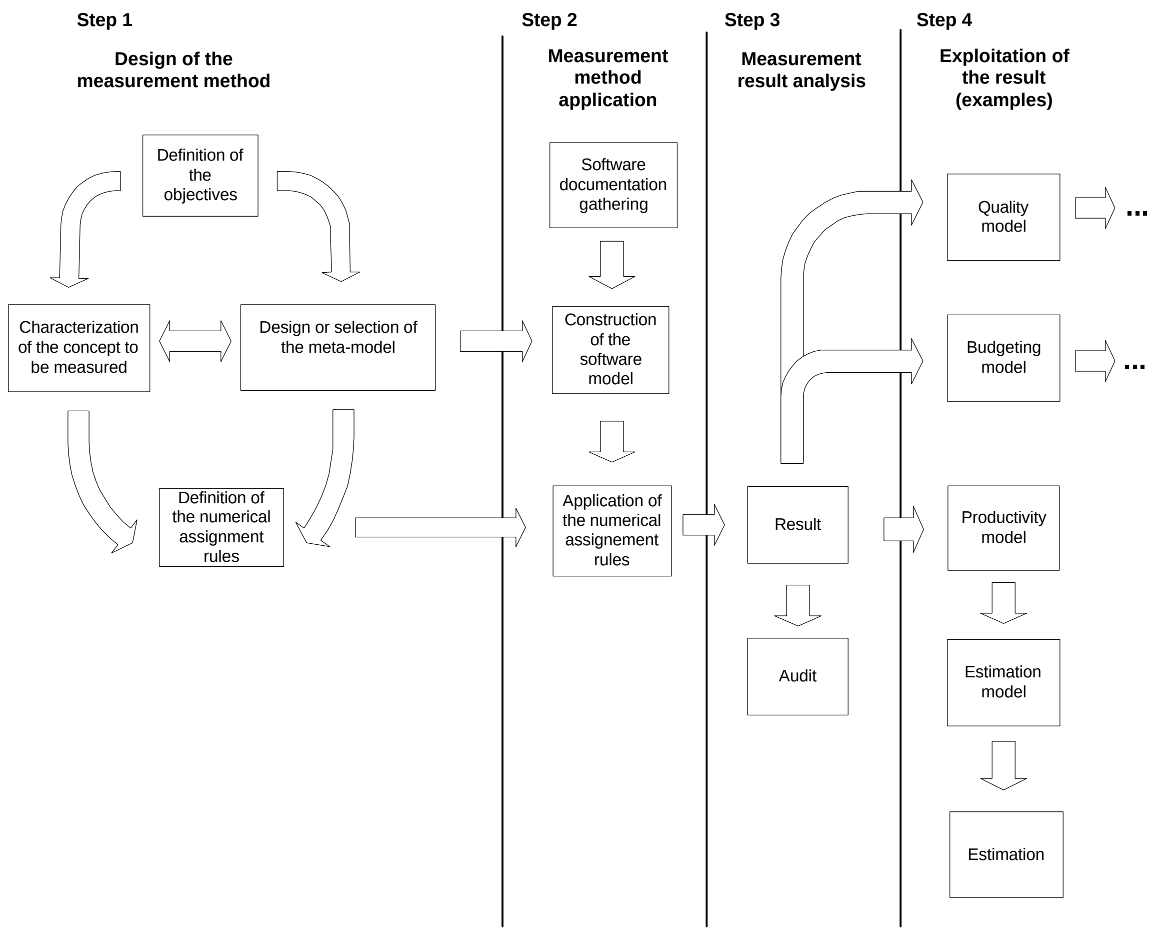
Step 1
Design of the measurement method
Definition of the objectives
Characterization of the concept to be measured
Design or selection of the meta-model
Definition of the numerical assignment rules
Step 2
Measurement method application
Software documentation gathering
Construction of the software model
Application of the numerical assignement rules
Step 3
Measurement result analysis
Result
Audit
Step 4
Exploitation of the result (examples)
Quality model
...
Budgeting model
...
Productivity model
Estimation model
Estimation

Figure 8: Measurement Process - Detailed Model (Jacquet and Abran 1997)

Such further field-testing will also permit the accumulation of enough empirical data to allow the development of meaningful productivity and estimation models.

7. Summary and Conclusion

This paper presented an approach, called Full Function Points (FFP), to adapt FPA so that it could take into account the specific characteristics of real-time software. This approach is based on the observation that real-time software displays the following specific data and transactional characteristics:

- Data: Existence of a large number of single control data, that is, data characterized by the fact that they occur once and only once in the whole application. This data is used to control, directly or indirectly, the behavior of other applications or mechanical devices.
- Transactions: The number of subprocesses in a given real-time process varies a great deal from one process to another. This is not the case in the MIS domain, where processes of the same type have a relatively constant number of subprocesses.

New data and transactional function types were therefore introduced to account for these characteristics. FFP was field-tested with positive results: according to the real-time practitioners, FFP met its stated objective of adequately measuring the functional requirements of their real-time software. In these practitioners' opinion, experimental implementation was performed objectively, precisely and in an auditable manner, in such a way that someone else with the same set of rules would come up with the same results. These same practitioners concurred that FFP measurement procedures and rules are based on current practices as to what is effectively being documented about their real-time applications.

This paper has also illustrated the use and tailoring of Basili’s framework for empirical research to the context of exploratory research. This framework was helpful in structuring the research process and enforced thoroughness and rigor throughout the investigation. The complete framework for this research is illustrated in the following table.

Project Definition			
Motivation	**Object**	**Purpose**	**Users of Research**
Understand and adapt the study object	Function Point Analysis (FPA)	Adapt FPA to take into account the specific functional characteristics of real-time software	Software measurement program manager, designers, consultants and researchers

Project Planning		
Project Steps	**Project Inputs**	**Project Deliverables**
Literature Review on: ! Utilization of FPA to measure real-time software • Other attempts to adapt FPA to real-time software • Utilization of these proposals	• Articles • Books • Reports	• Literature review report
Exploratory test of FPA and other proposal(s)	• Report on literature review • FPA Counting Practices Manual	• Report on exploratory test
Design of a prototype of an extension to FPA for real-time software	• Report on literature review • Report on exploratory test • Our experience with FPA	Prototype: • Description of concepts • Draft of counting procedures and rules
Field-test of the prototype	• Prototype • Documentation application measured • Knowledge of application specialist(s)	• Internal counting reports
Analysis of measurement process and measurement results	• Internal counting reports	• Report on the results of the analysis
Production of a standard for the measurement process	• Prototype • Report on the results of the analysis	• Counting Practices Manual

Project Operation					
Literature Review	**Exploratory test**	**Design of a prototype**	**Field-test of the prototype**	**Analysis of results**	**Production of standard**
Utilization of FPA to measure real-time software: • Few publications • No data, only observations. **Other attempts to adapt FPA to real-time software** Three types of solutions: • Addition of functions • Approximation of final count • Orthodox approach Extensions proposed: • Feature Points (Jones, 1991) • Asset-R (Reifer, 1990) • 3D FP (Whitmire, 1992) • Application Features (Mukhopadhyay Kerke, 1992), **Utilization of these proposals** • No reports are supported by empirical data	**3D-FP tests:** • More interesting proposal • 2 applications measured. Problems encountered: – Procedures and rules not detailed enough for counting purposes – Documentation required for counting not available at the industrial sites **FPA tests:** Characteristics of real-time that are not adequately tackled by FPA: – Variable number of sub-processes of a single processes – Many transient data Key strengths of FPA from an instrumentation perspective to be kept: replicatability, objectivity, auditability, practicality, transferability	The approach proposed, called Full Function Points (FFP), introduces 6 function types: • Data function types: – Updated Control Group – Read-Only Control Group • Transactional function types: – External Control Entry – External Control Exit – Internal Control Read – Internal Control Write	• Three real-time applications were measured with FPP and FPA • Three types of professionals were present at the counting sessions: – Measured application specialists – FFP specialists – Observers • The tests were conducted between Dec-1996 and March-1997 • A fourth test was conducted by an industrial partner using the FFP counting manual only and without the assistance of the FFP specialists	**Measurement results** • FFP produce more points than FPA, confirming the observation that FPA generates low counts. FFP results represent for the application specialists a more relevant measure. • The observation that real-time processes have a variable number of sub-processes was also confirmed. There were processes with only 3 sub-processes, while others had 50. **Measurement process** • FFP seem easier to understand than FPA • FFP concepts and procedures are well described in the FFP manual. An industrial partner was able to count using the documentation only. • FFP counting effort is similar to FPA. More function types have to be counted, but they are easier to identify.	A Counting Practices Manual was produced. It has the following structure: • Brief overview of FPA • Definition of real-time software • Limitations of FPA in a real-time environment • Detailed description of FFP • Detailed counting procedure and rules • Counting example • Observations about the use of FFP in the industry. This public document can be found at the following Web Sites: http://www.info.uqam.ca/Labo_Recherche/lrgl.html and http://www.lmagl.qc.ca. Other documents in English, French and Japanese are also available at these addresses

Project Interpretation		
Interpretation Context	**Extrapolation of Results**	**Further Work**
Study Purpose: • The study purpose was reached as proposed in the project definition **Field of Research:** • Important contribution since there is a need for a functional size measurement technique in the real-time domain to conduct, for ex., productivity analysis and estimation. Other approaches to adapt FPA have been proposed, but it seems that none of these approaches has succeeded in gaining market acceptance.	**Representativeness of field testing:** • Positive factor: – Actual “pure” real-time applications were measured at four different industrial sites. – Application specialists and observers were present at the counting sessions. • Negative factor: – Only four applications were measured	• More field-tests of FFP are required • Degree of repetitiveness must be studied – IFPUG/M.I.T.-type studies required • Usefulness of FFP in productivity, estimation and quality models

Table 6 – Exploratory Research Framework

Acknowledgments

This project was jointly carried out by the Software Engineering Management Research Laboratory at the Université du Québec à Montréal and one of its industrial partners, the Software Engineering Laboratory in Applied Metrics (SELAM). The authors 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.

References

[1] A. Abran and P. N. Robillard, "Function Points Analysis: An Empirical Study of its Measurement Processes," *IEEE Transactions on Software Engineering*, vol. 22, pp. 895-909, 1996.

[2] A. J. Albrecht, "Measuring Application Development Productivity," presented at IBM Applications Development Symposium, Monterey, CA, 1979.

[3] A. J. Albrecht and J. E. J. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, vol. SE-9, pp. 639-648, 1983.

[4] J.-M. Desharnais, "Statistical Analysis on the Productivity of Data Processing with Development Projects using the Function Point Technique," in *Département d'informatique*. Montréal: Université du Québec à Montréal, 1988, pp. 58.

[5] C. Jones, *Applied software measurement - Assuring productivity and quality*. New York, NY: McGraw-Hill Inc., 1991.

[6] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, pp. 406-429, 1987.

[7] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*. Menlo Park: The Benjamin/Cummings Publishing Company, Inc., 1986.

[8] R. B. Grady, *Practical software metrics for project management and process improvement*. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1992.

[9] B. Hetzel, *Making Software Measurement Work - Building an Effective Measurement Program*. Boston: QED Software Evaluation Series, 1993.

[10] S. H. Kan, *Metrics and models in software quality engineering*. Readings, Massachusetts: Addison-Wesley Publishing Company, 1995.

[11] S. A. Whitmire, "3D Function Points: Scientific and Real-Time Extensions to Function Points," presented at Pacific Northwest Software Quality Conference, 1992.

[12] V. R. Basili, R. W. Selby, and D. H. Hutchens, "Experimentation in Software Engineering," *IEEE Transactions on Software Engineering [ISO]*, vol. SE-12, pp. 733-743, 1986.

[13] P. Bourque and V. Cote, "An experiment in software sizing with structured analysis metrics," *Journal of Systems and Software*, vol. 15, pp. 159-72, 1991.

[14] V. Côté, G. Métivier, P. Bourque, and J.-P. Jacquet, "Caractérisation des logiciels industriels de gestion," *Génie Logiciel*, vol. Actes des Neuvième journées internationales «Le génie logiciel et ses applications» (GL96), pp. 92-102, 1996.

[15] P. Bourque, M. Maya, and A. Abran, "A Sizing Measure for Adaptive Maintenance Work Products," presented at IFPUG Spring Conference, Atlanta, 1996.

[16] B. Kitchenham, P. Brereton, D. Budgen, S. Linkman, V. L. Almstrum, and S. L. Pfleeger, "Evaluation and assessment in software engineering," *Information and Software Technology*, vol. 39, pp. 731-734, 1997.

[17] D. J. Reifer, "Asset-R: A Function Point Sizing Tool for Scientific and Real-Time Systems," *Journal of Systems Software*, vol. 11, pp. 159-171, 1990.

[18] S. C. Whitmire, "An Introduction to 3D Function Points," *Software Development*, pp. 43-53, 1995.

[19] T. Mukhopadhyay and S. Kekre, "Software effort models for early estimation of process control applications," *IEEE Transactions on Software Engineering*, vol. 18, pp. 915-24, 1992.

[20] IFPUG, *Function Point Counting Practices Manual, Release 4.0*. Westerville. Ohio: International Function Point Users Group, 1994.

[21] J.-P. Jacquet and A. Abran, “From Software Metrics to Software Measurement Methods: A Process Model,” presented at Third International Symposium and Forum on Software Engineering Standards (ISESS'97), Walnut Creek, CA, 1997.

[22] D. Garmus, “Introduction to function point counting in a real-time environment,” *The Voice 1996 - A publication of the International Function Point Users Group*, vol. 1, pp. 27-29, 34, 1996.

[23] D. Garmus and D. Herron, *Measuring the Software Process: A Practical Guide to Functionnal Measuring*. Upper Saddle River, New Jersey: Prentice Hall, 1996.

[24] T. DeMarco, *Controlling software projects*. Englewood Cliffs, New Jersey: Yourdon Press, a Prentice Hall company, 1982.

[25] D. St-Pierre, M. Maya, A. Abran, J.-M. Desharnais, and P. Bourque, “Full Function Points: Function Points Extension for Real-Time Software - Counting Practices Manual,” Université du Québec à Montréal, Montréal, Technical 1997-04, September 1997.

[26] D. St-Pierre, M. Maya, A. Abran, and J.-M. Desharnais, “Adapting Function Points to Real-Time Software,” presented at IFPUG 1997 Fall Conference, Scottsdale, Arizona, 1997.

[27] C. F. Kemerer, “Reliability of function points measurement: a field experiment,” *Communications of the ACM*, vol. 36, pp. 85-97, 1993.

[28] S. Nishiyama and T. Furuyama, “The validity and applicability of function point analysis - as related to specification quality and ergonimics -,” presented at EOQ-SC'94, Basel, Switzerland, 1994.

[29] G. E. Wittig, P. M. Morris, G. R. Finnie, and E. E. Rudolph, “Formal Methodology to Establish Function Point Coefficients,” presented at IFPUG, Measuring New and Emerging Technologies in the Next Millennium, Scottsdale, Arizona, 1997.