Measuring *ALL the Software not just what the Business Uses*

Pam Morris and Jean-Marc Desharnais

Total Metrics and SELAM

Function Point Analysis (FPA) is used by organisations worldwide as one of the measures used to establish the baseline size of their software assets in outsourcing contracts. This paper introduces new techniques, which enable all the functionality delivered and worked on by the supplier to be included in the productivity performance monitoring of these contracts. Typically only the business applications layer can be measured using FPA. The infrastructure software e.g. Utilities, device drivers and gateway applications, are usually overlooked because FPA is not designed to, nor easily adapted to, measuring internal layers of functions not delivered to the business user. This new Full Function Point Technique, developed by the University of Quebec in Montreal and SELAM, is a refinement of the FPA technique. It is no longer limited to only measuring MIS type applications but was specifically designed to meet the needs of organisations that build and support infrastructure applications, real-time and embedded software.

1 Introduction

Outsourcing Information Technology's (IT) software development and maintenance activities has become increasingly popular as a means of enabling an organisation to more effectively focus on their core business activities.

The high cost and risk associated with these IT outsourcing contracts means that they need to be carefully monitored and managed. Both the client and the supplier need to establish a means by which the client's software assets can be quantified and the supplier's performance can be evaluated and compared to agree targets. The most common mechanism for providing these performance measurements is to measure the supplier's productivity rates in units of software product delivered per unit effort or units of software product delivered per unit cost.

The units of software product delivered are usually measured using a Functional Size Measurement (FSM) method called Function Point Analysis (FPA). FPA was developed by Alan Albrecht in the late 1970s and has since been refined by the International Function Point Users Group (IFPUG). Until recently the use of FPA has been concentrated within the business application domains i.e. software which delivers functionality to the human business user. These applications are typically commercial and management information systems (MIS) software. FPA has been found to very effective in measuring the functionality delivered by these types of applications, particularly when used for measuring the productivity of software development and as input into estimates for project resources and schedules. However, in recent years the need to quantify software has extended beyond just measuring business applications for these purposes. Many developers are working on real-time embedded and control software where the users are equipment rather than people. Other developers are building infrastructure software, which enable the business applications software to operate and have other applications as their users. With the advent of outsourcing, the client and supplier need to monitor *all* software worked on by the development teams, not only the applications which deliver software directly to the business users, but also include software that has other applications or equipment as its primary users. The functionality delivered by these 'other types' of applications does not behave in the same way as functionality delivered primarily to human users.

Many functional size measurement specialists have found that FPA is less effective when measuring software, which delivers functionality to 'users' other than the business user compared to when it is applied in its traditional domain for which it was designed. This creates a problem in outsourcing contracts where all the supported software needs to be included in the performance measures. This paper explores the reasons for these observed limitations of FPA and shows how they were overcome using a new functional size measurement technique called 'Full Function Points' (FFP). Using this FFP technique the authors were able to successfully measure these 'other types' of applications or 'non-business' software as they are referred to. The paper presents the results from a pilot project, which measured these non-business software applications, using both the FPA and FFP techniques. These types of applications constituted about one-tenth of the total applications for the outsourcing contract. Results obtained showed that the FFP measures have the potential to be used successfully for productivity monitoring and estimating, where FPA measures have been tried and were found to be lacking.

2 Types of Software

Software that is usually included within the scope of the MIS organisation can be categorised based on the types of service by software (Figure A).

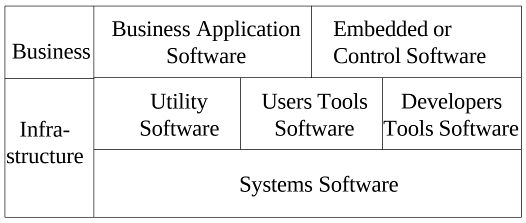
Business
Business Application Software
Embedded or Control Software
Infra-structure
Utility Software
Users Tools Software
Developers Tools Software
Systems Software

Figure A

2.1 Business Applications Software

These applications deliver functionality that supports the organisation's core business.

The users are primarily human business users, however a small proportion of the functionality may also be delivered to, or triggered by, other Business Applications. This type of software is typically business or commercial (MIS) software and would include Payroll applications, Accounts Receivable or Fleet Management systems.

2.2 Embedded or Control Software

These applications also deliver functionality, which supports the organisation's core business.

The users are primarily other software applications embedded in equipment. This type of software typically operates under strict timing conditions and is often referred to as Real-time software. Examples would include Equipment monitoring Systems, Telephone Switching Systems.

2.3 Utility Software

These applications deliver software that provides the infrastructure to support the Business Applications Software. The users are primarily Business Applications, which trigger the operation of the utilities but may include the developers or business administration people as the administrative users. Examples would include backup utilities (to ensure the data reliability of the Business Application) or archiving utilities (to optimise the performance of the Business Application). Other examples are installation and conversion software.

2.4 Users Tools Software

These applications are tools used by administrative users to create the functionality delivered by the Business Applications Software.

The users are primarily Business Applications Software, which utilise functionality delivered by the tools to enable them to deliver functionality to the business. Administrative human users of these tools may be from either the Business or IT. Examples would include Report Generators, Spreadsheets, and Word processors.

2.5 Developers Tools Software

These applications are tools used by developers to create the functionality delivered by the Business Applications Software. The users are primarily other applications, which are either generated by, or used as input to, the tools operation. Human Users may also include IT developers as administrative users. Examples would include, Code Generators, Testing software, New Product Generators etc.

2.6 Systems Software

These applications enable all other types of software to operate and deliver functionality. The users are primarily other applications with a limited interface to Human IT operational staff. Examples would include operating systems, printer drivers, protocol converters, and presentation software.

3 FPA and FFP Concepts

This section compares the differences between the basic concepts of FPA and FFP. FFP was developed by the University of Quebec in Montreal (UQAM) and the Software Engineering Laboratory in Applied Metrics (SELAM) to take into account functional characteristics specific to real-time software. However its concepts have been found to be equally applicable to other types of software particularly where the primary users are not human.

FPA was designed and refined for Business applications software, which usually constitutes about 70% - 80% of a commercial organisation's software portfolio. This section describes the basic concepts of FPA, which have contributed to it being able to be used to effectively measure Business Applications software over the past 20 years. Full details of the rules for applying FPA can be found in the IFPUG Counting Practices Manual 4.0

3.1 FPA concepts

The FPA technique measures functionality by quantifying the software's processes (inputs, outputs, enquires) and data files (internal and external). It is based on the following fundamental principles:

1. Functional size is the measure of the functionality delivered to the end business user. Only processes that send or receive data to, or from, the external user boundary are included in the measurement of functionality delivered to the user.
2. A process is required to have a predominant role of *either* inputting *or* extracting data. This predominant role determines the process type (input, output or enquiry).
3. The functional size of a process is directly related to the amount of data, which crosses the external user boundary of the software during the execution of the predominant side of the process.
4. An extremely complex process of a particular type can only be measured to have, at the most, double the functionality of the simplest process of that type.
5. Functionality delivered by stored data is a significant contributor to the overall functional size of the software.
6. Functionality changed is recorded as being the measure for the whole function irrespective of the proportion of the process being changed.

3.2 FFP concepts

1. The FFP technique measures functionality of the software by quantifying the software's sub-processes within each process and control data (internal and external). It is based on the following fundamental principles:
2. It measures functional size from the functional perspective instead of the external user view. I.e. measures the functionality required to be delivered by a process to the user of the process not just the functionality experienced directly by the user. Sub-processes that read and write the data to and from data groups are included in the measurement of functionality in addition to the sub-processes required to receive data (entry) and extract data (exit).
3. A process is not required to have a predominant role *per se*. In order to measure size it is only necessary to identify all the entry, exit, read and write sub-process types.
4. The functional size of a process is determined by measuring the size of individual sub-processes, which are not limited to those which only accept (entry) or extract (exit) data. The predominant role of a process to either extract or accept data does not influence its size.
5. An extremely complex process of a particular type can be sized accordingly by awarding proportionally more points. In theory, there is no limit to the number of points awarded for one

specific process.

6. Functionality delivered by stored data is a less significant contributor to the overall functional size of the software, than functionality delivered by the processes.
7. Functionality changed is recorded at the level of the sub-processes. Only a part of the process (identified by the sub-processes) is credited for the change.

The FFP technique has taken into consideration current industry practices in the design of real-time software and the design of what is currently documented regarding the user requirements from a functional perspective. It introduces new concepts for measuring data and transactional function types, which cater for the characteristics of non-business applications software[1]

Overview of FFP Technique

FFP, like FPA, measures functional size by evaluating transactional processes and logical groups of data. However unlike FPA, the transactional processes are evaluated at sub-process level. I.e. the sub-processes within a process are identified, classified and assessed. Points for functional size are awarded at sub-process level. The sub-processes within each transactional process can be categorised into one of four types:

- External Control Entry (ECE)
- External Control Exit (ECX)
- Internal Control Read (ICR)
- Internal Control Write (ICW)

The data groups, which contribute, to the overall size fall into two categories.

- Multiple record data groups, which can be either updated or only, read by the processes. These are similar to the Internal Logical files and External Interface files counted for FPA.
- Single Record (single occurrence) data groups. These data groups may be maintained by the processes (Updated Control Group - UCG) or only read by the processes (Read-only Control Group - RCG). The single occurrence data groups contain all instances of single control values used by the processes. There may be only one instance of a UCG or RCG per application.

[1] . A full description of the FFP technique is available in the Full Function Points: Counting Practices Manual Technical Report 1997-04

4 COMPARISON OF FPA and FFP FOR Non-Business Applications Software

The following section illustrates how the conceptual differences between FPA and FFP impact their capability to measure functional size in non-business applications software.

4.1. Identifying Functionality Delivered to the End User

In the past functional size measurements using FPA have usually only included functionality of processes that send or receive data to or from the external user boundary I.e. FPA is usually used to report the size of the functions delivered to the external business user. In contrast, the end user of the non- business applications software is primarily other software applications. Many outsourcing contracts are based on the payment for function points delivered (added, changed or deleted) to the end business user. Contention arises when the customer requests infrastructure functionality to be developed or changed and the charges for a contract are based on function points impacted. (E.g. A client request is to improve the performance of all database accesses. The developers do this by modifying the archiving software to archive selected data). The *quality* of the end user functionality (i.e. performance) is improved but the functional size of the business applications software is not impacted. FPA specialists consider performance as a general system characteristic rather than functionality changed in software. Problems with the contract arise when the supplier requires payment for the software changes but the client maintains that contract implies that only changes to function points delivered to the business will be paid for. Issues also arise when a project needs to build infrastructure software in addition to the Business Application software but the infrastructure software is not included in the project's functional size. Consequently, the impact on the effort for building the infrastructure software is not adequately catered for by the adjustment for the technical and quality features in the VAF of the Business Application. The FFP approach, from the functional perspective instead of external user view enables the functionality delivered by non-business Applications to be effectively measured. I.e. when estimating projects, it enables infrastructure software, which supports the Business Applications to be sized, and their productivity rates and resource estimates to be separately established.

4.2. Measuring internal and external sub-processes

FPA measures functionality by evaluating the amount of data, which crosses the external boundary during the processing of the predominant side of a process. The processes within Business Applications software tend to be primarily involved with inputting and extracting data so the amount of data movement is a good indicator of overall process size. However problems arise with non-business applications where there is significant internal processing compared to the processing required to move the data into or outside the boundary. This poses a problem in outsourcing contracts where the performance of the suppliers is measured in Function Points delivered. Since any process which has significant internal processing but little external visibility will not be accredited for the full amount of functionality it delivers. This will result in low recorded productivity rates for the project since although considerable effort may have been expended adding or changing the internal logic within a process, if it only accepts a minimal number data items then its function points will be low. For example, a process, which receives the coordinates of a radar system and only sends out confirmation or an alarm, may have significant internal processing to check positioning and exception conditions. The FPA measurement standards, which consist of measuring only the external data moverments, do not represent an adequate measure of the functionality delivered by this process.

In comparison FFP measures *all* the sub-processes within a process i.e. not only the data entering and exiting the process but the internal processes of reading and writing to the data groups. By measuring all the functionality that a process is required to deliver rather than just the external aspects of a process it captures its full complement of functionality for estimating and productivity comparisons.

4.3. Categorizing processes which do not have predominant role

FPA requires a process to have a predominant role of either inputting or extracting data. This predominant role determines the process type (input, output or enquiry). The processes within non-business Applications software tend not to have a predominant role making them difficult to categorize unequivocally into type, since they often:

- Accept data which enters one side of the application boundary, process it and send the results immediately externally across the boundary to another application. (e.g. translation process in a gateway protocol converter or extraction process in screen scraper software)
- Involve the processing of multiple and variable sets of incoming data and outgoing data. Neither side of the process is predominant. (e.g. processes in real-time equipment monitoring)

It is difficult to consistently measure these types of processes since the same process may be categorised into differently each time it is sized, resulting in a range of reported functional sizes for the same application. I.e. different people will categorise the same process as either an input, output or enquiry each of which will be allocated a different number of function points. These variations in results cause contractual problems when performance targets fall within the error boundaries of the measures.

In comparison FFP categorises sub-processes within processes. The number of points awarded is independent of the *type* of sub-process. Therefore any errors or variations in categorisation do not impact the measured size of a process.

4.4. Sensitivity to large variations in functionality delivered by Processes

The FPA rating scale used to award function points to a process increases by just over two fold between the points awarded for the simplest process (e.g. Low complexity input is awarded 3 points) and the points awarded for the most complex process (high complexity output is awarded 7 points). The coarseness the FPA measure is usable and acceptable for business applications where in most cases a two fold difference in size is representative of the range of functionality delivered by most processes.

However the processes delivered by non-business Applications software have been found in practice to deliver a much broader spectrum of functionality than a two-fold difference. However, no matter how complex the functionality delivered by a process FPA cannot award it points beyond the maximum number. This can cause vast errors in estimates of effort for a requirement to change a number of very complex processes. This is a concern in an outsourcing contract where a requirement of the contract is to provide fix price estimates on projects using historical productivity rates.

The FFP measure is not restricted to awarding a maximum number of points to any one process. It therefore much more sensitive to large variations in size of processes experienced in non-business applications.

4.5. Measuring Process Rich, Data Poor software

FPA measures the data groups accessed by the processes as one of the major contributors to functional size. FPA is based on the concept that maintaining and reporting on data is the software's primary role. It includes within its total size the function points awarded to the logical data groups accessed by the application. These data groups also contribute to the functionality attributed to each process that access the data groups. However non-business Applications do not have the same emphasis on stored and maintained data. Their processes often operate by referencing relatively static threshold values and parameter controls to make decisions on the data to output. The processes may involve multiple steps or sub-processes which reference and update fields in these data groups but the data itself is often minimal compared to the many steps to analyse it and react appropriately. The data input to the process is not usually permanently stored but used for the duration of the process. The permanent data, which is accessed, is relatively simple and usually consists of historical logs, threshold values or parameter controls.

Difficulties arise when using FPA to measure non-business Applications since FPA uses the amount of stored data as a significant factor in determining the functional size of the application. Where the stored data is simple but the processing of the stored values is complex, the functional size of the application is underestimated. In contrast, FFP determines the size of a process by measuring the number of unique *data accesses* rather than the number of unique *data groups.* Thus giving a better indication of size for applications that have a few groups of simple data but complex use of that data.

5 The layered applications

If the characteristics of the different types of software are considered, then they can be identified as being on a different level to that of the business user applications and to each other. In fact, they constitute different layers: the business user will see the utility software as providing the technical features of their software, while the user of utilities could consider the developer tools as the technical means of providing their software. The lowest layer in the non-inclusive list is the operating systems software.

Within an organisation, especially within a telecommunications organisation, it is possible to identify additional types of software. For this reason, it is necessary to find definitions and rules to assist the person measuring the software to identify the different layers of the software. The functionality delivered by software within one layer is not the same type of functionality delivered by software within a lower layer. Functionality delivered by software at different layers cannot be combined nor directly compared. Although the effort to build the different types of software can be combined and the productivity rates to build and support the different software types can be compared.

5.1 Definition of a layer

An item of software is typically structured in horizontal “layers”[2]. Each layer is a world unto itself. A layer does not need to know how its inputs are generated. It is just required to deal appropriately with the input when it is received. A layer perceives the layer below it as a set of primitives. Each layer "sees" the layers below but cannot see the layers above. Similarly, what happens to data output from a layer is irrelevant to the producer once that output has been dispatched. The internal operation of one layer does not need to be known by any other layer. Indeed, it is preferable that internal details be protected from outside alteration. Making one layer dependent upon internal organisation of another is very undesirable. Such dependency restricts the ability to maintain (i.e. change or enhance) the layer which is depended upon. This is the basic principle of information hiding. The organisation ought to be able to entirely re-construct a server layer without affecting clients, which use its primitives, so long as the software retains the same interface. For this reason, the organisation usually prevents a client layer from directly using the services of any subordinate layers except the one immediately below it.

[2] "Function Point Counting Practices for Highly Constrained Systems – Release 1.0”, United Kingdom Software Metrics Association (UKSMA), March 1993, UK.

5.2 Rules for identifying a layer

The following are the recommended rules for identifying a subordinate layer from a superior layer:

A. A subordinate layer is considered as a technical implementation by the superior layer.
B. A subordinate layer does not recognise the superior layer and has no functionality by itself to communicate with the superior layer.
C. A subordinate layer could work without the assistance of a superior layer. If the superior layer is not working properly, this will not affect the subordinate layer.
D. A subordinate layer is independent of the technology used by the superior layer.
E. A subordinate layer provides services to the superior layer and could be used by the superior layer.
F. A superior layer could not fully work if the subordinate layer was not working properly.
G. A superior layer does not necessarily use all the functionality of the subordinate layer.
H. A subordinate layer could be a superior layer from the perspective of a layer below it.
I. A layer always delivers functionality
J. A layer is software, not a piece of equipment. The software could be embedded, within the equipment.
K. From one layer to another the data is perceived differently.

6 FIELD RESULTS

6.1 Background

The following results are from an organisation in the telecommunications industry, which had outsourced the development and maintenance of the majority of their software applications. At least 10 –15% of these could be classified as non-business Applications and demonstrating the characteristics previously described. Previous attempts had been made at measuring the size of these non-business applications using FPA, but developers had found low correlation between the measured size of projects, with the actual of effort to develop the software and had abandoned its use for estimating. They had also experienced significant variance in the size of the applications when measured by different people. Both factors lowered their confidence in the reliability of the FPA measure and identified a need to find an alternative effective functional size measurement technique.

6.2 Method

In order to ensure consistency of interpretation, only one person was used to measure the applications. This person had 12 years experience in functional size measurement and was an IFPUG-certified function point specialist (CFPS) and co-author of the FFP technique.

FPP and FPA were used to count 5 application areas and 10 sub-applications within those areas. Of these 10 sub-applications, 5 could be described as non-business Applications software while the remaining 5 were mostly Business Applications software (B, C2, C3, D1, and D2). Both measurement techniques were used to measure 7 of the sub-applications. However, due to time constraints, the remaining 3 sub-applications were measured using only FPA (these sub-applications were entirely Business Application type (C3, D1, and D2). Measuring them with FPA only, was not considered to be a problem since our past experience had demonstrated that, for the Business Applications, the FFP sizes are comparable with those obtained by FPA for applications like these which have no complex processes.

6.3 Results

For most of the sub-applications FFP measured significantly more functionality than that measured using FPA. Table 1 shows that the total FPA size for all applications was 4067 function points and the total FFP size for all applications was 8370[3] points. Whilst we are not proposing that FFP points and FPA function points are equivalent, we believe that the results are significant for the following reasons:

- When counting typical Business applications with both techniques, the results were similar. This is demonstrated by the results for sub-Application C2 (FPA=878, FFP=896) and to a lesser extent sub- application B1(FPA=764, FFP=791).
- Some processes within the non-business sub-applications could not be measured using FPA because the functionality that they delivered could not be reliably categorised into the FPA Function types, and did not comply with the definition of an elementary process. This was the case for D3 (FPA=0, FFP=2604), A1 (FPA=210, FFP=794) and E1 (FPA=43, FFP=318).
- When counting complex processes, within non-business Applications, the difference between the size measured by the two techniques was found in all cases to be significant. Sub-applications of A, are typical examples where A1 the difference between the two measures is 74% (FPA=210, FFP=794) and for A2 the difference was 37% (FPA=115, FFP=183).

[3] assuming sub-applications C3,D1 and D2, (all of which are business applications) were the same size if measured by FFP as that measured by FPA

Application	Sub-applicn	FPA (fps)	FFP (points)	Difference	Difference %	Type
A	A1- L1	210	794	584	74%	Non-Business
	A2- L2	115	183	68	37%	Non-Business
B	B1-L1	764	791	27	3%	Mostly Business
C	C1-L3	272	676	404	60%	Non-Business [4]
	C2-L2[5]	878	896	18	2%	Business
	C3-L1	1273	N/C[6]	0	0%	Business
D	D1-L1	727	N/C	0	0%[6]	Business
	D2-L1	110	N/C	0	0%[6]	Business
	D3-L2	0	2604	2604	100%	Non-Business
E	E1-L1	43	318	275	86%	Non-Business
		4067	**8372**	**3980**	**51%**	**FFP total**

Table 1

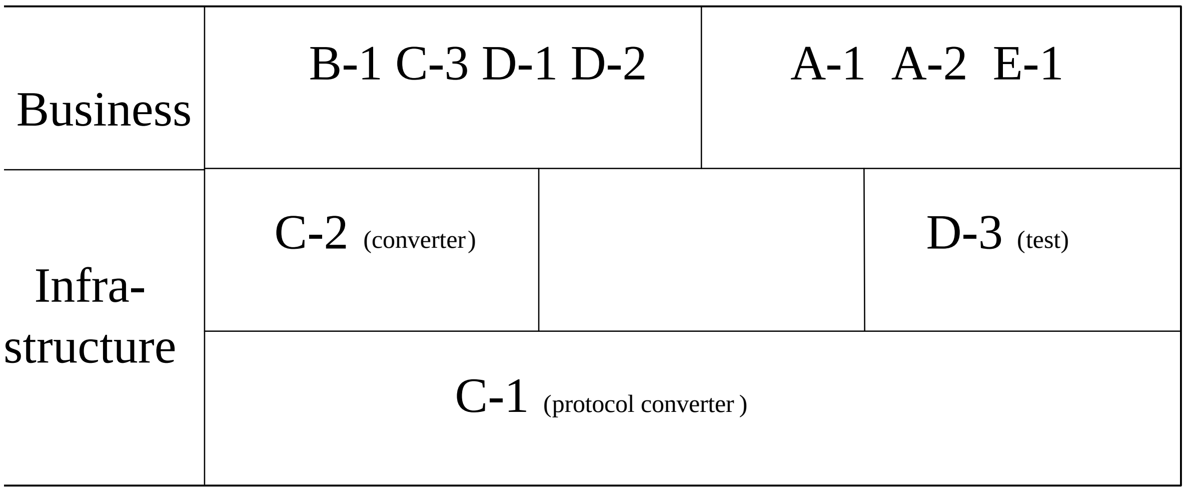

Figure B

[4] This system could be considered a batch system.
[5] We had difficulty categorising the processes as either an input, output or enquiry. However this did not have an impact on the overall size.
[6] N/C = Not counted

6.4 Discussion

The sub-applications identified within each of the application areas were separately identified if they delivered functionality to a different category of user. When these sub-applications delivered functionality hierarchically to each other they were identified as being in a particular ‘layer’. Where each layer was the ‘user’ for the layer below it. The top layer (L1) being that which delivered functionality to the core business. Applications within any layer may deliver functionality to peer applications within the same layer, however their primary user is applications in the layer directly above them in the hierarchy, i.e.:

- Layer 1 (L1) delivered core business functionality directly to an external user [either a person in the case of Business applications or equipment in the case of embedded (real-time) software.
- Layer 2 (L2) identifies infrastructure software, which delivers functionality to support its users, i.e. primarily the applications in L1.
- Layer *n* identifies infrastructure software, which delivers functionality to support its users, i.e. the applications in L*n-1*.

Hierarchically layered software, below Layer 1, provides the infrastructure for IT organisations. Although the users below the first layer are not directly the business users the business users are still able to take advantage of the infrastructure software because, without it, their applications would be inoperable. The infrastructure software delivers functionality indirectly to the business users via other applications. In the context of outsourcing an IT organisation, measuring software *only* from a business user perspective i.e. Layer 1, hides a good part of the actual functionality as shown with application C where 47% (1150 FFP) of the total functionality (2,423 FFP) was measured to be in infrastructure software. It is important in any outsourcing contract that the infrastructure software, which is developed and supported as part of the contract, is, identified separately from the software it services and its functional size is:

- Measured consistently and completely.
- Measured independently of the software it services.
- Analysed independently of the software it services. That is, it is not incorporated into the total size of the software of which it is a user or software that acts as its user. For estimating and productivity measures its size is used and reported independently.

The results of this preliminary field study showed that, for the applications measured, FFP was:

- More effective than FPA in capturing all the functionality delivered by non-business applications software.
- Was easier to apply more consistently than FPA.
- Needed a greater level of detail of the internal processing requirements of each process in order to measure it than that required by FPA.

7 Conclusion

All software developed and supported by the supplier in an outsourcing contract, needs to be able to be measured in order to provide input into performance monitoring. This software usually includes applications, which do not deliver functionality directly to the human business users. Experience showed that the FPA functional sizing technique was not well suited to measuring software that has other *software* as its primary users. This initial study indicated that the FFP technique provides a more effective method of measuring these infrastructure and real-time embedded and control software applications (non-business applications), than FPA. FFP's ability to measure:

- consistently across layers,
- in a repeatable way, by different people, and
- effectively i.e. measure all the functionality delivered by these types of applications,

make it a good candidate to be considered when measuring *all* the software involved in monitoring a contract.

8 Bibliography

Alain Abran, Marcela Maya, J-M. Desharnais, Denis St-Pierre, Adapting Function Points Analysis to Real-time Software, American Programmer, Fall, 1997.

Albrecht, A.J. (1979), Measuring Application Development Productivity, Proceedings of Joint Share Guide and IBM Application Development Symposium, October, 1997, pp. 83-92.

Desharnais, J.-M., Statistical Analysis on the Productivity of Data Processing with Development Projects using the Function Point Technique. Université du Québec à Montréal. 1988.

Desharnais Jean-Marc, St-Pierre Denis, Maya Marcela, Abran Alain, Bourque Pierre, Full Function Points: Counting Practices Manual, Technical Report 1997-04 UQAM and SELAM, Montreal, September, 1997.

Desharnais J.-M., St-Pierre D., Abran A., Gardner B., Definition of When Requirements Should be Used to Count Function Points in a Project, The Voice, International Function Point Users Group, July, 1996.

Desharnais J.-M., Validation Process for Industry Benchmarking Data, Invited Paper, Conference on Software Maintenance, IEEE, Montreal, September, 1993, pp. 371-372.

Desharnais J.-M., Morris P., Post Measurement Validation Procedure for Function Point Counts, Position Paper Forum on Software Engineering Standards Issues, October, 1996.

Galea, S. (1995), The Boeing Company: 3D Function Point Extensions, V2.0, Release 1.0, Boeing Information and Support Services, Research and Technology Software Engineering, June, 1995.

IEEE, (1990). IEEE Standard Computer Dictionary: A compilation of IEEE Standard Computer Glossaries, IEEE Std 610-1990, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1990.

IFPUG (1994). Function Point Counting Practices Manual, Release 4.0, International Function Point Users Group - IFPUG, Westerville, Ohio, 1994.

Illingworth, V. (1991) (editor), Dictionary of Computing, Oxford University Press, 3rd edition, 1991, 510 pages.

ISO/IEC/JTC1/SC7 Standard 14143 – Information Technology - Software Measurement – Functional Size Measurement – Part 1 : Definition of Concepts.

Jacquet, J.-P. and Abran, A. (1997), ‘From Software Metrics to Software Measurement Methods: A Process Model’, presented at the Third International Symposium and Forum on Software Engineering Standards, ISESS ‘97, Walnut Creek (CA), 1997.

Maya M., Abran A., Oligny S., St-Pierre D., Desharnais J.-M., Measuring the functional size of real-time software, ESCOM-ENCRESS 98, Rome, May 1998, 10 p.

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

St-Pierre, D., Maya, M., Abran, A. and Desharnais, J.-M. (1997a), Full Function Points: Function Points Extension for Real-Time Software - Concepts and Definitions, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Technical Report 1997-03, March, 1997, 18 pages.

St-Pierre, D., Maya, M., Abran, A. Desharnais, J.-M. and Bourque, P. (1997b), Full Function Points: Function Points Extension for Real-Time Software - Concepts, Definitions and Procedures, Software Engineering Management Research Laboratory, Université du Québec à Montréal, Technical Report 1997-04, September, 1997, 43 pages.

St-Pierre, D., Abran, A., Araki, M. and Desharnais, J.-M. (1997c), Adapting Function Points to Real-Time Software, IFPUG 1997 Fall Conference, Scottsdale, AZ, September 15-19, 1997.

Whitmire, S. A. (1992), ‘3-D Function Points: Scientific and Real-Time Extensions to Function Points’, Proceedings of the 1992 Pacific Northwest Software Quality Conference, June 1, 1992.

Authors

Jean-Marc Desharnais (Master degree in Administration, Master in Computer Management, CFPS) is a specialist in software engineering measurement. He is one of the co-authors of the FFP technique. He has carried out a number of software engineering researches projects covering assessment, budgeting and productivity evaluation. Mr. Desharnais has also evaluated productivity levels in several organisations and set up quantification programs to include the assessment, productivity, quality and budgeting of software maintenance. Mr. Desharnais holds Master's degrees in Computer Management and Public Administration. He is the Executive Director of the Software Engineering Laboratory in Applied Metrics (SELAM). A Certified Function Points Specialist since 1993, he has participated in several committees of the International Function Point Users Group (IFPUG) in the last ten years. He has been Vice-President of CIM since 1989.

Pam Morris

Ms Pam Morris (B.Sc., Dip. Ed., Grad. Dip. Computing, CFPS), is the Director of Consulting and Training for TOTAL METRICS Pty. Ltd Australia. She has extensive experience in the software development field, specialising in software process improvement and software metrics since 1989. She has consulted to a wide range of organisations both in Australia, New Zealand and the United Kingdom.

Ms Morris is a founding member of the Australian Software Metrics Association (ASMA), holding a position on the Executive Board and the Function Point Counting and Benchmarking Database Special Interest Groups. Ms Morris is the international project editor of the ISO Standard 14143 for Functional Size Measurement and is convenor of WG12 (the ISO/IEC standards group responsible for the development of functional size measurement standards). She plays an active role internationally in the development of the FPA technique and represents the ASMA on the International Function Point User Group (IFPUG) Counting Practices Committee. She has combined her consulting and tertiary teaching experiences to develop and present numerous Software Measurement and FPA training courses to over 200 organisations and 900 attendees in Australia and New Zealand since 1991.