# Comparision Study on methods of software size measure

Nitin M Shivale[1], Lalit V Patil[2],S D Joshi[3]

[1]*Department of Information Technology, University of Pune.* [3]*Bharti Vidyapeeth Deemed University*
[1]*STES's Smt. Kashibai Naval College of engineering Pune-41,*

[3]*Bharti Vidyapeeth Deemed University College of Engg, Dhankawadi, Pune – 43*
[1]nitinrajni3@gmail.com, [3]sdj@live.in

[2]*STES's Smt. Kashibai Naval College of engineering Pune-41,*
[2]lalitvpatil@gmail.com

*Abstract*—**In the component based software a development cost is low due to the integration of a component with each other's. The black-box nature of the components is improper for the traditional size measure that leads to the inaccuracy in the component based software estimation. Software size measure is a most important task in the project management such as planning and estimation. Over the time component attributes and relationships could change and differ for development environments that also create a problem in the accuracy of estimation. In this paper we compair various size measure methods. The accurate measuring the system level size in the component based software development generates good prediction of the cost estimation. Accurately measuring the component size method used the unified modelling language specification. We also suggest the importance of size measure by comparing the various size measure methods along with cost estimation models. At last considering a case study, the feasibility of the size measure method for component based system is proved.**

***Keywords*— Component Based Software Developmet (CBSD), Component Point (CP), Function Point (FP).**

## I. INTRODUCTION

An important factor in selecting a cost estimation model is the accuracy of its estimation result and that accuracy comes through the correct measure of the software size. Many estimation models have been proposed over the last 30 years and may be classified into two major categories: algorithmic and non-algorithmic [2]. Each has its own strength and weaknesses. Now a day's models is based on Generic algorithm, neural network simulation, soft computing, Fuzzy Logic modelling etc. Here we propose a new component fuzzy Cocomo-II (CF-Cocomo- II) model to improve the accuracy of CBSD cost estimation.

This paper is organized as follows: section II describes the background and related work with comparisons of various size measure methods results along with Cocomo model [1]. Section III describes proposed system with mathematical models and results. Section IV describes conclusion and finally future work.

## II. SOFTWARE DEVELOPMENT SIZE MEASURE METHODS

As the technology grows up in a rapid way, software has become the most crucial part of the computer system project. Accurate cost estimation is very much important to both developers and customers. It can help to classify and prioritize development projects with respect to an overall business plan. Many practitioners have struggled with three fundamental issues [3].

1. Which software cost estimation model to use for the accurate estimation?
2. Which software size measurement to use for this cost estimation model?
3. How we can say the estimation is good?

To answer the above different questions various estimation and size measurement methods have been established.

### *A. Traditional Method*

The LOC is the main size measures used in the traditional methods for whole software system. LOC means the number of lines of the delivered source code of the software, excluding comments & blank lines [4]. A typical method for estimating the code size is to use 'Expert Judgement' together with a technique called PERT [5].

$$S = \frac{Sl + Sh + 4Sm}{6}$$

Where,

- Sl = Lowest Possible Size
- Sh = highest Possible Size
- Sm = Most Likely Size

The general effort model, applicable to all application levels and modes is given by [2]. Representing a monolithic approach to cost estimation

$$E = A\,(EDSI)^{B} \times (EAF)$$

Where,

- E = an effort estimation expressed in man-month
- EDSI = Estimate of Delivered Source Instruction
- A, B = Constants [2]
- EAF = Effort adjustment factor

Example:-

An example of Human resource application using the Basic Cocomo model to estimate the effort required in developing an 8500 line program in the organic mode assuming EAF is nominal I: e 1 is given below.

$E = 3.2 (8500)^{1.05} \times (1)$

$= 30$ MM (1)

Pros:

- Expert judges the system hence fast estimation.
- An expert with the relevant experience can provide good estimation.

Cons:

- Dependent upon the expert.
- An early phase of software life cycle inapplicable to estimating the code size.
- Single Effort adjustment factor applies on the whole system.

*B. Component Based approach with SLOC.*

Today development practices characterized a software system as an independent component. These components may be internally developed reusable components, Commercial-Off-The-Shelf (COTS) component or newly developed software artifacts [14]. To accurately predict effort in CBSD a fine grained approach is needed to identify and classify the relevant cost factors. Bohem uses the intermediate Cocomo model to cost to individual components with 15 cost factors.

Example: Consider the Human Resource Application 8500 line project made up of the Component listed in Table I.

TABLE I
NOMINAL COMPONENT MAN-MONTH FOR COMPONENT

| Component Name | EDSI | % of Total | CMMnom |
|---|---|---|---|
| Employee | 2000 | 23.4 % | 7.06 |
| Job | 3000 | 35.3 % | 10.60 |
| Assignment | 3500 | 41.2 % | 12.36 |

Based on the 30 MM computed in equation (1) for E, the expected number of EDSI per man-month is given by:

(EDSI/MM) nom = 8500/30

= 283 EDSI/MM

Using the EDSI/MM, every component is then apportioned its contribution to the total. For example, the nominal component man-month (CMMnom) for the employee component is given by:

CMMnom = EDSIper comp / (EDSI/MM) nom

= 2000/283

= 7.06 CMMnom.

After computing CMMnom for each component, the effort adjustment factor (EAF) is calculated individually for each component. Thus we are able to account for variance among the cost factors for the different component. For example, the CMMadj for the Job Component is calculated by:

CMMadj = (CMMnom) x (EAF)

= (10.60) x (1.13)

= 11.98 CMMadj

TABLE II
COMPONENT COCOMO WITH EFFORT ADJUSTMENT

| Component Name | EDSI | % of Total | CMM nom | EAF | CMMadj |
|---|---|---|---|---|---|
| Employee | 2000 | 23.4 % | 7.06 | 0.89 | 6.28 |
| Job | 3000 | 35.3 % | 10.60 | 1.13 | 11.98 |
| Assignment | 3500 | 41.2 % | 12.36 | 1.05 | 12.98 |

By factoring the components separately through adjusting component man- month in Table II, a new estimate of 31.24 MM is found [11].

Pros:

- Software size and many other cost factors for estimating are considered.
- CBSE embodies the "buy, don't build" philosophy.
- Reuse, flexibility, Easy to maintain.

Cons:

- Unable to capture all of the factors that could potentially impact component based development ([8], [9]).
- Hardly to build the environment that is fitted to the component.
- Due to the Black-box nature of component accuracy of size measure is less.

*C. Component Based approach with Function Point*

Albrecht proposed function point for measuring the size of the procedural business system from the end user's point of view. The total number of function points depends on the counts of (in terms of format or processing logic) types in the five classes and 14 general system characteristics (GSC) as explained in [10]. Some of the terms used in function point namely Unadjusted Function Point (UFP), Total Degree of Influence (TDI), Value Adjustment Factor (VAF), and Function Point (FP).

$$VAF = 0.65 + (0.01 \times TDI)$$

$$FP = UFP \times VAF$$

Where TDI = Sum of 14 GSC is on scale 0-5.

UFP = Sum of all the complexities of five classes.

Example: Consider the Human Resource Application Project and computing the function point for the accurate size of the project to the estimation.

1. Computing the complexity of all five classes namely External Interface Files (EIFs), Internal Logical Files (ILFs), External Inputs (EI's), External Outputs (EO's) and External Queries as described in [7].
2. Secondly computing the Unadjusted Function Point count (UFP) as shown in Table III.

TABLE III
COMPUTING UNADJUSTED FUNCTION POINT (UFP) [7]

| Function Type | Function Complexity | Count | Weight | FP | FP % |
|---|---|---|---|---|---|
| Internal Logical Files (ILFs) | Low | 0 | 7 | 0 | 26 % |
| | Average | 3 | 10 | 30 | |
| | High | 0 | 15 | 0 | |
| External Interface Files (EIFs) | Low | 0 | 5 | 0 | 12 % |
| | Average | 2 | 7 | 14 | |
| | High | 0 | 10 | 0 | |
| External Inputs (EI's) | Low | 0 | 3 | 0 | 31 % |
| | Average | 9 | 4 | 36 | |
| | High | 0 | 6 | 0 | |
| External Outputs (EO's) | Low | 0 | 4 | 0 | 13 % |
| | Average | 3 | 5 | 15 | |
| | High | 0 | 7 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| External Queries (EQ's) | Low | 0 | 3 | 0 | 17 % |
| | Average | 5 | 4 | 20 | |
| | High | 0 | 6 | 0 | |
| Total Unadjusted Function Point Count (TUFPC) | | | | 115 | 100 |

3. Computing the value adjustment factor (VAF) in [7].

$$VAF = (0.65 + (0.01 \times 23))$$
$$= 0.88$$
$$FP = 115 \times 0.88$$
$$FP = 101$$

Now if the application is implemented in the 'C' & Pascal Language then the Size Measure (SM) as follows:

SM(C) = (101 x 125), SM (Pascal) = (101 x 85)
= 12625 = 8585
= (12.62) KLOC = (8.5) KLOC

4. Using the Bohem Cocomo-II Model for effort estimation and considering all the 17 effort multiplier factors EAF = 0.184295 in [12].

$$E (C) = 3.2 (12.62)^{1.05} \times 0.184295$$
$$= 45.84 \times 0.184295$$
$$= (8.44) MM$$

$$E (Pascal) = 3.2 (8.5)^{1.05} \times 0.184295$$
$$= 30.27 \times 0.184295$$
$$= (5.57) MM$$

Pros:

- A normalization factors for software.
- Applicability in the early phase of the software life cycle.

Cons:

- FPA criticizes as not being universally applicable to all types of software [13]. FPA doesn't capture all functional characteristics of real time software.
- FPA doesn't support the object oriented and component based software system size measures effectively.

### *D. Comparisons of Size Measure Methods*

The Table IV shows the result of all the methods for the cost estimation. Analysing this we are able to understand that which size measurement method gives us the accurate size for the estimation. As per the rule of IFPUG 1FP = 125 lines of code in 'C' Language and 1FP = 85 lines of code in PASCAL Language [6].

TABLE IV
RESULT OF SIZE MEASUREMENT METHODS

| **Method** | **Size Type** | **Language** | **Size** | **Estimate Model** | **Effort MM** |
|---|---|---|---|---|---|
| Traditional Method | SLOC | C | 8500 | Basic Cocomo | 30 |
| CBSD Method | SLOC | C | 8500 | Intermed iate Cocomo | 31.24 |
| CBSD Method | FPA | PASCAL | 101 | Cocomo II | 5.57 |
| | | C | 101 | Cocomo II | 8.44 |

### *E. Need of size measure methodology in CBSD*

As we have known the importance of size in effort estimation the above Table IV shows that the importance of accurate size measurement very clearly. Therefore it is quite clear that the effort estimation of CBSD is proportional to the size of the CBSS. Hence we required a size measure methodology which more precisely gives us the exact size of component based systems [15]. Which help us in the accurate estimation for building the software on time in the competitive market.

## III. CASE STUDY ON COMPONENET-BASED SYSTEM SIZE MEASURE

To overcome and address the problem discussed in need of size measure methodology in CBSD. A Case study of Global Positioning System with Component Point for correct size measure is considered as shown in Fig. 1. [17].

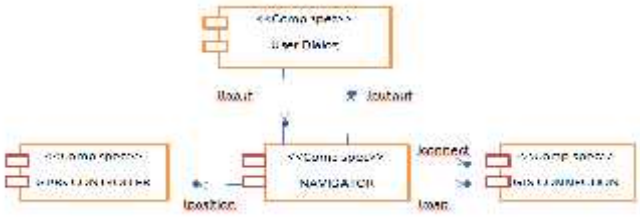

Fig. 1 GPS Component Specification

### *A. Component Point*

The idea behind CBSD is to reduce in house development with minimum bugs thereby minimizing the reload time and risk. Due to the black box nature of component the size and development effort of CBSS have been very much dependent on the number of components integrated. Whereas size is proportional to the number of components integrated. The Component Point Process as shown in Fig. 2.

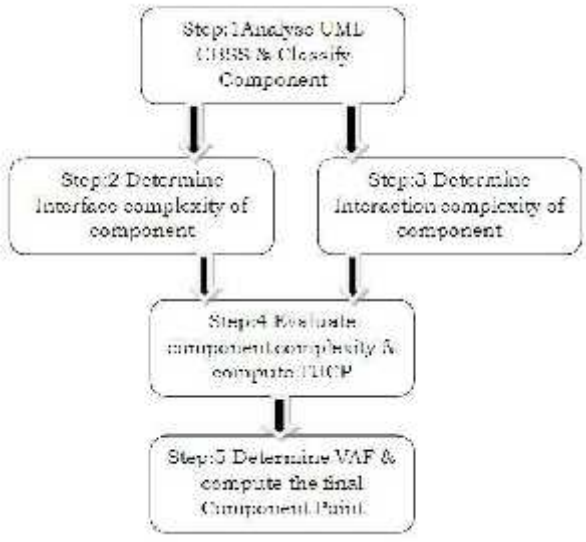


Fig. 2 Component Point Estimation Process

$$TUCP = \sum_{i=1}^{3} \sum_{j=1}^{3} Cij \times Wij$$

Where,

Cij = the number of Component of type i with complexity Level j

Wij = the weight given for the component type i with Complexity level j

TUCP = Total Unadjusted Component Point

VAF = 0.65 + (0.01 x TDI)

The final CP count is then computed as follows equation.

CP = TUCP x VAF

*B. Component Complexity of GPS [17].*

The Table V shows the Component complexity.

TABLE V
COMPONENT COMPLEXITY

| **Compnent** | **Component Type** | **IFCI** | **ITCI** | **Component Complexity Level** |
|---|---|---|---|---|
| User Dialog | User (UC) | 7 | 0.420 | Low |
| Navigator | Domain (DC) | 7 | 0.012 | Low |
| GPRS Controller | Service (SC) | 7 | 1.667 | Average |
| GIS Connection | Service (SC) | 6 | 2.300 | Average |

The Table VI shows the Unadjusted Component Point Count.

TABLE VI
UCP COUNT FOR THE GPS [17].

| **Compnent Type** | **Component Complexity Level** | | | **Total** |
|---|---|---|---|---|
| | **Low** | **Avg** | **High** | |
| DC | 1 x 3 = 3 | .. x 6 =.. | ..x 10 = .. | 3 |
| UC | 1 x 4 = 4 | .. x 7 = .. | ..x 12 = .. | 4 |
| SC | .. x 4.5 = .. | 2 x 7=14 | ..x 11.5 =. | 14 |
| UCP | | | | 21 |

The Table VII shows the Toal Degree of Influence.

TABLE VII
INFLUENCE OF GSC [17].

| **Sr.No** | **Characteristics** | **Influence** |
|---|---|---|
| 1. | Data and Online Services | 5 |
| 2. | Distributed Processing | 2 |
| 3. | System/Component Performance | 4 |
| 4. | Development Rigidity | 3 |
| 5. | User Friendliness | 4 |
| 6. | System Complexity | 3 |
| 7. | Installability | 1 |
| 8. | Operability | 3 |
| 9. | Maintainability | 2 |
| 10. | Multi-Site Use | 0 |
| 11. | System/Component Relaibility | 4 |
| 12. | System/Component Portability | 3 |
| 13. | Component immaturity | 4 |
| 14. | Lack of Component under support | 3 |
| TDI | | 41 |

*C. Result and Evaluation Method*

As an application given in [17] provides the

VAF = 0.65 + (0.01 x 41)

= 1.06

CP = TUCP x VAF

= 21 x 1.06

= 22.26 for the global positioning system development in CBSD. Applying the proposed methodology to it the results we get as follows.

$$EFFORTpm = 2.94 \times [22.26]^{E} \times \prod_{i=1}^{17} EMi$$

$$E = 0.91 + 0.01 \times \sum_{j=1}^{5} SFj$$

E = 1.097 and $\prod EM = 0.172459$

EFFORTpm = 15.2463758622

SCHEDULEmonths = 2.1455

AVG STAFFpeople = 7.1062

## IV. CONCLUSIONS & FUTURE WORK

There are so many models to estimate the results. Every size measure methods has it's own pros and cons. We have focused mainly the size measure for CBSD. Due to the black box nature of component traditional methods fails to capture the accurate size for estimation. Size measure is the important task in effort estimation. More accurate the size more accurate the effort estimation result.

COCOMO Post architecture model uses seventeen cost drivers to adjust the final result. Providing the accurate size to the model produces more accurate result.

## REFERENCES


[1] B. W. Boehm, *Software engineering Economics*, Englewood ciliffs. N. J: Prentice- Hall, 1981.

[2] C. E. Walston and C. P. Felix, "*A method of proposing measurement and estimation"*, IBM system journals, vol. 16, no. 1, pp, 55-73, 1977.

[3] F. J. Heemstra, "*Software Cost Estimation*", Information and software Technology, vol.34, no.10, pp. 627-639, 1992.

[4] N. E. Fenton and S. L. Pfleeger, *Software metrics: A rigorous and practical Approach*,PWS publishing company. 1997.

[5] J. D. Aron, *Estimating Resource for Large Programming Systems*, *NATO science* Committe, Roam, Italy, octomber 1969.

[6] IFPUG: International function Point Group, http://www.ifpug.org.

[7] Scribd: Digital library content, http://www.scribd.com/fp

[8] Leonhard C. Davis, J., "*Job-shop Development model: A Case study*", IEEE software, vol. 12, no. 2, PP.86-92, March, 1995.

[9] McConnell S., "*Less is More*", Software Development, vol. 5, no. 10, pp. 28-34, October, 1997.

[10] A. J. Albrecht, J. E. Gaffney, "*Software Function, source lines of codes, and development effort prediction: a software science validation*", IEEE Trans software Engg, SE. 9, pp. 639-648, 1983.

[11] Randy k Smith, A. Parrish, J. Hale, "*Cost estimation for Component based software development*", ACM 1-58113030-9/98/0004, 1998.

[12] T. N. Sharma, "*Analysis of software cost estimation using COCOMO-II*", IJSER, vol. 2, issue 6, June- 2011.

[13] Abran. A, Robillard. P. N., "*Function and points: a study of their measurement processes and scale transformations*", journal of system and software 25(2), 171-184.

[14] T. wijayasirivardhane, R. Lai, K. C. Kang, "*Effort estimation of Component-based software development- a survey*", IET software, vol. 5, iss. 2, pp. 216-228, 2011.

[15] Verner. J., Tate. G, "*A software size model*", IEEE Transaction on software engineering 18(4), 265-278, 1992.

[16] Iman Attarzadeh and seiw Howk ow, "*Improving Estimation Accuracy of the COCOMO II using an adaptive fuzzy logic Model*", IEEE International Conference on fuzzy systems, Taipei, Taiwan, June 27-30, 2011.

[17] T. Wijayasirivardhane, R. Lai, "*Component point: A System level size measure for component- based software Systems*." The journal of systems and software 83(2010) 2456-2470.

 