Function Point: A Quality Loom for the Effort Assessment of Software Systems

1. H. Azath and 2. R. S. D. Wahidabanu
1. King College of Technology, Namakkal, Tamil Nadu, INDIA
2. Government College of Engineering, Salem, Tamil Nadu, INDIA.

Summary
Accurate estimation of software development effort is critical in software engineering. Underestimates lead to time pressures that may compromise full functional development and thorough testing of software. In the existing systems, the effort and cost estimation are more concentrated only on the development of software systems alone and not on the quality coverage. Hence the quality assurance for the effort estimation is proposed in this paper. To assure this quality, the ISO 9126 quality factors are used. For weighing the factors, the function point metric is used as an estimation approach. The classification of software system for which the effort estimation is to be calculated based on the COCOMO model classes. An exhaustive literature survey reveals that attention is not paid to the following for estimating the effort: 1. Function point, 2. COCOMO classes of systems, and 3. ISO9126 quality factors. Thus by combining all the three parts, a new effort estimation method is developed as a research approach.

Key words:
COCOMO systems, effort estimation, function points, software quality.

I. INTRODUCTION

THE objective of project metrics is twofold. First, these metrics are used to minimize the development schedule by making the adjustments necessary to avoid delay and moderate potential problems and risks. Second, project metrics are used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality [16].

There are many effort estimation techniques for software system developments are available. But none of the models paid attention to the quality assurance coverage. However, some models concentrate only for the development of software that may cover few of quality assured factors and the quality consideration is not available for estimating the effort.

H. Azath is with the Regional Research Centre, Government College of Engineering, Zone 6, Anna University-Chennai, SALEM-636 011. (phone: 9865164505; e-mail: writetoazath@yahoo.com).
R. S. D. Wahidabanu is heading the Department of Electronics and Communication Engineering, Government College of Engineering, Zone 6, Anna University-Chennai, SALEM-636 011.

Manuscript received December 5, 2008
Manuscript revised December 20, 2008

So, this paper is fully focused on assuring the quality in effort estimation for software system development [1], [12], [19], [20]. In this paper the forthcoming sections are named as study variables, research approach and results, results comparisons, conclusion and future scope.

In the *study variables section*, the function point metric, COCOMO classes of systems and the ISO9126 quality factors are discussed [15]-[16].

The *research approach and results section* describes the usages of function point metric, the appliance of COCOMO classes of systems in the function point analysis and the weighing mechanism of ISO9126 quality factors. An example software system used to apply this proposed work is the CAD software [17]-[18].

In the *results comparisons section*, the effort (in terms of person-months) is used to compare the various results of some available models with the proposed result.

In the *conclusion and future scope section*, the results between proposed & existing scenario are compared and the possible extension of work are also discussed.

II. STUDY VARIABLES

A. Function points

Function-oriented software metrics is used to measure the functionality delivered by the application as a normalization value. The most widely used function-oriented metric is function point (FP). FP is a programming language independent, making it ideal for applications using conventional and nonprocedural languages. Moreover it is based on data that are more likely to be known early in the evolution of a project, making it more attractive as an estimation approach.

The accuracy of a software project estimate is depends on a number of things: (i) the degree to which the planner has properly estimated the size of the product to be built; (ii) the ability to translate the size estimate into human effort, calendar time, and cost expenses; (iii) the degree to which the project plan reflects the abilities of the software team; and (iv) the stability of product requirements and the environment that supports the software engineering effort.

The function point metric (FP), first proposed by Albrecht [ALB79], can be used effectively as a means for measuring the functionality delivered by a system. Using historical data, the FP can then be used to (i) estimate the cost or effort required to design, code, and test the software; (ii) predict the number of errors that will be encountered during testing, and (iii) forecast the number of components and/or the number of projected source lines in the implemented system.

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity. Information domain values are defined in the following manner:

Number of external inputs (Els): Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

Number of external outputs (EOs): Each external output is derived within the application and provides information to the user. In this context external output refers to reports, screens, error messages, and so on. 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 immediate software response in the form of an on-line output (often retrieved from an ILF).

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

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

Organizations that use function point methods can develop criteria for determining whether a particular entry is simple, average, or complex. Nonetheless, the determination of complexity is somewhat subjective.

To compute function points (FP), the following relationship is used:

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma (F_i)] \qquad (1)$$

Where count total is the sum of all FP entries as shown in Fig. 1. The F_i (i = 1 to 14) are value adjustment factors (VAF). The 0.65 and 0.01 are empirically derived constants.

The VAF is based on responses to the following questions:

1) Does the system require reliable backup and recovery?
2) Are specialized data communications required to transfer information to or from the application?
3) Are there distributed processing functions?
4) Is performance critical?
5) Will the system run in an existing, heavily utilized operational environment?
6) Does the system require on-line data entry?
7) Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8) Are the ILFs updated on-line?
9) Are the inputs, outputs, files, or inquiries complex?
10) Is the internal processing complex?
11) Is the code designed to be reusable?
12) Are conversion and installation included in the design?
13) Is the system designed for multiple installations in different organizations?
14) Is the application designed to facilitate change and for ease of use by the user?

Information Domain Value	*Count*		*Weighing factor*				
			Simple	*Average*	*Complex*		
External Inputs (EIs)		x	3	4	6	=	
External Outputs (EOs)		x	4	5	7	=	
External Inquiries (EQs)		x	3	4	6	=	
Internal Logical Files (ILFs)		x	7	10	15	=	
External Interface Files (EIFs)		x	5	7	10	=	
Count total →							

Fig. 1. Computing Function Points

Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential). The constant values in (1) and the weighing factors that are applied to information domain counts are determined empirically.

An estimation model for computer software uses empirically derived formulas to predict effort as a function of LOC or FP. The empirical data that support most estimation models are derived from a limited-sample of projects. For this reason, no estimation model is appropriate for all classes of software and in all the development environments. An estimation model should be calibrated to reflect local conditions.

Existing FP-oriented models include the following:

E = -91.4 + 0.355 FP	Albrecht and Gaffney model	(2)
E = -37 + 0.96 FP	Kemerer model	(3)
E = 0.054 x $FP^{1.353}$	SMPEEM	(4)

SMPEEM: Software Maintenance Project Effort Estimation Model [23].

Effort Validation

Every project has a defined number of people on the software team. As time allocation occurs, the project manager must ensure that not more than the allocated number of people should be scheduled at any given time.

For example, consider a project that has three assigned software engineers (e.g., three person-days are available per day of assigned effort). On a given day, seven concurrent tasks must be accomplished. Each task requires 0.50 *person- days* of effort. More *effort* has been allocated than there are people to do the work [2] – [6], [10], [13], [16], [22].

B. The COCOMO Model

The COCOMO (COnstructive COst MOdel) model is the most complete and thoroughly documented model used in effort estimation. The model provides detailed formulae for determining the development time schedule, overall development effort, effort breakdown by phase and activity, as well as maintenance effort [16], [22].

The COCOMO model relies on two assumptions. First, it is linked to the classic waterfall model of software development. Second, good management practice with no slack time are assumed. The model is developed in three versions of different level of detail: *basic, intermediate, and detailed*.

The overall modeling process has three classes of systems:

Embedded. This class of systems is characterized by tight constraints, changing environment, and unfamiliar surroundings. Good examples of embedded systems are real-time software systems (say, in avionics, aerospace, medicine).

Organic. This category includes all the systems that are small relative to project size and team size, and have a stable environment, familiar surroundings, and relaxed interfaces. These are simple business systems, data processing systems, and small libraries.

Semidetached. The software systems under this category are a mix of those of organic and embedded nature. Some examples of software of this class are operating systems, database management systems, and inventory management systems.

C. ISO 9126 Quality Factors

The ISO 9126 standard was developed in an attempt to identify quality attributes for computer software. The standard identifies six key quality attributes [16], [21]:

Functionality: The degree to which the software satisfies the stated needs as indicated by the following sub-attributes: suitability, accuracy, interoperability, compliance and security.

Reliability: The amount of time that the software is available for use as indicated by the following sub-attributes: maturity, fault tolerance, and recoverability.

Usability: The degree to which the software is easy to use as indicated by the following sub-attributes: understandability, learnability, and operability.

Efficiency: The degree to which the software makes optimal use of system resources as indicated by the following sub-attributes: time behavior and resource behavior.

Maintainability: The ease with which repair may be made to software as indicated by the following sub-attributes: analyzability, changeability, stability, and testability.

Portability: The ease with which the software can be moved from one environment to another as indicted by the following sub-attributes: adaptability, installability, conformance and replaceability.

III. RESEARCH APPROACH AND RESULTS

Function Points and the effort in person-months are computed for the CAD software [17]-[18]. The variable ‘a’ from the Fig. 2 to 7 denotes the classification of the software system as follows:

As shown in table 1, the starting value of ‘a’ begins from organic class and it is '0'. The reason is no additional effort required for simple organic systems. For other two classes it is incremented by one and two to differentiate the complexity & constraint level.

TABLE I
VALUES ASSIGNMENT FOR VARIABLE 'a'

System classification (Based on COCOMO model)	Value of 'a'
Embedded system	2 (tight constraints)
Semidetached	1 (both mixed)
Organic	0 (simple)

The CAD software is classified under the semidetached system. So the value of a=1 will be used in the following computations [16].

Factor	*Value*
Backup and recovery	4
Data Communications	2
Distributed processing	0
Performance critical	4
Existing operating environment	3
On-line data entry	4
Input transaction over multiple screens	5
ILFs updated online	3
Information domain values complex	5
Internal processing complex	5
Code designed for reuse	4
Conversion/installation in design	3
Multiple installations	5
Application designed for change	5
Value Adjustment Factor	*1.17*

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x $\Sigma(F_i)$]

$FP_{Estimated}$ = 30 x [1.17] = 35.1 (5)

Equation (5) is obtained from Fig. 2.

The Value Adjustment Factor against the 14 questions is allotted as above [16]. These values are common for all the 6 quality factors since the CAD software alone is considered for applying the research approach [18].

In the Function Point figures 2 to 7, the '*information domain value*' is taken from the ISO 9126 quality sub attributes. . Figure 2 to 7 (6 FP figures) are developed for each of the 6 major quality factors of ISO 9126.

Fig. 2 to 7 refers the function point computations for each of the 6 quality factors in ISO9126. The count value in the fig. from 2 to 7 will be either 1 or 0 to indicate the presence or absence of the attribute respectively.

After calculating the count values, the summation gives the overall FP estimated value. By substituting this FP estimated value in any estimation model, the effort in person-months will be reached. Equation (4) is taken for applying the research approach [16]. Other equations may also be used to apply the FP value.

Information Domain Value	*Count*		*Weighing Factor: Simple*	*Average*	*Complex*		
Suitability	1	x	7+a	**10+a**	15+a	=	11
Accuracy	1	x	5+a	**7+a**	10+a	=	8
Interoperability	1	x	4+a	**5+a**	7+a	=	6
Compliance	1	x	3+a	**4+a**	6+a	=	5
Security	0	x	3+a	**4+a**	6+a	=	0
Count total →							30

Fig. 2. Computing Function Points for the Functionality

Information Domain Value	*Count*		*Weighing Factor* *Simple*	*Average*	*Complex*		
Maturity	1	x	7+a	**10+a**	15+a	=	11
Fault Tolerance	1	x	5+a	**7+a**	10+a	=	8
Recoverability	1	x	**4+a**	5+a	7+a	=	5
Count total						→	24

Fig. 3. Computing Function Points for the Reliability

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x $\Sigma(F_i)$]
$FP_{Estimated}$ = 24 x [1.17] = 28.08 (6)

Equation (6) is obtained from Fig. 3.

Information Domain Value	*Count*		*Weighing Factor* *Simple*	*Average*	*Complex*		
Understandability	1	x	7+a	**10+a**	15+a	=	11
Learnability	1	x	5+a	**7+a**	10+a	=	8
Operability	1	x	4+a	**5+a**	7+a	=	6
Count total						→	25

Fig. 4. Computing Function Points for the Usability

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x $\Sigma(F_i)$]
$FP_{Estimated}$ = 25 x [1.17] = 29.25 (7)

Equation (7) is obtained from Fig. 4.

Information Domain Value	*Count*		*Simple*	*Average*	*Complex*		
			Weighing Factor				
Time Behavior	1	x	7+a	**10+a**	15+a	=	11
Resource Behavior	1	x	**5+a**	7+a	10+a	=	6
Count total						→	17

Fig. 5. Computing Function Points for the Efficiency

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x Σ(F_i)]
$FP_{Estimated}$ = 17 x [1.17] = 19.89 (8)

Equation (8) is obtained from Fig. 5.

Information Domain	*Count*		*Simple*	*Average*	*Complex*		
			Weighing Factor				
Analyzability	1	x	7+a	**10+a**	15+a	=	11
Changeability	1	x	5+a	**7+a**	10+a	=	8
Stability	1	x	4+a	**5+a**	7+a	=	6
Testability	1	x	3+a	**4+a**	6+a	=	5
Count total						→	30

Fig. 6. Computing Function Points for the Maintainability

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x Σ(F_i)]
$FP_{Estimated}$ = 30 x [1.17] = 35.1 (9)

Equation (9) is obtained from Fig. 6.

Information Domain Value	*Count*		*Weighing Factor*				
			Simple	*Average*	*Complex*		
Adaptability	1	x	7+a	**10+a**	15+a	=	11
Installability	1	x	5+a	**7+a**	10+a	=	8
Conformance	1	x	4+a	**5+a**	7+a	=	6
Replacability	1	x	3+a	**4+a**	6+a	=	5
Count total →							30

Fig. 7. Computing Function Points for the Portability

$FP_{Estimated}$ = Count total x [0.65 + 0.01 x $\Sigma(F_i)$]
$FP_{Estimated}$ = 30 x [1.17] = 35.1 (10)
Equation (10) is obtained from Fig. 7.

Therefore, the total $FP_{Estimated}$ is obtained from the summation of (5) to (10) is given by,
= 35.10 + 28.08 + 29.25 + 19.89 + 35.10 + 35.10
= 182.52 FP.

Equation (4)= $E = 0.054 \times FP^{1.353}$
$E = 0.054 \times 182.52^{1.353}$
$E = 0.054 \times 1146.97$
E = 61.93 person-months.

IV. Comparisons of Results

The existing results for the CAD software are as follows:
Based on the FP estimate the estimated effort is *58 person-months.*
Based on the LOC estimate the estimated effort is *54 person-months.*
Based on the Process-Based estimate the estimated effort is *46 person-months.*
Based on the Use-Case estimate the estimated effort is *68 person-months.*
Total estimated effort for the CAD software range from a low of 46 person-months (derived using a process-based estimation approach) to a high of 68 person-months (derived with use-case estimation). The average estimate (using all four approaches) is 56 person-months.

Based on the *proposing estimate the estimated effort is 62 person-months* [9].

V. Conclusion and Future Scope

Based on the above results, the proposed 62 person-months of effort is nearer value to the average result of other estimation models. And hence this type of estimation may be recommended for the software development.

The unique difference between the proposed and existing estimation of effort for the software system development is *the level of quality consideration*.

That is, existing estimations are using only few quality factors for effort estimation, but the proposed effort estimation covers the ISO9126 quality factors, which automatically reflects in the development of software.

Other metrics may be used to estimate the effort and substituting other quality factors can be explored as a future scope [8].

REFERENCES

[1] Tridas Mukhopadhyay and Sunder Kekre, “Software Effort Models for Early Estimation of Process Control Applications”, IEEE Trans. software Eng., vol. 18, no. 10, pp. 915–924, Oct 1992.

[2] Charles R. Symons, “Function Point Analysis: Difficulties and Improvements”, IEEE Trans. software Eng., vol. 14, no. 1, pp. 2-11, Jan 1988.

[3] Allan J. Albrecht and John E. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”, IEEE Trans. software Eng., vol. SE-9, no. 6, pp. 639-648, Nov 1983.

[4] D. R. Jeffery, G.C. Low and M. Barnes, “A Comparison of Function Point Counting Techniques”, IEEE Trans. software Eng., vol. 19, no. 5, pp. 529-532, May 1993.

[5] Raimo Rask, Petteri Laamanen and KalleLyytinen, “Simulation and Comparison of Albrecht’s Function Point and DeMarco’s Function Bang Metrics in a CASE Environment”, IEEE Trans. software Eng., vol. 19, no. 7, pp. 661-671, July 1993.

[6] Graham C. Low and D. Ross Jeffery, “Function Points in the Estimation and Evaluation of the Software Process”, IEEE Trans. software Eng., vol. 16, no. 1, pp. 64-71, Jan 1990.

[7] Magne Jorgensen and Martin Shepperd, “A Systematic Review of Software Development Cost Estimation Studies”, IEEE Trans. software Eng., vol. 33, no. 1, pp. 33-53, Jan 2007.

[8] Lionel C. Briand and Jurgen Wust, “Modeling Development Effort in Object-Oriented Systems Using Design Properties”, IEEE Trans. software Eng., vol. 27, no. 11, pp. 963-986, Nov 2001.

[9] June Verner and Graham Tate, “Estimating Size and Effort in Fourth-Generation Development”, IEEE Software, pp. 15-22, July 1988.

[10] Rajiv D. Banker, Robert J. Kauffman, Charles Wright and Dani Zweig, “Automating Output Size and Reuse Metrics in a Repository-Based Computer-Aided Software Engineering (CASE) Environment”, IEEE Trans. software Eng., vol. 20, no. 3, pp. 169-187, March 1994.

[11] Nader B.Ebrahimi, “How to Improve the Calibration of Cost Models”, IEEE Trans. software Eng., vol. 25, no. 1, pp. 136-140, Jan/Feb 1999.

[12] Chris F. Kemerer and Benjamin S. Porter, “Improving the Reliability of Function Point Measurement: An Empirical Study”, IEEE Trans. software Eng., vol. 18, no. 11, pp. 1011-1024, Nov 1992.

[13] Alain Abran and Pierre N. Robillard, “Function Points Analysis: An Empirical Study of Its Measurement Processes”, IEEE Trans. software Eng., vol. 22, no. 12, pp. 895-910, Dec 1996.

[14] Hoang Pham and Xuemei Zhang, “A Software Cost Model with Warranty and Risk Costs”, IEEE Trans. software Eng., vol. 48, no. 1, pp. 71-75, Jan 1999.

[15] Randy K. Smith, Joanne E. Hale and Allen S. Parrish, “An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation”, IEEE Trans. software Eng., vol. 27, no. 3, pp. 264-271, March 2001.

[16] Roger S. Pressman, “Software Engineering: A Practitioner’s Approach”, Mc Graw Hill International Edn., 6th Edn., 2005, pp. 464, 472, 653-710.

[17] RTI Project no. 7007.011, “The Economic Impacts of Inadequate Infrastructure for Software Testing”, for National Institute of Standards and Technology, pp. 1-309, May 2002.

[18] Martin Horwood, “CAD Data Quality”, for Infosys Technologies, pp. 14-16, May/June 2005.

[19] “IEEE Editorial Style Manual”, pp. 1-18, 4th edition.

[20] “Preparation of Papers for IEEE Transactions and Journals”, pp. 1-6, May 2007.

[21] Norman E. Fenton and Shari Lawrence Pfleeger, “Software Metrics: A Rigorous Practical Approach”, Thomson Asia Pvt. Ltd., Singapore, 2nd Edn., 2005, pp. 28, 343-344, 551-556.

[22] Hans Van Vliet, “Software Engineering: Principles and Practice”, John Wiley & Sons Ltd., 2nd Edn., 2004, pp. 158-160, 172-176.

[23] Yunsik Ahn, Jungseok Suh, Seungryeol, Kim and Hyunsoo Kim, “The Software maintenance project effort estimation model based on function points”, Journal of Software Maintenance and Evolution, 2006.

[24] Benediktsson, O., D. Dalcher, and K. Reed, “COCOMO-Based Effort Estimation for Iterative and Incremental Software Development”, Software Quality Journal., 2003, 11(4): pp. 265-281.

[25] Carmel, E., “Time-to-Completion Factors in Packaged Software Development”, Information and Software Technology, 1995, 37(9): pp. 515-520.

[26] Boehm, B., et al., “The COCOMO 2.0 Software Cost Estimation Model: A Status Report”, American Programmer, 1996, 9(7): pp. 2-17.

[27] Boehm, B., C. Abts, and S. Chulani, “Software Development Cost Estimation Approaches – A Survey”, Annals of Software Engineering, 2000, 10: pp. 177-205.

[28] Carson, C., “Using the TQC Problem-Solving Process to Develop an Improved Estimation Technique”, Texas Instruments Technical Journal, 1996, 13(6): pp. 101-106.

[29] Chrysler, E., “Some Basic Determinants of Computer Programming Productivity”, Communications of the ACM, 1978, 21(6): pp. 472-483.

[30] Edwards, J.S. and T.T. Moores, “A Conflict between the Use of Estimating and Planning Tools in the Management of Information Systems”, European Journal of Information Systems, 1994.

[31] Cuelenaere, A.M.E., M.J.I.M. Genuchten, and F.J. Heemstra, “Calibrating a Software Cost Estimation Model: Why and How” Information and Software Technology, 1987, 29(10): pp. 558-567.

H. Azath received the M.Tech degree in information technology from Punjabi University, Patiala, Punjab, India in 2004 and B.E. degree in computer science and engineering from Government College of Engineering, Tirunelveli, Tamil Nadu, India in 2001.

He is an ASSISTANT PROFESSOR & HEAD of Information Technology department of King College of Technology, Namakkal, Tamil Nadu, India.

The areas of interest are Software Engineering & Operating Systems. He became a Member of ISTE in 2005.

R. S. D. Wahidabanu received the Ph.D degree in application of neural networks from Anna University, Chennai, Tamil Nadu, India in 1998 and M.E degree in applied electronics in 1985 and B.E. degree in electronics and communication engineering in 1981 from Government College of Technology, Coimbatore, Tamil Nadu, India.

She is a PROFESSOR & HEAD of Electronics and Communication Engineering Department of Government College of Engineering, Salem, Tamil Nadu, India.

Dr. Wahidabanu became members of MISTE, MIE, MSSI, MCSI, in 2001 and MISOC in 2008.