



TenStep Supplemental Paper

12 February 2004

The Need for Function Points

People in the development world have a difficult time creating tangible metrics that tell how productive they are as individuals, teams or organizations. Of course you are productive. In fact, you may be the fastest, smartest, most productive developer in your field. But what does that really mean? Normally, you would like to quantify how much work you can produce in an hour or a day or a month, and then compare this to some average or standard. This is difficult, if not impossible, for two reasons. First, it's not at all clear how to measure what you are producing. Second, even if you found a way to measure what you produced, the chances are no one else is measuring what they produce in the same way. Therefore, you cannot compare yourself to others, or to some standard, in any meaningful way.

The concept of function points was introduced to try to resolve this dilemma. Function points by themselves do not show productivity. Function points are a way to measure the size of a software application. However saying that one application is 2,000 function points and another application is 4,000 function points does not tell you much, other than one application is twice the size of another. What really makes function points interesting is that they can help resolve the two problems that were posed above. First, function points provide a way to measure what you are producing, and second, they can be applied fairly consistently across different applications, people, teams and organizations. This allows for comparisons to take place.

A little history

The Greeks and Romans did not use function points. The concept of function points is relatively new, even in the brief history of software development. The original ideas were developed in the late 1970's by IBM as a way to gauge the relative size of a computer application. IBM originally wanted to develop a technique for estimating the effort required to build an application. It made intuitive sense that the bigger an application was, the more effort would be required to build it.

There were many ways to try to represent the size of an application. A developer might think in terms of lines of code. If two applications were both written in COBOL, it probably makes sense that a system with 20,000 lines of code is smaller than one with 40,000 lines of code. However, this approach rapidly falls apart when you bring in different languages. A 5,000 line COBOL program, for instance, may or may not be bigger than a 2,000 line Visual Basic program. Different coding styles also contribute to more or less lines of code for the same basic function.

Now let's look at the size of an application from a user perspective. The user sees reports, screens, and maybe data files. He/she also sees business functionality, interfaces to other applications, databases, tables, etc. It makes sense that an application with 40 screens and 10 reports is probably smaller than an application with 80 screens and 20 reports. This way of looking at size is independent of the underlying technology and development



TenStep Supplemental Paper

language. In fact, the application with the smaller number of screens might require more lines of code, but that is no longer relevant in the sizing calculations. This approach seemed to make sense to Allan Albrecht, who was the individual from IBM who came up with the original function point concepts in 1979.

The original concepts related to function point counting have been modified over time to take into account new best practices and new technology. For instance, in 1979, there was no such thing as web development. The function point counting rules now take web development, as well as client server development, into account. In the late 1980's, the International Function Point Users Group (IFPUG) was formed to further the advancement of function point counting around the world. They produce a Counting Point Manual that is currently released in version 4.1.1.

What are function points?

The details of function point counting are more complex than what can be described in one, or a series, of columns on the subject. In fact, the IFPUG Counting Manual is a couple hundred pages long. This book is just the beginning. Reading should be supplemented by training and lots of experience before a person should feel comfortable doing function point counting. However, in summary, the basic concepts around function points require you to look at five areas of the application:

1. **Internal logical files** include data that the users are responsible for maintaining themselves.
2. **External interface files** are files that are needed from other systems. This data is used for inquiry or reference and is not updated by your application.
3. **External input** refers to counts of the functions that allow the user to manipulate internal data. This is usually in the form of adds, changes or deletes.
4. **External outputs** are functions that allow the user to request an output through the manipulation of data in the system. This could be information on reports, output screens, webpages, etc. The key is that the data is processed or transformed into new information that is made available to the user.
5. **External inquiries** refer to information that is basically displayed directly from underlying tables. If you display information on a customer that comes directly from a customer file, then it is external inquiry data. If some fields were transformed based on internal program processing, it is more likely external output (#4 above).

Once the basic characteristics are determined, an algorithm is applied that weights certain areas higher than others. But once the basic counts are determined, the number of total function points can be determined easily by applying the formula.

Estimating size and effort using function points

When you capture function points on your first application, they are interesting, but of limited value. The value of function points grows as you accumulate some history. If you



TenStep Supplemental Paper

count function points per application, as well as the total effort required to build the application, you start to put together a correlation that you can utilize in estimating.

Let's say you have a number of historical projects, and you start to see a pattern that shows that it takes 5000 hours of effort to produce an application of 500 function points. (This relationship is for example purposes only.) You can start to leverage this historical information on future project estimates. Let's imagine that you are the project manager on a new application. When you prepare your initial high-level estimates for the project, function points are hard to leverage. However, you can estimate the work to gather business requirements. Depending on how you do your development, you will be able to validate the remaining effort by the end of the Analysis Phase or the beginning of the Design Phase. If you create a Conceptual Systems Design or Technical Systems Design, you will have identified the online screens, the batch reports, the major features, the major tables and the major interfaces. This is the information you need to calculate function points. When you add up all the appropriate information and do the appropriate calculations, you find that the application has 500 function points. You now have a better idea of the effort required. According to our initial pattern, you should be able to complete 500 function points with 5000 effort hours.

The more historical data you capture, along with demographics on the characteristics of the project, the more information you will have to provide very accurate estimates of the remaining work.

Using function points to determine productivity

The really neat thing about function points is that they provide some ability to get at the question of productivity. The general problem today is that you might know how many hours and dollars it takes to build an application, but you don't have the right information to make a ratio that allows you to compare people or projects or teams or organizations. Hours and dollars are the numerator of the ratio, but what is the denominator?

Function points give you something to put into the denominator. If you use function points, you can calculate dollars per function point (\$/FP) and effort hours per function point (hrs/FP). Function points work because they measure relative size and complexity, and because they are counted similarly and consistently on all projects and in all companies. (Maybe not exactly, but similarly.)

For example, let's say that you have 20 projects and you see a consistent trend that shows that it takes ten hours to build one function point. You can now start to make some rational observations. For instance, if a project team finishes an application of 100 function points in 1400 hours, you could say that they were not as productive, and you can look for reasons why. Let's say instead that you have three projects that utilize light development techniques, and they average 8 effort hours per function point. You would then have a pretty powerful and quantitative argument in favor of moving all development to that approach. Let's say you introduce a new testing tool and you see that you average 9.25 hours per function point. Again, now you are starting to have facts to back up your decisions on what is helpful and hurtful to the development environment.



TenStep Supplemental Paper

Using function points for benchmarking

A company can use function points to measure the internal trends in the development environment. You can also use function points to compare yourself with other companies. For instance, if you have a group of companies that all use function points, you may find that one company takes 14 hours per function point, one takes ten hours and one takes eight hours. Once the relative productivity has been established, the companies can look at the most efficient company to determine what development processes they have in place. The less efficient companies can make similar improvements and try to force their numbers down toward the more efficient company.

Why doesn't everyone use function points?

Function points finally give you a sense for relative size and complexity, allowing you to make internal and external comparisons and to track the trends over time. Based on this information, you might wonder why every company is not using them. There are a couple simple factors.

Lack of metrics sophistication. Many companies simply do not capture and track very many development metrics at all. Companies that don't normally collect organization-wide metrics are not going to jump into function points.

They are not easy to collect. Counting function points is not for the faint of heart. You definitely need to be trained. Your function point counters probably need to be certified as well. They also need experience. One certified counter said he believed that he needed a year of experience counting function points before he really felt comfortable. Since the purpose of function points is to allow comparisons from application to application, it is important that these counters be accurate and consistent.

Summary

Knowing the relative size of an application is interesting, but this is not where the value is. Most of the value comes from combining the function point count with additional information. Function points can then be used to help determine productivity, perform benchmarking studies and help estimate the effort associated with future project work.

Function points open up a wide variety of sizing and productivity metrics. If there were a way to automate the collection, perhaps the concept would be adopted by everyone. However, the complexity, cost and time required to collect and leverage the information makes them a fit for sophisticated companies that are advanced in their understanding and use of development metrics.