Case Study on Software Cost Estimation Techniques

Arpita Sharma¹ Mahender Kr. Beniwal² T. N. Sharma³

I. INTRODUCTION

According to the survey, only one third projects overrun their original estimates. To make exact prediction of software development is a critical issue and it is difficult to determine how much effort a project and project managers to analyze it. It is little bit difficult to make the estimation about the project and determine how much time and manpower a project requires in the beginning. System analysts are unable to make realistic hardware- software during the system design phase and software project personnel cannot exactly tell their proposed budget. So, it is optimistic related to software development and also it is inevitable because it compromises as a consequence. But in fact huge overruns results from inaccurate estimates that are believed to occur frequently. The entire process of developing a cost for software is not different from any other element of cost. However, aspects of the process are peculiar to software estimating. Some of the unique aspects are driven by the nature of software as a product. Other problems are created by the nature of the estimating methodologies.

Cost estimation is one of the most challenging tasks in project management. It estimates about needed sources and it includes estimating the size of the software product to be produced. It is very difficult to estimate the cost of software development. Many of the problems that hinder the development are responsible for the difficulty in estimating that effort. The first step to estimate and understand is to define the system to be estimated. Actually, software is invisible and intangible to understand the process that can't be seen or touched. Because, software grows and changes as it is written. When hardware design is inadequate or it fails to give any solution it is often attempted through changes to the software. It may occur late in the development process for the software growth.

Many software cost estimation methods are available including algorithmic methods after 20 years research. In fact, no one method is better or worse, their strength and weaknesses are complimentary to each other. So it is necessary to understand strength and weaknesses for estimating the projects. This paper includes popular methods of software cost estimation and their implementation on some applications.

A. Considerations in choosing the Estimation techniques:
The sections discussed below describe the major issues to be considered for choosing an estimation technique.

1) Size of the Project
In order to choose the best estimation technique, size of the project is one of the factors that can be considered. Now, it can be categorized into:

1 Small: Small projects are characterized as a project with five or fewer total technical staff. The statistical oriented techniques cannot be used in small projects while it can be used in larger projects, since variations in individual productivity drown out other factors. Small projects mostly use the same number of people on the team for the entire project. This invalidates some of the more algorithmically oriented large project estimation techniques. Therefore, the best estimation technique for small projects tend to be “bottom-up” techniques based on estimates created by the individuals who will actually do the work.

2 Large: Large projects comprises of a team of approximately 25 people or more that lasts 6 to 12 months or more. In case of large projects the techniques changes significantly from the beginning of the project to the end. In the beginning of the project, the best estimation approaches tend to be “top-down” techniques which are based on algorithms and statistics. When the specific team members are not yet known at that point of the project the above techniques are valid. Like when plans are based on a team that consists of say 11 senior engineers, 25 staff developers and 8 testers, rather than specific individuals. During the middle stages of the project, a combination of top down and bottom up techniques based on the project’s own historical data will produce the most accurate estimates. At the end of the project, bottom up techniques will provide the most accurate estimates.

3 Medium: Medium size projects consist of approximately 5 to 25 people and lasts 3 to 12 months. They have an advantage of being able to use all the estimation techniques that large projects as well as small projects cannot use.

B. What is being projected
After determining the features some projects focus on estimating the schedule and effort required to deliver those features. While other projects determine their budgets and development time frames and then focus on estimating that how many features they can deliver.

There are various estimation techniques which are applicable irrespective of what is being estimated. Some of the techniques are better suited for estimating how much effort a project will require, how much time a project will take or how many features can be delivered.

Estimating size means estimating the scope of technical work of a given feature set in units like lines of code, function points, or some other measure. And estimating features refers to estimating how many features can be delivered within schedule and budget constraints.
C. Software Development Style:
There are two basic development styles for the purposes of estimation. They are sequential and iterative. The main difference between these kinds of projects is the percentage of requirements they define early in the project as compared to the percentage they define after construction is underway. According to these criteria, the various common development approaches can be summarized below as:

1. **Extreme Programming:** Extreme programming defines only the requirements that will be developed in the next iteration, which typically lasts less than a month (Beck 2004). For estimation purposes, Extreme Programming is a highly iterative approach.

2. **Evolutionary Prototyping:** When requirements are unknown, then Evolutionary Prototyping is used. Also the main reason behind the use of evolutionary prototyping is to help define requirements (Mc Connell 1996). For estimation process, this is an iterative development style.

3. **Staged Delivery:** Staged delivery attempts to define the majority of its requirements prior to beginning the majority of construction (Mc Connell 1996). It uses iterations within design, construction, and test. So it is iterative. However, for estimation process, it is a sequential development style.

4. **Evolutionary delivery:** An evolutionary delivery project can define anywhere from “hardly any” to “most” of its requirements up front. The evolutionary delivery project can be either sequential or iterative based on which end of the scale the project falls on. Most evolutionary delivery projects leave enough requirements undefined at the beginning of construction that the approach as normally practiced is iterative.

5. **Rational unified Process:** The rational unified process (RUP) describes its stages as “iterations”. However, a nominal RUP project seeks to define about 80% of its requirements before construction begins. For estimation process, RUP is a sequential development style.

6. **Scrum:** Scrum is a development style in which a project team takes on a set of features that it can implement within a 30-day “sprint”. Once a sprint is started, the customer is not allowed to change requirements. For estimation process, Scrum is sequential, for an individual sprint. And for multiple sprints, Scrum is iterative, since features are not allocated for more than one sprint at a time.

II. **Effect of Development Style on Choice of Estimation Techniques**
Both sequential and iterative projects tend to start with top-down or statistically based estimation techniques and both migrate toward bottom up techniques. Iterative projects transition to refine their estimates more quickly using project specific data.

A. **Development Stage**
In a project, a team works in its own way, and develops information that supports more accurate estimates. Therefore, requirements become better understood, designs become more detailed, plans become firmer and the project itself generates productivity data that can be used to estimate the remainder of the project. The development stages can be divided as follows.

1. **Early:** The early stage is the period from the beginning of the project concept until requirements have been mostly defined, in case of sequential projects. And in case of iterative projects, early refers to the initial planning period.

2. **Middle:** Middle stage is the time period between initial planning and early construction. In case of sequential project, it is the time from requirements and architecture time until enough construction has been completed to generate project productivity data that can be used for estimation. And in case of iterative projects, middle refers to the first two to four iterations- the iteration that occur before the project can confidently base its estimates on its own productivity data.

3. **Late:** It refers to the time from mid- construction through release.

B. **Accuracy Possible:**
The accuracy of a technique can be defined as a function partly of the technique, partly of whether the technique is being applied to a suitable estimation problem, and partly of when in the project the technique is applied. Some estimation techniques produce high accuracy but at high cost. And others produce lower accuracy but at lower cost.

III. **Brief Introduction of the Applications Discussed Below**

1. **Just Weather:** This is a weather app, which provides you with current weather as well as weather forecast reports of your geo location. You can view other cities weather by just inserting City Name, Country Name. Just Weather also provides you with app widget for your home screen with update you with your current weather details.

2. **Stock Exchange:** Stock Exchange let user to add his/her previous investments. User can view own investment's current status. My Stocks let User view all other company stock info in tabular format as well as its graphical representation. User can compare different company stock variation. User can mail or message the current stock information.

3. **Unit Converter:** Unit Converter allows the user to get the converted form of units. It converts one unit into another. User can achieve the converted form of any unit which he wants like conversion of temperature from Celsius (°C) to Fahrenheit (°F).

4. **Call Manager:** Call Manager allows the user to keep track of all the call logs. This application allows the user to view all the missed calls, received calls, dialed calls. User can also delete any call log as per his requirement.

IV. **Application of Some of the Methods:**
In this section, some of the applications are being applied on various different type of application. Popular methods for estimation in software engineering include:

1. **Expert Judgment Method/Delphi Method**
2. **Analogy**
3 COCOMO II

A. Expert Judgment Method/Delphi Method:

Expert judgment techniques involve consulting with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. Generally, a group consensus technique, Delphi technique, is the best way to be used. The strengths and weaknesses are complementary to the strengths and weaknesses of algorithmic method. The method is applied on all four applications discussed above. A panel of 3 experts was made who are having experience of software development for at least 4 years. These experts are

1. Abhay Mathur (About 4 years of experience)
2. CB Singh (About 4 years of experience)
3. Shiv Kumar (About 4 years of experience)
4. Pratik Diwedi (About 15 years of experience)

After all round meetings and discussions the final estimation by experts is given as following table

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Name of Project</th>
<th>Technology</th>
<th>Estimated PM using Delphi Method</th>
<th>Actual PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Just Weather</td>
<td>Android</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>Stock Exchange</td>
<td>Android</td>
<td>2.5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Unit Converter</td>
<td>Android</td>
<td>12 days</td>
<td>16 days</td>
</tr>
<tr>
<td>4</td>
<td>Call Manager</td>
<td>Android</td>
<td>2</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 1: Final Estimation by Experts

B. Estimating by Analogy:

Estimating by analogy means comparing the proposed project to previously completed similar projects where the project development information is known. Actual data from the completed projects are extrapolated to estimate the proposed project. This method can be either at system-level or at the component-level.

Estimating by analogy is relatively straightforward. Actually in some respects, it is a systematic form of expert judgment since experts often search for analogous situations so as to inform their opinion.

On applying the analogies of projects, following results are produced.

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Name of Project</th>
<th>Technology</th>
<th>Estimated PM using Analogies Method</th>
<th>Actual PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Just Weather</td>
<td>Android</td>
<td>3</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>Stock Exchange</td>
<td>Android</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Unit Converter</td>
<td>Android</td>
<td>21 days</td>
<td>16 days</td>
</tr>
<tr>
<td>4</td>
<td>Call Manager</td>
<td>Android</td>
<td>2.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 2: Result

C. COCOMO II

COCOMO II when applied on four software projects produces the results as per the following table.

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Name of Project</th>
<th>Technology</th>
<th>Estimated PM using COCOMO II</th>
<th>Actual PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Just Weather</td>
<td>Android</td>
<td>21 days</td>
<td>16 days</td>
</tr>
<tr>
<td>2</td>
<td>Stock Exchange</td>
<td>Android</td>
<td>22 days</td>
<td>16 days</td>
</tr>
<tr>
<td>3</td>
<td>Unit Converter</td>
<td>Android</td>
<td>21 days</td>
<td>16 days</td>
</tr>
<tr>
<td>4</td>
<td>Call Manager</td>
<td>Android</td>
<td>22 days</td>
<td>16 days</td>
</tr>
</tbody>
</table>

Table 3: different values of 4 different projects

Above table is describing the various values which are being calculated for various projects. This table is showing different values of 4 different projects.

V. CONCLUSION

The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for project managers as well as system analysts and developers. There are many software cost estimation methods available including algorithmic methods, estimating by analogy, expert judgment method, top-down method, and bottom-up method. No one method is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. To understand their strengths and weaknesses is very important when you want to estimate your projects.

For a specific project to be estimated, which estimation methods should be used depend on the environment of the project. According to the weaknesses and strengths of the methods, you can choose some methods to be used. I think a combination of the expert judgment or analogy method and COCOMO II is the best approach that you can choose. For known projects and projects parts, we should use expert judgment method or analogy method if the similarities of them can be got, since it is fast and under these circumstances, reliable. For large, lesser known projects, it is better to use algorithmic model like COCOMO II.

VI. REFERENCES

