



Considerations in Project Cost Estimation

Prepared by

Randall Colville

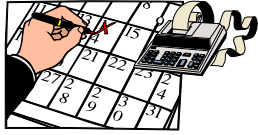
July 2008



Considerations in Project Cost Estimation

Table of Contents

Introduction.....	1
Estimation Techniques	2
Experience.....	2
Expert Judgement	2
Wideband Delphi Approach	2
Algorithm.....	3
Lines of Code.....	5
Function Points	6
Cost Estimation Approach Evaluation	8
Expert Judgement.....	8
Advantages.....	8
Disadvantages	8
Wideband Delphi Approach	8
Advantages.....	8
Disadvantages.....	8
Lines of Code	8
Advantages.....	8
Disadvantages	9
Function Points.....	9
Advantages.....	9
Disadvantages	9
Criteria 10	
Approach Evaluation	11
Recommendation	13
Introduction.....	13
Historical Data	13
Selection of Cost Model Tool	14



Considerations in Project Cost Estimation

Introduction

It is the responsibility of the project manager to make accurate estimations of size, effort and costs. There is a need for accurate estimates of effort and size at a very early stage in a project. However, when software cost estimates are done early in the software development process, the estimate can be based on wrong or incomplete requirements.

Size estimation is the first and most important part of the project planning phase. The size estimate dictates the rest of the project plans, including the cost quote, time, and scheduling of human and matching resources to complete the task.

A software cost estimate process is the set of techniques and procedures that an organization uses to arrive at an estimate. Why is cost estimation difficult or error prone?

- Cost estimation can require a significant amount of effort to perform it correctly
- Cost estimation is often done hurriedly, without an appreciation for the required project effort
- Experience is required for developing estimates, especially for larger projects
- Human bias (e.g., an estimator is likely to consider how long a certain portion of the system would take, and then to extrapolate this estimate to the rest of the system, ignoring the non-linear aspects of software development)

The causes of poor and inaccurate estimation include:

- Imprecise and drifting requirements
- New software projects are nearly always different from the previous
- Software practitioners don't collect enough information about past projects
- Estimates are forced to match the available resources and preset deadlines

Project estimates must include costs, schedule, and effort for the defined requirements. Even if cost is not an issue with respect to project funding, it is an important variable in assessing the real costs involved in development efforts. As the project unfolds, the actuals must be compared against the estimates, and remedial action taken when significant deviation is noted. This will assist senior management in tracking each project and ensuring that issues are resolved quickly, to the mutual satisfaction of the involved parties.



Considerations in Project Cost Estimation

Estimation Techniques

There are two main common types of estimation techniques: experience and algorithms. Before any estimation technique can be used, there must be some level of defined requirements. More detail in the requirements definition will result in more accurate estimates.

Experience

Expert Judgement

Expert Judgement is the most commonly used estimation method. About 62% of estimators/organizations use this intuitive technique. It involves consultation with one or more local experts who are knowledgeable about the development environment or application domain to estimate the effort required to complete a software project.

This method relies heavily on the experience of estimators and their knowledge in similar development environments and historically maintained databases on completed projects and the accuracy of these past projects. Typically, however, estimators do not refer to previous projects as it is too difficult to access or the expert cannot see how the information would help in the accuracy of the estimate. The majority of estimators use their memories of previous projects. A particular strength of using an expert is that they can raise unique strengths and weaknesses of the local organizational characteristics. This method is subjective and unstructured.

Wideband Delphi Approach

The Wideband Delphi technique gathers the opinions of a group of experts with the aim of producing an accurate unbiased estimate. It is a structured technique of expert judgement and is essentially a form based technique involving a multistep procedure

- a. Experts are issued the specification and estimation form by the coordinator
- b. A group meeting is held to discuss the product and estimation issues
- c. Experts produce an independent estimate
- d. Estimates are returned, indicating the median estimate and the expert's personal estimate
- e. Another group meeting is held to discuss results
- f. Experts prepare a revised independent estimate
- g. Steps c-f are repeated until a consensus is reached by the panel of experts



Considerations in Project Cost Estimation

This method is an attempt to remove biases and politics from an estimate, as the experts do not communicate about their particular estimate and the method filters out extreme opinions. The group discussion ensures that any estimation issues are not overlooked.

Algorithm

Algorithmic methods all have difficulty in producing accurate estimates in the early parts of a project. At the beginning of a project relatively little information is known. Generally, almost universally, problems with the requirements are blamed for inaccuracies with estimates. The following are some of the reasons for a lack of adequate requirements:

1. Users do not understand their requirements during the early stages of a project. Software projects are often undertaken when there is a recognition that a problem exists but no clear idea of the real problem, or the solution to the problem. Yet at this early stage of problem recognition, someone is expected to be able to write a requirement specification at a sufficiently detailed level such that an accurate cost estimate can be made. Estimates can be made at this stage, but it must be recognized that they are inherently inaccurate.
2. Requirements creep. As projects progress and the knowledge of the problem increases, users (and developers) request more and more features and changes to be included in the project. Thus, over the development of the project, new features work their way into the requirements.
3. Correct and complete requirements for complex systems are impossible to achieve. A complete statement of the requirements cannot be defined before development begins. Unless the system being developed is almost identical to a previously developed system, the requirements will invariably be wrong and/or incomplete. As a project evolves, users and developers gain a better understanding of the problem and of the solutions. As a better understanding of the problem being solved is gained, the requirements evolve. Since the requirements are probably wrong or incomplete at the beginning of a project, it is unlikely that the estimates based on those requirements will be accurate.
4. Requirements written by people who do not know how to write requirements. Many organizations have run into problems with requirements because they did not know how to write requirements.
5. Long development time, leading to requirements that are obsolete before the system is delivered. The rate of change in technology is so fast that any attempts to predict what the technology will be in a few years are doomed to failure. As the technology changes, so do the range of solutions to problems, and the users' expectations of the solutions. The customer is dissatisfied because the product does not satisfy the new requirements.

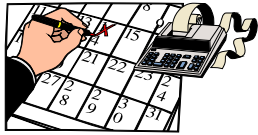


Considerations in Project Cost Estimation

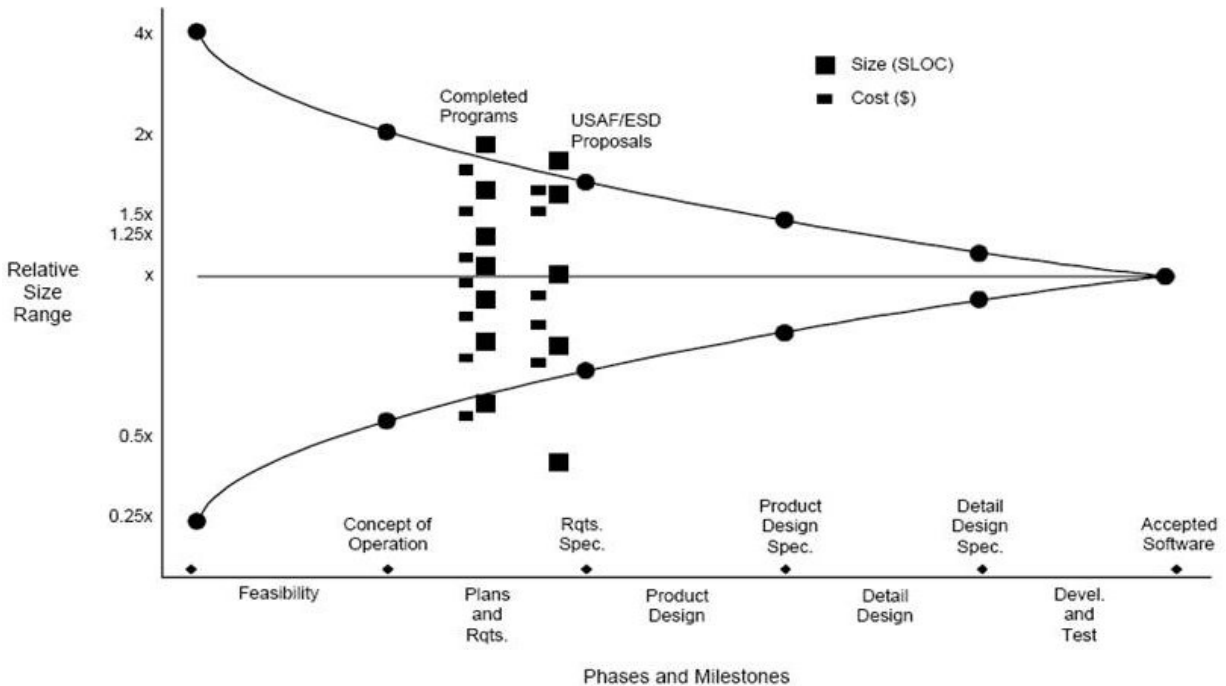
6. Large staff turnover for end users, resulting in changing requirements as new people arrive. Developing software systems requires a consistent users' base throughout the development cycle. If the users' base changes too frequently, requirements continually change, and it is difficult for developers to obtain consistent answers and comments from the end users.

As the project progresses, more detail is obtained and estimation becomes more accurate. For this reason, algorithmic methods typically use confidence intervals as part of their approach. A *confidence interval* gives a range of values which includes the estimated value (cost, duration, and/or effort about a project). The width of the confidence interval shows the level of certainty about the estimated value. A very wide interval may indicate that more data should be collected before anything very definite can be said. A graphic example of how the width of confidence intervals decrease from project initiation to project completion (as more detail becomes known about the project requirements) is shown below¹.

¹ *COCOMO II Model Definition Manual*, The Center for Software Engineering at USC.



Considerations in Project Cost Estimation



Lines of Code

One of the most effective direct measurements for software size is a count of the number of lines of source code that have been implemented. The time and effort involved in implementing a software program is linearly related to the software size in lines of code. This linear relationship is strongest for lower level programming languages (e.g., C), but still exists for the highest level fourth generation languages.

Lines of Code can be declared as **new**, **modified**, **reused**, and as a **base** to a program which is due to be modified. Furthermore, new code can be declared as **reusable**.

Lines of Code can also be easily applied to company statistics, such as productivity (LOC per programmer hour), defect rates (number per 1000 LOC). Lines of Code can also be used to determine the effectiveness of product reviews and inspections.



Considerations in Project Cost Estimation

Function Points

A Function Point value is a relative score that measures size of software in terms of its delivered functionality. Function points measure software size in terms of what type of transactions are processed between a software program and external entities like users, databases, and hardware devices. All these transactions constitute a program's functionality. The same productivity statistics involving LOC can be applied to function points.

The entry point for counting function points to a software project is a detailed requirements statement. It can be done without the need for a conceptual program design. The key to counting function points is the identification of user data groups. Various transactions will be performed on these data groups. These transactions must be identified, each of which falls into one of the following categories:

- a. **External Inputs:** The entry of data into the application domain from external entities constitutes an external input. External inputs can come from a user or another application. Input dialogs, interactive user input, as well as inputs from other hardware like a disk drive or modem are external inputs.
- b. **External Outputs:** The exit of data from the application domain to an external entity is an external output. The destination of an external output may be a user readable report on a screen or printer. Outputs to other applications, auxiliary storage or other hardware also constitute external output.
- c. **External Queries:** When an input or request generates an immediate output or answer, it constitutes an external query. They may come from users or other applications. Menus, embedded computer inquiries and online context sensitive help constitute external queries.
- d. **External Interface Files:** An external interface file is any major aggregate of data that enters or leaves the domain of an application. This includes shared files and messages.
- e. **Internal Logical Files:** Each major aggregate of user data stored within the application domain is an internal logical file.

Once raw function points are computed, influences on the application (environmental or application related) are questioned to incorporate as part of the overall estimate calculation:

- Data communication
- Distributed functions
- Performance objectives



Considerations in Project Cost Estimation

- Heavily used operational configuration
- Transaction rate
- On-line data entry
- End-user efficiency
- On-line updating
- Complex processing
- Reusability
- Ease of installation
- Ease of operations
- Multiple sites
- Facilitate changes



Considerations in Project Cost Estimation

Cost Estimation Approach Evaluation

An initial review of the the approaches show the following advantages and disadvantages:

Expert Judgement

Advantages

- Uses experience on past projects to assess factors on the new project
- Assessment of representativeness
- Adapt to exceptional circumstances

Disadvantages

- No better than the expertise and the objectivity of the estimator
- The estimate can be biased
- Incomplete recall (faulty history)

Wideband Delphi Approach

Advantages

- Removes the politics from an estimate
- The estimate is not usually biased
- Group discussion ensures estimation issues are not overlooked

Disadvantages

- Can be time consuming, as there are many people involved and meetings must be scheduled and attended
- The panel needs to be very experienced

Lines of Code

Advantages

- LOC is a direct measure of software size
- LOC is easily counted
- Counting can be easily automated
- Simplicity of categorizing code into reusable, new, changed, and reused



Considerations in Project Cost Estimation

Disadvantages

- Lack of agreement on standards in the software industry
- LOC counts vary with language
- Difficulty in visualizing Lines of Code early in the development
- The same automatic counter may not work for all programming styles in the same language
- A Line of Code is a strange term to most clients
- Some calculated statistics using LOC can be misleading

Function Points

Advantages

- They are appropriate and better seen in the early development phases
- A consistent and abstract measure, independent of language and design
- More easily understood by the client
- High support from user groups
- Less misleading when used for productivity data

Disadvantages

- Inconsistent counts are produced by inexperienced people. Training is an asset
- No reflection of language or programming style, which could affect productivity estimates
- Lack of historical data limits room for improvement of estimation
- Difficulty in counting function points for completed products for the sake of developing historical data
- Function point counting is very labor intensive
- The practice is subject to common bias
- Function point counting is difficult to automate



Considerations in Project Cost Estimation

Criteria

Estimating project costs requires repeatable processes. A repeatable process:

1. Must be objective
2. Applies throughout the project
3. Allows for all development phases
4. Is usable for software components
5. Easily adjusts for later projects
6. Evaluates accuracy
7. Permits statistical analysis
8. Is capable of being automated
9. Estimates traditional programming languages, data oriented development, and object oriented development
10. Is easy to use

These ten criteria are used for evaluating the approaches. They can also be used to evaluate specific cost estimation applications. The end result of this evaluation will be the production of a short list of tools appropriate to your organization.



Considerations in Project Cost Estimation

Approach Evaluation

Approach	Criteria										
	Objective	Full Project	All Development Phases	S/W Components	Later Projects	Accuracy Evaluation	Statistical Analysis	Automated	Traditional Development	New Development Approaches	Easy to Use
Experience											
Expert Judgment	N	Y	Y	Y	Y	Y	N	N	Y	N	Y
Wideband Delphi	Y	Y	Y	Y	Y	Y	N	N	Y	N	N
Algorithm											
Lines of Code	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Function Point	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N



Considerations in Project Cost Estimation

Algorithmic cost modeling is the most objective means of estimating project size. In the process, costs are analyzed using mathematical formulas linking costs or inputs with metrics to produce an estimated output. The formulas used in a formal model arise from the analysis of historical data. The accuracy of the model can be improved by calibrating the model to a specific development environment, which basically involves adjusting the weights of the metrics.

The Lines of Code (LOC) approach appears to be the best approach. It is objective, can be automated, and is easy to use. Approaches based on Function Points, while objective, are difficult to automate and are not easy to use. In addition, LOC models have an extensive historical database that are released annually to the public. This ensures the availability of a more accurate predictive model for estimating software project costs.

Current LOC approaches can estimate project size independent of the programming language used. An early design model is used in the early stages of a project when very little may be known about the size of the product to be developed, the nature of the target platform, the nature of the personnel to be involved in the project, or the detailed specifics of the process to be used. A post-architecture model is used when a software life-cycle architecture has been developed. This model takes advantage of the greater knowledge available later in the development process when requirements have been detailed.



Considerations in Project Cost Estimation

Recommendation

Introduction

In systems development, it is not uncommon to experience differences of 100% to 300% between estimates for one project, *even when the estimators are using proven project estimation techniques*. In fact, people get cynical when estimating projects, often taking a best guess and multiplying it by two or three (or even more).

Some people believe that software development will, by its nature, defy efforts to measure it. It is stated that software is an art, not a science, and the process of creating software is too complex and *creative* to be measured effectively.

To estimate and monitor projects effectively, the projects must first be measured. Both existing and new applications need to be measured. Measuring some software projects, however, can be easier than measuring other types of software projects. To understand why, look at what is involved in doing accurate size estimation for projects using multiple development platforms.

Estimation must take into consideration the different speeds at which different programmers code and make adjustments for different development platforms. Programmers just learning a language are less productive than more experienced programmers. Coding in C is faster than coding in assembler. Using a code generator for user interface programs is faster than coding them by hand. Estimation must also accurately measure a system's size. The measurement method chosen often needs to be applied across all platforms, which introduces further variances in the estimation process.

Accurately estimating projects is frustrated by the following factors:

1. Variations in programmers' skill
2. Variations in development platforms
3. Variations in project scope and size
4. Variations in project team size and organization

Historical Data

To be able to estimate projects effectively, you need a good record of previous projects. For this reason, the first step in estimating projects is maintaining an up-to-date and accurate record of prior projects. The more completed projects recorded, the more accurate the estimation will be. The larger the population sample, the less error the estimate will contain. For the estimates to be worthwhile, the database should contain at least 15 recorded projects.



Considerations in Project Cost Estimation

The project plan for the estimated project should be at a detailed, component-level. Time spent on each component must be tracked. The project estimate needs to be compared with actual time totals and the schedule and/or designs adjusted as needed. The project should be added to the project database when it is completed.

Most organizations do not have a good historical database of projects that can be used for estimation. They may start by using project management tools, such as Microsoft Project or ABT Project Workbench. However, the typical lack of discipline and consistency in capturing project information means that the data already contained in their project repository cannot be totally trusted. Once consistent and trusted data is captured in the repository it can be used as a source to develop project estimates. As more projects are captured, the estimates should become more accurate and precise. In the meantime, organizations need to access external databases that contain historical industry standards.

Selection of Cost Model Tool

An in-depth evaluation of cost estimation tools must be completed. The following is a list of the criteria that should be used in the tool review. Each of the criteria should be weighted *a priori* according to its importance to the organization.

- Cost
- Support (amount of time, warranty, support type)
- Ease of use
- Documentation (user's manual, system documentation)
- Training (cost, number of people, onsite/offsite, internal-power users)
- Management reporting (graphics, interface with word processors)
- Required detail (little available information, requirements defined in detail)
- "What-if" scenarios
- Longevity of vendor
- Credibility of vendor's clients
- Risk management
- Interface with the organization's project management tools (direct, indirect)
- Capability of handling multiple releases/projects
- Availability of up-to-date industry data
- Platform (mainframe, individual workstation, server)
- Security and backups
- Availability of product upgrades
- Ability to apply both LOC and Function Points
- Assistance to project tracking and oversight activities



Considerations in Project Cost Estimation

- Maintenance and comparison of performance metrics and benchmarks
- Maintenance of internal and external historical data
- Calibration of estimation based on historical data
- Speed of processing