Software Size Estimation Using Activity Point

S Densumite¹ and P Muenchaisri²

¹, ² Chulalongkorn University, Bangkok 10330, Thailand

Email: ¹ sharif.d@student.chula.ac.th, ² pornosiri.mu@chula.ac.th

Abstract. Software size is widely recognized as an important parameter for effort and cost estimation. Currently there are many methods for measuring software size including Source Line of Code (SLOC), Function Points (FP), Netherlands Software Metrics Users Association (NESMA), Common Software Measurement International Consortium (COSMIC), and Use Case Points (UCP). SLOC is physically counted after the software is developed. Other methods compute size from functional, technical, and/or environment aspects at early phase of software development. In this research, activity point approach is proposed to be another software size estimation method. Activity point is computed using activity diagram and adjusted with technical complexity factors (TCF), environment complexity factors (ECF), and people risk factors (PRF). An evaluation of the approach is present.

1. Introduction

Software size has been a challenge in software project management as an important parameter for effort and cost estimation. Software Line of Code (SLOC) is a software sizing physically counted after the software is developed. Many methods computed at early software development activities.

Allan Albrecht [1, 2] introduces Function Point Analysis (FPA) method to estimate software size in early phase which decomposes the complex system into simple systems. Unadjusted function point is adjusted with value adjustment factor (VAF) to represent software size as function points unit. Thirty years later, FPA has evolved into many methods such as IFPUG [3], NESMA [4] and COSMIC-FFP [5, 6]. Unified Modeling Language (UML) based object oriented metrics are widely used to create sizing model that inspired from function point including use case point (UCP) [7], sequence point (SP) [8], and class point (CP) [9].

UCP is a software size estimation approach in the early phase which uses transactions, scenarios, and actors to be main factors computed to software size, but it is not considered in use case relations that are directly affected to software size. This paper presents an approach to utilize activity diagrams named Activity Point (AP) which apply UCP in transactions, scenarios and use case relation to compute software size.

This paper is divided into 6 sections. Section 2 introduces related work. Section 3 provides an approach to calculate software size with AP approach. Section 4 describes a method to define weight factors. Section 5 presents an evaluation of the model. Finally, section 6 shows conclusions and future works.

2. Related work

A. Albrecht [1, 2] proposes Function Point Analysis (FPA) method to predict software size from functionalities which are composed of five components, including Internal Logical Files (ILF),
External Interface Files (EIF), External Inputs (EI), External Outputs (EO), and External Inquiry (EQ). Counting elements are multiplied with numerical rating and summed to give Unadjusted Function Point (UFP). UFP is adjusted by Technical Complexity Factor (TCF) to collect the result in function point unit size.

CFP approach presented in [5, 6] is applied to business application, real-time, and industrial software. The CFP approach is inspired from FPA and is mainly focused on data movement, which is classified into four types: entry, exit, read, and write. Software size will be replaced with the total number of data movements.

G Karner [7] proposes a model to measure software size using use case diagram named Use Case Point (UCP). UCP approach emphasizes in user interaction and use case transactions of each use case. UCP consists of two components: unadjusted actor point (UAP), and unadjusted use case weight (UUCW). The components are summed and adjusted with Technical Complexity Factor (TCF) and Environment Complexity Factor (ECF).

A. Sellami et al [8] use a set of procedures for sizing sequence diagrams at a high level of granularity using the COSMIC rules, and for the measurement of the structural size of sequence diagrams at a finer level of granularity. The approach uses three interactions in sequence diagram including Alternative (alt), which represents optional behavior that will be compiled in the system, Optional (op) operator that represents a behavioral choice that will be chosen follows the condition.

G. Costagliola [9] et al present sizing approach for object oriented products using class diagram called Class Point (CP). The approach has three steps. First, identifying and classifying the classes into four types namely problem domain, human interaction, data management, and task management type. Second, assigning a complexity level and weighting to each class. Finally, determining class point by adjusting the weighted sum from the previous step.

L Lavazza et al [10] improve UCP method using Path (P) and Transaction (T) which is described in the use case description to create a measurement model established by linear regression. G Robiolo et al [11] offer an alternative way by creating an effort model with linear regression using use case transactions and scenarios as main factors.

H. Madhavji et al [12] present quality factors which are impacted to software architecture in people issues, including procrastination, missing/late meetings, team members(s) not delivering sufficient work, poor planning, and group strategy.

3. Activity and use case diagram in two perspectives

The approach divides activities into two perspectives: high level and low level. High level of activity diagram presents use case’s relationships that are directly affected to software size such as “glue code” which includes activities or incidents. On the other hand, low level of activity diagram describes transactions and scenarios that are separately computed and weight with ideal number which includes activities or incident.

Fig. 1 describes activity perspectives that are designed from use case diagram. There are transaction perspective and use case relation perspective presented in different level of abstraction as follows.

- Fig 1.a and 1.b show an activity diagram at low level that separately describes use case description of UC1 and UC2. Activity/Action represents transaction and Path represent scenario of use case diagram.
- Fig 1.c shows a high level activity diagram that describes use case’s relation. Activity/Action represents each use case in use case diagram.
4. Activity and use case diagram in two perspectives

In this section, Activity Point approach is proposed to estimate software size in early phase. Fig. 2 describes Activity Point (AP) which is computed from Unadjusted Activity Point (UAP), Technical Complexity Factors (TCF), Environment Complexity Factors (ECF), and People Risk Factors (PRF). UAP is unadjusted size which is computed from Total Activity Point (TAP), and Total Activity Complexity Weight (TACW). Detail of each component is described in the following.

4.1. Total activity point
TAP is a factor, which presents size at low level, including TAW and TPW. The level of TAW and TPW is classified into 3 levels: simple, average, and complex. TAP is obtained by summation of TAW and TPW at row 1 in Table 1. A number of transactions in use case diagram is replaced by a number of activities or incidents. A number of scenarios is indicated by a number of paths in activity diagram. TAW is calculated from a Number of Activities or incidents (NA) level multiplied with weighting number at row 2 in Table 1. NA classification level and weight value are presented in Table 2. In the same way, TPW is computed from Number of Paths (NP) level multiplied with weighting number at row 3 Table 1. NP classification level and weight value are shown in Table 3.

4.2. Total activity complexity weight
TACW is a factor which represents a size at high level of activity diagram. The activity diagram will be transformed to Control Flow Graph (CFG) and specified complexity using McCabe Complexity approach. The total complexity represents size as Activity Complexity (AC). In addition, if the activity diagram consists of Folk and Join notation, there are possible cases as follows.
- Paths of fork/join are sequences. The paths will be combined to a single path as shown in Fig. 3 because it is equal to McCabe complexity result. The reason is parallel sequencing process does not affected to complexity increased.
- In contrast, Fig. 4 shows paths of fork/join including decision or loop in different path. The notations are connected to a single path that the notations sequence connected is insignificant meaning if the loop or decision comes first to calculating complexity.
- Synchronisation link between actions in different paths of fork/join is shown in Fig. 5 which the paths should be separated and linked together.

TACW is computed from AC and weighting number at row 4 in Table 1. AC is classified into three levels as simple, average, and complex which are shown in Table 4.
4.3. Unadjusted activity point

UAP is an initial software size computed during software development from TAP and TACW at row 5 Table 1. The number of TAP (N) in the formula is equal to the total number of use cases at low level. TACW is described as size at high level.

4.4. Activity point

Activity point (AP) is a software development size, which is computed from UAP multiplied with adjusting factors at row 6 in Table 1.

The adjusting factors include TCF, ECF, and PRF which are subsequently shown in Table 5-7. TCF, ECF, and PRF are scored from 0 (not related) to 5 (highly related). TCF at row 7 in Table 1 is an adjusting factor in technical consideration. ECF at row 8 in Table 1 is an adjusting factor based on environment consideration. TCF and ECF calculation are the same as UCP approach in [7]. People risk factor (PRF) is an additional people factor proposed at row 9 in Table 1 which is used to find impact factors affected to AP. Formula at row 9 in Table 1 is created from estimated effort without PRF since PRF is assumed to be 1. The ideal number of PRF comes from the multiplier of the addition of personal factor (PF) and the ideal number that makes the lowest average MRE from 7 project training data sets which is presented at row 10 Table 1. PF in Table 7 is inspired from [12]. The highest risk is 5.2 when PF score is 35.

4.5. Effort

An estimation effort model is provided at row 11 in Table 1 which uses AP as size and is created by linear regression including predictive variable, and independent variable which subsequently represent actual effort and AP.

| No. | Equation |
|-----|----------|
| 1   | \( TAP = TAW + TPW \) |
| 2   | \( TAW = \sum_{i=1}^{N} NA_{level} \times Weight \) |
| 3   | \( TPW = \sum_{i=1}^{N} NP_{level} \times Weight \) |
| 4   | \( TACW = \sum_{i=1}^{N} AC_{level} \times Weight \) |
| 5   | \( UAP = \sum_{i=1}^{N} TAP_{i} + TACW \) |
| 6   | \( AP = UAP \times TCF \times ECF \times PRF \) |
| 7   | \( TCF = 0.6 + TF/100 \) |
| 8   | \( ECF = 1.4 + (-0.03 \times EF) \) |
| 9   | \( PRF = 1 + (0.12 \times PF) \) |
| 10  | \( \text{Average MRE} = \frac{\sum_{i=1}^{N}(Actual\ Effort_{i} - Estimated\ Effort_{i})}{Estimated\ Effort_{i}} \) |
| 11  | \( \text{Effort} = a + b \times AP \) |

Table 2. Activity/Action weight category.

| NA Level | Description                  | Weight |
|----------|------------------------------|--------|
| Simple   | Simple steps in activity or action between 1 and 3. | 2      |
| Average  | More steps in activity or action between 4 and 7. | 3      |
| Complex  | Much steps in activity or action that more than or equal 8 | 4      |
### Table 3. Path weight category

| NP Level | Description                                              | Weight |
|----------|---------------------------------------------------------|--------|
| Simple   | Simple alternative path in activity between 1 and 3.    | 3      |
| Average  | More alternative path in activity between 4 and 7.      | 5      |
| Complex  | Much steps alternative path in activity between that more than or equal 8 | 9      |

### Table 4. Activity complexity weight category.

| AC Level | Description                                              | Weight |
|----------|---------------------------------------------------------|--------|
| Simple   | Less complexity from McCabe Complexity is 1.            | 9      |
| Average  | Much Complexity from McCabe Complexity between 2 and 3. | 14     |
| High     | High Complexity from McCabe Complexity more than or equal 4. | 18     |

### Table 5. Technical complexity factors.

| Factor | Description                                              | Weight |
|--------|---------------------------------------------------------|--------|
| T1     | Multi user experience and user interface design         | 2      |
| T2     | Response time/performance objectives                     | 1      |
| T3     | Server configuration                                     | 1      |
| T4     | Requirement complexity                                  | 1      |
| T5     | Support multi devices                                   | 0.5    |
| T6     | Big data scaling support                                | 0.5    |
| T7     | Multiple sites implement                                | 1      |
| T8     | Portability to other platforms                          | 2      |
| T9     | System maintenance                                      | 1      |
| T10    | Concurrent/parallel processing                           | 1      |
| T11    | Security features                                       | 1      |
| T12    | Access for third parties                                | 1      |
| T13    | End user training                                       | 1      |

### Table 6. Environment complexity factors

| Factor | Description                                              | Weight |
|--------|---------------------------------------------------------|--------|
| E1     | Familiarity with development process used               | 1.5    |
| E2     | Application experience                                  | 0.5    |
| E3     | Object-oriented experience of team                      | 1.0    |
| E4     | Lead analyst capability                                | 0.5    |
| E5     | Motivation of the team                                  | 1.0    |
| E6     | Stability of requirements                               | 2.0    |
| E7     | Part-time staff                                         | -1.0   |
| E8     | Difficult programming language                          | -1.0   |
Table 7. People risk factors.

| Factor | Description                                       | Weight |
|--------|---------------------------------------------------|--------|
| R1     | Team member(s) Procrastination.                   | 2      |
| R2     | Missing/late meeting                              | 0.5    |
| R3     | Team member(s) often not delivering adequate work.| 2      |
| R4     | Poor planning and group strategy.                | 1      |
| R5     | Customer Orientation                             | 1.5    |

5. Defining weight values

Weight values of factors in AP approach are calculated using the algorithm as shown in Fig. 6. The multiplier of NA level, NP level, and AC level in (2) - (4) are executed from training data sets of 7 projects including 4 enterprise software developed with JAVA, 2 mobile applications developed with JavaScript, and 1 e-commerce website developed with PHP. Number of activities/actions, paths, and complexity are calculated to find correlation with actual effort. The correlation indicates which factor is more affected to actual effort. The correlation of activities/actions, paths, and complexity with effort are subsequently resulted as 0.823, 0.844, and 0.665. It means NP is the most affected to actual effort comparing with others. Therefore, the proportionate percentage affecting factor with actual effort is 50 percent of TPW, 40 percent of TAW, and 10 percent of TACW. In Table 2 – 4, weight values are between 1 and 20. NA level, NP level, and AC level are weighted with the ideal number from the range which gives UAP the highest correlation with actual effort. UAP is a prior adjusted size that does not consider external adjusting factor. Therefore, adjusting factors should be considered as external factors to adjust UAP to be a final size as AP unit.

1. INPUT projects data include number of transactions, scenarios, activity complexity, and actual effort.
2. FIND correlation dependence with project’s actual effort.
3. SET percentage affected to effort of transactions, scenarios, and activity complexity from the correlation.
4. MULTIPLY actual effort with the percentage.
5. STORE representing transactions, scenarios, and activity complexity point of each project from previous step.
6. INITIAL preferred range of weight value.
7. FIND ideal weight value of each point that gives the closest to actual effort of each project with looping.
8. SET standard weight value by average ideal weight result.

Figure 6. Algorithm for defining weight values

6. Evaluation

Figure 1 is a demonstrating diagram which shows activity diagrams at low level and high level. At low level, there are a simple activity level in (1.a) and an average level in (1.b). The path level of low level in (1.a) and (1.b) are simple. TAW of (1.a) is equal to 2 and (1.b) is equal to 3. TPW of (1.a) and (1.b) are equal to 3. Then, TAP of (1.a) and (1.b) are subsequently equal to 5 and 6. At high level, McCabe complexity of (1.c) is equal to 3 as average level. Thus, TACW is equal to 14. TAP of (1.a), (1.b), and TACW are summed to be UAP that is equal to 25. Assuming that adjusting factor score is equal to 1 that gives result of TCF, ECF, and PRF is 0.74, 1.265, and 1.7 consecutively. Therefore, AP is calculated from multiplier of UAP, and adjusting factor which gives result of 39.50 and uses as an input to estimate software development effort.

Three projects as test data sets including 2 enterprise software developed with JAVA, and 1 mobile application developed with JavaScript are used to evaluate the AP approach. The test data sets are used to evaluate AP model. Average Magnitude Relative error (MRE) and Prediction Quality (PRED) are presented as quality factors of predictive expected effort. Table 8 shows the number of paths of
each project at low level. Table 9 presents the number of activity/action of each project at low level. Table 10 provides the number of activity complexity of each project at high level. Activity point, estimated effort, and actual effort are presented in Table 11 with the average MRE of 0.09. The average MRE is calculated from absolute RE of each project compared with actual effort and estimated effort. PRED(0.25) is resulted as 1 to present quality of AP which shows the percentage of project that MRE lower than 25% comes from 100% of total projects.

| Project | Quantity of Path Levels | Project | Quantity of activity/action levels |
|---------|-------------------------|---------|-----------------------------------|
|         | simple  | average | complex | simple  | average | complex |
| 1       | 6       | 6       | 3       | 1       | 3       | 12      |
| 2       | 4       | 3       | 2       | 2       | 3       | 6       |
| 3       | 16      | 5       | 0       | 2       | 7       | 10      |

Table 10. Data of activity complexity.

| Project | Quantity of Activity Complexity |
|---------|--------------------------------|
|         | simple | average | complex |
| 1       | 0      | 0       | 1       |
| 2       | 1      | 0       | 0       |
| 3       | 0      | 0       | 1       |

Table 11. Effort evaluation.

| Project | Activity Point | Estimated Effort | Actual Effort | MRE  |
|---------|----------------|-----------------|--------------|------|
| 1       | 341.65         | 286.19          | 301          | 0.05 |
| 2       | 77.47          | 84.09           | 102          | 0.21 |
| 3       | 89.65          | 93.41           | 92           | 0.02 |
| Average MRE |                |                 |              | 0.09 |
| PRED(0.25) |                |                 |              | 1    |

7. Conclusion
Software size estimation is a factor that many effort models use. This research considers in estimating effort using software size estimation based on activity diagram named Activity Point. The approach emphasizes metrics that are used in use case point approach and additional metrics proposed. The additional metrics focus on use case relation as activities or incidents, paths, and complexity in activity diagram. Unadjusted activity point is a prior adjusted size. A new adjusting factor named people risk factor is proposed. Technical Complexity Factor (TCF), and Environment Complexity Factors (ECF) are alike presented in use case point approach. Evaluation is presented with absolute mean relative error and PRED(25). The average MRE is calculated from absolute RE of three projects compare with actual effort and estimated effort which is resulted as 0.09. PRED(0.25) is resulted as 1.0 to present quality of AP which shows the percentage of project that MRE lower than 25% comes from 100% of total projects.

In the future works, the researchers will increase the efficiency of the model by working with more data sets, suitable weight value factors and adjusting factor scored criterion. A comparing with other size estimation approaches may be investigated. An automated tool supporting the proposed will be developed.
References

[1] Albrecht A J 1979 Measuring Application Development Productivity *Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17* (IBM Corporation 1979) pp. 83–92.

[2] Kim SE, Lively W, and Simmons D 2006 An Effort Estimation by UML Points in Early Stage of Software Development *Software Engineering Research*.

[3] International Function Point Users Group (IFPUG) January 2010 *The Function Point Counting Practices Manual version 4.3.1* [EB/OL]. www.ifpug.org.

[4] ISO/IEC24750 2005 *Software engineering-NESMA functional size measurement method version 2.1-Definitions and counting guidelines for the application of Function Points Analysis*[S] (Geneva: International Organization for Standardization-ISO)

[5] ISO/IEC19761 2003 *Software Engineering-COSMIC-FFP A Functional Size Measurement Method*[S] (Geneva: International Organization for Standardization-ISO)

[6] Abdullah NAS, Rusli NIA, and Ibrahim MF 2013 A case study in COSMIC functional size measurement: Angry Bird Mobile Application in *Open Systems (ICOS) 2013 IEEE Conference* pp. 139-44.

[7] Karner G 1993 Resource Estimation for Objectory Projects *Objective Systems SF AB*.

[8] Sellami A, Hakim H, Abran A, and Ben-Abdallah H 2015 A measurement method for sizing the structure of UML sequence diagrams *Information and Software Technology* vol. 59 pp. 222-32.

[9] Costagliola G, Ferrucci F, Tortora G, and Vitiello G 2005 Class point: an approach for the size estimation of object-oriented systems *IEEE Transactions on Software Engineering* vol. 31 pp. 52-74.

[10] Lavazza L and Robiolo G 2010 Introducing the evaluation of complexity in functional size measurement: a UML-based approach *the Proceedings of the 2010 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Bolzano-Bozen, Italy).

[11] Robiolo G and Oroscrom R 2012 *An Alternative Method Employing Uses Cases for Early Effort Estimation* (Universidad Austral y Universidad Argentina de la Empresa).

[12] Madhavji H and Wilding M 2009 The Impact of Non-Technical Factors on Software Architecture (Canada: University of Western Ontario and IBM Canada Ltd).