Estimating With Objects - Part III

Contents

This column is the third in a series about estimating. This month, we discuss size estimating. We also introduce the subject of proxies and describe how they can help you to make better estimates. The first column in July gave an overview of estimating and the August column talked about software size. If you have not read those columns, you should probably look at them first to understand the context for this discussion and to see how the various estimating topics in these columns relate. To repeat what I said in the last two 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 starts the discussion of how to make size estimates. As we discussed last month, to make a project plan, you need an estimate of development resources. To estimate resources, however, you generally need to estimate the size of the product you plan to build. The reason to start with an estimate of product size is that large products generally take more time to develop than small ones. Thus, when you have good historical productivity data, a good size estimate will provide a sound basis for estimating development hours.

The size estimating problem

In making a size estimate, you start with a generally vague statement of requirements and attempt to deduce the size of the program needed to produce the desired results. This size estimating process is difficult for several reasons. First, the requirements should describe what is wanted not what is to be built. While this will provide maximum design flexibility, it does complicate the estimating problem. The reason is that there are many possible ways to build a product to meet a single requirement. You must thus start by deciding how you will build this product.

Since this product presumably differs from products you have previously built, you cannot know precisely how to build it. You must therefore produce a design. The estimate, however, will

generally be needed before you can start the work. There is thus not time to produce a complete design. This initial design must then be conceptual, and only be based on a brief judgment of how you might build the product.

Once you have the conceptual design, you estimate the likely size of the finished product. The problem now is that it is difficult to look at a high-level program design and judge how many LOC it will likely require. If you have not built a similar product before, you will have to use an analogy. That is, you must compare the size of this program with the sizes of programs you have previously built. You then judge where this one falls in the known range of these prior programs.

Note that in this series of articles, we will generally use lines of code (LOC) as the size measure. This does not mean that other measures could not be used, just that I will not discuss them. If you want a further discussion of various size measures, see the July article or look at Chapter 4 in my book, *A Discipline for Software Engineering*.

The comparison problem

The problem now is that it is hard to compare one program with another. Even when they are small, there are generally many differences and variations. As programs get much larger, they are almost impossible to compare in any orderly or consistent way. The bigger the product, the harder it is to relate it to other products. The reason is that larger products have more functions.

A related problem concerns the trends of the software business. Every new product, even for previously provided functions, tends to be larger than its predecessors. The reason is that we keep adding new features and functions. Examples are the use of new graphical environments, network security, recovery problems, compatibility with prior products, and comprehensive help facilities. The list could go on and on.

One way to address this comparison issue would be to break the new product into pieces. This, in fact, is how other industries make estimates. In building construction, for example, builders first start with a specification. They then break the job into all its component parts and have the various building trade subcontractors estimate each part. While the software community rarely has the luxury of a completed specification, the approach of looking at the parts of the job is equally applicable. That is, we could break the product into multiple smaller parts and then estimate the size of each part. This is essentially a parts list approach where you start with a detailed design and a listing of all the needed parts. By adding up the known sizes of all the previously built parts and estimating the new parts, you have the completed estimate. This, however, still leaves the problem of estimating the size of each part.

Estimating part size

Here is where the PROBE method comes in. PROBE stands for proxy based estimating. Instead of directly using LOC as the size measure, we use a proxy, or a substitute. Consider again the

building example. While builders can make pretty good estimates when they know the square feet of a planned building, few home buyers are able to estimate the square feet they will need. They are more comfortable thinking in terms of rooms and whether the rooms are large or small. Thus, for a four bedroom house, they might want one moderately large bedroom, two medium bedrooms, and one moderately small bedroom. They would also want a kitchen, family room, bathrooms, and so forth.

With data on the relative sizes of various kinds of rooms, the builder could then give the buyers a reasonable estimate of the needed square feet. Experienced builders, of course, will not provide an estimate without a specification and a set of architectural drawings. When pressed, however, they will generally tell you the costs and sizes of similar houses they have built. You could then judge where your planned house would fall within this range. This, by the way, would not be a bad practice for software estimating: either estimate from a spec of provide data on prior work and let the customer judge the range. If the customers then wanted detailed estimates, they would have to pay for the specification and design work required to make them. Unfortunately, it will take the software community some time to build the skills and credibility to work in this way. Until we do, however, we will continue to have estimating and planning problems

In the builder example, rooms are a proxy for square feet of floor space. The PROBE method uses Objects as proxies for LOC. While we start by estimating the numbers of objects of a given type, PROBE uses LOC data for various types of objects. Then, once you decide how many objects of each type are needed, you can calculate the likely number of object LOC. Before describing how to do this, however, let's first talk about the criteria for a good proxy.

Selecting a proxy

The criteria for a good proxy are as follows:

- The proxy measure should closely relate to the effort required to develop the product.
- The proxy parts in a product can be automatically counted.
- The proxy should be easy to visualize at the beginning of the development process.
- The proxy can be customized to the needs of the organization.
- The proxy should be sensitive to implementation variations that have an impact on development cost, effort, or product size.

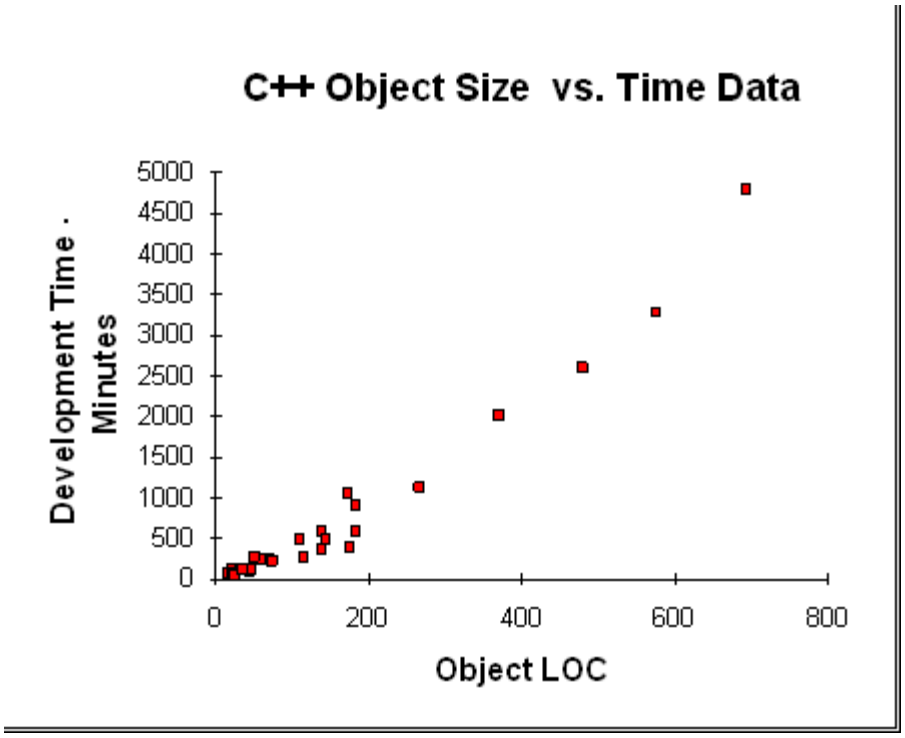
As noted in the following paragraphs, objects meet all the criteria for a proxy.

Relationship to development effort

To be useful, the proxy must have a demonstrably close relationship to the resources required to develop the product. By estimating the size of the proxy, one can then accurately judge the size of the job. The way to determine the effectiveness of a proxy is to obtain historical data on a number of prior products and check the correlation of the proxy values with the development

costs. If the correlation value of r is such that $r^2 \geq 0.5$, the proxy is judged to be a reasonable predictor of product size or development effort.

In the PSP, objects are used as the example proxy. In planning a new project, one then judges how many objects of each type will be needed and their relative sizes. An example of the relationship between object LOC and development times is shown on a scatter plot in the figure.



This figure shows the relationship between object LOC and development time for a family of 26 C++ programs. The correlation between size and development time is better than 0.95 and the significance is better than 0.005. Object LOC thus meet the PROBE requirement for close correlation to development effort.

The proxy parts in a product can be automatically counted

Since historical data are needed for making accurate estimates, it is desirable to have a large amount of proxy data. This requires that the proxy content of programs be automatically countable. The proxy should thus be a physical entity that can be precisely defined and algorithmically identified. Since objects are physical entities, both the number of objects and the object LOC can be automatically counted.

The proxy should be easily visualized at the beginning of the development process

The usefulness of a proxy depends on the degree to which it helps engineers visualize the size of the product to be produced. This in turn depends on the engineers' backgrounds and preferences. There will thus not likely be one best proxy for all purposes. With suitable historical data, one could even use several different proxies to make a single estimate or even different proxies at different points in the development cycle. The multiple regression method used with PROBE can be helpful for this purpose.

The principles of object-oriented design suggest that objects are good estimating proxies. During initial analysis and design, application entities should be used as the basis for selecting system objects. Here, an application entity is something that exists in the application environment. The object-based design process then selects program objects that model these real-world entities. These product objects can then be visualized during requirements analysis. Since objects are defined program elements, it is possible to precisely measure the size distribution of the objects in the historical database. By using these data, the LOC content of the objects in the proposed product can be estimated. Then, using PROBE, estimated object LOC are related to development hours and ultimate product size.

The proxy can be customized to the needs of the organization

Much of the difficulty organizations have with estimating methods comes from attempts to use data from one development group for planning the development work of another. It is important, therefore, to gather and use data that are relevant to the particular project being estimated and the people who will do the work. This suggests that large organizations keep size and resource databases for each major software product type, and that each engineering group gather and use data on their own work. Engineers should, in fact, keep their own personal estimating databases since this provides the most accurate basis for making estimates.

The proxy is sensitive to implementation choices

If the proxy counts are not sensitive to factors that affect development costs, then estimates using the proxy cannot be as accurate as estimates that reflect project differences. The proxies that closely reflect development conditions deal with physical properties of the product, while those that are most easily visualized at the beginning of a project are application entities. Examples of the latter are inputs, outputs, files, screens, and reports. Unfortunately, a good development estimate requires entities that closely relate to the product to be built and the people who will build it. Data are needed on proxy and product size for each implementation environment, each design style, each application domain, and each development group. It is thus essential that the project data include specification of the languages used, the design methods employed, and the

application domains covered. In making estimates, one should then only use historical data for projects that are similar to the one planned.

Coming soon

While there is more to size estimating, this will give you an idea of the approach. Next month's column describes how to gather and use proxy data. Following columns then explain how to use these data to estimate the size, the development time, and the schedule for a new program.

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