SOFTWARE COST EVALUATION USING UCP ANALYSIS

M.Bala Subramanian*, G. Rajarajeswari**

*Research Scholar, Department of Computer Science, Bharath Institute of Higher Education and Research, Bharath University, Chennai.
**Department of Information Technology, RRASE College of Engineering, Chennai.

Received on: 15.10.2016 Accepted on: 22.11.2016

Abstract

The information processing size, technical complexity factor, Environmental factors. The first two of these factors are intrinsic to the size of the system in the sense that they result directly from the requirements for the system to be delivered to the user and the for estimation purpose the third group of environmental factors are taken into account. In object-oriented software production, use cases describe functional requirements. The use case model therefore be used to predict the size of the future software system at an early development stage. This paper describes a simple approach to software cost estimation based on use case models, the ‘Use case Points methods’ (UCP). The method is not new, but has not become popular although it is easy to learn. Reliable estimates can be calculated in a short time with the aid of spreadsheet.

Use case points

Sizing method like Function point analysis are well known for there use in the SE. But, this has some serious limitation like counting the function points with experts. In 1993, ‘Use case points’ method for sizing and estimating the projects developed with the object oriented method proposed by Gustav Karner of Objectory (now a part of Rational Software) in 1993.

It is an extension and modification of A.J. Albrecht's work on Function Points. Karner's Use Case Points based estimation method should be used in conjunction with other estimating methods and the methodology is copyrighted with the rational software. The methodology is well defined in Schneider, Geri and Winter, Jason P, Applying Use Cases - A Practical Guide, Addison Wesley, 1998.

Estimation

An early estimate of effort based on use cases can be made when there is some understanding of the problem domain, system size and architecture at the stage at which the estimate is made. The use case points method is a software
sizing and estimation method based on the use case counts called ‘Use case points’ and they are best counted during the early phases of an object-oriented project that captures its scope with use cases.

Also, this model attempts to quantify areas such as Technical Complexity which are close to FP’s GSC and Environmental Complexity e.g. ease of use and programmer motivation.

**Use case model**

**Use case and Actors**

The Use case point’s components are as follows:

- Unadjusted Actor Weights (UAW)
- Unadjusted Use Case Weights (UUCW)
- Technical Complexity Factors (TCF)
- Environmental Factors (EF)

**Unadjusted Actor Weights (UAW)**

Actor is any entity that interacts with the system and is external to the system. Examples include:
Classify Actors into Simple, Average or Complex

- **Simple**: Another system with a well-defined API
- **Average**: Another system which interacts using a well-known protocol such as TCP/IP or User interacting with a system using a text-based interface
- **Complex**: User interacting with a system using a GUI-based interface or Another system without a well-defined API

Assign Weighting Factors for each Actor as per following table

<table>
<thead>
<tr>
<th>Actor Type</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>2</td>
</tr>
<tr>
<td>Complex</td>
<td>3</td>
</tr>
</tbody>
</table>

Total Unadjusted Actor weight (UAW) = \((\text{Number of simple actors} \times 1) + (\text{Number of medium actors} \times 2) + (\text{Number of complex actors} \times 3)\)

Unadjusted Use Case Weights (UUCW)

Use Case describes the things the user wants the system to do and it is an interaction between the actor and the system and it is internal to the system. The Use cases can be categorized as Transaction Based or Analysis Class Based.

They are classified by complexity into Simple, Average or Complex. Transaction based Weighting Factors, Based on number of transactions in the Use Case, including secondary scenarios. A transaction is an atomic set of activities.

It may help if visualized using Activity Diagrams, Analysis-Class based Weighting Factors, based on the number of Analysis Classes used to implement the Use Case.

Assign Weighting Factors for each Use Case as per following table
• Total Unadjusted Use Case Weights (UUCW) = (Number of simple Use Cases * 5) + (Number of medium Use Cases * 10) + (Number of complex Use Cases * 15)

• UAW = Σ (Use Cases * Weighting Factor)

<table>
<thead>
<tr>
<th>Use Case Type</th>
<th>Description</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>&lt; 3 Transactions</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>4 - 7 Transactions</td>
<td>10</td>
</tr>
<tr>
<td>Complex</td>
<td>&gt; 7 Transactions</td>
<td>15</td>
</tr>
</tbody>
</table>

Use Case Type

<table>
<thead>
<tr>
<th>Description</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>&lt; 5 Analysis Classes</td>
</tr>
<tr>
<td>Average</td>
<td>5 - 10 Analysis Classes</td>
</tr>
<tr>
<td>Complex</td>
<td>&gt; 10 Analysis Classes</td>
</tr>
</tbody>
</table>

The technical complexity factor is calculated by multiplying the value of each other by its weight and then adding all these numbers to get the sum called the TFactor. The following formulae is applied:

(i.e., Used to factor in Technical Complexity in the Project (as different from Functional Complexity represented by UUCP))

- TFactor = Σ (Tlevel Rating) * (Weighting Factor)
- TCF = 0.6 + (0.01*TFactor)

**Environmental Factors (EF)**

The environmental factor is calculated by multiplying the value of the each factor by its weight and adding the products to get the sum, called the EFactor. The following formulae is applied.

<table>
<thead>
<tr>
<th>Factor Description</th>
<th>Weighting Factor</th>
<th>Elevel Rating (0-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiar with Rational</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Unified Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application experience</td>
<td>0.5</td>
<td>0 - no experience</td>
</tr>
<tr>
<td>OO Experience</td>
<td>1</td>
<td>5 - Expert</td>
</tr>
<tr>
<td>Lead analyst capability</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>
Motivation

| 1 | 0- No motivation  
5- High Motivation |

Stable requirements

| 2 | 0-Extremely Unstable requirements  
5-Unchanging requirements |

Part-time workers

| -1 | 0-no part time staff  
5- all part time staff |

Difficult programming language

| 1 | 0-easy language  
5-difficult language |

Used to factor in experience & expertise level of the project team

- EFactor = Σ (Elevel Rating) * (Weighting Factor)
- E = 1.4 + (-0.03*EFactor)

**UCP Calculation Procedure**

Unadjusted Use Case Points (UUCP) = UAW + UUCW

Use Case Points (UCP) = UUCP*TCF*EF

**Effort Conversion**

From the assessment of Environmental Factors, count how many of the first six factors have a rating of <3, added to how many of the last 2 have a rating of >3. If the above total is 2 or less, then estimate 20 Person hours per UCP Else estimate 28 Person hours per UCP.

**UCP Conclusion**

This model is based on limited research, it is to be used in conjunction with other estimation methodologies. The use-case should be at the right granularity. It should not be too fine-grained or coarse-grained. Avoid the trap of considering alternate flows as separate use cases.
Buffers for Project Management overheads, Configuration Management overheads, Training etc. to be included separately. However, using this method of calculation means that even small adjustments of an environmental factor, for instance by half a point, can make a great difference to the estimate.

References


5. An introduction to Function Point Analysis, Rofer Heller, Vice president, A publication for information Technology Professional


9. Function point analysis Quick reference card, CPM version 4.3

10. “Function points as a measure of productivity,” in Proc. GUIDE 53 Meeting, Dallas, TX, Nov. 12, 1981.

11. Function Point Analysis: Difficulties and Improvements, CHARLES R. SYMONS, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO. 1, JANUARY 1988

12. FUNCTION POINT ESTIMATION METHODS: A COMPARATIVE OVERVIEW, Roberto Meli, Luca Santillo


15. IFPUG - Guidelines to Software Measurement, Release 1.1,


18. Integrating Function Point Project Information for Improving the accuracy of Effort Estimation


27. Procs Santillo L., Meli R. - Early Function Points: Some Practical Experiences of Use – ESCOMENCRESS


