david consulting group
TRUSTED ADVISOR

Is Automated Function Point Counting (AFPC) ready for "prime time?"

July 2011

What do we mean by "Automated Function Point Counting"?

"Function Points" (FPs) were first developed at IBM by Allan Albrecht and his colleagues at the end of the 1970's to estimate the functional size of software requirements in a statistically consistent way. Today, five methodologies for FP analysis are included in the ISO/IEC 14143 standard although it should be noted that each methodology can produce a very different number for the same problem.

For our purposes, one characteristic that all five recognized methodologies have in common is that they rely on human, rather than machine, analysis of the software problem to generate a count. Hence, consistency in counting relies on regular and rigorous certification of the human counter and the counter's experience of the software requirements scenarios being presented.

While automated tools have been around for some time to assist the human counter, automating the entire counting process has proved to be an elusive goal that has not been conclusively achieved to date. The major reasons for this failure have been:

- Without going too much into the details, the current methodologies rely on the identification of the boundary of the count (e.g. project or application) and the categorization of the structures of the data (e.g. internal or external files) and transactions (e.g. inputs, outputs, inquiries). While some of the categories and structures are unambiguous, there is plenty of call for judgment on the part of the human counter. While the counter has an extensive book of rules to guide his judgment, there is enough ambiguity left that the governing bodies for the different counting methods only claim consistency for fully -certified counters in the range of +/- 5-10%.
- No widespread use of a rigorous, machine-readable language for describing functional requirements.

How does AFPC work?

The AFPC solutions available today require some rigorous formatting of the software requirement. The most readily available input format is a programming language although there is a least one offering on the market which uses UML as its input. At the heart of the AFPC tools is a language parser associated with a set of rules for interpreting the structured language once parsed. Several of the AFPC's available started life as static code analyzers for measuring and reporting on code quality.

Whereas the five ISO-recognized counting approaches have rules and certifications to try to maximize consistency of judgments amongst the human counters, consistency is not an issue within any one AFPC toll – it will always count precisely the same way. Instead, the main issue of the AFPC tools is accuracy

or, more realistically, validity. Since, there is no universal standard for measuring a function point, the AFPC systems cannot be assessed for accuracy. Instead, we have to test their validity and the only realistic way to do that is to compare their count results to one of the ISO-recognized human counting techniques – typically, the IFPUG approach because it has the most longevity and the most robust underpinning of statistical studies to corroborate it's consistency and proportionality to such key development metrics as effort and duration.

Typically, the tool being used for AFPC has to be "configured" with initial values appropriate for the software application being counted and then "calibrated" to capture additional information about the application that emerges during the first few analysis runs. This configuration and calibration usually involves a certified function point specialist and different tool providers choose to make this either interactive or sequential with the analysis runs.

What are the practical constraints on AFPC solutions?

1. Post-development sizing of applications for portfolio management or maintenance estimating: this is the low-hanging fruit for AFPC and the only constraint is the degree to which the ISO-recognized counting methods contain human capabilities that cannot be captured in the AFPC approach. The best known example is the counting of "inquiries" which humans can interpret from the written requirements but computers cannot easily detect in the source code. Note that this constraint can be ignored – the AFPC ignores inquiries – which could result in a perfectly reasonable FP count except that it is hard to validate against the ISO –recognized counts.
2. Pre-development sizing of projects or change requests: The need for rigorously format input data – today, this limits the applicability of AFPC to scenarios where source code or UML is available so pre-development sizing of projects or change requests for estimating purposes is not possible yet . Human counters are still required.
3. Post-development sizing of projects and change requests for actuals versus estimates or calculation of other metrics such as productivity: The same techniques have been applied to this area as have been applied to application but with less success so far. The reasons for less success are not clear but may be related to the problems of application boundary judgments. Human counters are still required.

Who offers AFPC solutions?

Efforts at standardization:

- Consortium for IT Software Quality www.it-cisq.org

Commercial offerings based on source code analysis:

- CAST Software www.castsoftware.com
- OutSystems www.outsystems.com
- Relativity Technologies www.microfocus.com

Commercial offerings based on UML analysis:

- FPModeler www.functionpointmodeler.com

Is there any evidence of success?

Yes. DCG has acted as an independent participant in trials of the AFPC capability of the CAST Software system. The CAST Software system is a mature software suite whose primary function and value is the ability to analyze the quality of code. However, the CAST team have been evolving the AFPC capability for some time.

The latest calibration exercise that DCG participated had a goal of validating the CAST-generated AFPC to within +/- 10% of an IFPUG FP count. Twenty software applications were run through the tool after configuration of the software but nor calibration. There was an appreciable improvement in validity of the AFPC's for the first runs of the last applications analyzed over the first runs of the first applications analyzed but none of the first runs met the trial success threshold of +/-10%. The next step was to perform calibration of the tool for each application. Every application was able to meet the trail success criteria within 5 runs (with a recalibration after each run). As the trial progressed through the twenty applications, the number of runs needed to meet the trail success criteria fell from 5 to 2 or 3.

The need for configuration and calibration requires manual input which adds to the cost of the tool itself. However, these results represent a significant step forward because after the initial configuration and calibration cost, the AFPC analysis can be run frequently (e.g. after every release) and almost indefinitely into the future with very low cost and resource consumption compared to the use of human counters.

Is AFPC ready for "prime time"? Yes – with limited but worthwhile scope but human counters look likely to be necessary for many years to come.