

**TOPIC:**

**COST ESTIMATING IN THE SOFTWARE COMPANY**

*A Cost Engineering White Paper*

*RAMA*

*ESC – Lille, Paris Campus*

**Abstract :**

Now a days the cost estimating in the software company calculates the manpower requirement ,staffing construction finance, scheduling, construction management, negotiating contracts, value engineering, types of cost estimates, overhead, insurance, profit. Staffing levels based on team experience, application size, reusable materials, and other factors. Software cost estimation is simple in concept, but difficult and complex in reality. The difficulty and complexity required for successful estimates exceed the capabilities of most software project managers. The current *best practice* for software cost estimation is to use a combination of software cost estimation tools coupled with software project management tools. Software cost estimation was the difficulty encountered in completing large software applications on time and within budget<sup>1</sup>.

---

<sup>1</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.html>

## Table of Contents

<u>1. Introduction.....</u>	<u>3</u>
<u>2. Requirement .....</u>	<u>5</u>
<u>3. Staffing .....</u>	<u>7</u>
<u>4. Costs .....</u>	<u>8</u>
<u>5. Schedules .....</u>	<u>9</u>
<u>6. Quality .....</u>	<u>10</u>
<u>7. Risk .....</u>	<u>11</u>
<u>8. Conclusion .....</u>	<u>11</u>

## **1. INTRODUCTION**

Research on software cost estimation started independently in a number of companies and military organizations that built large software systems. Formal research into software cost estimation became necessary when software applications and systems software began to go beyond 100,000-source code statements in size. The main issue that led to formal research programs for software cost estimation was the difficulty encountered in completing large software applications on time and within budget. A secondary issue was the fact that when deployed, software applications often contained significant numbers of bugs or defects<sup>2</sup>.

Additional features found in some but not all software estimation tools include the following:

- Risk and value analysis.
- Estimation templates derived from historical data.
- Links to project management tools such as Artemis or Microsoft Project.
- Cost and time-to-complete estimates mixing historical data with projected data.

The software cost estimation industry and the project management tool industry originated as separate businesses. Project management tools began appearing around the 1960s, about 10 years before software cost estimation tools.

Although the two were originally separate businesses, they are now starting to join together technically.

Project management tools have no built-in expertise regarding software, as do software cost estimation tools. For example, if you wish to explore the quality and cost impact of an object-oriented programming language such as Smalltalk, a standard project management tool is not the right choice. By contrast, many software cost estimation tools have built-in tables of programming languages and will automatically adjust the estimate based on which language is selected for the application<sup>3</sup>.

---

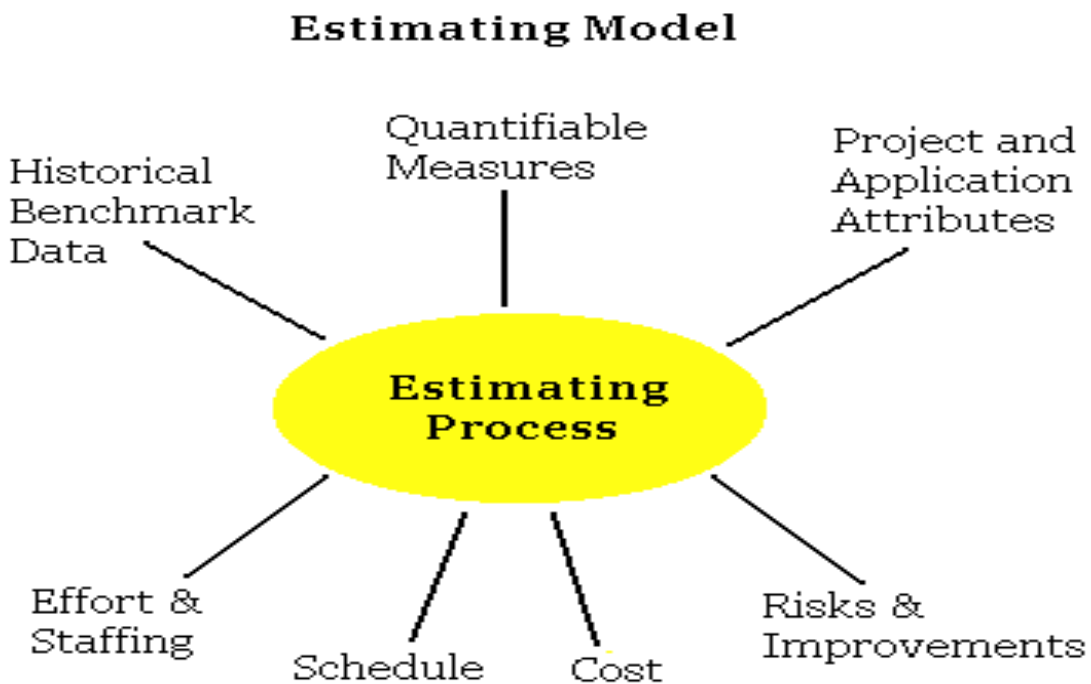
<sup>2</sup> <http://www.idiom.com/~zilla/Work/kcsest.pdf>

<sup>3</sup> <http://www.construx.com/resources/estimate/>

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. As a whole, the software industry doesn't estimate projects well and doesn't use estimates appropriately. We suffer far more than we should as a result and we need to focus some effort on improving the situation. Under-estimating a project leads to under-staffing it (resulting in staff burnout), under-scoping the quality assurance

Effort (running the risk of low quality deliverables), and setting too short a schedule (resulting in loss of credibility as deadlines are missed). For those who figure on avoiding this situation by generously padding the estimate, over-estimating a project can be just about as bad for the organization! If you give a project more resources than it really needs without sufficient scope controls it will use them. The project is then likely to cost more than it should (a negative impact on the bottom line), take longer to deliver than necessary (resulting in lost opportunities), and delay the use of your resources on the next project<sup>4</sup>.

5



*Figure of estimating model 1*

---

<sup>4 4</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.html>

<sup>5</sup> <http://www.qpmg.com/project.htm>

## **2. REQUIREMENT**

A Software Requirements is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies *at a particular point in time* (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

1. We evaluate the impact of your current requirements practices on your business and projects.
2. Then we develop a focused, custom solution - in line with your company goals - that you can use repeatedly. Our specialists conduct workshops, collaborate with you to develop new practices, or deliver interactive training sessions, depending on your need.
3. Then we measure the success and help you reallocate the resultant effort savings<sup>6</sup>.

Although staffing, effort, costs, and schedules are all important for the final estimate, a typical place to start estimating is with staffing levels. There are significant variations in staffing levels based on team experience, application size, reusable materials, and other factors<sup>7</sup>.

## **3. Software cost Estimation**

The four basic steps in software cost estimation are:

- 3.1 Estimate the size of the development product. This generally ends up in either Lines of Code (LOC) or Function Points (FP), but there are other possible units of measure. A discussion of the pros & cons of each is discussed in some of the material referenced at the end of this report.
- 3.2 Estimate the staff effort in person-months or person-hours.
- 3.3 Estimate the schedule in calendar months.
- 3.4 Estimate the project cost in dollars (or local currency)

---

<sup>6</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.html>

<sup>7</sup> <http://www.softwaretechnews.com/stn7-2/reifer.html>



### 3.1 Estimating size

An accurate estimate of the size of the software to be built is the first step to an effective estimate. Your source(s) of information regarding the scope of the project should, wherever possible, start with formal descriptions of the requirements - for example, a customer's requirements specification or request for proposal, a system specification, a software requirements specification. If you are [re-]estimating a project in later phases of the project's lifecycle, design documents can be used to provide additional detail. Don't let the lack of a formal scope specification stop you from doing an initial project estimate. A verbal description or a whiteboard outline is sometimes all you have to start with. In any case, you must communicate the level of risk and uncertainty in an estimate to all concerned and you must re-estimate the project as soon as more scope information is determined<sup>8</sup>.

Two main ways you can estimate product size are:

1) By analogy. Having done a similar project in the past and knowing its size, you estimate each major piece of the new project as a percentage of the size of a similar piece of the previous project. Estimate the total size of the new project by adding up the estimated sizes of each of the pieces. An experienced estimator can produce reasonably good size estimates by analogy if accurate size values are available for the previous project and if the new project is sufficiently similar to the previous one.

2) By counting product features and using an algorithmic approach such as Function Points to convert the count into an estimate of size. Macro-level

<sup>8</sup> <http://www.softwaretchnews.com/stn7-2/reifer.html>

“product features” may include the number of subsystems, classes/modules, methods/functions. More detailed “product features” may include the number of screens, dialogs, files, database tables, reports, messages, and so on<sup>9</sup>.

### **3.3 Estimating staffing**

Once you have an estimate of the size of your product, you can derive the effort estimate. This conversion from software size to total project effort can only be done if you have a defined software development lifecycle and development process that you follow to specify, design, develop, and test the software. A software development project involves far more than simply coding the software – in fact, coding is often the smallest part of the overall effort. Writing and reviewing documentation, implementing prototypes, designing the deliverables, and reviewing and testing the code take up the larger portion of overall project Effort. The project effort estimate requires you to identify and estimate, and then sum up all the activities you must perform to build a product of the estimated size.

There are two main ways to derive effort from size:

1) The best way is to use your organization’s own historical data to determine how much effort previous projects of the estimated size have taken. This, of course, assumes (a) your organization has been documenting actual results from previous projects, (b) that you have at least one past project of similar size (it is even better if you have several projects of similar size as this reinforces that you consistently need a certain level of effort to develop projects of a given size), and (c) that you will follow a similar development lifecycle, use a similar development methodology, use similar tools, and use a team with similar skills and experience for the new project<sup>10</sup>.

2) If you don’t have historical data from your own organization because you haven’t started collecting it yet or because your new project is very different in one or more key aspects, you can use a mature and generally accepted algorithmic approach such as Barry Boehm’s COCOMO model or the Putnam Methodology to convert a size estimate into an effort estimate. These models have been derived by studying a significant number of completed projects from various organizations to see how their project sizes mapped into total project

---

<sup>9</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.htm>

<sup>10</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.htm>

effort. These “industry data” models may not be as accurate as your own historical data, but they can give you useful ballpark effort estimates.<sup>11</sup>

#### **4. Estimating Cost**

The fundamental equation for estimating the cost of a software activity is simple in concept, but very tricky in real life:

$$\text{Effort} \times (\text{Salary} + \text{Burden}) = \text{Cost}$$

A basic problem is that software staff compensation levels vary by about a ratio of 3-to-1 in the United States and by more than 10-to-1 when considering global Compensation levels for any given job category. For example, here in the United States there are significant ranges in average compensation by industry and also by geographic region. Programmers in a large bank in mid-town Manhattan or San Francisco will average more than \$80,000 per year, but programmers in a retail store environment in the rural South might average less than \$45,000 per year. There are also major variations in the burden rates or overhead structures that companies apply in order to recover expenses such as rent, mortgages, taxes, benefits, and the like. The burden rates in the United States can vary from less than 15 percent for small home-based enterprises to more than 300 percent for major corporations. When the variance in basic staff compensation is compounded with the variance in burden rates, the overall Cost differences are notable indeed.

There are many factors to consider when estimating the total cost of a project. These include labor, hardware and software purchases or rentals, travel for meeting or testing purposes, telecommunications (e.g., long distance phone calls, video-conferences, dedicated lines for testing, etc.), training courses, office space, and so on. Exactly how you estimate total project cost will depend on how your organization allocates costs. Some costs may not be allocated to individual projects and may be taken care of by adding an overhead value to labor rates (\$ per hour). Often, a software development project manager will only estimate the labor cost and identify any additional project costs not considered “overhead” by the organization. The simplest labor cost can be obtained by multiplying the project’s effort estimate (in hours) by a general Labor rate (\$ per hour). A more accurate labor cost would result from using a specific labor rate for each staff position (e.g., Technical, QA, Project Management, Documentation, Support, etc.). You would have to determine what percentage of total project effort should be allocated to each position.

---

<sup>11</sup> <http://www.spc.ca/downloads/resources/estimate/estbasics.pdf>

## 5. Estimating schedule

The third step in estimating a software development project is to determine the project schedule from the effort estimate. This generally involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (this is the “staffing profile”). Once you have this information, you need to lay it out into a calendar schedule. Again, historical data from your organization’s past projects or industry data models can be used to predict the number of people you will need for a project of a given size and how work can be broken down into a schedule. If you have nothing else, a schedule estimation rule of thumb [McConnell 1996] can be used to get a rough idea of the total calendar time required: Schedule in months = 3.0 \* (effort-months) <sup>1/3</sup> Opinions vary as to whether 2.0 or 2.5 or even 4.0 should be used in place of the 3.0 value – only by trying it out will you see what works for you<sup>12</sup>.

Estimating software schedules has been a troublesome topic because most large software projects tend to run late. Close analysis of reported schedule errors indicates three root causes for missed schedules:

- 1) conservative or accurate schedule projections are arbitrarily overruled by clients or senior executives,
- 2) creeping requirements are not handled proactively,
- 3) early quality control is inadequate, and the project runs late when testing begins.

Formal schedule estimation is an area where cost estimation tools and project management tools frequently overlap. Often the cost estimation tool will handle high-level scheduling of the whole project, but the intricate calculations involving dependencies, staff availability, and resource leveling will be done by the project management tool. A basic equation for estimating the schedule of any given development activity follows: Effort/Staff = Time Period Using this general equation, an activity that requires eight person-months of effort and has four people assigned to it can be finished in two calendar months, i.e.: 8 Months/4 People = 2 Calendar Months In real life, schedule estimating is one of the most difficult parts of the software estimation process. Many highly Complex topics must be dealt with such as the following:

---

<sup>12</sup> <http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.htm>

- An activity's dependencies upon previous activities.
- Overlapping or concurrent activities.
- The critical path through the sequence of activities.
- Less than full-time staff availability.
- Number of shifts worked per day.
- Number of effective work hours per shift.
- Paid or unpaid overtime applied to the activity.
- Interruptions such as travel, meetings, training, or illness.
- Number of time zones for projects in multiple cities. It is at the point of determining software schedules when software cost estimation tools and project management tools come together. The normal mode of operation is that the software cost estimation tool will handle sizing, activity selection, effort estimation, cost estimation, and approximate scheduling by phase or activity. Then the software cost estimation tool will export its results to the project management tool for fine tuning, critical path analysis, and adjusting the details of individual work assignments.

## **6. Quality and risk**

### **Software Risk Management and Cost Estimation**

Project and program managers require accurate and reliable cost estimates to allocate and control project resources, and to make realistic bids on external contracts. They also need to determine whether, for a given system size, the cost of a prospective project is too high to redefine the project or put in place appropriate contingency plans (i.e., cost risk assessment). Different techniques for cost estimation have been discussed in the literature: algorithmic and parametric models, expert judgment, analogy, and rules of thumb to name a few. However, they suffer from a number of drawbacks. This tutorial is designed to introduce you to various cost estimation techniques and how you can employ them in the context of software project risk management<sup>13</sup>.

Achieving high productivity at the expense of quality is the wrong approach. Yet, many organizations have tried to do this in the commercial world at the customer's expense. To prevent this from occurring, we must view productivity from a quality point-of-view. To accomplish this feat, we must view productivity by normalizing the quality of the products generated upon delivery to the field using metrics like so many software errors per thousand ESLOC.

---

<sup>13</sup> <http://www.estec.esa.nl/conferences/fontainebleau/>

While there have been many papers published on quality metrics and measurement, few of them have put such error baselines into the public domain. There are many reasons for this omission. First, there is a great deal of confusion over just what an error is<sup>14</sup>.

## **7. Conclusion**

Software cost estimation is simple in concept, but difficult and complex in reality. The difficulty and complexity required for successful estimates exceed the capabilities of most software project managers. As a result, manual estimates are not sufficient for large applications above roughly 1,000 function points in size.

Commercial software cost estimation tools can often outperform manual human estimates in terms of accuracy and always in terms of speed and cost effectiveness. However, no method of estimation is totally error free. The current *best practice* for software cost estimation is to use a combination of software cost estimation tools coupled with software project management tools, under the careful guidance of experienced software project managers and estimation specialists.

## **References :**

<http://www.ecfc.u-net.com/cost/models.htm>  
<http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.pdf>  
<http://www.qpmg.com/projest.htm>  
<http://www.softwaretechnews.com/stn7-2/reifer.html>  
<http://www.spc.ca/downloads/resources/estimate/estbasics.pdf>  
<http://www.stsc.hill.af.mil/crosstalk/2002/06/jones.html>  
<http://www.estec.esa.nl/conferences/fontainebleau>  
<http://www.softwaretechnews.com/stn7-2/reifer.html>  
<http://www.idiom.com/~zilla/Work/kcsest.pdf>  
<http://www.construx.com/resources/estimate/>

---

<sup>14</sup> <http://www.softwaretechnews.com/stn7-2/reifer.html>