IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 8, August 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

# Software Sizing with Use case point

**Sangeetha K [1], Prof. Pankaj Dalal[2]**

[1] M Tech Scholar (Software Engineering), Department of Computer Engineering, Shrinathji Institute of Technology & Engineering, Nathdwara-313301, India

[2] Associate Professor, Department of Computer Engineering, Shrinathji Institute of Technology & Engineering, Nathdwara-313301, India

## Abstract

Software sizing is one of the critical activities in software project management. Effective software sizing help project managers to manage and control software projects successfully. In software industry, the technology is growing rapidly. It is impractical to change the measurement rules every time in accordance with the technology. Today's software applications use different technologies, tools and several programming languages. Hence, software systems continue to grow in size and complexity. Now it became very difficult to assess, and manage software products. There are so many methods to measure, and communicate size and productivity in the industry for effective project management. Use case point is one of such method in this category. It is a well-documented approach for software sizing in accordance with the use case diagram of the software. By using UCP method, we are able to produce a reliable estimate very early in the development cycle. This paper shows how effective will be the UCP method in software sizing.

Keywords: Software sizing, UCP (Use case Point), TCF

(Technical Complexity Factor), ECF (Environmental Complexity Factor), UAW (Unadjusted Actor Weight), UUCW (Unadjusted Use Case Weight)

## I. Introduction

Software sizing is a critical activity for planning and monitoring software project development focusing on time and budget [13]. Today's software applications are very heterogeneous in nature and use different technologies, tools, and several programming languages [2]. It became very difficult to understand and assess the software products. Software size is extremely important in determining and controlling costs, schedules, quality and productivity in software project management.

Software applications grow and become more complex. Therefore a common method needs to be used and established in the industry to understand, measure and communicate size and productivity. Software sizing is one of the challenging, and critical activities in the software development process. Effective software sizing is necessary for successful completion accordance with the budget and time.
In practice, it is tedious process for software professionals to measure the software size methodically.
This paper provides an insight view of Use Case Point Analysis and its benefits in software sizing. Hence, helps in estimating productivity and cost.

## II. Related work

Different studies have been published during the last 30 years comparing modelling techniques for software cost estimation.

Prior to 1970, effort estimation was done manually by using Thumb rules or some algorithms which were based on Trial and error [1]. During early 1970's the first automated software estimation prototyping composite model COCOMO (Constructive Cost Model) had developed by Barry Boehm [1].

Function Point Analysis (FPA) for estimating the size and development effort had been developed in 1975. 1984, IBM had done a major revision of his function point metric which is basis of today's function points [1]. 1993, the new version of COCOMO was introduced called COCOMO 2.0 which emerged in 1994 [4].

In the mid-1990s, Jim Rumbaugh, Grady Booch and Ivar Jacobson of Rational Software Corporation developed the Unified Modelling Language (UML) as notation and methodology for developing object oriented software. At the same time, Gustav Karner, from Rational Software Corporation, created a software project estimating method based on Use Case Points, with the same way that FPA assigns points to functions, and included statistical weighted modifiers[8][9][11]. Karner's technique is incorporated into RUP. Use Cases, as defined by UML, describes what the actors want the system to do and have proven to be an easy method to capture

the scope of a project early in the project lifecycle. For our use, we got the opportunity to create estimates early in the project lifecycle. However, like COCOMO and FPA, the accuracy of estimates created using the UCP estimating technique is largely dependent upon a large amount of relevant historical metrics. UCP calculation requires the use case diagram and use case descriptions. This description contains the steps of how a use case is executed. These steps are known as term 'transaction' of use case.

Researchers have worked on use case point method and Effort Estimation based on use case model.

Arnold et.al [3] proposed that the use of the use case point method is accepted to estimate the size. They compared the object oriented methods like OOSE, OMT, OOAD, UML, ROOM, OBA, and Syntropy.

Smith J proposed a framework to estimate LOC from use case diagram [12]. The framework takes input as use case level, size and complexity, for different categories of system and does not resort to fine-grain functional decomposition.

Periyasamy and Aditi Ghode focus on the internal details of use cases. The relationship between the entities in a use case diagram is used for internal details, and hence closely estimates the size of the software product to be developed. The authors explained the original UCP model with additional information obtained from use case narratives also changes some parameters value[10].

Hasan O et.al reports the development of new linear use case based software cost estimation model applicable in the very early stages of software development being based on simple metric. Results showed that accuracy of estimates varies between 43% and 55% of actual effort of historical test projects. These values outperformed those of well-known models when applied in the same context. This work is extending to improve the performance of the proposed model when considering the effect of non-functional requirements [7].

## III. Software Sizing

Software sizing is an activity in software engineering that is used to estimate the size of a software application or component in order to aid in other software project management activities such as estimating or tracking. Estimating is the process of forecasting or approximating the time and cost of completing project.

### Use case point method

UCP was first proposed by Gustav Karner in 1993 that developed from Function Point Analysis for object-oriented applications [5] [6]. UCP calculation process requires the use case diagram and use case descriptions. It is a well-documented approach for estimating software development activities. It is based on the same principles of Function Point estimation. It has turned out that Use Case Point (UCP) estimation is as reliable as Function Point estimation. It provides the ability to estimate the productivity of a software project from its use cases. By using UCP estimation we are able to produce a reliable estimate very early in the development cycle

The use case point of an application is calculated as follows [8]:

**Step 1: Calculating unadjusted actor weights (UAW).**The actors of each use case are categorized into simple, medium, or complex. Each actor is put into the groups with the criteria as presented in Table1

UAW is obtained as the sum of the weights of each actor

$$\mathbf{UAW} = \sum_{i=1}^{n} \mathbf{ActorWeight}_i \qquad \text{Eqn 1}$$

Where, n = number of actor

Actor Weight = weight of each actor category

TABLE 1: CLASSIFICATION OF ACTOR

| Category of Actor | Description | Weight |
|---|---|---|
| Simple | System through API | 1 |
| Average | System through Communication protocol | 2 |
| Complex | Human actor through GUI | 3 |

**Step 2: Calculating unadjusted use case weights (UUCW).** The UUCW expressed the use case complexity that is measured by the number of transactions in a use case. Like the actors, each use case in the system are grouped into simple, medium, or complex according to the criteria listed in Table 2

The UUCW is obtained by adding the weight of each use case using Eqn 2.

$$\mathbf{UUCW} = \sum_{i=1}^{n} \mathbf{UsecaseWeight}_i \qquad \text{Eqn 2}$$

Where, n = number of use case

Use Case Weight = weight of each use case category

TABLE 2: CLASSIFICATION OF USE CASE

| Use case Category | Description | Weight |
|---|---|---|
| Simple | Up to 3 transactions | 5 |
| Average | 4-7 transactions | 10 |
| Complex | More than 7 transactions | 15 |

**Step 3: Calculating unadjusted use case points (UUCP).** The UUCP is obtained from the sum of the Unadjusted Use Case Weights (UUCW) and Unadjusted Actor Weights (UAW) as in Eqn. 3

**UUCP = UUCW + UAW** **Eqn 3**

**Step 4: Calculating technical complexity factor (TCF).** The TCF is one of the main things needed to estimate the size of software in order to take into account the technical considerations of the system. According to Roy K Clemmons, thirteen standard technical factors exist to estimate technical complexity on a project[14]. This is calculated by assigning a score between 0 (factor not relevant) to 5 (important factor) for each of the thirteen technical factors listed in Table 3.

TABLE 3: TECHNICAL FACTOR WEIGHT [14]

| Ti | Technical Factor | Multiplier |
|---|---|---|
| 1 | Distributed System Required | 2 |
| 2 | Response Time Is Important | 1 |
| 3 | End User Efficiency | 1 |
| 4 | Complex Internal Processing Required | 1 |
| 5 | Reusable Code Must Be A Focus | 1 |
| 6 | Installation Ease | 0.5 |
| 7 | Usability | 0.5 |
| 8 | Cross-Platform Support | 2 |
| 9 | Easy To Change | 1 |
| 10 | Highly Concurrent | 1 |
| 11 | Custom Security | 1 |
| 12 | Dependence On Third-Party Code | 1 |
| 13 | User Training | 1 |

This score is multiplied by the weighted value assigned to each factor. The Technical Factor (TF) is obtained from sum of multiplying score and weight as shown below.

$$\mathbf{TF = \sum_{i=1}^{13} Score * Weight} \quad \textbf{Eqn 4}$$

Technical Complexity Factor (TCF), is calculated from TF using the Eqn 5 [8]

**TCF = 0.6 + (0.01 * TF)** Eqn 5

**Step 5: Calculating environmental complexity factor (ECF).** The ECF is another factor that applied to estimate software size by counting environmental considerations of the system. It is calculated by assigning a score between 0 (no experience) to 5 (expert) for each 8 environmental factors that are listed in the Table 4.

TABLE 4: ENVIRONMENTAL FACTOR WEIGHT

| $E_i$ | Environmental Factor | Weight |
|---|---|---|
| 1 | Familiarity With The Project | 1.5 |
| 2 | Application Experience | 0.5 |
| 3 | OO Programming Experience | 1 |
| 4 | Lead Analyst Capability | 0.5 |
| 5 | Motivation | 1 |
| 6 | Stable Requirements | 2 |
| 7 | Part Time Staff | -1 |
| 8 | Difficult Programming Language | -1 |

Similar to the calculation of the TCF, this score is multiplied by weighted value of each factor. The Environmental Factor (EF) is obtained by adding the product of score and weight as in Eqn. 6. Furthermore, this value will be used to obtain Environmental Complexity Factor (ECF). It given the formula for the calculation of the ECF is Eqn 7 [8].

$$\mathbf{EF = \sum_{i=1}^{8} Score * Weight} \quad \textbf{Eqn 6}$$

**ECF = 1.4 + (-0.03 * EF)** **Eqn 7**

**Step 6: Calculating the use case point (UCP).** The UCP is obtained by multiplying unadjusted use cases points and technical complexity factors and the environmental complexity factor according to the Eqn 8 [8].

**UCP = UUCP * TCF * ECF** **Eqn 8**

**Step 7: Final step in UCP method is calculating Effort**. Effort value is obtained by multiplying the value of UCP and constant ER in staff-hours/UCP (Eqn 9). Researchers typically use a value of ER = 20 staff-hours/UCP as proposed by Karner [9].

**Effort = UCP * ER** **Eqn 9**

## IV. Data Set

In our work we have studied ten different systems along with their technical requirement as well as various environmental factors, the development

teams' experience and knowledge. From the use case diagram of these systems, we have estimated the actor weight as well as use case weight.

By using this unadjusted actor weight and use case weight along with different technical and environmental factors of the system the use case point count were calculated. These are shown in the table 5:

TABLE 5: USE CASE POINT MATRIX

| Sr. No | Proj ID | UCP | | | | |
|---|---|---|---|---|---|---|
| | | UUCW | UAW | TCF | ECF | Count |
| 1 | A | 75 | 6 | 0.875 | 0.89 | 63.08 |
| 2 | B | 60 | 6 | 1.05 | 0.92 | 66.65 |
| 3 | C | 90 | 13 | 1.05 | 0.935 | 101.1 |
| 4 | D | 75 | 7 | 1.04 | 0.935 | 79.7 |
| 5 | E | 100 | 9 | 0.98 | 0.92 | 98.3 |
| 6 | F | 55 | 6 | 1.095 | 0.875 | 58.4 |
| 7 | G | 60 | 9 | 0.97 | 0.965 | 64.59 |
| 8 | H | 95 | 6 | 1.04 | 0.965 | 101.4 |
| 9 | I | 65 | 5 | 0.98 | 0.89 | 61.05 |
| 10 | J | 90 | 12 | 0.965 | 0.935 | 92.03 |

From this data we have calculated the development effort for different systems, which is shown in the table 6

TABLE 6: DEVELOPMENT EFFORT MATRIX

| Sr. No | System | UCP | |
|---|---|---|---|
| | | System Size (Count) | Effort (Person-months) |
| 1 | A | 63.08 | 7.01 |
| 2 | B | 66.65 | 7.41 |
| 3 | C | 101.1 | 11.23 |
| 4 | D | 79.7 | 8.86 |
| 5 | E | 98.30 | 10.92 |
| 6 | F | 57.90 | 6.43 |
| 7 | G | 64.59 | 7.18 |
| 8 | H | 101.10 | 11.23 |
| 9 | I | 61.05 | 6.78 |
| 10 | J | 92.03 | 10.23 |

## V. Effect of system characteristics on the estimation of size

We have analysed the effect of the different technical and environmental characteristics on the estimation of size.

In use case point analysis, 13 technical factors and 8 environmental factors are considering for adjusting the unadjusted UCP to UCP.

The result of the correlation analysis performed on unadjusted UCP and Adjusted UCP is given in the table 7:

TABLE 7: CORRELATION BETWEEN UUCP & UCP

| | *UUCP* | *UCP* |
|---|---|---|
| UUCP | 1 | |
| UCP | 0.968988 | 1 |

The correlation analysis result shows the effect of system characteristics on the estimation of project size. The system characteristics exhibit a very strong relationship on the estimate of project size.

The graphical representation of this effect is shown in the following Fig 1.

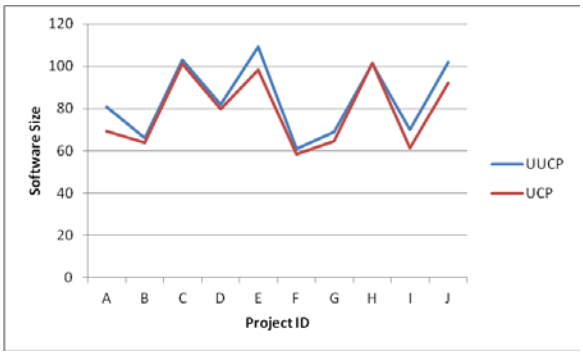


FIG 1. GRAPHICAL COMPARISON OF UUCP &UCP

The graph shows how the different system characteristics are affecting the estimation of project size. The X-axis contains the different project IDs and Y axis contains the values for the size of project estimated under use case point method. The graph also shows the variation in size of the project with and without considering the various system characteristics.

## VI. Conclusion

A number of different models and effort estimation methods have been developed in the past four decades. This clearly indicates the awareness among the researchers of the need to improve effort estimation in software engineering. Many factors have impact on the software development process. These factors are not only human, technical but also political and their impact can never be fully predicted. The even insufficiently accurate estimates are far better than none. We have illustrated result of use case point method for measuring the size in the estimation process in our work. If the estimation is done accurately, it decreases error. The requirement & design is more clearly shown in use case point. The relationship among the matrix of these methods shows the practical results. Hence we conclude that

the estimation process through use case point is better. The estimation using use case point shows practical reality of development. Hence the accuracy of estimation depends upon method used for estimation also.

## References

[1]. A. J. Albrecht. Function Point Counting Practices Manual, Release 4.0. International Function Points Users Group, 1994.
[2]. Albrecht, A.J. Measuring application development productivity. In: SHARE/GUIDE: Proceedings of the IBM Applications Development Symposium, (October 1979) 83-92.
[3]. Arnold, P. and Pedross, P. Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department. Forging New Links. IEEE Computer. Soc, Los Alamitos, CA, USA, pp. 490-493. 1998.
[4]. Banker, R. D., H. Chang, et al. (1994). "Evidence on economies of scale in software development." Information and Software Technology 36(5): 275-282.
[5]. Charles Symons 1991. Software Sizing and Estimation Mark II function Points (Function Point Analysis), Wiley 1991.
[6]. Chatzoglou, P. D. and L. A. Macaulay (1998). "A rule-based approach to developing software development prediction models." Automated Software Engineering 5(2): 211-243.
[7]. Hasan.O. Farahneh, Ayman A. Issa:A Linear Use Case Based Software Cost Estimation Model, World Academy of Science, Engineering and Technology Vol:5 2011-01-27
[8]. Ibarra, G. I., & Vilain, P. (2010). Software Estimation Based on Use Case Size. Brazilian Symposium on Software Engineering, (pp. 178-187).
[9]. Karner, G. (1993). Resource Estimation for Objectory Projects. Objective Systems SF AB.
[10]. Periyasamy and AditiGhode Department of Computer Science University of Wisconsin-La Crosse La Crosse, WI 54601 Cost Estimation using extended Use Case Point (e-UCP) Model ,IEEE 2009.
[11]. Shinji Kusumoto, FumikazuMatukawa, Katsuro Inoue, Shigeo Hanabusa, YuusukeMaegawa, Estimating Effort by Use Case Points: Method, Tool and Case Study
[12]. Smith, J. The Estimation of Effort Based on Use Cases. Rational Software, White paper. 1999.
[13]. C. Jones, *Software Estimating Rules of Thumb*, 2007.
[14]. Article by Roy K Clemmons “Project Estimation with Use Case Points”