



# Managing (the Size of) Your Projects

## A Project Management Look at Function Points

Carol A. Dekkers  
*Quality Plus Technologies, Inc.*

*This article is an introduction, as well as a refresher, to readers who need to update their knowledge about function points (FPs). It includes many of the concepts presented in my presentation, "Requirements Are (the Size of) the Problem," at the 1998 Software Technology Conference in Salt Lake City, Utah.*

The use of FPs is becoming a widely supported method to create meaningful software metrics. In the past year, there has been a proliferation of software industry conferences with an increased emphasis on software measurement and FP-based metrics. Even tracks within traditionally non-measurement conferences now feature software measurement and FP presentations (the American Society for Quality's International Conference on Software Quality, the Quality Assurance Institute's International Quality Conference, the Software Technology Conference, and The MITRE Corporation's Software Engineering and Economics Conference, to name a few).

Prominent systems consultants are also increasingly vocal and supportive of FP-based metrics. For example, Howard Rubin based many of his workflow measurements on FPs (see *IT Metric Strategies*, December 1997), Capers Jones frequently highlights FPs as a basis for year 2000, productivity, and quality metrics; and Ed Yourdon discussed FPs in his presentation at the Applications of Software Measurement Conference (Atlanta, Ga., October 1997).

As president of the International Function Point Users Group (IFPUG), I appreciate the visibility generated for FPs by leading software practitioners, but I am disappointed that many professionals remain unfamiliar or misinformed about what functional size can and cannot do. There are even professionals who believe that by implementing FPs they can throw away all other software measures—they think FPs are *the* measurement program, but they are not.

FPs provide *one* additional project management tool, but they are not the entire toolkit. This article outlines the major components involved in FP count-

ing, including counting examples. It ends with a look at the misconceptions about what FPs can and cannot do. (Other published articles such as "Demystifying Function Points – Understanding the Terminology" are available from the author.) This article also provides project managers with enough knowledge about FPs and their usage to make informed decisions about when and how to implement functional sizing on their projects.

### What Are Function Points?

FPs measure the size of a software project's work *output* or work product rather than measure internal features such as lines of code (LOC). FPs evaluate the size of the functional user requirements that are supported or delivered by the software. In simplest terms, FPs measure *what* the software must do from an external, user perspective, irrespective of how the software is constructed. Similar to the way that a building's square measurement reflects the floor plan size, FPs reflect the size of the software's functional user requirements.

However, to know only the square foot size of a building is insufficient to manage a construction project. Obviously, the construction of a 20,000 square-foot airplane hangar will be different from a 20,000 square-foot office building. In the same manner, to know only the FP size of a system is insufficient to manage a system development project: A 2,000 FP client-server financial project will be quite different from a 2,000 FP aircraft avionics project.

I will carry the construction analogy one step further. To exclusively count the LOC (which many military departments do) is similar to counting the total number of construction pieces, e.g., 850,000 individual components that weigh

15,000 tons. This counting method is more indicative of construction methods than the functions (FPs) contained within a building. FPs are similar to summing up the square footage of functional items on a floor plan, e.g., the square footage of four sets of 10 x 15-foot restrooms, nine 12 x 20-foot meeting rooms, one 30 x 20-foot boiler room, two 10 x 40-foot staircases. FPs, like square feet, provide a normalized *size*, based on summing up the component functions that the resultant product must provide.

Because—no matter how large or how small—most software is developed to address user requirements, it can be measured in FPs. Some practitioners use the analogy of a "black box" to describe how FPs measure software independent of the inner workings of the software.

The process to calculate FPs is maintained by IFPUG and documented in its *Function Point Counting Practices Manual* (currently in release 4.0). Unlike LOC, FPs are independent of the physical implementation and languages used to develop the software, and they remain consistent no matter what development language or technique is used.

The fit between FPs and software development can be described analogously with square feet and construction (Table 1).

### Why Use Function Points?

FPs provide an objective project size for use in estimating equations (together with other factors) or to normalize productivity or quality ratios. The value in using FPs lies in the ratios and normalized comparisons between ratios. Process improvements can be found when normalized ratios are compared and their underlying project attributes evaluated.

Metric	Construction Units of Measure	When Is It Important to Measure?	IT Units of Measure	When Is It Important to Measure?
Estimated Project Size	Square feet.	During floor plans stage.	FP.	During requirements or contract stage.
Unit Cost, Overall Cost	Cost per square foot, total cost.	During construction contract negotiation.	Cost per FP, cost.	Before go or no go development decision.
Estimated Work Effort	Man-months.	Throughout construction or whenever change occurs.	Hours or man-months.	Throughout development or whenever change occurs.
Size of Change Orders	Square feet, cost (impact).	Whenever change is identified.	FP, cost, or hours (impact).	Whenever change is identified.

Table 1. *Function points as a construction analogy.*

FPs provide a standard, normalized measure of the work product or *functional size* of software. Together with other measures, FP-based software metrics highlight process improvement opportunities and can increase estimating and prediction accuracy.

### The Key to Counting Function Points: “Think Logical”

A fundamental feature of FP counting is that everything is counted from a *logical* user perspective, based on functional user requirements.<sup>1</sup> This is a paradigm shift for developers who are excellent at programming and physical configuration management. It does not matter to the functional size (FPs) whether it takes one thousand lines of COBOL code and eight subroutine calls or 100 lines of C++ code to perform a given business function; the FP count remains the same because the user function is the same.

Because excellent developers are akin to excellent plumbers involved in home construction, it takes a change in their mind-set to remove themselves from the physical implementation and look only at the floor plan. At this point, perhaps I have lost some developers who may think, “But the plumbing is important to keeping the house functioning. If we do not count other aspects of the software like the development language, we cannot accurately predict how long it will take to build.” This assertion is absolutely correct, but functional size (FPs) is not the same as work effort. Here is the relationship:

**Size (in FPs):** An *independent* measure of the software’s logical size. (Like the total room count and square footage of the finished building, which are constant regardless of construction methods.)

**Work effort (in hours):** A *dependent* measure of how long the software will take to develop, equal to a function of size, language, platform, skills, methods, team size, risks, and many other variables.

**Productivity (in hours per FP):** A *dependent* result, dependent on all the same factors as work effort. *Note that an independent variable (FP) divided by a dependent variable (hours) yields a dependent result.*

This means that just as the quality of the raw materials, piping configuration, and house layout affects the *work effort* it will take to plumb a house, so, too, will the language and other attributes affect software development time. However, regardless of *how* the house is designed and constructed, the functional size of the house stays the same. With software, the software *size* (in FPs) is independent of the language, skills, physical configuration, and other factors used in the development. When you use FPs, you are talking only about the software size.

### What Gets Counted in Function Point Counting?

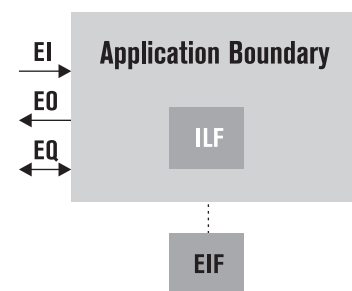
To count FPs, evaluate the following five *logical* components of the software based on the user requirements (Figure 1):<sup>2</sup>

- **Internal logical files (ILFs)** – Logical, persistent entities maintained by the software application.
- **External interface files (EIFs)** – Logical, persistent entities referenced only by this software application but which are maintained by another software application.
- **External inputs (EIs)** – Logical, elementary business processes that cross into the application boundary to maintain the data on an ILF or to ensure compliance with user business requirements, e.g., control data.
- **External outputs (EOs)** – Logical, elementary business processes that result in data leaving the application boundary to meet a user requirement, e.g., reports, data files, tapes, and screen alerts.
- **External queries (EQs)** – Logical, elementary business processes that consist of a data “trigger” followed by a retrieval of data that leaves the application boundary.

The five types of logical components counted are not the same as physical components. Discussion of ILFs, for example, does not refer to the physical files or data sets. ILF refers to the logical, persistent entities maintained through a standardized function of the software—they are the stand-alone, logical entities that typically would appear on an entity relationship diagram. For example, in a human resources application, an associate or employee entity would be maintained. This entity would be counted as an ILF.

Another illustration of counting logical components is when referring to EIs, which are the logical, elementary business processes that maintain the data on an ILF or that control processing.

Figure 1. *IFPUG function point components in relation to the application.*



## Making Adjusted FP Counts

Without getting into *IFPUG Function Point Counting Practices Manual* specifics of how to rate components as low, average, or high complexity, the following illustrates how to arrive at the unadjusted FP count for an application or development project. Following are the functional user requirements.

- Create an application to store and maintain employee records consisting of the following data fields: name, number, rank, street address, city, state, ZIP code, date of birth, telephone number, office assigned, and date the employee data was last modified.
- The application must provide a means to add new employees, update employee information, terminate employees, and merge two employee records.
- The application must provide a weekly report on paper that lists which employees' information has changed during the past week.
- The application must provide a means to browse employee data.
- No data outside the application is referenced, and all data validation edits are done using hard coded (not modifiable) data.

The FP components to be counted based on the above include

- **One ILF** for the employee data because it is a persistent logical entity maintained by the application. Based on an evaluation of the data elements and logical record types

(contained in the counting practices manual), this ILF would be categorized low and be worth seven FPs.

- **Four EI processes:** One EI each for add employee, update employee, terminate employee, and merge employee records. Assuming each one is of low complexity (each requires only one logical entity and requires fewer than 16 data elements), each EI would be worth three FPs for a total of 12 FPs.
- **One EO process:** The weekly report would be categorized as an external output and typically would consist of fewer than 20 data elements and require only the employee logical file. Based on the counting rules, this external output would be classified low and be worth four FPs.
- **One EQ process.** The process to browse the employee data would be classified as an EQ. Based on the number of data elements (fewer than 20) and the number of logical files accessed (the employee ILF), this EQ would be classified low and be worth three FPs.

Total components from the above four points: 26 unadjusted FPs. The final step would be to determine the value adjustment factor based on the user business constraints evaluated per the *Function Point Counting Practices Manual*. Guidelines are provided in the manual to help FP practitioners properly evaluate the adjustment factor.

The logical business process of adding an associate would be one user function; therefore, in function point counting, you would count one EI. The size in FPs for this one EI would be the same regardless of how we physically implemented it because in every implementation, it performs one logical user function. For example, the count for

“add associate” is the same regardless of the number of screens, keystrokes, batch programs or pop-up data windows needed to complete the process.

### What Is Involved in Function Point Counting?

The basic steps<sup>3</sup> involved in function point counting include

- Determine type of count, e.g., new development project, application or base count, or enhancement project count.
- Identify the application boundary, i.e., what functions must the software perform? This creates a context diagram for the application or project.
- Count the data function types.
  - ILFs – logical data groups maintained within the application boundary.

- EIFs – used only for reference by the application.
- Count the transactional function types.
  - EIs – data entry processes and controlled inputs.
  - EOs – e.g., reports.
  - EQs – e.g., question-and-answer pair that results in data retrieval.
- Evaluate the complexity of each of the five function types identified above. IFPUG provides several simple matrices to determine whether a function is low, average, or high, based on data element types (user recognizable, non-recursive data fields), record element types (subsets of user-recognizable data), and file types referenced (number of logical data groupings required to complete a process). Table 2 summarizes the number of FPs assigned to each function type. Following the IFPUG guidelines, count and rate all the identified functions and add the FPs together. The resulting number is the *unadjusted FP count*.

- Determine the value adjustment factor (VAF), which reflects the user problem complexity for the developed software. The VAF is calculated via an equation ( $VAF = 0.65 + (\text{sum of GSCs} \times .01)$ ) and the evaluation of the following 14 general system characteristics (GSCs). Specific evaluation guidelines for the following GSCs are provided in the *IFPUG Function Point Counting Practices Manual*.

- Data Communication.
- Distributed Data Processing.
- Performance.
- Heavily Used Configuration.
- Transaction Rate.
- On-Line Data Entry.
- End-User Efficiency.
- On-Line Update.
- Complex Processing.
- Reusability.
- Installation Ease.
- Operational Ease.
- Multiple Sites.
- Facilitate Change.
- Calculate the final adjusted FP count (adjusted function count = unadjusted FP count x VAF).

Table 2. *Unadjusted FP values by component.*

Function Type	Low	Average	High
EI	x 3	x 4	x 6
EO	x 4	x 5	x 7
EQ	x 3	x 4	x 6
ILF	x 7	x 10	x 15
EIF	x 5	x 7	x 10

## The Logical Boundary

One of the first steps of counting FPs is to identify the logical boundary around a software application. This "boundary" separates the software from the user domain. (Users can be people, things, other software applications, departments, other organizations, etc.) As such, the software may span several platforms and include both batch and on-line processes. The boundary is not drawn around the software in terms of how the system is implemented but rather in terms of how an experienced user would view the software. This means that a single application boundary can encompass several hardware platforms, e.g., both main-frame and PC hardware used to provide an accounts receivable application would be included within the application boundary.

### Where Do Function Points Fit In?

Once the adjusted FP count for a project or application has been created, it becomes the size of the work product. Just as the total functional size of a house does not equal the speed at which a house can be built or its construction time, the FP size does not equal productivity or work effort. FPs measure the size of *what* the software does, rather than *how* it is developed and implemented. This means that given a common set of logical user requirements, the FP size of the software will be the same whether it is developed using COBOL or DB2 or

using rapid application development or structured development methods.

## Where Can I Learn More About Function Points?

If you are going to get serious about software measurement or FPs or just want further information on how to get started with a measurement program, contact IFPUG or me. Incorporated in 1986, IFPUG is a not-for-profit users' group that has become a leader in establishing and publishing function point-related documents, including the *Function Point Counting Practices Manual*, the *Guidelines to Software Measurement* (currently in release 1.1), *Function Points as Assets* guide, and several detailed FP case studies. IFPUG remains a volunteer organization (with a small, paid administrative staff), is active in International Organization for Standardization (ISO) standards, sponsors conferences and workshops, and certifies FP training, counters (certified function point specialists), and FP software. Currently, paid membership represents over 30 countries worldwide. ♦

### About the Author

**Carol A. Dekkers** is president of Quality Plus Technologies, Inc., a management consulting firm specializing in training and consulting in function points, software metrics, requirements, and estimation process improvement. She is president of the IFPUG board of directors



and is a project editor within the ISO functional size measurement workgroup (ISO/IEC/JTC1/SC7 WG12). She is a frequent presenter and trainer at both U.S. and international quality and measurement conferences and is credentialed as a certified management consultant, a certified function point specialist, a professional engineer (Canada), and an Information Systems professional.

Quality Plus Technologies, Inc.  
8430 Egret Lane  
Seminole, FL 33776  
Voice: 727-393-6048  
Fax: 727-393-8732  
E-mail: [dekkers@qualityplustech.com](mailto:dekkers@qualityplustech.com)  
Internet: <http://www.qualityplustech.com>

### Notes

1. If you do not have "functional user requirements," does that mean you cannot count FPs? There are always requirements, although they may not be fully articulated by users or documented in a clear and complete fashion. In the early stages of software development, you may have to estimate the requirements or make assumptions about the user requirements and subsequently base your count on those assumptions. (See "Requirements Are [the Size of] the Problem," *ITMetric Strategies*, March 1998, which further explores the topic.)
2. The components listed are taken from the IFPUG *Function Point Counting Practices Manual*, Version 4.0, February 1994. The explanatory text in italics below each component is my wording to describe each component. For further information or to obtain the manual, contact the IFPUG administrative office in Westerville, Ohio at 614-895-7130 or visit the Web site at <http://www.ifpug.org>.
3. These steps condense the full details of function point counting included in the *Function Point Counting Practices Manual*, Version 4.0. Additionally, there are full case studies of FP counts, done at differing phases of application development, that can also be ordered through the IFPUG office.

## Types of Function Point Counts

The full details of FP counting procedure is contained in the IFPUG *Function Point Counting Practices Manual*, Version 4.0. There are two major types of FP counts:

- **Application or base FP count:** This count is the size of an installed base application. (Think of it in terms of total square feet of an existing house). The base size in FP is a point-in-time snapshot of the current size of an application. This number is useful whenever comparisons are required between different applications, e.g., defects divided by base FPs.
- **Project or enhancement FP count:** This count reflects the size of the functional "area touched" by an enhancement project. An enhancement project count is the result of summing the new functions added in the project plus the functions removed from the application by a project plus the functions changed by the project. (Think of this count in terms of a renovation project where the square foot of the project equals the sum of the area of a new living room, a removed bathroom, and a remodeled kitchen). This count is useful in project-based metrics, e.g., relative cost in dollars divided by development FP.

As the project is finished, the application or base FP count must be updated by the net, i.e., new minus removed plus the net difference in the changed functions.