# Contents

# Chapter 33: Estimation for Software Projects

Software project management comprises of a number of tasks before the actual project can begin. Project managers estimate the time it will take to work on a project with collaboration with the software team and the resources that would be required to complete the project etc. Project planning is an integral part of software development, it correlates with the creation of blue print before we actually start building a house.

Firstly the team defines the scope of the work, in later stages the scope is decomposed into functional requirements and mapped to timelines for the project. Project planning enables software engineers to efficiently work on projects and estimate how much time it will take to execute a certain task.

In an excellent guide to "software project survival," Steve McConnell

[McC98] presents a real-world view of project planning:

Many technical workers would rather do technical work than spend time planning.

Many technical managers do not have sufficient training in technical management

to feel confident that their planning will improve a project's outcome.

Since neither party wants to do planning, it often doesn't get done. But failure to plan is one of the most

critical mistakes a project can make . . .

effective planning is needed to resolve problems upstream [early in the project] at

low cost, rather than downstream [late in the project] at high cost. The average project

spends 80 percent of its time on rework—fixing mistakes that were made earlier

in the project.

## Observations on Estimations

The Estimation for the time taken to complete a project begins with an initial commitment, generally the commitment is revised several times over the life cycle of the project but it provides the starting point. The time taken to complete a project also depends upon the requirements, complexity, and the experience of the team working on the project. Estimating the time taken to complete a project requires a lot of experience along with good past data to use for predictions.

## Software scope and feasibility

Software scope defines the features and functions delivered to the users at the end of the software completion process. Its very important to determine the feasibility of functions and scope of the project before the development team actually start to build the software.

Teams who rush past these steps realize mid-way in the project that the software has come at a standstill, little do they know that it was set up for failure from the start.

## Resources

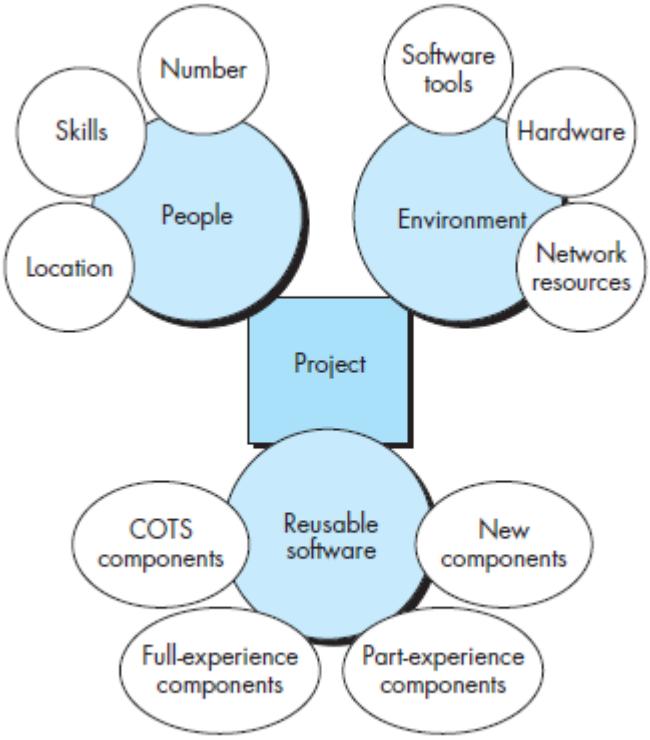


*Figure 1Resources required to build a software*

This stage comprises of determining the various kinds of resources used in the project such as managerial (Project Manager, product owner) as well as technical resources (Software engineer, QA Engineer, DevOps engineer etc.).

The team also determines the reusable/opensource resources to be used in the project as well as the environmental resources required (Specific hardware/software if any) for the project.

## Software project estimation:

There are several methods to estimate the project cost and effort such as waiting till the end of project completion (not very practical), basing the effort and estimated time on the basis of past data (but past data is not always a good measure to predict future developments), use decomposition techniques to estimate the project cost and estimate or use a hybrid approach.

Generally we go with the second, the third option or with a hybrid approach. We don't tend to go with the first option since it is not very practical in real world scenarios.

## Decomposition Techniques

### Software Sizing

This phase is the first challenge for the project planner, the software estimate will only be as good as the 'software size' estimated at the start of the project, in this phase the software is generally broken down into function points (FP), with each FP representing the application domain, functionality delivered etc.

In this approach size of the product can also be measure via LOC (lines of code).

### Problem based estimation

In this phase we make three kind of estimates, optimistic estimate, most likely estimate and a pessimistic estimate. The expected value for the estimate is determined by the formula:

$$S = \frac{s_{opt} + 4s_m + s_{pess}}{6}$$

*Figure 2Formula for calculating software estimates.*

### Process based estimation

This is one of the most used techniques, in process based estimation the processes of the software are decomposed into relatively small set of activities actions.

One the problem functions and processes are modeled we estimate the effort and time (person-months) that will be required to complete the software activity for each software function.

### Reconciling Estimates

The estimation techniques above generally produce different estimation costs, the project planner has to reconcile on the data produced and come up with a single figure cost/estimation for the project. If the estimation of one method varies greatly from the other then the estimation costs should be reevaluated.

## The Make/Buy decision

In a lot of software areas, software engineers are faced with the question whether to buy a software or build it from the ground up. Generally, acquiring a ready made software is much cheaper than making your own software, but in that case other decisions such as whether the bought software will be based on subscription model/one-time buy or some other model etc are involved.

To overcome these challenges software engineers use the following techniques:

## Decision trees

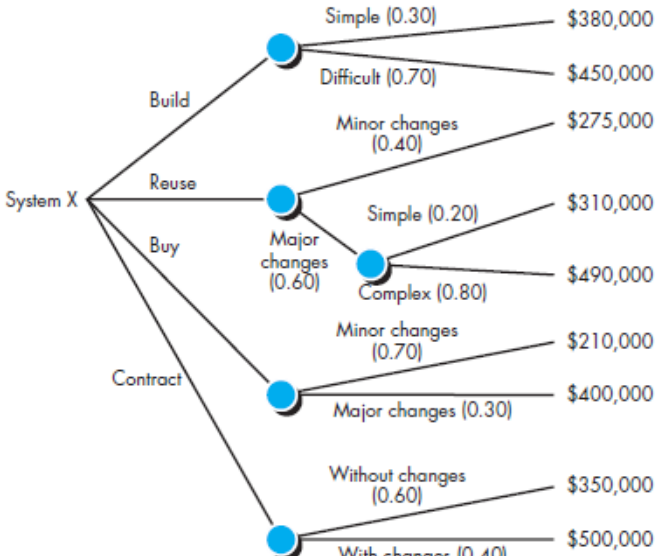


*Figure 3 A decision tree*

Engineers use decision trees and statistical analysis to compute the costs about whether it'll be cheaper to buy an off-the-shelf software or make your own software.

## Outsourcing

The term outsourcing refers to giving the software development work to a third party that does the work at a lower cost, with world becoming increasingly online connected and remote work increasing, the general trend is shifting towards outsourcing.

Overall, the software development industry is extremely competitive at every level and the only way to thrive is to create quality software at a lower cost than your competitors.