# Function point

From Wikipedia, the free encyclopedia

A **function point** is a unit of measurement to express the amount of business functionality an information system (as a product) provides to a user. The cost (in dollars or hours) of a single unit is calculated from past projects.[1]

As of 2012, there are five recognized ISO standards for functionally sizing software:

- COSMIC: ISO/IEC 19761:2011 Software engineering. A functional size measurement method.
- FiSMA: ISO/IEC 29881:2008 Information technology - Software and systems engineering - FiSMA 1.1 functional size measurement method.
- IFPUG: ISO/IEC 20926:2009 Software and systems engineering - Software measurement - IFPUG functional size measurement method
- Mark-II: ISO/IEC 20968:2002 Software engineering - Ml II Function Point Analysis - Counting Practices Manual
- NESMA: ISO/IEC 24570:2005 (http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37289) Software engineering - NESMA function size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis

More recently, OMG produced a standard for Automated Function Point (AFP) counting. Announced in 2012,[2] it's available to the public for download at www.omg.org (http://www.omg.org)

Automated FP standard by OMG replicates the IFPUG process by detecting data and transaction functions, and distinguish internal and external logical files to calculate function points. The push for adoption was led by Dr. Bill Curtis, Director of the Consortium for IT Software Quality (CISQ), Capers Jones and David Herron, member of the IFPUG's Management and Reporting Committee, co-author of *The IFPUG Guide to IT and Software Measurement*,[3] and one of the original founders of the David Consulting Group (http://www.davidconsultinggroup.com/about), an independent consulting firm expert in Function Point.

## Contents

## Introduction

Function points were defined in 1979 in *Measuring Application Development Productivity* by Allan Albrecht at

IBM.[4] The functional user requirements of the software are identified and each one is categorized into one of five types: outputs, inquiries, inputs, internal files, and external interfaces. Once the function is identified and categorized into a type, it is then assessed for complexity and assigned a number of function points. Each of these functional user requirements maps to an end-user business function, such as a data entry for an Input or a user query for an Inquiry. This distinction is important because it tends to make the functions measured in function points map easily into user-oriented requirements, but it also tends to hide internal functions (e.g. algorithms), which also require resources to implement, however, there is no ISO recognized FSM Method that includes algorithmic complexity in the sizing result. Recently there have been different approaches proposed to deal with this perceived weakness, implemented in several commercial software products. The variations of the Albrecht based IFPUG method designed to make up for this (and other weaknesses) include:

- Early and easy function points - Adjusts for problem and data complexity with two questions that yield a somewhat subjective complexity measurement; simplifies measurement by eliminating the need to count data elements.
- Engineering function points :- Elements (variable names) and operators (e.g., arithmetic, equality/inequality, Boolean) are counted. This variation highlights computational function.[5] The intent is similar to that of the operator/operand-based Halstead Complexity Measures.
- Bang measure - Defines a function metric based on twelve primitive (simple) counts that affect or show Bang, defined as "the measure of true function to be delivered as perceived by the user." Bang measure may be helpful in evaluating a software unit's value in terms of how much useful function it provides, although there is little evidence in the literature of such application. The use of Bang measure could apply when re-engineering (either complete or piecewise) is being considered, as discussed in Maintenance of Operational Systems—An Overview.
- Feature points - Adds changes to improve applicability to systems with significant internal processing (e.g., operating systems, communications systems). This allows accounting for functions not readily perceivable by the user, but essential for proper operation.
- Weighted Micro Function Points - One of the newer models (2009) which adjusts function points using weights derived from program flow complexity, operand and operator vocabulary, object usage, and algorithmic intricacy.

## Benefits

The use of function points in favor of lines of code seek to address several additional issues:

- The risk of "inflation" of the created lines of code, and thus reducing the value of the measurement system, if developers are incentivized to be more productive. FP advocates refer to this as measuring the size of the solution instead of the size of the problem.
- Lines of Code (LOC) measures reward low level languages because more lines of code are needed to deliver a similar amount of functionality to a higher level language. C. Jones offers a method of correcting this in his work.[6]
- LOC measures are not useful during early project phases where estimating the number of lines of code that will be delivered is challenging. However, Function Points can be derived from requirements and therefore are useful in methods such as estimation by proxy.

## Criticism

Albrecht observed in his research that Function Points were highly correlated to lines of code,[7] which has resulted in a questioning of the value of such a measure if a more objective measure, namely counting lines of code, is available. In addition, there have been multiple attempts to address perceived shortcomings with the measure by augmenting the counting regimen.[8] [9] [10] [11] [12] [13] Others have offered solutions to circumvent the challenges by developing alternative methods which create a proxy for the amount of functionality delivered.[14]

## See also

- Weighted Micro Function Points
- Source lines of code
- Software development effort estimation
- Software Sizing
- Mark II method
- Comparison of development estimation software
- COSMIC software sizing

## References

1. ^ Thomas Cutting, Estimating Lessons Learned in Project Management - Traditional (http://www.pmhut.com/estimating-lessons-learned-in-project-management-traditional), Retrieved on May 28, 2010
2. ^ OMG Adopts Automated Function Point Specification, January 17, 2013 (http://news.silobreaker.com/omg-adopts-automated-function-point-specification-5_2266545367346577623)
3. ^ IFPUG (2012). *The IFPUG Guide to IT and Software Measurement*. Auerbach Publication. ISBN 1439869308.
4. ^ A. J. Albrecht, “Measuring Application Development Productivity,” Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979), pp. 83–92.
5. ^ Engineering Function Points and Tracking System, Software Technology Support Center (http://www.stsc.hill.af.mil/crosstalk/1994/11/xt94d11e.asp), Retrieved on May 14, 2008
6. ^ Jones, C. Applied Software Measurement: Assuring Productivity and Quality. McGraw-Hill. June 1996.
7. ^ Albrecht, A. Software Function, Source Lines of Code, and Development Effort Estimation - A Software Science Validation. 1983.
8. ^ Symons, C.R. “Function point analysis: difficulties and improvements.” IEEE Transactions on Software Engineering. January 1988. pp. 2-111.
9. ^ Hemmstra, F. and Kusters R. "Function point analysis: evaluation of a software cost estimation model." European Journal of Information Systems. 1991. Vol 1, No 4. pp 229-237.
10. ^ Jeffery, R and Stathis, J. "Specification-based software sizing: An empirical investigation of function metrics." Proceedings of the Eighteenth Annual Software Engineering Workshop. 1993. p 97-115.
11. ^ Symons, C. Software sizing and estimating: Mk II FPA (Function Point Analysis). John Wiley & Sons, Inc. New York, NY, USA.1991
12. ^ Demarco, T. "An algorithm for sizing software products." ACM Sigmetrics Performance Evaluation Review. 1984. Volume 12, Issue 2. pp 13-22.
13. ^ Jeffrey, D.R, Low, G.C. and Barnes, M. "A comparison of function point counting techniques." IEEE Transactions on Software Engineering. 1993. Volume 19, Issue 5. pp 529-532.
14. ^ Schwartz, Adam. "Using Test Cases To Size Systems: A Case Study." 2012 Ninth International Conference on Information Technology- New Generations. April 2012. pp 242-246.

## External links

- The International Function Point Users Group (IFPUG) (http://www.ifpug.org/)
- The Common Software Measurement International Consortium (http://www.cosmicon.com/)
- The Netherlands Software Metrics users Association (NESMA) (http://www.nesma.nl/english/)
- ISBSG (International Software Benchmarking Standards Group) - http://www.isbsg.org
- MAIN (Metrics Associations' International Network) - http://www.mai-net.org
- Function Points: A New Way of Looking at Tools (http://doi.ieeecomputersociety.org/10.1109/MC.1994.10088)
- Function Point Analysis (http://foldoc.org/?Function+Point+Analysis) in FOLDOC

Retrieved from "http://en.wikipedia.org/w/index.php?title=Function_point&oldid=571054520"
Categories: Software metrics | Software engineering costs