The Strengths and Weaknesses of Functional Measure as a Good Negotiation Tool for New Development Projects

Jean-Marc Desharnais, Peter Westwood and Tony Rollo

Abstract

The user/contractor negotiations when constructing a software application are not easy. Many factors need to be considered (cost, product deliver, technical issues, competition, risk, etc.). In this article our focus is on the delivery of the product through the different phases of a project, using functional measure. To follow a product though different phases, a measure that can cope with preliminary requirements must be used. So, functional measure is preferable to lines of code (1). But lines of code are not totally eliminated. Some have used FPA (Function Point Analysis) to predict lines of code (1,2) or to predict FPA from lines of code (backfire) (3). However there has been very little study, to our knowledge, that analyses the performance of FPA as a predictor, from the user requirements through to the implementation of the software (2).

The following article relates our experiences measuring, at different project phases (early, intermediary and final), two software development projects using different methodologies for counting and validating function points. We found that it was necessary to consider different types of users when counting software, not only the end user. What one user could see as technical could be seen as a functional requirement by another. This challenges the common view about what functional measure is measuring. In both projects a report generator was seen as the solution for the many reports identified in the requirements. This resulted in a lower number of function points. Other less obvious infrastructure processes were constructed to improve the quality of the product. We were not able to do the count of those processes based on FPA actual rules. Those processes are considered as technical.

The final result for one project had no more than 6% difference in the number of function points between early, intermediary and final report. What was surprising for the supplier was the fact that the number of function points went down. There were many disagreements between the client and the supplier about the extension of the requirements caused by the construction of infrastructure processes not considered by FPA.

The intermediary result of the second project showed minus 16% difference between the requirements and the second count when considering only the number of function points. For this particular project, it was decided to look more closely at the infrastructure processes considering different types of users. We used a measurement methodology that takes in account the different type of users (COSMIC-FFP[1]). As a result, at the intermediary stage, the difference between the requirements count and the second count was 2%. Our conclusion is that, in order to control the cost of a software development project, the measurement methodology must consider the different types of users

1. Main Steps for the Measurement

To improve user/developer negotiations, the Government of Canada suggested Function Point Analysis (FPA) for new development projects. In 1998, we were asked to measure, at the preliminary requirements phase, the size of two different project products, before the suppliers were selected. A second measure was done later for the two projects after the

[1] COSMIC-FFP Measurement Manual, Field trials version 2.0, October 1999. The document is available at s: www.lrgl.uqam.ca.

suppliers were chosen and had documented their own solutions, based on the requirements. Finally, one of the two projects was measured when finished. The final measurement phase for the second project is due Fall 2001.

The first step of the measure was at the preliminary **requirements phase**. The information was at a high level, from the user point of view. For example, for one project, the security was described in 12 statements like this one:

- "The solution must provide the facilities to be able to control the functions that any user can access".

For the audit there was only 10 statements like this one:

- "The solution must record the user, date and time, location information, and specific transactions attempted for all activities on the system".

The requirements were clearly stated form the point of view of different users. There was no reference about the way to implement a specific function. This respects "l'esprit et la lettre" of the functional measure. The documents, with all the preliminary requirements, were part of the contracts between the contractor (for a public bid) and the client (a specific Governmental Agency).

The result of the measure was an estimate of the project product size based mainly on a hypothesis about what the users really wanted at the end of the project. In order to measure the product size a specialist in software measure was helped by the client2 to interpret the requirements.

The next step was to measure the contractor's functional specifications. At this phase the architecture was well described along with some technical solutions. There was a complete entity-relationship model available, together with descriptions of the different functionalities based on the requirements describing the manipulation of the data within different processes (read, write, entry and exit).

The result of this measure was a more precise estimate of the size based on how the architects and analysts saw the requirements. This phase helped to determine if the specialist in measurement, who was helped by the client, had the same understanding of the requirements as the developer.. This helped to clarify the requirements and their interpretation. The requirements were an important reference through the development as they were the basis of the contractual agreement.

The final step was to measure **the software as delivered** (or almost delivered) when the code was tested and the user has at least almost finished the system test. If a system is delivered on time within the budget and within 20% of the predicted size, most will consider it to be a success. (It is also necessary to consider the quality of the deliverables - function points do not measure this.)

2. Methodologies Used

Two methodologies were used: Function Point Analysis and COSMIC-FFP. The following explains the different steps of their use, and why and how a second measure was used.

2.1. Methodologies

We used Function Point Analysis version 4.1 for our count for the two projects. We used COSMIC-FFP for a partial count in one of the projects.

The International Function Point User Group (IFPUG) released version 4.1 in January 1999 (7). Interpretations of the rules are based on industry rules jointly used by Australians

[2] The client is a Government Manager who knows the requirements for the software very well.

and Canadians in the past few years (5) incorporating some recent interpretation from the IFPUG Counting Practice Committee (e.g.: how to differentiate a query from an output).

For one project we also used COSMIC-FFP methodology (6) developed by the University of Quebec in Montreal (UQAM) and industry partners. You can find the Guide explaining this methodology at www.lrgl.uqam.ca/.

At the end of each count with FPA, we used the validation process from Desharnais and Morris to ensure the quality of the count. This method is described in (8) and (9).

For the counting, we followed the different steps, which are a part of the IFPUG Counting Practice Manual (7), except the ones in bold referring to the adjustment factor. Scientific papers (12) already suggested the limits of the adjustment factors. For this reason we did not use them.

- Purpose of the count
- Scope and Boundary
- Data functions count
- Transactional functions count
- FPA unadjusted
- **Adjustment factor**
- **FPA adjusted**

The boundary of the software is linked to the purpose of the measure[3]. In these cases the purpose was to measure what need to be developed by the contractor, excluding the packages used by the contractor, to implement the solution. This is an important consideration. For example, if the contractor used a report generator already on the market, instead of developing it, there are no function points to measure except the functionality developed to adapt the package. The package is then outside the boundary. We also measured what was developed to convert the data from one application (the old one) to the new application. This corresponds to the definition of measurement of a development project with FPA (IFPUG 4.1, Chapter 4).

2.2. Why use COSMIC-FFP

Considering that we need to measure all of the software, not just a part of it, FPA rules are not complete (10) when identifying only one level of software to be measured. The FPA rules do not consider more than one layer of software and clearly reject all other layers seen as "technical" software because it is part of the solution, not of the product[4]. This paper challenges this view considering that a solution for one user is a product for another. Within COSMIC-FFP version 2, it is possible to take into account different parts of the software using the concept of "layer"(6). Our definition of layer is based on the concept described in version 2 of COSMIC-FFP[5].

There are 10 principles to determine if there is more than one layer within a software product. The main principle is that from one layer to another, the data is seen from a different perspective. An example of two different layers is the report generator versus the reports. The developer using a report generator is interested in the parameters used to construct a report, while the end user is interested in the different attributes showed in the report. Another example is the functionality of the language used to develop a software or macro (in this perspective MS Excel is a language) versus the functionalities delivered by the software to the end user. By analogy, to a manager, the accounting report provides information on

[3] IFPUG Counting Practice Manual uses the word "count" while ISO 14143 uses the word measure instead.

[4] "Measure software development and maintenance independently of technology used for implementation" IFPUG 4.1, page 3-2, January 1999.

[5] (6) pages 23-24.

expenditures against a budget, but to an archivist the same physical document is a piece of paper to place on the shelve in the right place.

One of the reasons why FPA does not deal with infrastructure processes is because the FPA rules do not apply within infrastructure processes:
• FPA makes a distinction between different types of transactional functions (INPUT, OUTPUT and INQUIRY). An infrastructure process often deals with more than one transactional function, which mean that an INPUT could be just a part of the elementary process. The definition of the INPUT is then in contradiction with the intent of measuring a transactional function and the definition of an elementary process[6]. • An infrastructure transactional function can be very simple or extremely complex. The actual IFPUG rules do not provide more than 7 points for a transactional function. This leads to some problems with infrastructure software and real-time software (10) • Adding the data functions to the transactional functions dilutes the relationship between what is perceived as being done for the user and the size.

From this perspective, we found for both projects that different pieces of the software product were developed not only at the end user level but also for different types of users. Within an organization there is more then one type of user. Some users are technical relative to other users. For example, if an organization has the need to maintain a security system, this is definitively not a task for the end user, but another type of (technical) user. Also, the maintenance of the parameters for the report generator is relatively technical, even if it is functionality delivered by the contractor. The audit trail is also relatively technical for the end user. We classified each piece of software by layers (10).

Those considerations could be also very important for the cost per function point. At a lower level of layer (e.g.: constructing a report generator) we suspect that the cost per function point is higher. At a higher level or end-user level (e.g.: producing the report) the expected cost per function point is lower. As long as the number of FPs for the lower level is relatively low, the margin of error for the price is low enough to permit an average cost for the entire project. However contractors would probably like to see more refinements to the price structure. Those considerations are linked to the concept of functional reuse and the role of infrastructure to improve the value of software. The concept was already presented at the ESCOM-SCOPE conference April 2000 in Munich (11).

3. Results and Comments on Results

3.1. Project 1

Project 1	FPA from Requirements	Diff	FPA Contractor first solution	Diff	FPA Final count
With no Report Generator	**1037**	3%	**1073**	2%	1060
With Report Generator	951	4%	987	2%	**974**

The difference is less than 5% in all cases. Every measure was produced from a non-adjusted base. The measures in **bold** were the ones presented to management at different phases of the project. We did not count the Report Generator as a solution at the Requirements phase, neither did the contractor, which explains the differences. As already

[6] An *elementary process* is the smallest unit of activity that is meaningful to the user(s).

mentioned, the final measure (974) is lower than the two earlier measures (1037 and 1073) mainly because of new agreements between the contractor and the user to use a report generator tool instead of programming directly the 30 different reports. The number of function points for the reports at the requirement phase was 150 FPA. The number of points to adapt the report generator was 64 points even though the report generator (an infrastructure project) gave more opportunities for additional functionality for the end user. In fact, instead of having just 30 reports, the possibilities are for many more. This shows that just adding the Report Generator functionality (or any infrastructure functionality) could be misleading and lead to misunderstanding between the user and the contractor. The second project shows more clearly this problem.

3.2. Project 2

Project 2	From Requirements	Diff	Contractor's first solution	Diff	Final count
First count with contractor's first solution	1189	-16%	993	NA	NA
Second count (including COSMIC FFP)	1189	2%	1210	NA	**NA**
Second count including additional user functionality	**1468**	2%	1489	NA	**NA**

The final measure has not been done yet. For reasons that will be explained below, two intermediary measures were performed ("first count with contractor's first solution", "second count".)

From the preliminary requirements we measured 1189 function points. It was already clear the software required was multi-layered. We performed the Contractor's first solution measure when the contractor's solution was specified. When we finished the Contractor's first solution measure of the original preliminary requirements with 993 points (counting only the end-user level or client layer) there were a lot of discussions between the client and the contractor. The contractor believed that his solution was giving more functionality and quality than the one strictly counted at the end-user level. . The client conceded that that this was probably so, but he needed some objective method of measuring the additional functionality. So it was decided to measure not only the functionality provided to the end user, but also the functionality provided to different (technical) users. COSMIC-FFP was used to measure the layered software defined by the contractor in a second measure. It resulted in 217 points being added, without any extra user defined functionality being added.

In addition there was clearly some additional user functionality that had been requested by the client. This was measured at 279 points.

Those who are interested in the equivalence between an FPA measure and a COSMIC-FFP measure should refer to an article already published by Morris and Desharnais (10). At the client level for an MIS system the result is roughly the same. For a real-time system and an infrastructure project with complex processes there is a difference.

Without measuring the impact of the layered software, and ignoring the added user functionality, there was a 16% difference between the function point measure at the preliminary requirements stage and that at the contractor's first solution stage. By taking into account the impact of the layered software, and accounting for the added user functionality the 16% difference reduced to a 2% difference. Once that 2% difference conclusion had been made then the client and the contractor could conclude their agreement.

3.3. Comparison between the requirements and the components at different phases

The exercise showed that the function point method is generally good at predicting the size of new software development at the end-user level. The margin of error between the different measures from Requirements to the Functional Analysis was between 2% and 16%. Even if the margin suggested by the literature is 20% the contractor and the client did not consider that precise enough.

3.4. Considerations regarding the experience of the users, the contractor and the counter

The following conditions, combined together, helped the success of the count:

- The client had a good idea of the system he wanted.
- The client and the contractor were experienced in the software domain.
- The requirements were relatively clear to understand at the functional level.
- The contractor agreed to follow the process and give the information to the expert in functional measure. The expert in functional measure had then the opportunity to talk directly to people who knew the requirements and the client needs.
- The expert in functional measure counter is experienced at measuring FPA.
- The expert in functional measures counting was the same from the beginning of the process.
- The expert in functional measures counting had many years of experience in data processing.
- The client and the contractor respected the functional measure counting expertise of the expert.

3.5. Measuring Effort

Considering a development between 2 to 5 person years, the 5 person weeks taken for the measurement of each project, is relatively low. Base on our experience, measuring 250 points per day is the average in a normal measurement environment (this assumes: the availability of quality documentation, and the availability of the subject matter expert). When the task must be auditable and needs multiple verifications and validations before delivering the results, the number of function points per day will be lower. The approval of the contractor and the client at each step takes additional time, and needs to be considered.

4. Conclusion

From this experience, our conclusion about the strengths and weaknesses of using only FPA are the following.

Strengths:

- It is possible to compare the requirements with the results at different steps of the exercise.
- FPA gives good results globally when comparing the number of function points by phase.
- The counting is relatively reliable when using a validation method.
- FPA gives to the user a way to know if what will be delivered is what they asked for in the requirements.
- FPA can be used as a tool to discuss the final cost of the contract.
- FPA can be used as a tool to give more insight for some requirements.
- FPA can be used as a tool to check the "scope creep" of the project during the different phases of the development.

Weaknesses

Using FPA:

- It is not possible to determine the impact of a particular change in a defined process to the global project.
- There is no direct relation between a specific number of points (e.g.: adding a new report) and the effort to realize the task. This drives to a certain misunderstanding between the developer and the FPA measurer. The measure does not take into account the extreme complexity of some processes.
- We cannot measure the quality of software when delivered. The choice of some tools could accelerate the development of the software, however slowing down drastically the use of the software. (Quality criteria need to be included in a contract, they are independent of functionality and Function Point counts.)
- It is not possible to determine the impact of one layer on another (10).
- There is no explanation on how to deal with different of levels of software such as: report generators, drivers developed for a specific user, using macros to derive reports instead of constructing different reports, constructing generic functionality usable by different parts of the software, etc.

COSMIC-FFP concept of layers is a direct answers to the last two points. COSMIC-FFP measures the data movement of the processes instead of using tables, and adding transactional functions and data functions is a direct answer to the first four points (10). The size of a functional process is not limited by a number in a table, but by the number of data movements. Taking account the different layers helps to understand quality – for example: providing a generalized report generator instead of a certain number of individually programmed reports - but it is not sufficient

5. Future Research

Some more research is needed in order to make functional measure more efficient:

- An evaluation should contain some quality measure, not just functional measure. The main quality measures that could be considered are: speed of the software considering a specific environment (equipment, network, etc.); the volume of transactions that should be managed by the system; the number of defects in the delivered software, the quality of the documentation for the user. Each criterion should be documented in the requirements.
- The functional measure should be more related to the size of each process, instead of having a global count. This will give more insight into the impact of a particular change in a process.
- There is a need to improve the interpretation of the rules.

6. References

[1] Betteridge, R, “Function Points vs. Lines of Code”, System Development, p. 4-6, 1990

[2] Betteridge, R, Successful experience of using function points to estimate project costs early in the life-cycle, Information and Software Technology, Vol. 34, no 10, p. 655-658, 1992

[3] Henderson, G.S., “The application of function points to predict source lines of code for software development”, Faculty of the School and Logistics of the Air Force Institute of Technology, Air University, 190 pages, 1992.

[4] Jones, C., “Backfiring: Converting Lines of Code to Function Points”, IEEE Computer, Vol. 28, no 11, p. 87-88, November 1995.

[5] Resolving FPA Count Issues Resolution Guidelines conformant with IFPUG 4.1 – Counting Practices Manual, Copyright Total Metrics, Version 1.7, Australia

[6] COSMIC-FFP Measurement Manual, Field trials version 2.0, October 1999.
[7] IFPUG, Counting Practice Manual, Version 4.1, 1999.
[8] Morris P., Desharnais J-M, “Validation of Function Points – An Experimental Perspective”, IFPUG, 1996, pp. 167-184.
[9] Desharnais J.,M., Morris P., “Post Measurement Validation Procedure for Function Point Counts”, Position Paper Forum on Software Engineering Standards Issues, October, 1996.
[10] Morris P., Desharnais J-M, “New Methods for Measuring Function Points in Outsourcing Contracts”, IFPUG, Orlando 1998.
[11] Ho, V.T., Abran A., Oligny S., “Using COSMIC-FFP to Quantify Functional Reuse in Software Development”, ESCOM-SCOPE 2000, April 18-20, 2000.
[12] Kemerer C.F., Reliability of Function Points Measurement: A Field Experiment, MIT, December 1990.