AUTOMATION OF FUNCTION POINTS COUNTING: FEASIBILITY AND ACCURACY ?

Alain Abran and Keith Paton

Abstract

The rules governing Function Point Analysis (FPA) are described in textual format in the Counting Practices Manual by the International Function Point Users Group. These textual descriptions need to be transformed into very structured requirements if the tools to be built have some chance to produce accurate results. Ideally it should be possible to represent the FP rules into decision tables that can then be programmed and fully tested. By using a formal notation system and decision tables, FPA counting process can be expressed in 17 tasks of which 12 are algorithmic and five require human judgment. For the function points counters it captures the mechanism of counting function points as defined in the CPM; for the tool builder it provides the detail required to start building the physical design of an interactive tool for counting function points.

NOTE

This presentation at the IFPUG industrial conference presents some of the concepts documented in the full research paper « A Formal Notation for the Rules of Function Point Analysis » submitted to the ACM Transactions on Software Engineering and Methodology - ACM TOSEM.

Software Engineering Management Research lab.
DEPARTEMENT D'INFORMATIQUE
UNIVERSITE DU QUEBEC A MONTREAL
C.P. 8888 SUCCURSALE CENTRE-VILLE
MONTREAL (QUEBEC) CANADA H3C 3P8

PHONE: (1) 514 - 987-3000 (8900)
FAX: (1) 514 - 987-8477

MAILTO:abran.alain@uqam.ca
http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

1 Introduction

This introduction defines the problem to which Counting Function Points is a solution, shows why that problem is important, and explains how to recognize a solution.

Albrecht (1979) proposed that
1. each software system has a size (in function points)
2. the size of a software system (in function points) can be measured from its detailed design and constraints

Since the method was introduced by Albrecht (1979), the official version has been that defined by the International Function Point Users Group. The current definition of the method is version 4.0 of the Counting Practices Manual (IFPUG 1994); this will be denoted the CPM. The rules and examples now run to over 200 pages and a lengthy period of study is required to master them, and even a Certification procedure.

This presentation proposes that one way of solving the functional size measurement problem is to embody the rules of the CPM in a structure of decision tables

Function Point Analysis was introduced by Albrecht (1979) and is now guided by the International Function Point Users Group which has modified the FP measurement standards over the years, presenting the latest version each time in an updated version of the Counting Practices Manual (the CPM).

The method lacks a formal basis, since the object to be counted is nowhere defined. It is clear from the CPM that the object to be measured must include such things as files, records, fields, processes and that it must be known which processes read and write which fields in which records in which files.

Some software tools have been devised to help users count function points as shown in table 1, for example (Mendes et al. 1996).

Tool	*Developer*
Autopoint	Integrated Software Specialists
FP Analyst	Cayenne Software Inc.
Composer FP Report	Texas Instruments
LINC Logical and Informat Network Compiler	Unisys - Brazil
Via Recap	VIASOFT Inc.

Table 1: Tools for Counting Function Points

Although a full critical review of these tools has yet to be published, it is worth noting that
- ◊ They are proprietary; they work only for designs expressed in a particular notation defined by the tool and not open to the public.

◊ Some of the tools suggest that CPM can be completely automated.
Proprietary tools cannot be used for designs imported from some other place, notably a general CASE tool. This presentation shows that CPM can not be completely automated but that several key steps require human judgment.
In what follows a structured analysis of the counting rules is presented using the specification techniques of decision tables for the parts that require no judgment. All proposed decision tables are justified by referring to the CPM.

The presentation is designed for three classes of audience, the function points expert, the function points standard committee member and the tool builder. For the function points expert it captures the mechanism of counting function points as defined in the CPM; for the standards committee member, it provides a structured analysis of the rules highlighting duplications or gaps; for the tool builder it provides the detail required to start building the physical design of an interactive tool.

2 Function Points Counting Tasks

Since this presentation is about the methodology of FPA, the tasks in counting Function Points are as follows:

1	Identify the original system, the enhanced system and the conversion system
2	Identify the status for each file
3	Identify the status for each process
4	Decide which processes are **required by** this application
5	Decide which files are **required by** this application
6	Identify the attribute **type** for each relevant file
7	Identify the attribute **type** for each relevant process
8	Decide which relevant processes are **unique**
9	Decide which relevant files are **unique**
10	Compute the attributes **ftrs** and **dets** for each unique relevant process
11	Compute the attributes **rets** and **dets** for each unique relevant, file
12	Compute the attribute **count** for each unique, relevant process
13	Compute the attribute **count** for each unique relevant file
14	Compute the unadjusted function points ADD, CHGA, CFP, DEL for the parts added, modified, used for conversion and deleted
15	Decide on the values of the 14 attributes in the group **general characteristic** for the systems both before and after the enhancement
16	Compute the value adjustment factors VAFB, VAFA before and after the enhancement
17	Compute the attribute adjusted count for the enhancement

Table 2 : Tasks in Counting Function Points

A step starting "Decide" indicates one where the decision must be taken by the user; one starting "Compute" indicates an algorithmic activity carried out by the tool. In the detailed analysis it has been shown *why* the "decide" step can not in principle be carried out by the tool and *how* the "Compute": step can be carried out by the tool. At this stage it is sufficient to note that with five "Decide" tasks and 11 "Compute" tasks the task is clearly one that calls for a co-operation between user and machine. If the tool were not available the user would have to do the 11 "compute" tasks manually; if the user were not available the five "Decide" tasks would be impossible to carry out (in all situations, including enhancements).

The table below provides a two-way index between the 17 tasks given above and the sections of the CPM. Thus the row "identify boundary" indicates that identify boundary supports tasks 2 and 3; the column 4 indicates that task 4 is supported by the CPM sections EI definition, EO definition and EQ definition.

	Task as given in this text																
the CPM page theme	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
3 type of count	√																
4 identify boundary		√	√														
5-6 ILF identification					√	√											
5-6 EIF identification					√	√											
5-7 ILF counting rules									√		√		√				
5-7 EIF counting rules									√		√		√				
6-4 EI definition				√			√										
6-6 EI counting rules								√		√		√					
6-16 EO definition				√			√										
6-6 EO counting rules								√		√		√					
6-28 EQ definition				√			√										
6-30 EQ counting rules								√		√		√					
7-3 VAF calculation															√	√	
8-10 Enhancement Project FP calculation														√			√

Table 3 : Cross-index between the CPM and this text.

3 Decision Tables

3.1 Notation Formalism for the Software

The presentation describes a methodology of counting function points through decision tables. In the case of counting function points, the *object to be measured* must be some representation of the software system, although the CPM nowhere defines just how the software system is to be

represented for the purpose of measuring its size. The first requirement is to express the design of the software in some notation that can be understood by a computer program.
The idea that a design can be expressed as a picture in which boxes represent stored data and bubbles represent the processes that manipulate that stored data dates back at least to DeMarco (1979) and the presence of this notation in many CASE tools suggests that it is a powerful unifying force in system design. Figure 1 is an example.

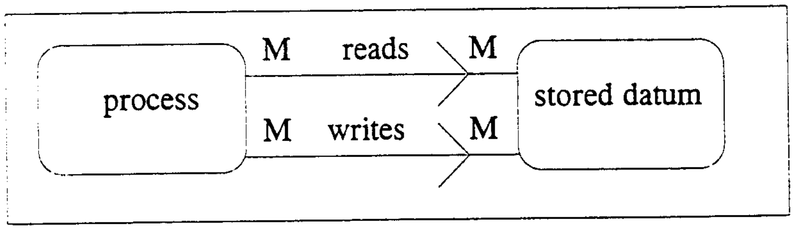


Figure 1 : Notation Formalism

In the general form, any process can have many inputs and many outputs and any stored datum can be read by many processes and written by many processes
Any *design* can thus be summarized as a quadruple *(P,S,R,W)* where

◊ *P* is a set of processes
◊ *S* is a set of stores
◊ *R* is the reads relation from *P* to *S*
◊ *W* is the writes relation from *P* to *S*

A quadruple (P,S,R,W) represents a valid design when and only when

1. every process reads at least one store *and* writes at least one store
2. every store is read by at least one process *or* written by at least one process

The description of the notation has followed DeMarco in using the terms "process" and "store"; to be consistent with the CPM, we replace the word "store" by the word "file" from now on.

3.2 Defining the Boundary

The CPM explains that the first step in counting function points is to define a boundary between the items (processes, stores) to be counted and the items not to be counted. It explains this in text, thus

> Use the system external specifications or get a system flow chart and draw a boundary round it to highlight which parts are internal and which parts are external to the application.

Many different boundaries can be drawn but from the point of view of function points, the only thing that matters is which processes and stores are inside and which are outside the boundary. In the main presentation, we follow the language of the CPM and talk about processes and files being inside and outside a boundary.

The processes in P are placed inside or outside B by following this rule:

If p is MODIFIED or ADDED or DELETED or USED FOR CONVERSION then p is inside B otherwise it is outside B.

Likewise, the files in F are placed inside or outside B by following this rule

If f is MODIFIED or ADDED or DELETED or USED FOR CONVERSION
then f is inside B; otherwise it is outside B.

The CPM defines the function point count as the sum of the function point count over the items added, modified or deleted or used for conversion Since the interior of B has been defined to contain exactly the items added, modified, deleted or used for conversion, it follows that the function point count is the sum of the function point count over the interior of B.

3.3 Classifying the Files

Given the boundary B, the files are classified according to the table below:

f is inside B	f is outside B	
	f is read by some p inside B *or* *f is written by some p inside B*	*f is read by no p inside B* *and* *f is written by no p inside B*
f is an internal logical file ILF	f is an external interface file EIF	f is an ignored file IGF

Table 4 : Different types of store

3.4 Classifying the Processes

3.4.1 Classifying the Processes by the CPM

The CPM provides rules for recognizing three types of process,

- ◊ the external input,
- ◊ the external output
- ◊ the external inquiry.

It is implicit that the rules should allow the user to classify a given process, that is assign to it at most one of these three types. It is clear from the rules of the CPM that the classification has much to do with the answers to these questions

- ◊ does the process read outside B?
- ◊ does the process write outside B?
- ◊ does the process read inside B?
- ◊ does the process write inside B?

There are therefore 16 cases to consider. Each cell in table 5 shows an example of each case and quotes a rule number from the CPM to show the categories into which this case cannot fall. The notation 6.31.7 means chapter 6 of the CPM, page 31, bullet 7 and so on.

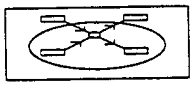
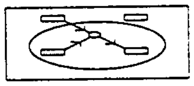
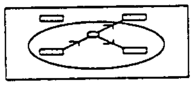
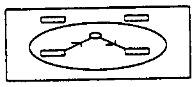
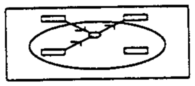
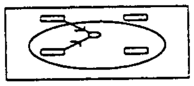
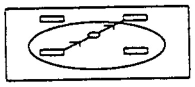
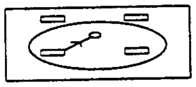
		reads outside B?	yes	yes	no	no
		writes outside B?	yes	no	yes	no
reads inside B?	writes inside B?					
yes	yes		 not an EQ (6.31.7)	 not an EO (6.19.1) not an EQ (6.31.7)	 not an EI (6.7.1) not an EQ (6.31.1)	 not an EI (6.7.1) not an EO (6.19.1) not an EQ (6.31.1)
yes	no		 not an EI (6.7.2)	 not an EI (6.7.2) not an EO (6.19.1) not an EQ (6.31.7)	 not an EI (6.7.1) not an EQ (6.31.1)	 not an EI (6.7.1) not an EO (6.19.1) not an EQ (6.31.1)

Table 5 : Detailed Classification of Processes (8 of 16 possibilities)

Table 6 summarizes the situation so far.

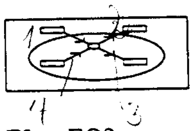
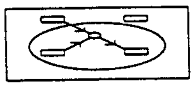
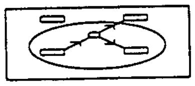
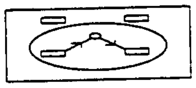
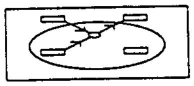
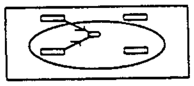
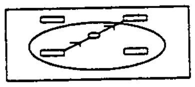
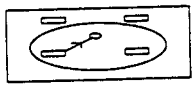
		reads outside B?	yes	yes	no	no
		writes outside B?	yes	no	yes	no
reads inside B?	writes inside B?					
yes	yes		 EI or EO?	 EI?	 EO?	 NIL
yes	no		 EO or EQ?	 NIL	 EO?	 NIL

Table 6 : Intermediate Classification of Processes (8 of 16)

Of the 8 possibilities,

◊ three are definitely NIL

◊ three have one possibility remaining (four EO and two EI)
◊ one is ambiguous (EI or EO)
◊ one is ambiguous (EO or EQ).

The NIL cases in the right hand column can be ignored. Table 7 shows 6 of the remaining 6 cases together with a suggested classification.

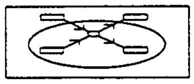
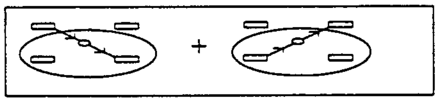
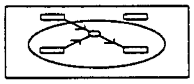
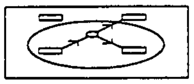
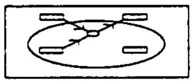
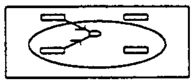
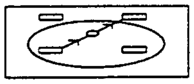
 EI or EO? This is a double process expressed as the sum of an EI and an EO, thus. 	 EI? This is an EI, an update of existing data, as in new balance = old balance + deposit	 EO? This is an EO; it writes to an ILF, an activity not prohibited by the rules
 EO or EQ? This is an EQ (It could also be an EO; it reads from outside and reads from inside, neither being prohibited by the rules	 NIL	 EO? This is an EO

Table 7 : Further Classification of Processes (6 of 12)

This yields the classification of table 8:

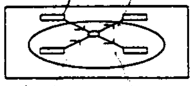
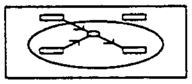
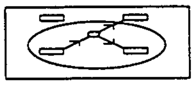
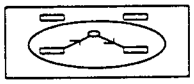
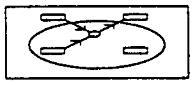
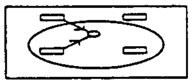
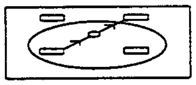
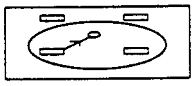
		reads outside B?	yes	yes	no	no
		writes outside B?	yes	no	yes	no
reads inside B?	**writes inside B?**					
yes	yes		 DOUBLE	 INPUT	 OUTPUT	 NIL
yes	no		 INQUIRY	 NIL	 OUTPUT	 NIL

Table 8 Final Classification by the CPM (8 of 16)

Although all cases have now been classified, extra logic not found in the CPM has been required. In particular, in three places we have labeled a case as a NIL because we believe it contravenes the intention but not the letter of the CPM. This suggests that a simpler classification is needed

3.4.2 Classifying the Processes -a simplified approach

A simpler classification of the processes can be based on eight rules:

1 A process must read or write outside B
2 A process must read or write inside B
3 A process must read somewhere
4 A process must write somewhere

These first four rules reject nine of the 16 candidates. These lead to seven surviving cases. Now consider the next four rules which lead to the final classification of table 7:

5 A surviving process that reads outside B but does not write outside B is an INPUT
6 A surviving process that writes outside B but does not read outside B is an OUTPUT
7 A surviving process that reads and writes outside B is an INQUIRY if it reads inside B and an INPUT if it writes inside B
8 A surviving process that carries on all four activities is a DOUBLE PROCESS and must be decomposed as an input plus an output.

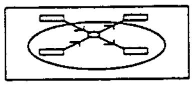
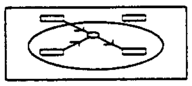
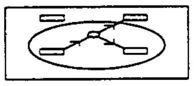
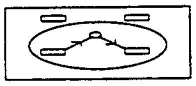
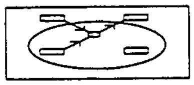
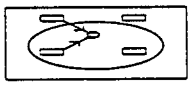
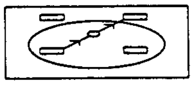
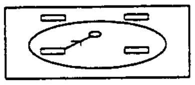
		reads outside B?	yes	yes	no	no
		writes outside B?	yes	no	yes	no
reads inside B?	**writes inside B?**					
yes	yes		 DOUBLE (8)	 INPUT (5)	 OUTPUT (6)	 NIL (1)
yes	no		 INQUIRY (7)	 NIL (4)	 OUTPUT (6)	 NIL (1)

Table 9 : Final Classification of Processes using Simplified Rules (8 of 16)

Thus 8 rules cover all cases. Rules 1-6 are very straightforward. This leaves just rule 7 which says that an inquiry must not write an ILF and rule 8 on the double process.

Table 6 shows the classifications by the CPM directly, table 9 that by the rules 1-8 of the simplified scheme defined by the eight rules above. The two classifications are identical but the unified scheme follows fewer, simpler rules that allow the reader to see the reasons behind most of the rules. It thus lays extra stress on the two exceptions, namely the double process and the rule that an inquiry is not allowed to write to an ILF.

3.5 Comments

This presentation does not address the identification procedures of unique files and processes, nor the steps for the assignment of weights. This is however addressed in Abran & Paton. (1997).

4 Observations

Section 2 showed how to break down the activity of counting function points into 17 tasks, each labeled as algorithmic or requiring judgment. Section 3 also traced the 17 tasks back to the CPM. Section 3 transformed the relevant parts of the CPM into processing logic suitable for the parts of a tool to count function points.

It has been illustrated that the activity of counting Function Points, as currently documented in the CPM, can be regarded as 17 tasks, six of which require human judgment and 11 of which can be automated. It is therefore concluded that no automatic tool for counting Function Points is at present possible with the current set of rules and documented standards. However there is scope for an interactive tool that will carry out book-keeping and arithmetic while allowing the user to exercise the human judgment required.

The analysis of this presentation suggests the following four applications:

1. Reviewing existing tools based on CPM 4.0
2. Building a validation protocol to facilitate the verification of tools that claim to have automated the Function Points counting process.
3. Extending the rules to include uncodified practice used by expert counters.
4. Building a tool with which a standards body can investigate the effect of changing the rules

First, this presentation breaks FPA into 17 tasks of which 12 are algorithmic and five are interactive. This ratio can guide us when we review existing tools for FPA; it prompts us to wonder why some tools can claim to do the whole process automatically.

Second, a validation protocol could be built to facilitate the verification of tools that claim to have automated the Function Points counting process.

Third, expert counters believe that counting Function Points depends not only on the rules of the CPM but also on what we may call "uncodified practice", pieces of expertise developed over the years and not written down anywhere.

Fourth, we may distinguish between the *rule-users*, those who count function points to measure the size of a software system and the *rule-makers*, those who devise the rules on which FPA is based. The rule-users count with the most recent CPM, supplemented by some version of the "uncodified practice" referred to above.

5 References

- Abran, A., Paton, K. (1997), *A Formal Notation for the Rules of Function Point Analysis,* submitted to the ACM Transactions on Software Engineering and Methodology - ACM TOSEM.
- Abran, A. (1994) *Analyse du Processus de Mesure des Points de Fonction, (*Ph D thesis) École Polytechnique de Montréal.
- Albrecht A.J. (1979) *Measuring Application Development Productivity*, Proceedings IBM Applications Development Symposium, Monterey, CA.
- DeMarco T. (1979) *Structured System Specification* Yourdon Press, New York.
- IFPUG (1994) *Counting Practices Manual 4.0*, International Function Points Users Group., Columbus, Ohio.
- ◊McMenamin S.J. and Palmer J.F. (1981) *Essential System Design*, Yourdon Press, New York
- Mendes, O, Abran, A., Bourque, P., *Function Tools Market Survey - 1996*, http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html, 1996
- Rudolph E.E. (1989) *Precision of Function Point Counts*, IFPUG Spring Conference, San Diego, CA.