World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

A Case Study to Assess the Validity of Function Points

Neelam Bawane nee' Singhal, and C. V. Srikrishna

***Abstract*—**Many metrics were proposed to evaluate the characteristics of the analysis and design model of a given product which in turn help to assess the quality of the product. Function point metric is a measure of the 'functionality' delivery by the software. This paper presents an analysis of a set of programs of a project developed in C++ through Function Points metric. Function points are measured for a Data Flow Diagram (DFD) of the case developed at initial stage. Lines of Codes (LOCs) and possible errors are calculated with the help of measured Function Points (FPs). The calculations are performed using suitable established functions. Calculated LOCs and errors are compared with actual LOCs and errors found at the time of analysis & design review, implementation and testing. It has been observed that actual found errors are more than calculated errors. On the basis of analysis and observations, authors conclude that function point provides useful insight and helps to analyze the drawbacks in the development process.

***Keywords*—**Function Points, Data Flow Diagram, Lines of Codes.

I. Introduction

QUALITY assurance is to verify that applicable procedures and standards are being followed. Quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities. To achieve a high quality product, we want to correct as many as errors as possible before the end users encounter them and declare as defects.

Software metrics provide a quantitative way to assess the quality of internal product attributes thereby enabling a software engineer to assess quality before the product is built. Metric is a quantitative measure of the degree to which a system, component or process possesses a given attribute [13].

Many metrics were proposed to evaluate the characteristics of object-oriented design, i) evaluate the characteristics of UML, ii) estimate the development effort, and for several other purposes. Among the proposed metrics, a number of approaches undertook the adaptation of the principles of Function Point Analysis to object-oriented systems.

This paper aims at developing some experimental evidence concerning Function Point as object-oriented metrics through the application of such metrics to a set of programs. The measured programs were developed in a quite homogeneous environment, with no relevant external bias affecting the results of measurements. The present analysis is restricted to relatively small size of the program.

However, the development process and the quality of the product were quite representative for applications to typical object-oriented systems. Thus the results presented here can be considered valid for average object-oriented products.

The paper is organized in two parts.

First part describes the literature survey about metrics specifically about the function point metrics. Second part describes a case study which shows the validity of function points (calculated for DFD at analysis phase) to calculate the LOCs and possible errors in design and implementation.

Neelam Bawane is with PES Institute of Technology, Bangalore, India (phone: 09945516241; e-mail: neelambawane@yahoo.com).

C. V. Srikrishna is with PES Institute of Technology, Bangalore, India (phone: 09448107190; e-mail: cvsrikrishna@yahoo.co.in).

II. Function-based Metrics

The function point metric (FP), first proposed by Albrecht, can be used effectively as a means for measuring the functionality delivered by a system.

Function Points are a measure of the size of computer applications. The size is measured from a functional, or user point of view. It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application. Regardless of language, development method, or hardware platform used, the number of function points for a system will remain constant [2]. The only variable is the amount of effort needed to deliver a given set of function points. Although function point metric does not satisfy the consistent and objective attribute which should be present in effective software metric, it provides useful insight and widely used [3].

Function points can also be used to predict the number of possible errors likely to occur at different phases such as analysis& design review, unit and integration testing. Function points [5], [24] are derived using an empirical relationship based on countable measures of software's information domain and assessments of software complexity. Information domain values are defined in the following manner:

Number of external inputs (EIs): Each external input originates from a user or is transmitted from another application. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately. E.g. transaction types

Number of external outputs (EOs): Each external output is derived within the application and provides the information to the user, e.g. reports, screens error messages etc. Individual data items within a report are not counted separately.

Number of external inquiries (EQs): An external inquiry is defined as an online input that results in the generation of some intermediate software response in the form of an online output.

Number of internal logical files (ILFs): Each internal logical file is a logical grouping of data that resides within the application boundary and is maintained via external inputs.

Number of external interface files (ELFs): Each external logical file is a logical grouping of data that resides external to the application but provides the data that may be of use to the application.

A complexity value is associated with each count. The determination of complexity is subjective. Organizations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex.

The complexity classification of each component is based on a set of standards that define complexity in terms of objective guidelines.

First, the function counts (FCs) can be calculated with the help of weighting factors based on the equation (1) & Table I:

$$FC = \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} * x_{ij} \quad (1)$$

where w_{ij} are the weighting factors of the five components by complexity level (low, average, high) and xij are the numbers of each component in the application.

TABLE I
COMPUTING FUNCTION POINTS

Information Domain Value	Count	Weighting factor			
		Simple	Average	Complex	Total
External Inputs (EIs)	No. X	3	4	6 =	
External Outputs (EOs)	No. X	4	5	7 =	
External Inquiries (EQs)	No. X	3	4	6 =	
Internal Logical Files (ILFs)	No. X	7	10	15 =	
External interface Files (EIFs)	No. X	5	7	10 =	
Count total	→				

The Fi (i=1 to 14) are value adjustment factors (VAF) based on responses to the following characteristics that ranges from 0 (not important or applicable) to 5 (absolutely essential).

- Data communications
- Distributed functions
- Performance
- Heavily used configuration
- Transaction rate
- Online data entry
- End-user efficiency
- Online update
- Complex processing
- Reusability
- Installation ease
- Operational ease
- Multiple sites
- Facilitation of change

Second, the value adjustment factor (VAF) can be calculated by summing up the scores (ranging from 0 to 5) for these characteristics based on the equation (2):

$$VAF = 0.65 + 0.01 \sum_{i=1}^{14} ci \quad (2)$$

where the value adjustment factor (VAF) is the score for general system characteristic. Finally, the number of function points is obtained by multiplying function counts and the value adjustment factor using equation (3):

$$FP = FC * VAF \quad (3)$$

Based on the projected FP value derived from the analysis model, the project team can estimate the overall implemented size of the Project. Past data indicate that one FP translates into 60 lines of code if an object oriented language is used.

These historical data provide the project manager with important planning information that is based on the analysis model rather than preliminary estimates. Past projects have also found an average of three errors per function point during analysis and design reviews and four errors per function point during unit and integration testing. These data can help software engineers assess the completeness of their review and testing activities

III. A CASE STUDY – LIBRARY SYSTEM

Problem Definition

Present case study is carried out to assess the following:

Can function point analysis be used as the method for calculating the software size in terms of LOCs and possible errors?

Method of Evaluation

The informal requirements for the basic version of the library management programs were the following:

A register of accredited users is maintained by an administrator, who can add, remove and change user privileges. Users get access to the system via the typical login/logout mechanism. A logged-in user can search for the books in the catalogues based on author names or titles of the books. Librarian can issue and return the books after verifying the member and book details.

The specifications of the library system were considered and measured. The specification of the system was given to students by means of an informal text. In order to support the computation of Function Points, we translated the original

specifications into a data flow diagram. In order to remain as independent as possible from the UML-based techniques, the measurement was based on the data-flow diagrams. A simplified version of such DFD is reported in Fig. 1.

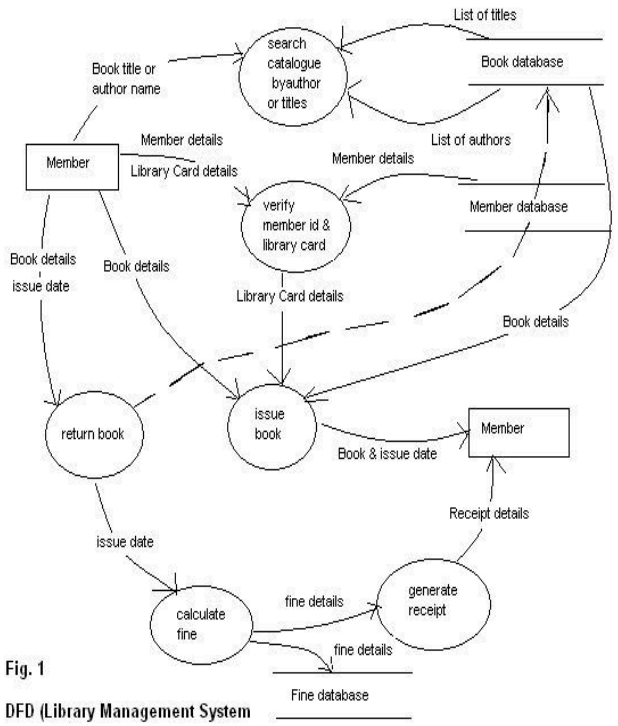

Fig. 1
DFD (Library Management System

Function point computation

Function types were identified, and their complexity was evaluated: the values are reported in Table I. Function points are derived using an empirical relationship [Eq.1] based on countable measure of software's information domain and assessments of software complexity [20], [24]

Referring to DFD (Fig. 1)

Number of external inputs (EIs): 8

- Member_id
- Member_name
- Book_id
- Book_title
- Author_name
- Publisher_name
- Library_card_no
- Issue_date

Number of external outputs (EOs): 3

- Issue_date
- return_date
- fine_amount

Number of external inquiries (EQs): 2

- Search catalogue by author
- Search catalogue by book title

Number of internal logical files (ILFs): 5

- Book database
- Member database
- Library_card details database
- Issue_return database
- Fine database

Number of external interface files (ELFs): 1

- Member

TABLE II
COMPUTING FUNCTION POINTS

Function type	Low	Average	High	Total
EIs	8 * 3	0 * 4	0 * 6	24
Eos	3 * 4	0 *5	0 * 7	12
EQs	2 * 3	0 * 4	0 * 6	06
ILFs	3 * 5	0 * 7	0 * 15	15
ELFs	1 * 7	0 * 10	0 * 10	07
Total FPs				64

Each of 14 questions (mentioned in literature) is answered using a scale of 0 to 5. We assume that ∑Ci is 48 (a moderately complex product). The count total must be adjusted using equation (2) & (3):

$$FP = counttotal * [0.065 + 0.01 * \sum C_i]$$

count total =64 from Table II

Therefore,

$$FP = 64 * [0.65 + (0.01 * 48)]$$
$$= 72.32 \approx 72$$

Assumptions & Results

Past date indicate that one FP translate into 60 times of code (if an OOP language is to be used)

LOCs = 60 * 72
= 4320 (approximately)

Past project have found an average of 3 errors per function point during analysis and design reviews and 4 errors per function point during unit and integration testing.

Thus, possible number of errors in analysis and design reviews should be 3*72 i.e. 216. At the time of testing possible number of errors should be 4*72 i.e. 288. Thus total possible number of errors should be 504.

Verification of Results

After implementation it was found that lines of code are 4870, which is more than calculated LOCs (on the basis of FPs in analysis phase) by a value of 550.

Errors found at the time of analysis and design reviews are 196 and errors found at the time of testing are 325. Thus total errors found are 521 which is more than calculated by a value of 17.

IV. Conclusion

Above analysis and observations shows that function points is an important tool to measure the probable errors at the all the development stages. However, errors found may be more if development process is not matured, thus an indication to improve the process.

Acknowledgment

The authors acknowledge the support given by the management to do the research work.

References

[1] Alain Abran, Pierre N. Robillard, "Function Points Analysis: An Empirical Study of Its Measurement Processes", IEEE Transactions on Software Engineering,, Volume 22, Issue 12, December 1996
[2] C. R. Symons, "Function Point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, Volume 14, Issue 1, January 1988
[3] Chris F. Kemerer, "Reliability of function points measurement: a field experiment", Communications of the ACM, Volume 36, Issue 2, February 1993
[4] Chris F. Kemerer, Benjamin S. Porter, "Improving the Reliability of Function Point Measurement: An Empirical Study", IEEE Transactions on Software Engineering, Volume18, Issue 11, November 1992
[5] Futrell, Robert T., Shafer Donald F. and Shafer, Linda I., "Quality Software Project Management," Pearson Education Pte. Ltd., Delhi, 1st edition, 2002.
[6] G. Antoniol, R. Fiutem, C. Lokan, "Object-Oriented Function Points: An Empirical Validation", Empirical Software Engineering, Volume 8, Issue 3, September 2003
[7] Giuliano Antoniol, Chris Lokan, Gianluigi Caldiera, Roberto Fiutem, "A Function Point-Like Measure for Object-Oriented Software Empirical Software Engineering, Volume4, Issue 3, September 1999
[8] Godbole, Nina S., "Software Quality Assurance: Principles and Practices", Alpha Science International Ltd., 2004.
[9] Graham C. Low, D. R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Transactions on Software Engineering, Volume 16, Issue 1, January 1990
[10] http://en.wikipedia.org/wiki/Function_point
[11] http://en.wikipedia.org/wiki/Software_testing
[12] http://jira.atlassian.com/secure/attachment/17146/sqa+activities.txt
[13] http://www.isb.wa.gov/policies/portfolio/tr25/tr25_l2e.html
[14] http://www.softwareqatest.com/qatfaq1.html
[15] http://www.softwareqatest.com/qatfaq2.html
[16] http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=6331
[17] IEEE Standards Collection: Software Engineering, IEEE Standard
[18] J. E. Matson, B. E. Barrett, J. M. Mellichamp, "Software Development Cost Estimation Using Function Points", IEEE Transactions on Software Engineering, Volume 20, Issue 4, April 1994
[19] Jalote, Pankaj, "CMM in Practice," Pearson Education Pte. Ltd., 1st edition. 2004.
[20] Kan, Stephen H., "Metrics and Models in Software Quality Engineering," Pearson Education Pte. Ltd., Delhi, 2nd edition, 2004.
[21] McCall, J. P. Richards and G Walters, "Factors in software Quality," NTIS AA-A049-014, 015, 055 Nov 1977
[22] Mohammed Abdullah Al-Hajri, Abdul Azim Abdul Ghani, Md Nasir Sulaiman, Mohd Hasan Selamat, "Modification of standard function point complexity weights system", IEEE Transactions on Software Engineering, Volume 19, Issue 7, July 1993
[23] Mohammed Abdullah Al-Hajri, Abdul Azim Abdul Ghani, Md Nasir Sulaiman, Mohd Hasan Selamat, "Modification of standard function point complexity weights system", Journal of Systems and Software, Volume74, Issue 2, January 2005
[24] Pressman, Roger S., "Software Engineering: A Practitioner Approach," McGraw-Hill Companies, Inc., 4th edition, 1997.
[25] R. Rask, P. Laamanen, K. Lyyttinen, "Simulation and Comparison of Albrecht's Function Point and DeMarco's Function Bang Metrics in a CASE Environment", IEEE Transactions on Software Engineering, Volume 19, Issue 7, July 1993
[26] Sebastian Kiebusch, Bogdan Franczyk, "Process family points versus (full) function points", EDSER '06: Proceedings of the 2006 international workshop on Economics driven software engineering research, May 2006
[27] Shinji Kusumoto, Masahiro Imagawa, Katsuro Inoue, Shuuma Morimoto, Kouji Matsusita, Michio Tsuda, "Function point measurement from Java programs May 2002 ICSE '02: Proceedings of the 24th International Conference on Software Engineering", Journal of Systems and Software, Volume 74, Issue 2, January 2005
[28] Silvia Abrahão, Geert Poels, "Experimental evaluation of an object-oriented function point measurement procedure", Information and Software Technology, Volume 49, Issue 4, April 2007
[29] Wei Xia, Luiz Fernando Capretz, Danny Ho, Faheem Ahmed, "A new calibration for Function Point complexity weights", Information and Software Technology, Volume 50, Issue 7-8, June 2008
[30] Wendy W.Peng, Dolores R. Wallace, "Software Error Analysis", NIST Special Publication 500-209