CHAPTER 7

INTRODUCTION TO HYBRID TOOL

7.1 INTRODUCTION

A hybrid tool has been proposed for effort estimation, which employs a software effort estimation based on Use Case Models: The “Use Case Points Method.” The hybrid tool focuses on three basic parameters. 1. Software cost estimation, 2. Benchmarking and 3. Risk Assessment. The Use Case Model can be used to predict the size of the future software system at an early development stage to estimate the effort in the early phase of software development, as well as software engineering development, Evgeni et al (2002).

7.2 USE CASE POINT METHOD

This section briefly explains the procedure how Use case point [UCP] method has been implemented in the hybrid tool.

7.2.1 Use Case Model

The first and the foremost step is to calculate Use Case Point (UCP) from Use Case Model, Grady Booch et al (1999). The Use Case Model mainly consists of two documents, system or sub system documents & use case documents contains the following description of items: system name, risk factors, system – level use case diagram, architecture diagram, subsystem descriptions, use case name, brief description, context diagram, preconditions, flow of events, post conditions, subordinate use case diagrams, subordinate use cases, activity diagram, view of participating classes,
sequence diagrams, user interface, business rules, special requirements & other artifacts.

From the above specified information, the hybrid model focuses mainly on two parameters system – level use case diagram and flow of events. System – level use case diagram includes one or more use case diagrams showing all the use cases and actors in the system, JeanMarc Jaezequel et al (2002). Flow of events includes a section for the normal path and each alternative path in each use case. Figure 7.1 shows the use case diagram for banking application. Figure 7.2 shows the flow of event file.

Figure 7.1 Use case diagram for banking application

1. A session is started when the customer wants to check the balance, deposit funds, withdraw cash and transfer funds from bank.

2. A new session is started when a customer inserts an ATM card into the card reader slot of the machine.

3. The ATM pulls the card into the machine and reads it.
4. If the reader cannot read the card due to improper insertion or damaged stripe, the card is ejected, an error screen is displayed, and the screen is aborted.

5. The customer is asked to enter his/her PIN, and is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case.

6. In case of failure of the ATM machine, ATM technician repairs the system

.................

.................

.................

.................

Figure 7.2 Flow of Events (Session Use Case)

7.2.2 Counting use case point

Intuitively, UCP is measured by counting the number of actors and transactions included in the flow of events with some weight, Jonao Araujo et al (2002). A transaction is an event that occurs between an actor and the target system, the event being performed entirely or not at all. But, in this method the effort estimation is calculated by applying the following procedure.
Procedure

7.2.2.1 Counting Actors weight

The actors in the use case are categorized as simple, average or complex. A simple actor represents another system with a defined API. An average actor is either another system that interacts through a protocol such as TCP/IP or it is a person interacting through a text based interface. A complex actor is a person interacting through a GUI interface.

Table 7.1 Counting Actor’s weight

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Program Interface</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>Interactive, or Protocol Driver</td>
<td>2</td>
</tr>
<tr>
<td>Complex</td>
<td>Graphical User Interface</td>
<td>3</td>
</tr>
</tbody>
</table>

The number of each actor type that the target software includes is calculated and then each number is multiplied by a weighting factor shown in Table 7.1. Finally, actor’s weight is calculated by adding those values together.

Procedure

7.2.2.2 Counting use case weights

Each Use case should be categorized into simple, average or complex based on the number of transactions including the alternative paths. A simple use case has 3 or fewer transactions, an average use case has 4 to 7 transactions and a complex use case has more than 7 transactions. Then, the number of each use case type is counted in the target software and then each number is multiplied by a weighting factor shown in Table 7.2.
Table 7.2 Transaction Based Weighting Factors

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>3 or fewer transactions</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>4 to 7 transactions</td>
<td>10</td>
</tr>
<tr>
<td>Complex</td>
<td>More than 7 transactions</td>
<td>15</td>
</tr>
</tbody>
</table>

Finally, use case weight is calculated by adding these values together.

Procedure

7.2.2.3 Calculating unadjusted use case points:

It is calculated by adding the total weight for actors to the total for use cases.

Table 7.3 Weighting Factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>Distributed System</td>
<td>3</td>
</tr>
<tr>
<td>T₂</td>
<td>Response or Throughput Performance Objectives</td>
<td>4</td>
</tr>
<tr>
<td>T₃</td>
<td>End – User Efficiency (online)</td>
<td>5</td>
</tr>
<tr>
<td>T₄</td>
<td>Complex Internal Processing</td>
<td>2</td>
</tr>
<tr>
<td>T₅</td>
<td>Code must be readable</td>
<td>3</td>
</tr>
<tr>
<td>T₆</td>
<td>Easy to install</td>
<td>5</td>
</tr>
<tr>
<td>T₇</td>
<td>Easy to use</td>
<td>5</td>
</tr>
<tr>
<td>T₈</td>
<td>Portable</td>
<td>2</td>
</tr>
<tr>
<td>T₉</td>
<td>Easy to Change</td>
<td>5</td>
</tr>
<tr>
<td>T₁₀</td>
<td>Concurrent</td>
<td>1</td>
</tr>
<tr>
<td>T₁₁</td>
<td>Includes special security features</td>
<td>4</td>
</tr>
<tr>
<td>T₁₂</td>
<td>Provides direct access for third parties</td>
<td>2</td>
</tr>
<tr>
<td>T₁₃</td>
<td>User training facilities required</td>
<td>2</td>
</tr>
</tbody>
</table>
Procedure

7.2.2.4 Weighting technical and environmental factors

The UUCP are adjusted based on the values assigned to a number of technical and environmental factors shown in Tables 7.3 & 7.4.

Table 7.4 Technical Factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
<td>Familiar with the Rational Unified Process</td>
<td>4</td>
</tr>
<tr>
<td>F₂</td>
<td>Application Experience</td>
<td>3</td>
</tr>
<tr>
<td>F₃</td>
<td>Object – Oriented Experience</td>
<td>2</td>
</tr>
<tr>
<td>F₄</td>
<td>Lead Analyst Capability</td>
<td>3</td>
</tr>
<tr>
<td>F₅</td>
<td>Motivation</td>
<td>5</td>
</tr>
<tr>
<td>F₆</td>
<td>Stable Requirements</td>
<td>4</td>
</tr>
<tr>
<td>F₇</td>
<td>Part – Time Workers</td>
<td>3</td>
</tr>
<tr>
<td>F₈</td>
<td>Difficult Programming Language</td>
<td>3</td>
</tr>
</tbody>
</table>

Method:

Each factor is assigned a value between 0 and 5 depending on its assumed influence on the project. A rating of 0 means the factor is irrelevant for this project and 5 means it is essential.

Calculation of TCF

It is calculated by multiplying the value of each factor \(T_{1} - T_{13}\) in Table 7.3 by its weight and then adding all these numbers to get the sum called the T Factor. Finally, the following formula is applied:

\[
TCF = 0.6 + (0.01 \times \text{T Factor})
\] (7.1)
Calculation of Environmental Factor

It is calculated accordingly by multiplying the value of each factor (F₁ – F₈) in Table 7.4 by its weight and adding all the products to get the sum called the E Factor. Finally, the following formula is applied:

\[ EF = 1.4 \times (-0.03 \times \text{E Factor}) \]  \hspace{1cm} (7.2)

Procedure

7.2.2.5 Calculating UCP

Use Case Point (Adjusted) is calculated by

\[ \text{UCP} = \text{UUCP} \times \text{TCF} \times EF \]  \hspace{1cm} (7.3)

Figure 7.3 shows the use case effort model.

Figure 7.3 Calculating Effort
Procedure

7.2.2.6 Estimating Effort

By multiplying the specific value (man – hours) by the UCP, the effort can be easily calculated.

7.3 AN AUTOMATED TOOL FOR ESTIMATING USE CASE POINT

7.3.1 Overview

The objective of the questionnaire in Appendix – I is to collect information about project data from various managers working in different software companies. This project data has been utilized to compare the effort estimated by the hybrid tool proposed in the thesis and the effort given by the project managers for a particular project. In order to effectively introduce Use Case Point Method to the software development, a Use Case Point measurement tool was created. There were several existing tools available which are based on Use Case Models but in all these existing models, it is necessary to judge the complexity of actors and Use cases manually. The judgement is the most important part in software cost estimation. In order to obtain the entire procedure described in section 7.2 automatically, it is mandatory to describe a set of rules to classify the weight for actor and use case in section 7.2. Also, it is necessary to write the Use – Case Model in machine – readable format. The use case model is written in XMI [XML Metadata Interchange], Wai Ming et al (1999). The reason for choosing this type of file format is because most case tools for writing UML diagrams support to export them as XMI files.
7.3.2 Rules for weighting actors

As described in section 7.2, weight for each action is determined by the interface between actor and the target software. But, the interface information will not be available in the actor description. Only the name of the actor will be available. So, it is very much essential to create a protocol which determines the complexity of actor.

Step 1: Classification based on actor’s name

At the initial stage of the classification it is to be determined whether the actor is a person or an external system based on the name of the actor. A list of keywords has been created to determine the same.

For example the keywords “system” and “server” are used in the system’s name.

Keywords for step 1 (KL<sub>a</sub>): System, Server, Application, Tool.

The models have been developed with a minimal set of keywords. As on later stages, the new keywords will be updated automatically and can be used for later projects.

Step 2: Classification based on Keywords included in Use case

Classification is based upon on the flow of events to which the actor is relevant. Initially, a set of three keywords for each complexity was used. Next the tool will extract all words included in the flow of events and then match them with each keyword in the lists. Finally, the actor’s weight is assigned as per the complexity for the keyword list that is most fitted to the words in the flow of events.
Keywords

Average Actor (System) (KL_{aa}) : Message, Mail, Send

simple actor (KL_{sa}) : Request, Send, Inform.

Average Actor (Person) (KL_{ap}) : Command, Text, I/P, CUI

Complex actor (KL_{ca}) : Press, Push, Select, Show, GUI, Window

Step 3: Classification based on experience date

If it is not possible to determine the actor’s weight at step 2, then it can be determined based on the experience data. The experience data includes the information about the Use Case Model and the Use case Point developed in the past software projects.

7.4 RULES FOR WEIGHTING USE CASES

As described in section 7.3, the complexity of use case is determined by the number of transactions. So, the hybrid model focuses on the flow of events in the Use Case Model. The simplest way to count the transaction is to count the number of events. There are no standard procedures or protocols to write the flow of events and it is also quite possible that several transactions are described in one event. So, because of this limitation several guidelines to write events in use case model have been proposed. Two basic guidelines for writing flow of events file were framed:

\[(G_1) \rightarrow \text{Use a Simple Grammar} \]

\[(G_2) \rightarrow \text{Include a reasonable set of actions.}\]
Based on the above said guidelines, the hybrid tool analyzes the events using the morphological analysis and syntactic analysis. Through this analysis, the information of morpheme from the statement and dependency relation between words in the statement can be determined. The hybrid tool conducts the morphological analysis for all statements and gets the information of the subject word and predicate word for each statement.

The following rules were then applied:

**RULE U – 1:**

Regard each set of the subject and predicate words as candidates of a transaction.

**RULE U – 2:**

Identify among the candidates, the one that is related to actor’s operation and system response as a transaction.

For each use case, the above - said rules were applied and based on these rules, the number of transactions will be determined. Then, based on the number of transactions I have determined the Complexity of each use – case is determined based on the number of transactions. Figure 7.4 diagrammatically explains the working principle of use case model in the hybrid model.
Figure 7.4 An Automated Hybrid Tool

The Automated Hybrid model shown in Figure 7.4 calculates the Use case Point [UCP] from use case models written in XMI files and the effort estimation has been extended to risk assessment and the detailed result has been explained in the next chapter.