Introduction to Fast Function Points

Author

William H. Roetzheim
13518 Jamul Drive, Jamul, CA 91935
William@costXpert.com
619.917.4917 voice
619.374.7311 fax

Abstract

In one way or another all stakeholders, project managers, end users, and executives must work with software size metrics, however, many organizations do not have the patience or budget to fully implement IFPUG function point counting techniques. Fast Function Points were developed by the Cost Xpert Group, as an extension of work by Software Productivity Research, Inc. (SPR), to support approximate function point estimates with a minimal of effort and training. Fast Function Points supplement rather than replace IFPUG function points, but are particularly useful at early stages of the software lifecycle. This talk introduces Fast Function Points, describes their proper use, and presents case studies.

The Need for Shortcuts

Function Point counting, as originally introduced by Albrecht in 1979, was aimed at a relatively homogenous set of applications and way of building software. Screens were CICS "green screens" or something similar, sequential records on magnetic tapes were used for input, output, temporary storage, and so on. The function point counting methodology was easy, and intuitively understood by managers and end users with little or no computer software background, basically involving counting of inputs, outputs, inquiries, and master files and multiplying each of these values times a fixed multiplier.

Over time, the function point counting methodology became increasingly complex to support software that was increasingly complex. Files were broken down by internal files versus interface files. Each element that was counted received an individual complexity adjustment. Project level complexity adjustments were used to make global adjustments to the function point totals. The end result was a counting methodology documented in ISO/IEC 20926 that yields precise answers for a wide variety of projects, but which suffers from the following disadvantages:

- Learning the counting methodology has gone from a few hours of explanation to a class lasting weeks combined with formal certification;

- The methodology has gone from something readily understood by end users and easily used by end users and project managers for scope control during requirement definition, to something that end users and project managers understand only at a conceptual level;

- Function point counts have gone from something accomplished in a number of hours from a requirement specification or existing application to something taking weeks or sometimes months of skilled labor;

- Revisions to function point counts throughout the requirement and design phase were easy and simple and they could be accomplished in near real-time, but now updates to function point counts are typically accomplished only at major milestones because of the effort involved; and

- Function points are often used as inputs to cost estimating tools, however, if the project level complexity adjustments are made during the function point counts these may overlap with similar effort adjustments made within the estimating tool.

None of this is meant to discourage organizations from making formal function point counts per ISO/IEC 20926 and using certified function point counting specialists. This remains the most reliable method of ensuring an accurate, consistent function point count. However, we believe that there is a place in the software development world (and lifecycle) for informal approaches that will approximate the function point values obtained through more formal approaches.

Pioneering Attempts

The individual that has probably done more to promote function points in the corporate world than any other person is Capers Jones, and he was one of the first to recognize the above issues and to propose a solution. In October 1985 Software Productivity Research (SPR) introduced a function point counting methodology that harkened back to Albrecht's original 1979 approach in its simplicity, and released a tool (SPQR SIZER/FP) to assist in the counting. The SPR methodology used average values for inputs, outputs, inquiries, data files, and interfaces combined with a simplified three factor complexity adjustment. In his book, *Applied Software Measurement,* Capers claims that the simplified SPR approach yields function point counts that average within 1.5 percent of the more formal IBM techniques. Note that this technique is *not* the same as feature points, also introduced by Capers Jones, which were a largely unsuccessful attempt to extend the domain of function points into algorithmically intense domains such as embedded development.

Unfortunately, to the best of our knowledge, the SPR methodology was not subsequently modified in any major ways to compensate for on-going changes in user interface metaphors and typical file storage techniques.

The Development of Fast Function Points

Cost Xpert fully supports ISO/IEC 20926 counted function points as an input to the estimating process; however, most of our users prefer a simplified approach we call Fast Function Points for estimates by end users, project managers, and others with minimal training in function point counting techniques.

At a high level, the concept is as follows:

1. Assume that in the original systems counted by Albrecht, his average values were appropriate.

2. Relate modern program and user interface constructs back to the "green screen" days of the 1970s and develop simplified counting rules that adjust backward from modern constructs to the equivalent constructs back then.

3. User terms that are understandable to the end users and which relate to physical items in a requirement specification or design, even if those terms do not precisely match the terms of ISO function points.

4. Perform all project level complexity adjustments within the cost estimating tool itself, rather than adjusting function point counts using project level complexity adjustments.

5. Plan for and use mixed mode counting approaches, where part of an application is sized using function points and other parts are sized using a different metric.

In Fast Function Points, counts are made in terms of inputs, outputs, tables, interface files, and messages. Let's look at how each of the five Fast Function Point metric constructs are applied, then come back to further justification for this approach.

Counting Inputs

Inputs are defined as a data entry screens which are the functional equivalent of a CICS input screen. For an ordinary data input screen, each screen is counted as one input. CICS did not support tabbed notebooks, and the data entry fields from each tab would normally have been separate screens in a CICS application, so each tab in a tabbed notebook counts as one Input. CICS did not normally incorporate vertical scroll bars to support long, scrolling screens. If a data entry screen extends beyond a physical screen boundary, it is counted as more than one screen (the number based on the number of physical screens). The number of inputs, adjusted as just described, is multiplied by four, to arrive at input related function points.

Counting Outputs

Outputs are defined as reports, whether to the screen or to the printer. Outputs are assumed to consist of static data combined with dynamic textual data that comes from simple calculations or database queries. Sophisticated graphical outputs such as would be found in a report that included business graphics are adjusted using the best judgment of the counter based on the complexity of the graphical output, and are typically counted as two to three outputs for each actual output. Very sophisticated graphics (e.g., a video game), and very sophisticated calculations (e.g., satellite ephemeral data) are not counted using Fast Function Points, but instead are counted using source lines of code (SLOC), class-method points, or something similar. The number of outputs, adjusted as just described, is multiplied by five to arrive at output related function points.

Counting Tables

Third normal form tables in the database are counted. First and second normal form tables are mentally normalized, and their third normal form equivalent is counted. Temporary tables are counted using the same approach. Lookup tables that consist of nothing more than look-up values for a combo-box or similar construct are not counted (these were originally counted in early versions of Fast Function Point counting, but we found that the counts and estimates were more accurate if they were not counted). Tables (files) that are created by the database management system (DBMS) and that contain indexing data only are not counted. The number of tables, adjusted as just described, is multiplied by ten to arrive at table related function points.

Counting Interfaces

Record oriented interfaces (whether file based, socket based, or anything similar) are counted, however, we do not count the number of interfaces (or file types), but rather, count the number of record formats the flow up and down the interface into or out of the system. So, a single file that contained a header and then a sequential collection of one record format would count as two interfaces. A single file that supported five different internal record formats would count as five interfaces. In effect, we ask the question, “How would this have been implemented in a tape based sequential file system?” and adjust accordingly. The number of interface, adjusted as just described, is multiplied by seven to arrive at interface related function points.

Counting Messages

Messages are synchronous or nearly synchronous transactions into or out of the system. Examples include Windows messages, remote procedure calls, simple object access protocol (SOAP) published services, and so on. Each unique service that comes in or goes out through the application boundary to another application is counted. We do not count standard messages coming and going as part of the operating environment itself, only communication with applications external to the operating system. the number of messages, adjusted as just described, is multiplied by four to arrive at message related function points.

Range Estimates and Totals

For each of the five components of the count, estimates for new development may be expressed in terms of a best case estimate, an expected estimate, and a worst case estimate. The mean and standard deviation can then be calculated. The total function point is simply the sum of the function points contributed by each of the five components of the estimate.

Validation of Fast Function Points

Capers published data showing that the simplified SPR counting methodology was within 1.5% of the more formal (and difficult) full function point counting approach. Our research indicates the Fast Function Points are typically within 5% of full function point counts. More significantly (to us, at least), we find that effort and schedule estimates made using Fast Function Points are typically within 5% to 10% of actual values observed on real-world projects. We believe that whatever errors are introduced by Fast Function Points into the estimating process are more than offset by the improved ease of use and convenience of this approach, especially when our customers always have the option of using ISO function point counting if desired.

Some Case Studies

Fast Function Points were used to count a large (over 10,000 Function Point) social services system for a State agency. The fast function point count was completed by a team of three over a three-day period. In parallel, an IFPUG certified counting specialist and an assistant made a count of the same system, requiring six weeks of work.   The results were:

- The Fast Function Point count was within 3% of the IFPUG count;
- The predicted SLOC count was within 10% of the observed SLOC count.

On another project, the customer contracted for a fixed number of function points in software development. Throughout the requirement phase, the fast function point count was updated twice per day and the results were posted to the requirement definition team. The customer effectively prioritized requirements during the requirement phase to remain within the function point budget allocated. The system was then built to the defined requirements. The overall development was completed within 2% of the budget.

Conclusions

Fast Function Points are a simplified counting approach which is easy to learn, easy to understand, and easy to apply. The results are acceptably close to IFPUG function point counts for most purposes.

Biography

William Roetzheim is the founder of the Cost Xpert Group and has been involved in software estimation, project management, and metrics for over twenty-five years. He is the author of 15 technical books, over 100 papers, and holds two patents. Mr. Roetzheim has an MBA and has completed course work for a Masters degree in Computer Science. He can be reached at William@costXpert.com.