



Software Sizing with Function Points

Function Point Counting can be conducted for **any software** platform, technology and environment. DCG simply requires copies of current systems documentation or access to a systems expert, who can answer questions regarding the applications to be counted. Counting can be performed independent of training or in combination with any of our courses.

Function Point Auditing is a more rigorous review of your Function Point counts and can be customized for your requirements. DCG auditing engagements typically include: (1) Auditing as an independent third party auditor for outsourcing agreements, (2) Auditing project and/or application counts for consistency and accuracy within an organization, (3) Auditing project counts in tandem with function point process review, to ensure not only accuracy of the counts but also the integrity and effectiveness of the process used to collect and count the data.

Function Point Baselining will accurately size your **current software application portfolio** using Function Point counting techniques. Each application will be sized, aged and categorized, culminating in a measured baseline, providing you with critical resource support information.

Function Point Approximation will **quickly and effectively size** your current software application portfolio based upon the accuracy level the client seeks. DCG conducted a detailed study during 1999 to determine the cost and accuracy of various counting techniques including full function point counting, approximation, estimating and backfiring. The results illustrated below indicate that it is not always practical or necessary to invest in full counting for a given project or application.

COST COMPARISONS FOR FUNCTION POINT COUNTING

Count Type	Accuracy	Cost
IFPUG	+/- 5%	1-3 days
IFPUG-Limited	+/- 25%	1-3 days
Approximation	+/- 35%	½ day
Ratio	+/- 50%	< ½ day
Expert	+/- 50%	< ½ day
Delphi	+/- 100%	< 1/4 day
Backfire	+/- 100-400%	Varies

Variations occurs based on language levels

Note: The cost is based on an average size application (250-1200 FP) and will vary for applications outside that size range

Alternative Sizing

If readers are planning to use the backfire method, an increase in accuracy can be achieved (relatively speaking) by using current backfire values (based upon IFPUG Counting Practices Manual 4.1 or 4.2). The conversion rates which follow reflect changes and clarifications to

www.davidconsultinggroup.com

Copyright © 2006 David Consulting Group
1770 E. Lancaster Ave, Suite 15
Paoli, PA 19301
+1.610.644.2856 (F) +1.866.293.0120



Software Sizing with Function Points

counting rules that have occurred since Release 3.0; e.g., counting each level of HELP rather than each occurrence, not counting error messages as EOs, not counting different formats or sorts for EIs, EOs or EQs, not counting super-files, not counting detail/summary data separately when it is produced in one report, etc. In referencing other backfire numbers, readers should ensure that they have been updated from those numbers published prior to 4.1.

Function Point Approximation will quickly and effectively size a current software application portfolio based upon the accuracy level the being sought. DCG conducted a detailed study during 1999 to determine the cost and accuracy of various counting techniques including full function point counting, approximation, estimating and backfiring. The results illustrated below indicate that it is not always practical or necessary to invest in full counting for a given project or application.

Cost Comparisons for Function Point Counting		
Count Type	Accuracy	Cost
IFPUG	+/- 5%	1-3 days
IFPUG - Limited	+/- 25%	1-3 days
Approximation	+/- 35%	1/2 day
Ratio	+/- 50%	< 1/2 day
Expert	+/- 50%	< 1/2 day
Delphi	+/- 100%	< 1/4 day
Backfire	+/- 100% - 400% *	varies
* Variation occurs based upon language levels		
Note: The cost is based on an average size application (250-1200 FP) and will vary for applications outside of that size range.		

The study conducted in 1999 was supported and participated in by several DCG client companies. The focus of the study was to determine the cost and accuracy of various counting techniques. Among those techniques included were full function point counting, approximation, estimating and backfiring. As a result of the study, DCG recommends that all baseline counts be accomplished with full disclosure as to the accuracy of the method being utilized. It is not always practical or necessary to invest in the full counting of each application in an organization's portfolio. The following high level overview of counting criteria can be used by organizations to discuss and determine which counting method is right for a given application.

- IFPUG Detailed - a complete and detailed count; typically performed on highly visible systems, systems that are core to the business or systems that may be undergoing frequent change requests
- IFPUG Limited - similar to "Detailed" in that accuracy is a primary concern, but average weightings are applied
- Approximation - the most robust of the approximation methods, used when accuracy is not of primary concern, but full functionality needs to be recognized

www.davidconsultinggroup.com

Copyright © 2006 David Consulting Group
1770 E. Lancaster Ave, Suite 15
Paoli, PA 19301
+1.610.644.2856 (F) +1.866.293.0120



Software Sizing with Function Points

- Ratio - typically used in instances where all data can be identified and logically parsed into user identifiable groupings
- Expert - used in cases of commercial off-the-shelf packages or with common applications where the consultant is familiar with similar types of applications
- Delphi - the least effective of the approximating and estimating techniques, Delphi can be used when an organization's portfolio has a certain percentage of "Detailed" or "Limited" counts available
- Backfire Calibration - a backfire method that utilizes a customized backfire value; used when accuracy is not an issue, but a sense of overall functionality being supported is necessary
- Backfire Calculation - same as "Calibration"; only industry backfire values are used

Lines of Code Counting

There are two methods for counting lines of code: Logical and Physical. A Logical count includes each logical statement that is terminated by a source statement delimiter such as a semicolon, colon or period (regardless of how many there are and regardless of whether the delimiter is on one physical line of code or after numerous physical lines of code). A Physical count includes each line of code that is terminated physically (by hitting the ENTER key of a computer keyboard, which completes the current line and moves the cursor to the next line). Table 1 sets out the accuracy of each type of LOC count on a scale of 1 to 4 with 1 being most accurate and 4 least accurate:

Accuracy Scale	Type of Count
1	Logical Count with tool support
2	Logical Count without tool support (hand count)
3	Physical Count with tool support
4	Physical Count without tool support (hand count)

In Performing a Logical Count:

- Count each statement terminated by a source statement delimiter such as a semicolon, colon or period.

In performing a Physical Count:

- Do count all executable lines used for actions such as the combination If, Then, Else statements or mathematical formulas
- Do count all data definitions used to identify information types
- Do not count comments used to inform readers of the code
- Do not count blank lines used to separate sections visually



Software Sizing with Function Points

The following table provides estimates of how many lines of code correspond to a single function point for a variety of programming languages.

Lines of Code Conversion Ratio to Function Points for Legacy Systems

Source Code Language (LOC Per Function Point)	Conversion Ratio
Basic Assembly	575
JCL	400
Macro Assembly	400
C	225
Cobol 74 (Cobol I)	220
FORTRAN	210
Cobol 85 (Cobol II)	175
Pascal	160
PL/1	126
RPG I	120
RPG II/III	110
Natural	100
C++	80
Java	80
dBase III	60
Focus	60
Clipper	60
Oracle	60
Sybase	60
dBase IV	55
Perl	50
JavaScript	50
VBScript	50
Shell Script	50
SAS	50
APL	50

Higher Level Languages are not included because of their significantly high divergent rates.

DCG does not consider backfiring as a recommended approach for sizing software project deliverables. Backfiring should be used only when sizing an organization's installed applications. Even then, the numbers should be validated by sampling (performing actual Function Point



david consulting group



Software Sizing with Function Points

counts and comparing them to Lines of Code) and validating applications within the installed base at that organization

To assist in your further evaluation on how you can use function points to size software, contact us at 610.644.2856 or send inquiry e-mail to info@davidconsultinggroup.com