Early functional size estimation with IFPUG unit modified

Juan J. Cuadrado-Gallego, Pablo Rodríguez-Soria, Alfonso González, Dácil Castelo
Computer Science Department
University of Alcalá
Madrid, Spain
jjcg@uah.es

Saahil Hakimuddin
Department of Computer Science and Engineering
Manipal Institute of Technology
Manipal, Karnataka, India
saahil.in@gmail.com

***Abstract*—Nowadays functional size measurement is a strategic key to deal with the management of software systems development. The origin of this importance is the fact that functional size measurement is the main input variable in software effort estimation systems. Nevertheless, to obtain precise functional size measurements it is not only necessary to have a lot of information of the system to be developed, but also software project planning is one of the early stages in the project. To solve this difficulty, one of the main software management research technique is centered in the study of methods to obtain precise functional size measurements early in the development phase for early functional size estimation. The functional size unit selected to do the study has been IFPUG because is the most widely used method.**

Keywords - Software Management; Software Process; Software Measurement; Functional Size, IFPUG

I. INTRODUCTION

The first unit used to measure the size of software products was given by the number of source lines of code (SLOC). Although this unit is useful when it is used to analyze different aspects such as error ratios or team productivity ratios, from a software project management point of view, SLOC suffers from the fact that it can only be measured once the software has been built.

For that reason, the definition of a magnitude able to measure software, for management issues, early in the project lifecycle became essential. One of the most significant figures for managing a software project is its functional size. Derivable from the projects' functional user requirements (FUR), it is possible to estimate the amount of human and material resources needed, time and costs required. This is the main aspect for the project development success.

Function points and the related measurement method was given by IBM's researcher Allan Albrecht, first by himself [1] and then working with his collaborator John Gaffney [2]. This measurement unit can be applied when the documentation is available during project's early phases, such as the software requirements specification and analysis phases. The enacting need for a software measurement unit such as the one proposed by Albrecht, able to solve software projects management issues, together with the success derived from its first applications, were the reasons for the foundation, in 1986, of the International Function Point Users Group (IFPUG), whose main goals are to promote the usage of this measure and to control the evolution of the measurement standard definition. The method was developed to measure the amount of functionality to be delivered to end users as perceived from their viewpoint.

Since its appearance in 1979, several variants have been produced during the years. This changed the name from Albrecht's Function Points to IFPUG Function Points Analysis (FPA). Since then, several versions of IFPUG FPA have been published. Among those methods, four have been recognized by the International Organization for Standardization (ISO) de jure standards:

1. IFPUG v.4.1, 1998. Standard ISO/IEC 20926 [3]

   This method will be described in detail in the next section.

2. NESMA v.2.1, Standard ISO/IEC 24570 [4]

The first description of the NESMA FPA is presented on a manual published by the Netherlands Software Metrics Association (NESMA), where it explains how to apply the IFPUG measurement unit, in particular to software developed as a part of maintenance projects. This manual, of which five versions has been published, has had a large impact on the software industry; that is the reason why the NESMA's Function Points are considered both a measurement unit and an international standard. NESMA FPA represents a minor variation from the IFPUG method and therefore it is possible to consider the two related functional size units as equivalent.

3. MK II v.1.3.1, Standard ISO/IEC 20968

Inspired in the IFPUG FPA, but with some foundations introducing noteworthy differences from them, MK II's Function Points were published by Charles Symons [5] as a new unit proposed for software functional size measurement. The MK II unit obtained widespread reach, mainly in the nineties, not only in the United Kingdom where it originated from, but also in many other countries. The reasons behind its success lay in the belief that this unit improves upon IFPUG FPA in a way so as to consider the internal complexities on data handling, a key aspect of business. All these reasons led to the promotion of this measurement unit

978-0-7695-4147-1/10 $26.00 © 2010 IEEE
DOI 10.1109/ICIS.2010.12

as an international standard. Nevertheless, the fact that Charles Symons participated in the development of COSMIC unit also, and is currently working in its development and sponsorship, should introduce certain skepticism about the future of the acceptance and usage of the MK II unit.

4. COSMIC v2.2. Standard ISO/IEC 19761[6]

To organize the execution of the tasks that led to the definition of the new measure, some experts established in 1998 the Common Software Measurement International Consortium, COSMIC, whose first outcome was the definition of the measure in 1999. Since its first publication, the interest in the new unit among both, the academic community and the industry, was enormous, reaching vast diffusion and utilization in very short time, with three new versions published later, including the one which has been standardized. COSMIC Function Points – that represent a 2nd generation FSM method - are the result of the pursuit of the international group of experts in software functional size measurement, to find a measurement unit capable of being successfully applied to the greatest possible number of software types and, specially, to real time software, where the application of the IFPUG unit is really hard.

The growing interest in industrial organization created by COSMIC can be verified by the growing number of projects included in the ISBSG repositories (one of the most important global repositories of data about software projects) which has grown from less than 50 projects in the 8th edition up to 110 ones in the latest one (the 10th edition, January 2007), with a growing rate more than 100% in five years; another noteworthy statistical data is the growing number of measurement experts certified in COSMIC, which, too, has grown by more than 200% in the last 2 years; all these facts reflects the relevance of research activities on it.

This situation is the result of the contribution of three factors: firstly, its wide scope of applicability, since the unit can be used to measure many different kinds of software; secondly, the clarity of its concepts, making the unit easy to use and to learn to use; and finally, the low cost resulting from using this unit.

The paper is organized as follows: Section 2 presents the IFPUG main features. Section 3 presents a review of previous published studies. Section 4 shows the conclusions for this work and outlines future research issues, and paper ends with a list of references used.

II. IFPUG METHOD

Function Points is functional measurement method based on Lineal equations. It was published for the first time in 1979 by Alan Albrech. The Function Points method is developed as an alternative to the estimation of the software product size through SLOC. Function Points have a rather major level of abstraction in comparison the SLOC, attending to aspects such as the number of input transaction types or the number of different reports generated by the system.

When they were presented, function points constituted a complete model for effort estimation and the equation that is gathered here corresponds to this model. Currently this estimation method of function points is used to determine the size of the software that is going to be developed, which will be used as an input variable for some other specific model of effort estimation.

Function Points represents some advantages against the SLOC; for example these can be estimated earlier in the life cycle since it is only necessary to have the requisites definition document, which is very interesting if function points are used as input in an effort estimation model, along with development time, since these two data could be known with a good approach and also very quickly. Another advantage is that they can be calculated by non technical members of the development team. Also, function points avoid the effects of the coding language and other differences in the implementation.

The calculation of Function Points is performed in two phases:

Classify the user´s functions under its category and calculate the not fitted function points by attending to the level of information processed by each function, which can be simple, medium and complex. For each level and function, pertaining to its category, there will be a natural number corresponding to the assignable function points for this function.

Currently there are 5 function categories (In the first article only four were defined):

1. Internal Logical Files (ILF)

   An internal logical file is a user recognizable group of logically related data or control information maintained within the boundary of the application being measured.

2. External Interface Files (EIF)

   An external interface file is a user recognizable group of logically related data or control information which is referenced by the application being measured, but maintained within the boundary of another application.

3. External Input (EI)

   An elementary process that processes data or control information sent from outside the boundary.

4. External Inquiry (EQ)

   An elementary process that sends data or control information only outside the boundary, using data retrieval.

5. External Output (EO)

   An elementary process that sends data or control information outside the boundary and includes additional processing beyond that of an external inquiry.

To establish the complexity of ILF and EIF the following rules must be followed:

1. Assign each identified ILF/EIF a functional complexity based upon the number of Data Element Types (DET) and Record Element Types (RET) associated with the ILF or EIF.
2. Count a DET for each unique user recognizable, which is a non-repeated field maintained in or retrieved from the data function through the execution of all elementary processes within the counting scope.
3. Count one RET for each data function. Count an additional RET for each of the following logical subgroups of the data function that contains more than one DET.
    a. Associative entity with non key attributes.
    b. Unique Sub-type.
    c. Attribute entity, in a relationship other than mandatory 1-1.

The complexity matrix for ILF and EIF is:

1	RET	1-19 DET(Low)	20-50 DET(Low)	+51 DET(Avg)
2-5	RET	1-19 DET(Low)	20-50 DET(Avg)	+51 DET(High)
+6	RET	1-19 DET(Avg)	20-50 DET(High)	+51 DET(High)

To establish the complexity of EI/EQ/EO the following rules must be followed:

1. Assign each identified EI/EQ/EO a functional complexity based upon the number of Data Element Types (DET) and File Types Referenced (FTR) associated with the the transactional function.
2. Review every DET (field) that crosses (enters/exits) the boundary. Count only one DET for each user recognizable, which is a non repeated attribute, that crossed the boundary during the processing of the transactional function.
3. Count one FTR for each unique data function that is accessed (read from and/or written to) by the transactional function.

The complexity matrix for EI is:

0-1	FTR	1-4 DET(Low)	5-15 DET(Low)	+16 DET(Avg)
2	FTR	1-4 DET(Low)	5-15 DET(Avg)	+16 DET(High)
+3	FTR	1-4 DET(Avg)	5-15 DET(High)	+16 DET(High)

The complexity matrix for EQ and EO is:

0-1	FTR	1-5 DET(Low)	6-19 DET(Low)	+20 DET(Avg)
2-3	FTR	1-5 DET(Low)	6-19 DET(Avg)	+20 DET(High)
+4	FTR	1-5 DET(Avg)	6-19 DET(High)	+20 DET(High)

After establishing the functions and their complexities, the function points counting weights are:

ILF	Low 7	Avg 10	High 15
EIF	Low 5	Avg 7	High 10
EI	Low 3	Avg 4	High 6
EO	Low 4	Avg 5	High 7
EQ	Low 3	Avg 4	High 6

The natural number reflects the number of function points.

By fitting the function points attending to the application complexity. 14 complexities features were analyzed:

1. Data communications (C1)
2. Distributed Data Processing (C2)
3. Performance (C3)
4. Heavily used configuration (C4)
5. Transaction rate (C5)
6. On-line data Input (C6)
7. End user efficiency (C7)
8. On-line data update (C8)
9. Complex processing (C9)
10. Reusability (C10)
11. Ease of Installation (C11)
12. Ease of Operation (C12)
13. Multiple localization (C13)
14. Change of facility (C14)

Each one with a variation range:

- Not present or without influence = 0
- Insignificant influence = 1
- Moderate influence = 2
- Medium influence = 3
- Significant influence = 4
- Decisive influence = 5

The adjusted Function Points calculation can oscillate in ± 35 % from the original Function Points calculation.

III. PRLIMINARY STUDIES

Some preliminary studies can be found in the literature, some of the main ones being published by Meli and Santillo of Italy and also some of others published by Asensio et al. of Spain.

The ones from Italy are:

- "Early and Extended Function Points: a new method for function points estimation" [7]
- "Early function points: some practical experiences of use" [8]
- "Early and Quick function points analysis" [9]
- "E&Q: an Early & Quick Approach to Functional Size Measurement Methods" [10]

All of these studies, more or less, reflect the same IFPUG early measurement model. Its main characteristics are:

1. The model proposed 4 sub-models or aggregation levels corresponding to the level of detail with which the system is known.
2. The first aggregation level is applied when user requirements are sufficient known to apply the IFPUG standard unit described in immediately previous section. The numbers used are exactly the same as enumerated in that section.
3. The second aggregation level is applied when the knowledge of the systems to be developed is enough to identify most of the IFPUG functions, but not the complexity of each one. For other functions, it is only possible to identify them like data functions (ILF o EIF) or transactional functions (EI, EQ or EO). For that level the function points counting weights are:

ILF	Min 7,4	Most Likely 7,7	Máx 8,1
EIF	Min 5,2	Most Likely 5,4	Máx 5,7
DataFunction	Min 6,4	Most Likely 7,1	Máx 7,8
EI	Min 4	Most Likely 4,2	Máx 4,4
EO	Min 3,7	Most Likely 3,9	Máx 4,1
EQ	Min 4,9	Most Likely 5,2	Máx 5,4
EOEQFunction	Min 4,1	Most Likely 4,6	Máx 5
TransacFunction	Min 4	Most Likely 4,4	Máx 4,8

4. For the third and the fourth aggregation level, new and very different concepts from IFPUG concepts are introduced and the description and results of those are out of the scope of this paper.

The one from Spain is

- "MTPF Function Points Measure Early Method" [11]

The main characteristics of this method are:

1. Establish two different numbers - CILFEIF as a sum of ILF and EIF; and CEIEOEQ as a sum of EO, EI and EQ.
2. Establish a high degree of correlation between the number of IFPUG function points and these two magnitudes.
3. Since, CILFEIF is related with the number of entities, the model proposes not to count the number of ILF and EIF but the number of entities and, for the same reason, not to count the number of EI, EO, and EQ but the number of processes.
4. Define the concepts of Entity and assign them different weights in order of themselves and their multiplicity. And define the variable CENT to measure entities.
5. Define the concepts of Elemental Process, Micro Function and Macro Function to measure processes. And define the variable CPRO to measure processes.

IV. CONCLUSIONS AND FUTURE WORK

This paper presents the importance of functional size measurement to obtain software projects effort estimations. And from that point the importance that an early estimation could possibly would have.

Starting from that point the performance of IFPUG function points are presented and based on it a review of the main proposals that could be found in the literature to perform early software functional size measurements with IFPUG, is stated.

Considering this paper as a beginning, the team is now researching a new and more usable method to obtain software functional size measurements early in the software projects' life cycles.

REFERENCES

[1] Albrecht A. J., "Measuring application development productivity," en Proc. Joint SHARE, GUIDE, and IBM Application Development Symp., IBM, pp. 83-92.

[2] Albrecht A. J. & Gaffney J. E., "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Trans. Software Eng., vol. 9, no. 6, pp. 639-647.

[3] ISO/IEC 20926: 2003, Software engineering IFPUG 4.1 Unadjusted functional size measurement Method. Counting practices manual International Standardization Organization, ISO, Ginebra, 2003.

[4] NESMA, "Definitions and counting guidelines for the application of function points analysis. A practical manual 2.2", Nederlandse Software Metrieken Associatie

[5] Symons C., "Function Point Analysis: Difficulties and Improvements," IEEE Transactions on Software Engineering, vol. 14, no. 1, pp 2-11.

[6] Common Software Measurement International Consortium, "COSMIC-FFP Meaurement Manual 3.0

[7] Meli, R. "Early and Extended Function Points: a new method for function points estimation ," IFPUG Fall Conference, Arizona, USA, September 1997

[8] Santillo, L. Meli, R. "Early function points: some practical experiences of use ," ESCOM, Roma, 1998

[9] Meli, "Early and Quick function points analysis," Roma, 2002

[10] Santillo, L "E&Q: an Early & Quick Approach to Functional Size Measurement Methods," IWSM, Montreal, 2005

[11] Asensio, "MTPF Function Points Measure Early Method," IWSM, Montreal, 2005