

## **Appendix H**

# **Counting Rules for Function Points and Feature Points**

---

# Content

<b>H.1 Function Points</b> .....	H-3
<b>H.2 Feature Points</b> .....	H-4
<b>H.3 Conclusion</b> .....	H-4

---

## H.1 Function Points

One of the challenges facing a software developer is determining the quantity of software to be developed. Traditionally, the measurement used is lines of code. The number of lines of code required to implement an algorithm or function can vary greatly from programmer to programmer. Allan Albrecht of IBM developed an alternative to lines of code in 1979 and called the new measure Function Points. Albrecht revised this measure over the years and there is now an international group that controls the standard for function points, the International Function Point Users Group (IFPUG).

A function point is a synthetic metric comprised of the weighted totals of:

- Inputs
- Outputs
- Logical Files
- Interfaces
- Inquiries

Weighting is determined by using a multiplier for each of the parameters above based upon low, medium, or high complexity.

Further refinement by IBM in 1984 included 14 influential complexity factors, each with a range of 1 to 5 (with 0 being used if the factor is not present). The factors are:

- Data communications
- Distributed functions
- Performance objectives
- 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

The sum of these influential complexity factors is multiplied by 0.01 and added to a constant of 0.65 to determine a complexity multiplier. The raw function point number is multiplied by the complexity multiplier to achieve a final function point total that is a measure of the size and complexity of the software product.

In summary, the function point technique provides an objective, comparative measure which assists in the evaluation, planning, management and control of software production.

More information on function points can be found on the [IFPUG](#) web page or in chapter 2 of Capers Jones book, Applied Software Measurement.

---

## H.2 Feature Points

Function points work well in measuring management information system software, but do not do well in characterizing real-time systems, operating systems, process control systems, communications systems, embedded systems, or engineering systems. This is because these systems have few inputs and outputs while they have high algorithmic complexity.

To overcome this, Software Productivity Research (SPR) created a measure known as the Feature Point metric to distinguish it from the function point metric. It introduces a parameter, algorithms, in addition to the five standard function point parameters. The algorithms parameter is assigned a default weight of 3. The Feature Point method also reduces weight for logical files from the average value of 10 down to an average value of 7.

More information on Function and Feature Points can be found on the [Software Productivity Research \(SPR\)](#) web page or in Chapter 2 of Capers Jones book, Applied Software Measurement.

---

## H.3 Conclusion

Software developers are no longer left to the metric of lines of code to estimate software size. Function points or feature points can successfully be used as alternatives, and may often be a better measurement. However, be aware that several of the popular software effort estimation models require lines of code rather than function or feature points as an input. Another consideration is the use of function points or feature points as normalizers in organizational metrics. For across-the-organization comparisons and for comparison with the rest of the world. Lines of code are usually used as the measurement normalizer. Capers Jones book does include a table in Chapter 2 that provides conversion factors from function points to source code statements for 50 selected languages. No such table is available for feature points.