Estimating With Objects - Part VIII

Contents

This column is the eighth in a series about estimating. If you are wondering if this series will ever end, hang in, it is almost over. The next two columns conclude the method discussion, and then there is one column on the results engineers have obtained by using the PROBE method in practice.

If you are new to this series of columns on Estimating with Objects, the first was in the July 1996 *Object Currents* issue. The prior columns in this series gave an overview of estimating and defined some of the steps in making size and resource estimates. If you have not read these earlier columns, you should look at them first to understand the context for this discussion and to see how these various estimating topics relate. To repeat what I have said in previous columns, the estimating method described here is called PROBE. If you want to quickly learn more about PROBE, you should read my book A Discipline for Software Engineering, from Addison Wesley. This book introduces the Personal Software Process (PSP)^{SM}, which is an orderly and defined way for software engineers to do their work.

This column continues the discussion of how to make software estimates. To make a project plan, you need a resource estimate and, to estimate resources, you need to estimate the size of the product you plan to build. Also, to make a good size estimate, you need historical data on the sizes of the programs you have previously written. This and the previous columns describe how to gather these data and how to use them to make the size and resource estimates. In this column, we talk about how you can make accurate estimates and how to judge the accuracy of the estimates you have made.

Resource estimating

Before we discuss estimating accuracy, it is important to first describe how to make resource estimates. If you recall, the December column discussed the regression method, and how it helps you estimate program size. Starting with an estimate of the object LOC in the program, use historical data on estimated object LOC and actual new and changed LOC to build a regression model mapping estimated object LOC to actual program size. With these regression parameters β_0 and β_1, the model is:

New and Changed LOC = $\beta_0 + \beta_1$(Estimated Object LOC)

Once you have estimated the object LOC, you can just plug it into this equation to get the number of LOC to be developed.

It turns out that you could just as well have used this same method to estimate the time to develop the program. Then, you would again start by estimating the program's object LOC as before, but now you would use data on estimated object LOC and actual development hours to calculate different values of the β_0 and β_1 regression parameters. Then, using these new parameters, you could calculate the likely development time for the new program as follows:

Development Time = $\beta_0 + \beta_1$(Estimated Object LOC)

The methods for estimating development time and new and changed LOC are identical. In both cases, you start with an estimate of object LOC, but you use different data to calculate the regression parameters.

Estimating accuracy

The first rule of estimating is that errors are normal and should be expected. Do not expect to make accurate estimates for small jobs. This is not because accurate estimating is impossible, just that accurate estimates for small jobs are unlikely. The reason is that there is considerable variation in everyday work. When you do a small task, lots of things change. For example, in driving to work every day, you probably spend about the same average amount of time. Occasionally, however, it rains or snows, there may be an accident, or you might stop for gas. If you measured your commuting time, some days would vary considerably from the norm. On average, however, you would take about the same amount of time. You can thus accurately judge the average time for a week or so, but you can rarely be accurate for each day, at least very far in advance.

This suggests a strategy for estimating. First, learn to make balanced estimates. That is, when you make 10 estimates, about as many will be overestimates as underestimates. This, it turns out, is the most critical step, and it can only be done by using historical data. That is, when you make an estimate, you compare this job with prior jobs, and assume this job will take the same average time as prior similar jobs. Then you regularly adjust your estimates to compensate for consistent under- and overestimates. When you do this, you will make balanced estimates.

This, it turns out, is precisely what linear regression does. By following the PROBE method, your estimates are based on your historical data. And the linear regression method automatically adjusts for your consistent under- and overestimates.

Estimating in detail

Now that you have learned to make balanced estimates, the next step is to estimate in detail. This means, when estimating a large job, break the task into multiple smaller parts, and then estimate each part separately. Now, if your estimates are balanced, and if these estimates had no common biases, you would expect the total job estimate to be pretty accurate. One example of a common bias would be when you were highly motivated to make a low estimate.

As we noted in http://www.sigs.com/publications/docs/oc/9611/oc9611.c.humphrey.html ([**Note**: Because this document or Web site is no longer available online, the link to it was removed from this file.] Part V of this series in November, the reason these multiple small estimates lead to more accuracy is that estimating accuracy improves by the square root of the number of the estimate's parts. The small fluctuations of the detailed estimates tend to cancel out, and the composite is much more accurate than the individual parts. Your estimating strategy is therefore to break jobs into parts, to estimate each of the parts, to determine the accuracy of each of the part estimates, and then to combine these parts into the total estimate.

Judging estimate accuracy

Now, how can you judge how good your estimate is? This question is answered in two parts. First, find the accuracy of the detailed part estimates and second, combine these estimate parts into the total job.

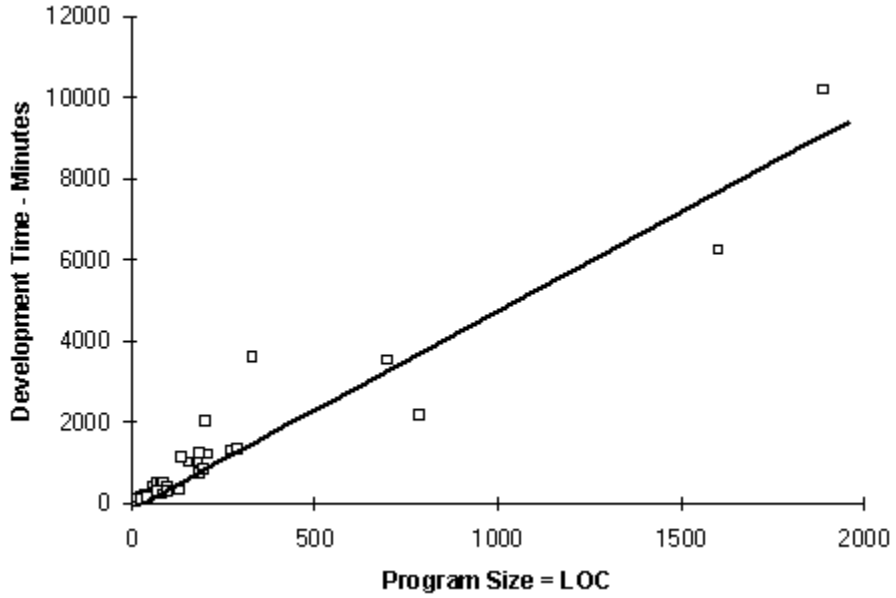

Figure 1: Regression Calculation

The prediction interval provides a way to judge the accuracy of an individual estimate. The basic idea is shown in Figure 1. Here, the regression calculations produced the line, and this line is

what you use to make your estimates. The problem, however, is clear from the figure: the actual data points fluctuate around the regression line. Thus, rather than being precise, the regression line is only an approximation. Assuming that you continue to estimate as in the past, you would expect the amount of variation around the regression line to be pretty much like what you had before. All the prediction interval does is to calculate the likelihood of one of these data points falling within a prescribed range of the regression line.

The prediction interval is calculated as shown in the following equation.

$$\text{Range} = t(\alpha/2, n-2)\sigma\sqrt{1+\frac{1}{n}+\frac{(x_k - x_{avg})^2}{\sum_{i=1}^{n}(x_i - x_{avg})^2}}$$

Range =

The prediction interval is then the estimate value plus and minus the range. Here, x and y are the data points from the prior projects. For example, $x_1$might be the estimated object LOC for project 1 and $y_1$the development time for that project. $x_2$and $y_2$would then be these data for project 2, and so forth. n is the number of projects for which you have data, and $x_k$is the new estimate value.

The σterm is the standard deviation of the data around the regression line, and its square is calculated as follows:

$$\sigma^2 = \left(\frac{1}{n-2}\right)\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

In the range equation, the t term is a value from the t distribution, which adjusts for the amount of data you have. The value of the t distribution can be found in standard statistical tables where you select the t value you want based on the number of points in your data set and the width of the prediction interval you seek. The n-2 term means you look in the row for n-2 degrees of freedom. For 10 data points, for example, you would look on row 8. The $\alpha/2$term means you use a double-sided t distribution and look for the p columns. Thus, for a 90% prediction interval, you would look in the p = .90 column and for a 70% prediction interval, you would look in the p = .70 columns. If the t distribution table is single-sided, which is common, you would find the t value for 90% in the p = .95 column and the t value for 70% in the p = .85 column.

The meaning of the prediction interval

If you seek a 90% prediction interval, for example, roughly 90% of your estimates should fall within the interval. This would tell you the likely range around your estimate that the actual value will fall 90% of the time. Thus, looking at Figure 2, you can see that about 90% of the 35 data points are within the 90% prediction interval. This is not just good luck, the prediction

interval is calculated to ensure that this is the case. Note that the prediction interval merely tells you the likely error in the estimate. It assumes that this estimate has the same general behavior as prior estimates.

To find out more about the t distribution or the prediction interval, look in a standard statistical text, or consult Appendix A in my book, http://www.awprofessional.com *A Discipline for Software Engineering.*

Combining multiple estimates

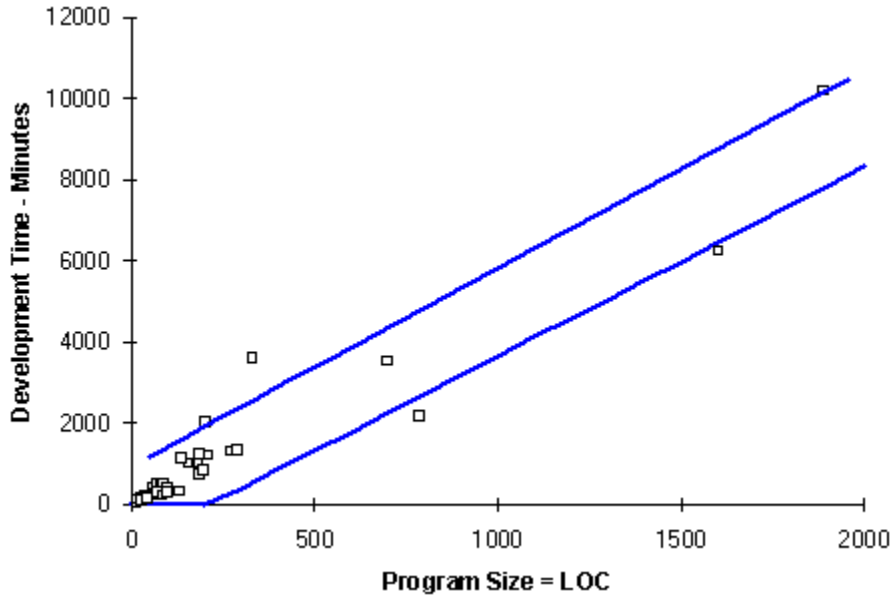

Figure 2: Prediction Interval -- 90% Finally, now that you have the estimates for the program parts, and you have the prediction intervals for each of these estimates, how do you combine the part estimates to get the final estimate, and how do you combine the part prediction intervals into an overall prediction interval?

To get the final estimate, add up the individual part estimates. You can see how to do this from the table. Here, the final total estimate is merely the sum of the individual values, or 506 LOC.

Program #	**Estimated LOC**	**70% Estimate Range**	**Range*Range**
1	118	48	2304
2	86	35	1225
3	171	73	5329
4	78	29	841

5	53	27	729
Total	506		10068

To get the prediction interval, a little more work is required. While there are complicated ways to do this, you can treat a 70% prediction range much like a standard deviation. With the 70% ranges for each of the estimates as in the table, square the intervals to get the equivalent of a variance. Then add the variance values and take the square root of the result. This gives the range of the prediction interval for the combined estimate as 100 LOC. To repeat, you add the part estimates to get the final estimate, but you combine the squares of the prediction ranges and take the square root. This is why estimate accuracy improves as the square root of the number of parts in the estimate.

Next month's topics

We have now covered almost all the estimating topics. Over the next two months we will talk about making and tracking schedules. Then, in May's column, we conclude this estimating series with a discussion of results engineers have achieved using the PROBE method in practice.

An invitation to readers

In these columns, I discuss software process issues and the impact of processes on engineers. I am, however, most interested in addressing issues you feel are important. So please drop me a note with your comments and suggestions. Depending on the mail volume, I may not be able to answer you directly, but I will read your notes and consider them when I plan future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu