How To Estimate Program Size

In my last article on Cost of Quality, I started out by blithely stating, "Let's say we're going to write a system of 25,000 Lines of Code." Teri (a perceptive reader) called me on it! She wrote:

> *If a new system is built, how do you guess at how many lines of code there will be? You possibly can guess at the number of programs from looking at the requirements but how do you guess the amount of lines involved for each program?*

It's an important question, and doesn't have a quick and easy answer. This was what I told Teri.

Why Lines of Code?

In my article I used Lines of Code (LOC) as the unit for measuring the size of software. Some people embrace this unit and others are adamantly against it. There are a variety of units you can use, including LOC and Function Points. So the first step is to choose the unit you will use to measure the size of your software. A good unit will have these attributes:

- **It correlates with the effort involved** in producing the software, defect rates and anything else we would want to use the estimates for. Both LOC and Function Points correlate nicely with these things.

- **It can be measured objectively in existing code.** LOC is quite easy to measure; LOC counters abound. Function Points cannot be measured using a tool, and counting them manually in existing code is not an exact science.

- **It can be estimated by professionals.** Function Points were designed precisely for estimating systems before they are built. Using LOC for this purpose is less straightforward, but it can be done handily using the techniques I will describe below.

These things lead me to choose LOC as my unit of choice. (And I am not alone.) But it is a professional preference. If you prefer a different unit, then by all means, use it.

After we have chosen a unit of measure, we can use it to do two things: Compile our historical data, and use that data to estimate each new system we will build.

Compiling Historical Data

We all have libraries full of code written by our developers. Those libraries are goldmines! Let's start digging.

1. <u>**Measure your existing software using your chosen unit.**</u> You could measure *each source file*, but this is likely not your best option because there are usually multiple components in each file. Measuring *each component* instead (e.g. each function or class) gives you a lot of data for the next two steps, which is a good thing. It also allows you to be more precise, which will make for better precision in the estimates based on your measurements.

2. <u>**Categorize each component you measured.**</u> The categories need to be things developers can visualize based on requirements (e.g., an I/O function or a certain type of a class). How many

categories should you have? Too many will make it difficult for developers to slot individual components. Too few will give you less precision. My recommendation: you should have around a dozen categories. Of course, the size of your library will be a factor, because you need to have several examples of each category for the next step.

3. **Document the size ranges for each category.** If your data set is large enough to use statistical techniques, compute the Mean and Standard Deviation for each category. For smaller data sets (the norm when you are just getting started), just capture the Min, Mean, and Max sizes for each category.

Now you have a list of a dozen or so categories of software component that you find in your systems, and for each, you have basic size ranges. This gives you the data you need to estimate a new project.

Estimating the Size of a System

This estimating process can only be accomplished by development professionals. (Sorry! Project Managers must stand by and watch, or facilitate!) It works best as a workshop in which the whole development team can discuss the requirements and come to consensus on the estimates. If the team has not yet been assigned, the estimates should be prepared by at least two competent development professionals. Based on the requirements, the developers compile two lists:

1. **Components that will change.** If the project is to change or enhance an existing system, or if a new system will be build using an existing one as a starting point, the developers must identify each existing component that will be changed. For each of these components they must:

   o *measure its existing size;*
   o *estimate how much of it will be changed (only a ball park estimate, e.g. around 30%); and*
   o *compute the size of the change by multiplying the existing component size by the estimated percentage change.*

2. **New components to be built.** The developers then forecast the new components they expect to build. For each of these components they must:

   o *identify the category from your historical data that it best fits into;*
   o *determine whether it would be a "very small", "small", "medium", "large", or "very large" example of that category; and*
   o *assign an estimated size based on category, the small-medium-large judgment, and your historical data. This is where the size of your historical data set comes in. If you have enough data to use statistical methods, then your size estimate for each component will be:*

   "Very small" = Mean − 2 Standard Deviations

   "Small" = Mean − 1 Standard Deviation

   "Medium" = Mean

   "Large" = Mean + 1 Standard Deviation

   "Very large" = Mean + 2 Standard Deviations

   If your historical data set is small so that you only have Min, Mean and Max values, then your size estimate for each component will be:

> *"Very small" = Half of Min*
>
> *"Small" = Min*
>
> *"Medium" = Mean*
>
> *"Large" = Max*
>
> *"Very large" = Twice Max*

Either way, the developers must apply their best judgment to the "Very small" and "Very large" estimates. If the estimates look unrealistic, the developers should adjust them.

With the developer estimates in hand, you can easily compute the expected size of the new system. Just add up the estimated new and changed component sizes to get the total project size. You may want to inflate the total estimate if the team believes they have not identified all of the components.

Better Estimates Next Time!

At the end of the project, you will want to add all of the newly created components to your historical data. (More data is always better!) Be sure to categorize them properly and use the actual final sizes, not the estimates! You will also want to update the historical data for any existing components that have changed.

Then re-compute the size ranges for each category. After doing a few projects (or one big one), you will likely have enough data to compute Mean and Standard Deviation for your categories, which will give you better estimates than Min, Mean and Max.

Finally, the developers can improve their ability to use this sort of estimating technique by comparing their initial list of components and estimated sizes with what they ended up with at the end of the project. They should specifically look for:

- Components they built or changed, but did not identify during estimation
- Components they identified during estimation, but did not build or change
- Size estimates that were way off (e.g. anything that was estimated "Medium" but ended up being "Very Large")

These errors should all be documented as lessons learned on the project. Then those lessons should be reviewed immediately before the next project is estimated.

Using techniques like these, you too will be able to blithely comment on the estimated size of your projects!