

## A Case For Software Estimation

A recent search of the World Wide Web identified over 2100 sites that describe over 5000 reasons that software projects fail, ranging from the poor use of technology to lack of communication to management inattention. While these factors can contribute to the failure of a project, the most important reasons include (1) a lack of understanding of the requirements of a project, (2) insufficient time or discipline to plan the project properly from the first day, and (3) a loss of focus when the project is underway.

Adequately planning a software development project requires knowledge of at least two key values: the time and effort. Ad-hoc planning strategies attempt to estimate these two values straight away as functions of past experience and expert judgment. Structured methods for project planning attempt to calculate these values as functions of estimated product-related and project-related values (parameters). A key parameter present in the vast majority of these methods is software size. Assuming that size is dynamic and that planning (including replanning or re-baselining) is a process that should continue throughout the life of the project implies, for structured planning strategies, that size estimation itself should be a continuous process.

Almost all software-intensive projects start with the potential for some degree of failure. The first risk of any software project is rooted in the initial estimate used to forecast needed resources. The basic equation traditionally used to estimate the project effort is:

$$\text{Effort} = \frac{(\text{Size X Complexity})}{\text{Productivity}}$$

At the beginning of any project the three variables in the equation are unknown. Over the years, estimation methods and models that significantly lower the initial estimation risks have been developed. However, failure to realize the potential imprecise nature of initial estimates and effectively manage and control the risks associated with them certainly is a major contributor to downstream problems, including project failure.

If used properly, estimates can provide basic constraints that potentially limit the options available when planning a project. The estimates also identify the resource limitations that must be considered when scheduling a project, which in turn may dictate the selection of methods and tools. Budget and resource constraints affect a project's schedule, the phasing of activities, the logical relationships of the work activities, and the structuring and packaging of the products. In addition, the estimate determines the options available to increase the quality of the products, either precluding or enabling the use of practices such as structured inspections or enhanced testing.

Software estimation methods have been applied with varying degrees of success, to small and large projects. Popular estimation methods include:

Analogy; Compare project with past similar projects

Expert Judgment; Consult with one or more experts

Top-down estimation; A hierarchical decomposition of the system into progressively smaller components is used to estimate the size of a software component

Bottom-up estimation; Individuals assess each component; estimates are summed to calculate the total estimate

Design to Cost; Uses expert judgment to determine how much functionality can be provided for a given budget

Parametric models; Perform overall estimate using design parameters and mathematical algorithms

As discussed earlier, one of the main inputs to software development effort is software size. If size is estimated correctly, the effort estimate will be realistic and will translate to a realistic cost estimate. In the physical world, size is a measure of volume or mass. In the software world, size is a measure of functionality. Various expressions of size include measures such as lines of code, number of features, or functions, function points and their derivatives, SEER function-based sizing, use cases, and objects. Additionally, the amount of rework of existing systems is a key size measure for modifications. Software size is the main driver of software development effort, cost and schedule via parametric models, and base productivity measures.

An effective software estimate provides the information needed to design a workable software development plan. Many elements are involved in determining the structure of a project, including requirements, architecture, quality provisions, and staffing mix. Perhaps the most important element in the success or failure of a project is the initial estimate of its scope, in terms of both the time and cost that will be required. The initial estimate drives every aspect of the project, constrains the actions that can be taken in the development or upgrade of a product, and limits available options. Although many people think they can estimate project scope on their engineering or management experience, most off-the-cuff estimates are incorrect and are most often based on simple assumptions and over-optimism, or worse, are made to accord with what others want to hear. Needless to say, such estimates often lead to disaster.

A sound and reliable estimate is the linchpin of a realistic software project plan. The assumptions, requirements, and projections on which the estimate is based allow you to plan a project or define a product with a realistic understanding of the limits that constrain what can actually be done. A software estimation process that is integrated with the software development process can help projects establish realistic and credible plans to implement the project requirements and satisfy commitments. It also can support other management activities by providing accurate and timely planning information.

Any software project that wants to be successful requires realistic, credible plans that describe how the project will meet specified objectives, and credible plans must be based on accurate estimates of the required effort, duration, and cost of the project. Realistic plans will also describe how the resources that are required to undertake the initiative in accordance with the schedule will be secured.

The problems inherent in accurately estimating the resources required to develop software have been understood and have received significant attention for the past 20 years, and many tools and methods have been developed to address them. As a result, many people have high expectations that the software delivery process has been improved such that these problems have been removed altogether. But the tools are not used widely enough, and overzealous managers still attempt to misuse them to justify unreasonable plans that result in insufficient resources to develop a quality product on time and within budget.

A software project estimate is the most knowledgeable statement that can be made at a particular point in time regarding effort, cost, schedule and risk. A complete estimate covers definitions, uncertainties, ground rules, and assumptions.

Many project managers have unrealistic expectations about estimates. The definition of the verb *estimate* is to produce a statement of the approximate value of some quantity. Estimates are based upon incomplete, imperfect knowledge and assumptions about the future. For these reasons, many estimates of software costs tend to be too low due to omissions of important product functions and project activities. Most importantly, however, all estimates have uncertainty. There is no such thing as a precise, single-value estimate. Managers should always ask how large the uncertainty of an estimate is. A manager can use the size of this uncertainty in conjunction with other factors such as perceived risks, funding constraints, and business objectives to make decisions about a project.

Cost estimates are projections of required effort, time and staffing levels. Because all estimates, particularly those made at the beginning of a project, are based on assumptions, they should be considered probabilistic. Cost estimates in particular should provide a range with an indication of accuracy, i.e., least, probable, and most, with the least and most values representing the upper and lower bounds of the projected cost.

Ideally an estimate should be produced using a process, such as the ten-step process described in the book *Software Sizing, Estimation, and Risk Management* by Daniel D. Galorath and Michael W. Evans.