EXHIBIT M

Contents

Table 3.1 (continued) Estimation Measures

Issue	*Source Lines of Code*	*Function Points*
Postmortem use	Useful; easy to compare before-and-after results	Useful: easy to compare before-and-after results; easy to see exactly where variation occurred
Expense of estimate	Low: rapid and very inexpensive	Moderate to high; slower and potentially expensive
Ability to estimate with automated tools	High: tools such as SEER-AccuScope can estimate lines with relative ease	High; tools such as SEER-AccuScope can estimate function points with relative ease

When to Use SLOC

SLOC has been the dominant method for sizing complicated, real time or embedded systems and works well for hand-generated systems in general. Use lines of code when SLOC-based historical data exists, when the development organization is comfortable with SLOC estimates, when add-ons to existing systems allow counting of actual SLOC in a system, and as a relatively easy check on other methods.

The great strength of SLOC is that it is easy to obtain. All other factors aside, it remains a fairly accurate predictor of development effort. By comparing code counts from past projects against a "rough order of magnitude" estimate for a proposed project, you can gain your first real understanding of project scope. By pairing SLOC estimates with other development factors, you will generally have enough information to develop a reliable estimate.

SLOC counts provide a firm indication of the volume of software generated, which is a first critical step for making comparisons and predictions. Despite the dominance of SLOC measures, some confusion exists regarding *which types* of lines to count, which has led to difficulty in comparing methods of counting SLOC. However, within the past several years, code counting methods have become more standardized. See Chapter 5 for a detailed discussion of lines of code.

When to Use Function Points

It is best to develop estimates based on function points when your project is largely comprised of information technology and the system's functions

are adequately specified. Alternatively, you can estimate a function point count using other means such as SEER-AccuScope to estimate size. In addition, you should use function points when sizing by SLOC could be misleading. For example, code generators can automatically generate many lines of code, which makes the number of lines generated an unreliable predictor of the amount of effort required. The great strength of function point counts is that they are developed directly from specifications, independent of implementation, which means estimates of project scope are more comparable across projects.

Counting function points is a sophisticated method that cannot be done automatically. There are few shortcuts; you must ensure that it will be done properly by assigning adequately trained and experienced personnel. If you have counts from previous similar projects, be sure to study those counts carefully to ensure that the work performed on your current project is consistent with the method used on those projects. Although experienced counters can accomplish this method fairly quickly and efficiently, it is very important that the counts be performed correctly and consistently.

The function point counting process should be put in perspective. An experienced function point counter is usually able to count project specifications amounting to about 600 function points per day. As a typical MIS database project of this size might take a bit more than a year to develop and consume a bit more than a hundred person-months of effort, using function points does not necessarily involve a big investment in up-front planning. Many large organizations therefore keep a function point counter on staff or hire outside consultants when necessary. By using the SEER function-based sizing method, function points can be estimated without conducting detailed counts. Finally, it is important to understand that a combination of SLOC and function-based sizing can be the most appropriate way of estimating the size of an existing system to which functional enhancements are being made.

Steps to Estimating Software Size

Managers of software development projects are responsible for ascertaining progress, risk, productivity, and a host of other factors that are critical to success of their projects. Two important factors are the size of the product and the subsequent effort that will be required to develop it. If you want to contain the risk of unexpected cost growth for your project, it is essential that you use a software sizing method that is consistent and repeatable.

In order to usefully apply the general concepts and techniques of managing risk to your software engineering project, it is also essential that you regularly reestimate the size of the product and the associated cost of the project as project conditions change or product specifications

Function Point Counting Checklist

When presented with a function point count, carefully consider the following information.

Sanity Counts

Function point counts vary with counters. It is therefore useful to have a second counter double check (conduct a sanity count) of some part of a count before the entire count has been completed. By doing this, counting methodology questions can be resolved early. Alternately, using a sizing model can approximate the count quickly.

Lack of Convergence

Unless a counter is very experienced, a function point count should be conducted over several iterations. The more experience a counter has, the better he or she will understand the technique and be able to converge on a reliable number.

Double Counting

Keep careful track of requirements and the resulting function point counts to make sure that counts have not been replicated. The following are some typical sources of double counting:

- Referencing a File More than Once
  Files should be counted in relation to the boundary of the entire application. For example, an external interface file is created completely outside the application boundary. On the other hand, an internal logical file is created completely within the application boundary. It should be clear that external and internal logical files are counted only once in a particular application, regardless of how often they are used.
- Confusing Designer's and User's Perspectives
  Counting occurs from the user's perspective, which means that certain architectural details may remain hidden from the count and will legitimately not be counted. Another source of perspective-related confusion is the difference between physical and logical files. Sometimes, what the user sees as one logical file may actually reside in several physical files, while the opposite may also be

true. In either case, conduct the count as the user would see the components.

Sample and Statistical Concerns

When validating estimates based on project histories, be sure to assess the samples and statistics used and their associated risks.

Probability Level

The acquisition type of the project can determine the probability level of the estimate. When comparing contractor estimates with your own, be aware that a contractor's acceptable risk varies with the type of contract. A contractor may choose an estimate other than at the 50 percent probability. (Internal developments may also have the same issues with the use of probability.)

On *cost plus* jobs, a developer's overriding interest is in winning the contract, and so he may offer a more daring estimate. An estimate probability of 40 percent and even 30 percent may be chosen and could lead to a 60 or 70 percent chance of a cost overrun.

On a *fixed price* award, a developer bears the expense of a cost overrun and so is more fearful of bidding lower than practical. For such projects, an estimate probability of 60 percent or even 80 percent might be chosen.

Compare your estimate range and the actual bid. If the bid lies at the probability level in your estimate range that would be predicted by the type of contract, then the estimates probably agree. You can sometimes directly ask a contractor what probability level was chosen, but this information may or may not be available to you.

Falsely Bounded Risk

A risk analysis may be populated with engineers' and estimators' assumptions that may not admit the full range of possible outcomes. One useful countermeasure is comparing assumptions from different people and program components (such as separate computer programs) so that you can reconcile inconsistencies.

Bias

You must be careful to obtain a balanced sample, one that is not biased in any way, and particularly by factors that are unrelated to the project you are analyzing. Sample bias is a particular problem with small samples.

Try to obtain a sample with characteristics that are consistent with the project you are estimating.

Outliers

When assembling a sample to help you validate the estimate of a current project, some values can lie well outside the common range; these are called outliers. Outliers must be separately examined. An outlier may be an unrepresentative event that should be ignored or it may offer special lessons. An outlier that is valuable should not necessarily be included in the main sample and in the sample statistics. It may instead be used to offer instructive lessons outside the conventional statistical analysis.

Costs

Are Staff Costs Fully Burdened?

Ask the developer what its staff costs are, and make certain that the monthly staff rate has been agreed to. Understand whether costs are fully burdened or whether additional charges will be incurred; if so, ensure that these costs are included in the complete cost estimate.

How Many Hours Are in a Staff Month?

A common United States standard is 152 hours per staff month. If a developer's hours per month vary from what the estimate uses, you must normalize the results to make a viable comparison. Multiply all effort-month figures by the developer's hours per month and then divide this number by the hours per month that you are using. (Automated cost models like SEER-SEM do this automatically.)

Staff and Effort Accounting

Does Overtime Count?

If the development staff is planning to use unreported overtime, this will cause variation from the estimate.

What Level of Management Participates?

Account for labor directly applied to this project only. Confirm that the developer has not included upper level (executive) management.