Estimating With Objects - Part V

Contents

[The size estimating strategy](#)
[The conceptual design](#)
[Subdividing the conceptual design into components](#)
[Subdividing the components](#)
[Making the size estimate](#)
[Calculating total object LOC](#)
[Estimating large programs](#)
[Estimating time](#)
[The next month's topics](#)
[An invitation to readers](#)

This column is the fifth in a series about estimating. The first was in the July 1996 issue. This month, we talk about how to produce the conceptual design, define the objects in that conceptual design, and estimate the LOC they contain. The prior columns in this series gave an overview of estimating, defined software size, introduced the subject of proxies, and showed how to categorize size data so you can use them in size estimating. 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 said in the 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 size 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 column describes how to use these data to make the size estimate.

The size estimating strategy

To start the PROBE estimating process, you need to follow the methods described last month to categorize the size data on the objects you have previously developed. With data on enough objects and methods, you can classify them into types and calculate the standard deviation of their method LOC. You can then determine the number of LOC in very large, large, medium, small, or very small objects of each type. It is also important that you use data on the objects you have personally written. Programming styles vary among individuals, and your estimates will be most accurate when you use your personal data.

The next step is to use these historical size data to judge the size of the new program. In essence, you follow the normal size estimating practice in other fields: compare the objects you plan for

the new product with the objects you have previously developed. By examining the sizes of similar objects, you can accurately judge the size of the new ones. Then, using these data, you can calculate the likely size of the new program.

The conceptual design

The first step in making an estimate is to produce a conceptual design of the product you plan to build. The reason you have to do this is that the requirements do not describe the product, they only describe what the user wants. Unless the user is very specific about how to meet these needs, the requirements will leave you considerable design flexibility. Requirements that specify the design or even the design approach are overly restrictive and often lead to poor results.

The figure shows how to make a conceptual design and use it to estimate object LOC. First, dream up, create, copy, or somehow produce the overall design approach. While there are no proven best ways to do this, there are a lot of methods. If you are not familiar with object-oriented design methods, I suggest you look around. There are plenty of books on the subject and many are very helpful. They will likely give you some good ideas about how to approach the design problem. The object-oriented design methods that I have looked at are all consistent with the PROBE method described here.

Figure 1: Estimating Object LOC
Figures in this file are displayed in a separate browser window. This window will remain open to display figures in this file, although it might be hidden behind other browser windows.

Note that the conceptual design is not the actual design. That will take more time to develop. Your intent here is only to produce a conceptual approach. This should be something you are reasonably confident you can build. When you later do the actual design, you will want to take enough time to consider alternatives. You should probably also examine performance, size, reuse, usability, testing strategy, and many other issues.

Subdividing the conceptual design into components

Next, subdivide the conceptual design into component parts. In doing this, you essentially act like a magician. In principle, you say, If I had components that did the following, I would know how to build this product. While it helps to have previously built such components, that is not necessary. All you need is to take the time to think through what each component is supposed to do.

After you have subdivided the product into components, look at these components to see if they are like components you have previously built. For relatively small products, these first level components will probably be at the object level. You can then compare them directly with previously developed objects.

Subdividing the components

If, however, these components are not yet at a low enough level to compare with prior data, you will need to do the same magic trick all over again. For each component, think what subcomponents you would need to build it. Again, invent the subcomponents just as before. Do this for each subcomponent and, each time, examine the results to see if you are at a detailed enough level. The key is to see if you can compare these subcomponents with the data on your prior objects.

For very large products, you may have to repeat this subdivision process several times. Once you finally arrive at the level where you can make the comparison with your historical data, however, stop! You have enough detail to make the size estimate.

Making the size estimate

Now that you have a sufficiently detailed conceptual design, examine each object to judge how big it is. First, decide on the object type and estimate how many methods it will likely have. You could even name each method to make sure you know what that object does. This will help you avoid gross estimating errors.

Next, with the object types and their numbers of methods, judge where this new object falls in size relative to the objects in your historical database. Unless you are pretty sure that the new object will be much larger or smaller than normal, it is a good idea to stick with average object sizes. This will help you avoid making consistent overestimates or underestimates.

Of course, there can be complications. You might, for example, need to use some objects of types for which you have no historical data. Here, the best you can do is to make comparisons with those of your objects that are most similar. Alternately, you could compare this new object with someone else's objects of a similar type. Another possible complication is a new object with methods of several different types. Here, you merely judge the size of each method by itself. When you have no prior data, however, you are really guessing and will likely have a larger estimating error. But you can still follow the PROBE method.

Calculating total object LOC

Now you are ready to look at your historical data for objects of the same type. If you judge the new object is probably about average, then pick the average LOC per method from the data. If, however, you judge that it is likely very large, pick the very large LOC per method from the data.

Once you have the LOC/method for each object, the next step is to multiply the LOC/method by the number of methods to get the LOC for each object. Finally, total up these estimated object LOC to get the total estimated object LOC for the new program.

With the total estimated object LOC, you nearly have the size estimate. The next and final step is to use linear regression to convert this object LOC estimate into a program estimate. This, however, is the subject of next month's column.

Estimating large programs

You can follow this identical strategy for estimating the sizes of even very large programs. Since this could involve a significant amount of work, however, you might ask some other engineers to estimate some of the parts. For really large systems, you could even use large engineering teams to make the estimates. In fact, if all the engineers on these teams have estimating experience, and if they all have data on the objects they have previously developed, this is the way to get very accurate estimates.

The reason that estimating in parts produces better estimates is that likely estimating errors are reduced by the square root of the number of independent parts. Thus, for a traditional single estimate error of 60%, if you have 9 engineers independently estimate the parts, your expected error would be one third as large, or 20%.

Estimating time

One commonly voiced concern with the PROBE method is that it can take a fair amount of time, particularly for large programs. After you use PROBE for a while, however, you will find that it takes less time than you expect. I have found that I can estimate programs of several thousand LOC in only about an hour or so. Historical data and a defined estimating method help you to make quite large estimates very quickly.

To get an idea of how long people in other fields take to estimate complex products, talk to some engineering groups. Ask them how long it takes to estimate the development of a new computer, the design and construction of an apartment building, or the development of a new jet engine. Accurate estimating requires detail, and detail takes time. Competent estimates for larger programs will take proportionately more time. If you are not willing to take the time to make a good estimate, then you might as well forget about estimating methods. While guessing is not very accurate, it takes very little time.

The next month's topics

You now have a pretty good idea of how to make a size estimate. This and the last four columns have described how to handle the most difficult parts of the estimating job. From now on, you use these data to actually produce the estimate. The next several columns explain these final steps.

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