

Software Estimation

Why is estimation so important?

Example: You have been asked to bid on painting a 5 room house. How much will this cost and when will it be done? You can't tour the house, your bid has to be in by the end of the day and the house is out of town. How do you do this? Is it even smart to bid?

Questions:

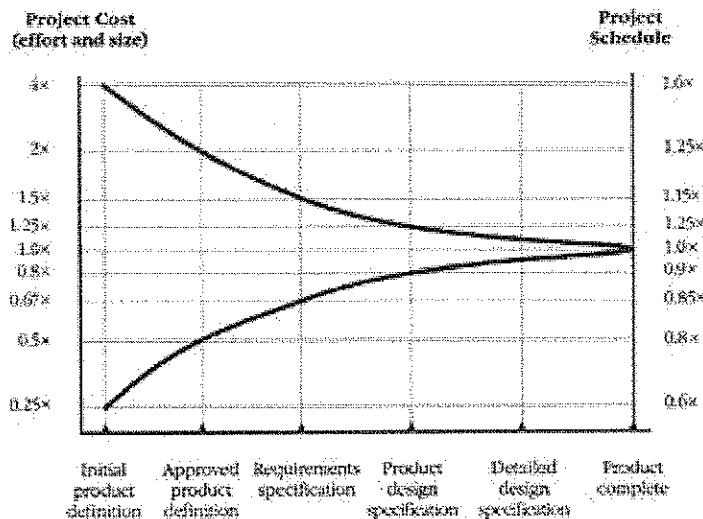
Based on experience 5 rooms costs \$300/room and takes 1 day/room to do, you win the bid with \$1500 in 5 days time.

1. What are some issues you might find when you show up at the job?
2. How likely are you to be on time and within budget?
3. What are some lessons we can learn from this about software development?
4. Is it wise to provide an estimate without first seeing (specifically) what work needs to be done?

Software Development Story

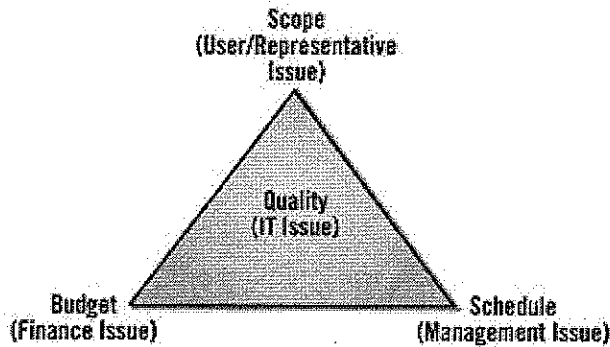
Software development is:

- Like getting fitted for glasses -- a process of gradual refinement. Estimates MUST be refined during the project lifecycle
From Rapid Development by Steve McConnell



- The more detailed and precise the requirements, the better. You have to know what "it" is to build it.

- Based on the Golden Triangle of Software Development
From <http://www.sdmagazine.com/documents/s=826/sdm0303g/>



Questions:

1. Who knows the software development story the best? Programmers, Managers, Customers?
- 2.

Estimation Overview

There are three main components to any estimate:

1. Size (Scope)
2. Effort (Schedule)
3. Cost (Budget)

There is a lot of confusion between *effort* and *duration*. *Effort* represents a unit of work which is then used to calculate a *duration*, which represents time on a schedule. For example, the paint job is based on the idea that it 1 person can paint an average room in a day. So, effort = 1 room/day. If we have 10 people and 10 rooms, it should take a job duration=10 days.

From Rapid Development:

What Do People Usually Mean by "Estimate"?

""[The common definition of estimate is] 'An estimate is the most optimistic prediction that has a non-zero probability of coming true.'

"Accepting this definition leads irrevocably toward a method called what's-the-earliest-date-by-which-you-can't-prove-you-won't-be-finished estimating."

—Tom DeMarco

Questions:

1. How do you define the term estimate?
2. How much value should be put in any estimate?
3. Why is the distinction between effort and duration important?
4. Can we trust that effort can so directly be translated to duration as in the paint example above? Why or why not?

Estimation Tips

1. Avoid off-the-cuff estimates.
2. Allow time for the estimate, and plan it.
3. Use data from previous projects.
4. Use developer-based estimates.
5. Estimate by walk-through. (*average estimation by comparison*)
6. Estimate by categories.
7. Estimate at a low level of detail. (*the more you know, the better you can estimate*)
8. Don't omit common tasks. (*estimation itself, testing, holidays, documentation, slack time, etc.*)
9. Use software estimation tools. (*PERT, Function Point Analysis, Historical, COCOMO, Lines of Code, Table look up, etc.*)
10. Avoid giving precise estimates, at least early on. Use ranges, + and – qualifiers, and confidence levels to balance accuracy and precision.

Questions:

1. What is the best estimation tool to use?
2. What is the most important factor in any estimate?
3. What is the difference between accuracy and precision?