IFPUG function points or COSMIC function points?

Gianfranco Lanza

Abstract

The choice of the more suitable metric to obtain the functional measure of a software application is not an easy task as it could appear.
Nowadays a software application is often composed by many modules, each of them with its own characteristics, its own programming languages and so on.
In some cases one metric could appear more suitable to measure one particular piece of software than another, so in the process of measuring a whole application it's possible to apply different metrics: obviously what we can't do is to add one measure to the other (we can't add inch with cm!).
In our organization we are starting to use COSMIC function points to measure part of the application and to use IFPUG function points to measure other parts.
One of the possible problem is: "when the management ask us only one measure of the functional dimension of the application what can we do?"
It's possible to obtain only one measure using the conversion factors from COSMIC to IFPUG but it would be better to maintain two distinct measures.
We are starting to compare the two metrics in different environments: in some cases there is a significant difference in the functional measure.
Using a metric is mainly a matter of culture. It's important to know what a metric can give us and what not, but above all, what is our goal.
If, for example, our goal is to obtain a prevision of effort and costs we have to know the productivity of each metric in the different environments: so we have to collect data!
This paper illustrates how we can use both the metrics in our application.

1. Introduction

Every day we have to do with metrics to satisfy our needs. For example how many times a day do we look at our watch to know what time is it?
We choose the right metric to satisfy our goal. If the goal is to know what time is it, we use the metric "time". If the goal is to weigh some fruits we use the metric that give us the weigh.
Sometimes the metrics give us measures that represent the same concept but in different measure unit (i.e. inch and cm for length). The metrics were born to measure and we have to measure to know. Every metric gives us an information, the functional metrics give us information about the dimension of our software products. In every moment of our life we have to use many metrics; in the world of the applications sizing is it sufficient to use one single metric? Can one single metric be sufficient to answer all our questions, all our needs? Today, the architecture of our software applications is complex and full of software components, each one with its own characteristics, programming languages and so on. It's absolutely normal that we can use different metrics according to the nature of the software component that we have to measure.

2. User's point of view

The dimensional measure of a software application is influenced by the choose of the user; the user's point of view is very important to determine our measure. The choose of the user

depends also by the goal that we have. If we have to measure our software application because our client wants to know the functional dimension of the software he'll buy, the user is our client. If we have to measure the software application to have information about the effort to develop the application, the user could be different . In a client server architecture we should like to know the functional dimension of the client component and the functional dimension of the server component, as they were two different applications, we could have two different borders, simply because we could have different data about productivity for the client component and for the server component. If our application is built with business services, we could have to measure separately these business services for knowing the effort of developing them, if some of these business services are already done, we would like to know how much reuse we will have. In an application we can have different user's point of view that give us different measures; the important thing is that we have not to mix them!

3. Elementary Process

To determine the functional dimension of an application is necessary to identify the elementary processes, independently by the metric we use. The elementary process is the smallest unit of activity, which satisfies all of the following:

1. is meaningful to the user;
2. constitutes a complete transaction
3. is self contained
4. leaves the business of the application being counted in a consistent state.[1]

The elementary process depends by the user's point of view and as at the end of the process the system has to be in a consistent state, it can't be divided into subprocesses. In an application, if we change the user's point of view, we can have different elementary processes.

4. Batch Processes

In some cases a batch process is a stream of processes, each of them linked together to constitute a unique elementary process, whichever user's point of view we have. If something of wrong happens during the execution of one process, the whole batch process has to be restored. In the figure 1 is represented the schema of this type of batch

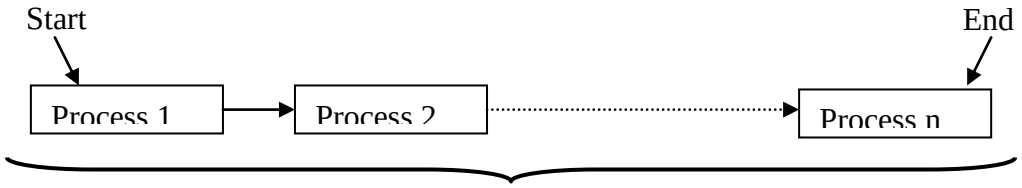

Figure 1 – Bath process

4.1. Batch processes using IFPUG function points

If we use the IFPUG function points, we have to consider one unique elementary process for one batch process (in fact we can't divide the elementary process in different elementary

processes because the system wouldn't be in a consistent state at the end of each process of the stream; this independently by the user's point of view).
Then we have to discover the primary intent of this process, if it is to maintain some logical files or change the system behavior, or if it is to expose data outside the boundary of our application. In the first case we have an EI, otherwise an EO or EQ.
To establish the complexity of our IFPUG functions we have to identify the FTR (File Type Referenced) and the DET (Data Element Type); we could have from 3 to 7 IFPUG function points, according to the complexity and the type of the transactional function.
We have to add this function points to the number of the data functions (ILF or EIF) function points but very often these batch processes are inside the boundary of a Client-Server or WEB application, the data functions that they reference are the same counted in the Client-Server or Web application; so being inside the same boundary we can't count twice our data functions!
We have also to observe that in an application of great dimension we can have a great number (more than 50) of these batch processes. At the end of the process, in any case, we have a number of function points for each of this process that are very similar, probably with a difference of three, four function points. Is this measure useful to us? Is it sufficiently significant to apply, for example, a model to obtain the cost and the effort of the batch processes developing?
Probably the answer is no!

4.2. Batch processes using COSMIC function points

If we use COSMIC function points we have also to identify the elementary process by the user's point of view. In this case the elementary process is the same of that identified in the IFPUG function points process. In COSMIC function points there aren't data functions but there are objects of interest and data movements, each data movement has one object of interest.[2] There are four type of data movements: Entry, Exit, Read and Write. In a batch process with a stream of processes linked together to constitute a single elementary process there are a multitude of data movements. In the figure 2 is illustrated an example.The weight of each data movement is one single function point. There is no limit to the number of data movements in a elementary process (at least one entry and one of the others), so we can have functional processes with a strong difference in function points, according to the number of data movements. Is this measure useful to us? Is it sufficiently significant to apply, for example, a model to obtain the cost and the effort of the batch process developing?
Probably the answer is yes!

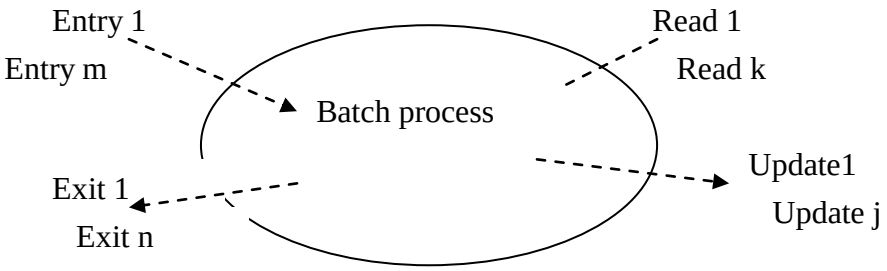

Figure 2 – Data Movements

4.3. A real case

In the Table 1 is represented the result in IFPUG function points and in Table 2 of COSMIC function points of the functional measure of the batch processes of a business application. The number of IFPUG function points of each batch process is obtained by the sum of the relative transactional functions with the portion of the Data functions weight (the entire Data Function weight is of 228 FP / 32 Batch Processes = 7,125 FP).
The number of COSMIC Function Point is obtained by the number of data movements of each batch process.

Table 1– The IFPUG function points counting result

Name	Function	FP
Print and Update BATCH		
CTTRIS000-002	EOH	7
CTTRIS000-003	EOH	7
CTTRIS000-008	EOH	7
CTTRIS500-001	EOH	7
CTTRIS500-003	EOA	5
CTTRIS600-001	EOH	7
CTTRTP000-002	EOH	7
CTTRTP000-003	EOH	7
CTTRTP000-005	EOH	7
CTTRTP000-008	EOH	7
CTTRTP000-009	EOH	7
CTTRTP000-010	EOH	7
CTTRTP000-011	EOH	7
CTTRTP000-012	EOH	7
CTTRTP000-013	EOH	7
CTTRTP000-014	EOH	7
CTTRTR500-001	EOH	7
CTTRTR600-001	EOH	7
CTTRTR600-002	EOH	7
CTTRTR600-004	EOH	7
CTTRTR800-001	EOH	7
CTTRTR800-002	EOH	7
CTTRTS000-001	EOH	7
CTTRTS700-001	EOH	7
CTTRTS700-002	EOH	7
CTTRTS700-008	EOH	7
CTTRTU000-001	EOH	7
CTTRTU000-002	EOH	7
CTTRTU000-003	EOH	7
CTTRTU000-008	EOH	7
CTTRTS700-001	EOH	7
CTTRTS700-002	EOH	7
Total =		222

Table 2 – The COSMIC function points counting result

Name	Entry	Exit	Read	Write	Total
Print and Update BATCH					
CTTRIS000-002	1	5	5	10	21
CTTRIS000-003	1	19	5	10	35
CTTRIS000-008	1	8	6	3	18
CTTRIS500-001	1	8	6	10	25
CTTRIS500-003	1	4	2	1	8
CTTRIS600-001	1	9	11	14	35
CTTRTP000-002	1	27	15	7	50
CTTRTP000-003	1	29	16	8	54
CTTRTP000-005	1	5	4	1	11
CTTRTP000-008	1	43	16	8	68
CTTRTP000-009	1	35	15	7	58
CTTRTP000-010	1	26	15	7	49
CTTRTP000-011	1	22	14	6	43
CTTRTP000-012	1	27	14	6	48
CTTRTP000-013	1	28	14	6	49
CTTRTP000-014	1	27	15	7	50
CTTRTR500-001	1	15	9	9	34
CTTRTR600-001	1	17	9	5	32
CTTRTR600-002	1	11	9	5	26
CTTRTR600-004	1	9	9	5	24
CTTRTR800-001	1	15	12	1	29
CTTRTR800-002	1	22	13	1	37
CTTRTS000-001	1	3	4	2	10
CTTRTS700-001	1	1	3	6	11
CTTRTS700-002	1	1	3	4	9
CTTRTS700-008	1	4	3	1	9
CTTRTU000-001	1	8	12	5	26
CTTRTU000-002	1	8	12	5	26
CTTRTU000-003	1	16	5	3	25
CTTRTU000-008	1	3	3	1	8
CTTRTS700-001	1	1	3	6	11
CTTRTS700-002	1	0	3	4	8
				Total =	**947**

As we can see there is a strong difference between the two measures, the difference is represented in the graphics in Figure 3 (IFPUG FP) and Figure 4 (COSMIC FP).
There is a strong difference as total sum of the batch processes but there is, above all, a very strong difference between the batch processes themselves.

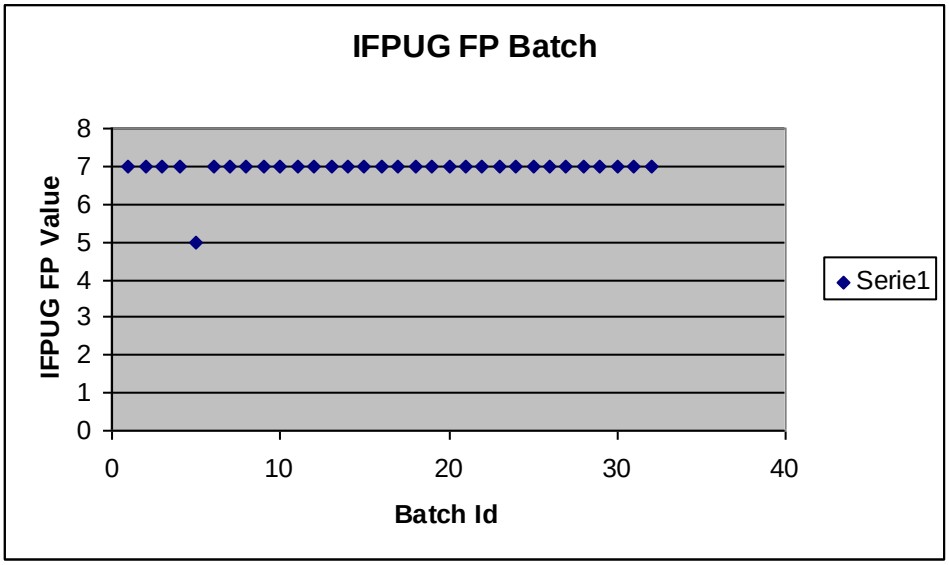

Figure 3 – Distribution of IFPUG FP Batch

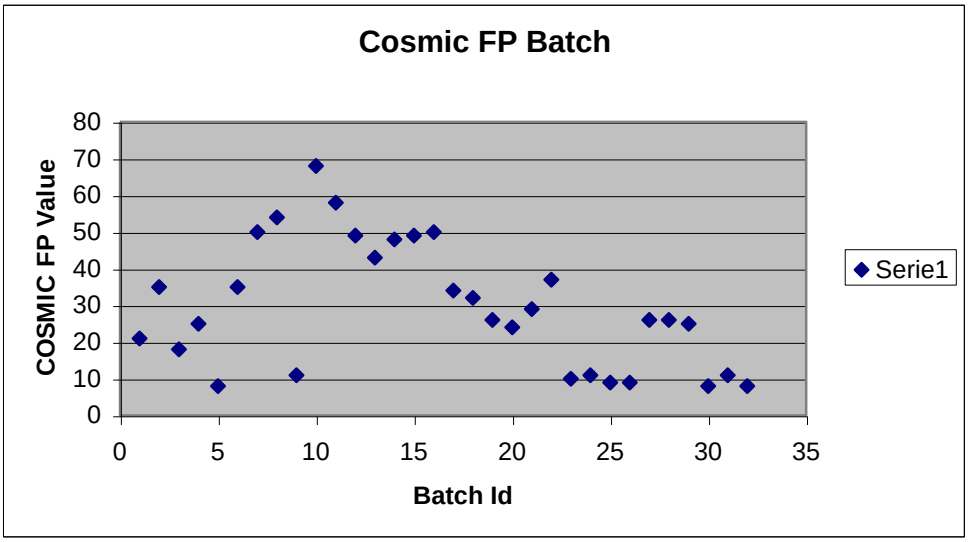

Figure 4 – Distribution of COSMIC FP Batch

5. SOA Architecture and Reuse

Nowadays, many applications are developed in a SOA architecture, some pieces of software are available as business services. This type of architecture should allow a certain amount of reuse in the application developing; the question is: "can we measure this amount of reuse?"

In some cases the SOA approach allows to have a business service as a functional process recognizable by the user (i.e. the authentication procedure), in other cases business services are part of a functional process that contain them (i.e. the reading of some external data, maintained by another application, during the flow of an elementary process). It would be useful to calculate a functional measure of the amount of the reusable software. When the business service available in a SOA architecture is a functional process and it is recognizable by the user, the measure can be calculated as one elementary process, so it shouldn't be a problem.

When the business service is contained in a functional process, it is more difficult to measure it as an elementary process. In both cases we have some reuse: this means that some operation that we will have to do are already done by a piece of software available. The problem is only how to measure this piece of software through a metric.
While in the first case we can use both COSMIC and IFPUG function points for the measure of the elementary process, in the second case, without the presence of an elementary process, the use of COSMIC function points through the identification of data movements can help us; the fact is: every data movements has associated an object of interest that have to be meaningful for the user, so which data movements can we count? It can depend on the user's point of view.
In the Figure 5 there is a representation of the two cases.

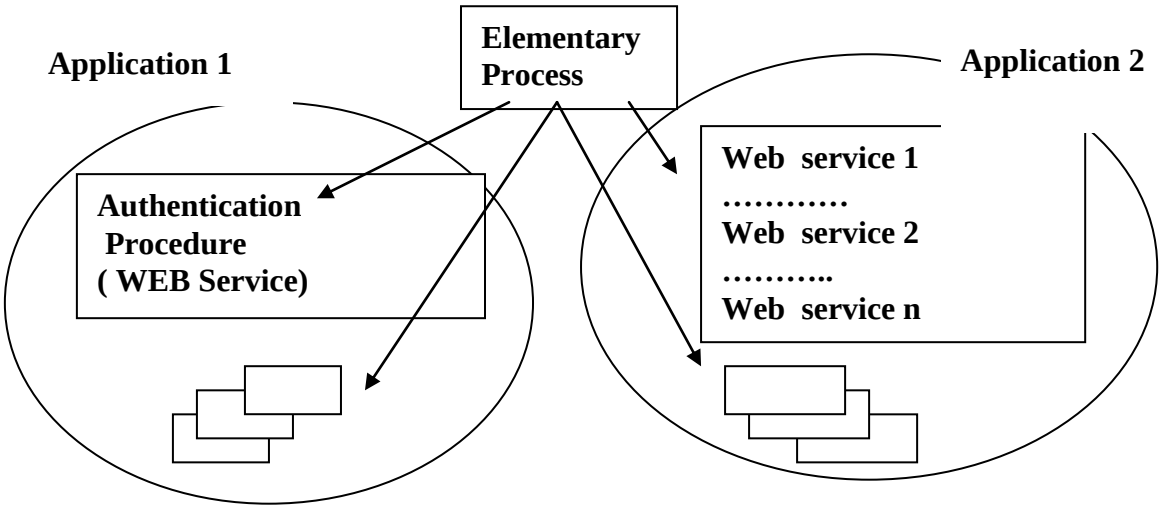

Figure 5

5.1. A Real Case

We illustrate a Use Case of inserting personal data into a register of birth. During the elementary process of inserting, the system has to reference many external interface files, to control if the person is already present, if he is present in other registers of birth and so on.
The first operation of the elementary process is to control if the user has the privilege to insert the person, otherwise an error message is displayed, then the system can proceed with the insert function.
As the insert function can be activate by more than one business application a business service has been built to perform this operation.
The user knows all the controls needed (they are functional requirements), but the fact to create a business service is not a specific requirement.
In the following Figure 6 we can see the COSMIC function points and IFPUG function points counting of the elementary process of inserting a person.
In the IFPUG counting there are also the Logical Data Files used in the EI.

COSMIC FP Counting

	Entry	Exit	Read	Write	FP	Data movements
PersonaFisica - Inserimento	2	1	2	1	**6**	Entry: PF, Residence Data Read: ID Utente Read : Privilege; Write: PF (business service inserisciPersonaFisica) Exit: message
inserisciPersonaFisica (business service)	2	1	7	3	**13**	Entry: PF **Read : PF in GMS Read: NAO; Read: BPR, Read: Topo; Read SITAD, Read: SAS Write: Anagrafica PF , Residenza** Write: Log; Exit: result Entry:ID Utente Iride; **Read: Iride**

IFPUG FP Counting

		FTR/RET	DET	FP
Persona Fisica - Inserimento	**EI**	8	23	6
GMS	EIF	1	8	5
NAO	EIF	1	15	5
BPR	EIF	1	6	5
Topo	EIF	1	5	5
SOTAD	EIF	1	5	5
SAS	EIF	1	6	5
Anagrafica PF	ILF	2	21	10
Iride	EIF	1	6	5

Figure 6

If we have to implement an application that performs the inserting function using the business service already available, we can notice that in IFPUG function points there is one transactional function EI that performs the inserting operation. It's impossible to weigh in function points the reuse of this function, the only thing that we can do is to apply a percentage to the 6 function points (a corrective factor, i.e. using the Nesma Enhancement Process)[3].

If we use the COSMIC function points we see that is possible to identify the data movements that we have not to implement. In the Figure 6 they are represented in bold: there are seven data read movements(Read : PF in GMS Read: NAO; Read: BPR, Read: Topo; Read SITAD, Read: SAS**,** Read: Iride) and two data write movements (Write: Anagrafica PF , Residenza)

for a sum of nine COSMIC function points. If we consider the nine data movements minus the write data movement of the call of business service, we can consider a reuse of **eight COSMIC function points**.
In this manner we can consider a certain amount of function points of reuse for every elementary process.

6. Complex Functions

In some cases, in a GUI application, there are complex procedures that are activated through buttons. Each of these procedures is a unique elementary process from the user's point of view. Generally they reference many FTRs but they haven't always many DETs. If the application is not so big they can have a strong impact about productivity and, consequently, on the effort of software developing. It's not so easy to evaluate their impact.
We illustrate a real case of an application of 190 IFPUG function points that has the very complex function "Esegui Verifica", activated by a button. The procedure is an EO transactional function.

			IFPUG FP			
	Function		FTR	DET	Compl.	FP
	.....					
Procedimenti						
	Vsualizzazione elenco procedimenti per azienda	EQ	1	1	Bassa	3,0
	Dettaglio procedimento	EQ	2	2	Media	4,0
	Nuovo procedimento	EI	2	5	Media	4,0
	Modifica	EI	2	3	Bassa	3,0
	Richiedente	EQ	1	1	Media	4,0
	Effluenti prodotti	EO	1	1	Bassa	4,0
	Dettaglio effluente	EO	1	1	Bassa	4,0
	Modifica effluente	EI	1	3	Bassa	3,0
	Annulla	EI	1	2	Bassa	3,0
	Elimina	EI	1	2	Bassa	3,0
	Stampa	EQ	2	2	Alta	6,0
	Revoca stampa	EI	1	2	Bassa	3,0
	Esegui verifica	**EO**	**6**	**23**	**Alta**	**7,0**
U.P.A.						
	Elenco UPA	EO	1	1	Bassa	4,0
	.....					

Figure 7

In the Figure 7 there is an abstract from the IFPUG function points counting, while in Figure 8 there is the COSMIC function points counting of "Esegui Verifica" of the same function.

As we can see the value in IFPUG function points of "Esegui Verifica" is seven while the value in COSMIC function points of the same function is 25 function points.

	COSMIC FP					
	Entry	Exit	Read	Write	FP	
Esegui Verifica	**1**	**12**	**12**	**0**	**25**	Entry: Procedimento; Read: anagrafe aziende, anagrafe tributaria, infocamere,Iter, dati, efflenti prodotti, UPA, tecnica colturale, tipologie di smaltimento, dichiarazioni, allegati, controlli, Exit:anagrafe aziende, anagrafe tributaria, infocamere,Iter,

Figure 8

Also in this case the number of COSMIC function points seems to weigh in a more significant manner the weigh of the function.

7. How do we manage IFPUG and COSMIC FP measures in the same application?

When part of a business application has been measured in IFPUG FP and other parts in COSMIC function points the problem is: "Which is the real measure of the whole application?"

We can't obviously sum IFPUG measure with COSMIC measure!

If there is a need to have a whole measure of the application, we can apply a conversion factor from COSMIC to IFPUG (i.e. Desharnais: CFP = 1,0*IFPUG –3; van Heringeen: CFP = 1,22*IFPUG – 64)

Can we trust these factors? According to the previous examples seen it's difficult to trust them completely.

It would be better to consider the two measures separately, each of them with its own characteristics.

To consider the two measures separately means naturally that we will have different indicators for each of them (i.e. Productivity, Defective…)

We can use Literature Data (ISBSG, Capers Jones,…) about some indicators. In the case of COSMIC the historical data available are not so many as in IFPUG function points, so, perhaps, it can be difficult to trust them at the moment. The better thing would be, in any case, to collect own data into a repository and to build own indicators.

8. Conclusions

Using metrics is undoubtedly a matter of culture

To use different metrics in an application gives the project manager more suitable information to control the software process. However we have to pay more attention in using metrics. Someone says that metrics are like spies, under pressure they tell you what you want!

Naturally we have to use the right metric in the right place and at the right moment, without any prejudice but, above all, we have to collect data for understanding and knowing what they can exactly give us, which indicators are useful to us.

This process is a delicate one, it has to be performed by metrics experts people that guide all the metric process and avoid a wrong use of it.

The most important thing, in any case, is that metrics can help us but they are not the solution of all our problems.

9. References

[1] IFPUG, “Function Point: Manuale delle Regole di Conteggio”,versione 4.2.

[2] COSMIC, “The COSMIC Functional Size Measurement Method” version 3.0.

[3] NESMA, "Function Point Analysis for Software Enhancement”, version 1.0.