A New Software Metric to Complement Function Points

The Software Non-functional Assessment Process (SNAP)

Charley Tichenor

Abstract. Sizing software requirements is an essential best practice in software project management for forecasting the work effort required for software development projects (and other related metrics). Arguably, the currently most accurate software metric for measuring the size of software is the International Function Point Users Group (IFPUG) "function point," which has the ISO standard ISO/IEC 20926:2009. Function points basically measure the size of the data flow and storage through the software, which we define in this paper as "functional" requirements. But function points do not measure other software requirements, which also require work effort resources. IFPUG has recently completed a successful beta test of a new method to assess the size of other, "nonfunctional" requirements, which when used in conjunction with function points should further increase the accuracy of software forecasting. The authors believe that this Software Non-functional Assessment Process v. 2.0 (SNAP) is ready to enter industry and academia for initial practice and further research.

Introduction

Forecasting the cost to produce software has been transformed from an art into largely a science through a methodology called function point analysis. Function point analysis basically quantifies the volume of data flow and storage through the software application; based on this measurement the cost required to develop the software can be quantitatively forecast. Years of experience with function points has shown it to be a robust methodology [1]. Yet, one wonders if a complementary software metric could be developed and used along with function points so that data flow and storage, and other aspects of the software that function points do not consider can be measured. Combining these measurements should improve the quality of software development cost forecasting (and other software metrics).

One proposed complementary metric is from SNAP. IFPUG, through its Non-functional Sizing Standards Committee, SNAP Project Team, developed a procedure for SNAP and wrote the SNAP "Assessment Practices Manual," now in version 2.1 [2]. During August and September 2012, the SNAP team conducted a beta test to measure how well SNAP 2.0 correlated with work effort. This beta test was successful, and the purpose of this paper is to share the results of this beta test. We will discuss:

a. **What function points are**
b. **What SNAP is**
c. **Why SNAP may be important**
d. **How the beta test was conducted**
e. **What the results were**
f. **Areas for future research**

Review of the Related Literature

IFPUG is the largest software metric association in the world, with more than 1,000 members and affiliates in 24 countries. The non-profit International Software Benchmark Standards Group (ISBSG) has become the largest source of benchmark data, with more than 5,000 projects available. New benchmarks are being added at a rate of perhaps 500 projects per year. All of the ISBSG data is based on function point metrics [3].

IFPUG maintains arguably the most widely used functional software sizing metric in the world, the IFPUG "function point" (in this paper, we will always refer to the unadjusted function point). The IFPUG Counting Practices Manual [4] is one standard for measuring functional requirements, and is recognized by the ISO.

ISO/IEC 20926:2009 specifies the set of definitions, rules and steps for applying the IFPUG Functional Size Measurement method. ISO/IEC 20926:2009 is conformant with all mandatory provisions of ISO/IEC 14143-1:2007. It can be applied to all functional domains and is fully convertible to prior editions of IFPUG sizing methods. … ISO/IEC 20926:2009 can be applied by anyone requiring a measurement of functional size. Persons experienced with the method will find ISO/IEC 20926:2009 to be a useful reference [5].

A function point is like a "chunk" of software. It is similar in concept to a "square foot" of house size, a "kilometer" of distance, a "gallon" of gasoline, or a "degree Kelvin" of temperature. According to IFPUG's Counting Practices Manual, function points are assigned to different components of software according to the user's viewpoint (rather than the programmer's viewpoint). IFPUG recognizes five different types of software components, listed in the table below, that are basically measures of the data flow and storage through the software. Also listed are their relative sizes in terms of function points and based on their complexity levels.

	Low	Average	High
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6
Internal Logical File	7	10	15
External Interface File	5	7	10

Table 1

For example, an input screen process for entering data into an application might be measured as a low complexity external input worth three function points, and a high complexity external interface file is counted as 10 function points. The IFPUG Counting Practices Manual has repeatable standards for how to count function points and determining whether a component has low, average, or high complexity.

Data Operations	Technical Environment
Data entry validations	Multiple platforms
Extensive logical and mathematical operations	Database technology
Data formatting	Batch process
Internal data movement	
Delivering added value to users by data configuration	
Interface Design	**Architecture**
User interface methods	Mission critical/real time systems
Help methods	Component based software
Multiple input methods	Multiple input/output interfaces
Multiple output methods	

Table 2

Here is how we can use function points for forecasting the cost to develop software. First, as an analogy, suppose that a customer wants to build a new house in a certain community. Suppose further that a typical house in that community is built at a cost averaging $300 per square foot. If the customer wants a new house of 1,000 square feet, then a good estimate of its cost will be about $300,000. Suppose we are considering building a new software application. Before we start building it we want to forecast its cost. A qualified function point analyst starts by examining the software's data requirements. Then, using the standards in the IFPUG Counting Practices Manual, the analyst counts each instance of the components in Table 1 that are anticipated to be in the software, and then totals their values for the final function point count. (adapted from [6]).

This function point size correlates with development cost. The original paper showing that function point size correlates with development cost was published in 1977 by Dr. Allan Albrecht in his paper "Measuring Application Development Productivity [7]." This paper was the publication of the results of his research team's development of the initial version of the function point methodology at IBM. The team correlated function point size of various IBM applications with their corresponding work effort, and found the correlation to be statistically significant. Since the publication of this paper, numerous organizations have developed function point-based software productivity models to help them forecast software development costs. Some companies have compiled large amounts of such data from government, industry, and other sources, and built commercial software estimation tools which use function points and other productivity indicators (such as software language used, skill of the programming team, project management tools used, etc.) to help clients forecast their software development costs.

Now we can forecast the cost to develop this software. Suppose that the function point analyst identified the software's components from Table 1 and counted a total of 1,000 function points. Suppose further that a typical application of this type is built at a cost averaging $300 per function point. A good estimate of its total development cost is therefore about $300,000.

A reading of the IFPUG Counting Practices Manual indicates that function points are basically a measure of the size of the data flow and storage through the software. For this paper, we define these software requirements as "functional" requirements. The cost estimate of $300,000 for developing 1,000 function points of software is based on data flow and storage size—the functional requirements for the software.

Let us return to our house cost forecasting analogy. A new house of 1,000 square feet in size in this Community should typically cost about $300,000, but the particular house design this customer wants is a little different than "typical." Suppose that this customer also wants to add hardwood floors (instead of typically carpeted floors), a wood-burning fireplace, a refrigerator with an extra large freezer, and extensive wiring to support a special home entertainment system. We improve the cost estimate for this house by factoring in the additional costs of these extras.

Now, suppose we want our software cost estimate to factor in software requirements which are not included as functional requirements in the IFPUG Counting Practices Manual. Let us consider certain requirements within the following categories and their subcategories. These are from the SNAP Assessment Practices Manual (refer to Table 2).

In this paper, we define these kinds of software requirements as "non-functional" requirements because they are not included in the ISO standard function point methodology in the IFPUG Counting Practices Manual yet require additional work effort to develop. We want to assess the size of these non-functional requirements for applications. We also want to know if non-functional size statistically correlates to the corresponding work effort—like function points do. This was the fundamental paradigm of the SNAP beta test.

We want to base the beta test analytics on statistical methods. We include the notions of random sampling, regression models, the F test, p-values, the Runs test, and the Spearman test. Basic Statistics books (for example, [8]) treat these. The next paragraphs will discuss the intended testing analytics.

For the beta test, random sampling means that we collect SNAP sizes from a wide variety of applications across the world. As much as possible with the resources we have, we want to have a sample that represents the software development industry.

Regression is a way to find the correlation between two variables. In this beta test, we want to determine if there is correlation between the SNAP sizes of the applications and their corresponding work efforts. We believe that as the SNAP size increases, the work effort to build those SNAP sizes should also steadily increase.

Statisticians often look for several indicators to measure the degree of strength of the relationship within a set of two variables, in this case, the SNAP size and corresponding work effort. If there is causation, then one indicator (in this case) would be the degree to which SNAP size accounts for the amount of

work effort. This is measured by the r^2 statistic. For example (assuming causation), if our data's r^2 is measured to be .75, then we conclude that SNAP size accounts for 75% of the reason for the work effort.

Another statistic is the associated p-value for this, also called "Significance F" in Excel. The p-value is the probability that we are wrong in concluding that SNAP size is correlated to work effort. If the p-value is .05, then we are 5% sure that we are wrong in concluding such a correlation, or put another way, we are 95% sure that we have statistical significance.

There are some technical assumptions in the standard regression process. One is that the data points are randomly scattered about the regression line. We can test for this using the Runs test, and we are comfortable that the model passes the Runs test if its p-value is below .05.

We also want to test for correlation using the Spearman test. This is a nonparametric test for rank correlation and makes no technical assumptions about the distribution of the data, other than it is randomly scattered about the regression line. This is a "worst case scenario" test we use should we have doubts about the validity of the standard regression test.

The final statistical test is for compliance with Benford's Law. Benford's Law is an interesting statistical test. Software development is a human stimulus and response activity. Part of the overall stimulus for developing software is the need for the non-functional requirements. The response is the number of SNAP points generated. If this occurs, then we can look at the leading digits of the SNAP size. For example, if the SNAP size is 483, then we would consider the leading digit of "4." Benford's Law says that in these stimulus and response situations, the distribution of the leading digits is logarithmic, as in the table below, i.e., 30.1% of the SNAP sizes should start with the number "1," 17.6% of the sizes should start with "2," and so forth until we should measure "9" as the leading digit in about 4.6% of the SNAP sizes [9].

First Digit	Percentage of Occurrences
1	31.10%
2	17.60%
3	12.50%
4	9.70%
5	7.90%
6	6.70%
7	5.80%
8	5.10%
9	4.60%

Table 3

This compliance with Benford's Law happens with function points. A study presented at the 2009 Fourth International Software Measurement & Analysis conference [10] showed that for a large internationally collected sample of function point counts (more than 3,000 function point counts from ISBSG, Victoria, Australia), their leading digits followed the distribution predicted by Benford's Law almost exactly.

Although the SNAP sample will be much smaller, we hope to see good convergence towards Benford's Law.

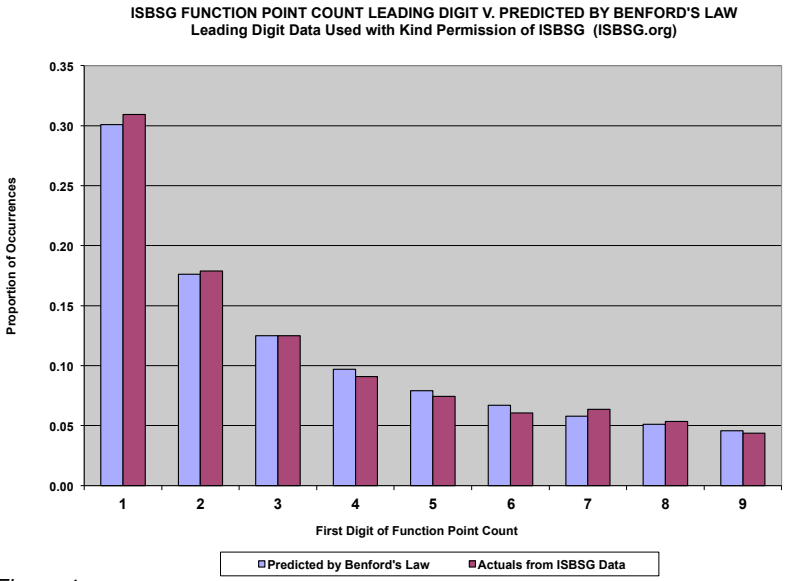

Figure 1

Research Design and Methodology

The purpose of this beta test was to repeat and extend the spirit of Dr. Allan Albrecht's statistical analysis of the early function point methodology for the SNAP methodology. Dr. Albrecht's research showed that software size measured in function points correlated with work effort for the applications tested. In a similar manner, based on data collected from the beta test, our research will hopefully determine the degree to which SNAP sizes correlate with corresponding work effort. Here is our research design and methodology.

Use version 2.0 of the SNAP manual as the basic reference.

Develop a standard SNAP data collection spreadsheet, largely based on last year's spreadsheet. This new spreadsheet had four worksheets:

1. "Basic Instructions" worksheet, which provides detailed instructions for data collection for the SNAP counter.
2. "Application Data" worksheet, for entering descriptive data.
3. "SNAP Counting Sheet," for entering the SNAP points. This worksheet permits the SNAP counter to enter only basic data per SNAP item, such as "DETs," "FTRs," "person-hours," and other data described by the SNAP training. The worksheet then automatically calculates SNAP points. All calculation cells are locked.
4. "Recap" worksheet, which automatically totals the SNAP sizes and work effort.

Issue a call for volunteer SNAP counters, and train them. This training will be done both using written materials (primarily the SNAP Assessment Practices Manual) and by telephone. The counters will choose the applications to size. Hopefully, this call for volunteers will result in a wide variety of countries represented and application types chosen.

Conduct all SNAP sizing at the application boundary level—"application boundary" as defined in the IFPUG Counting Practices Manual.

Collect at least 30 applications' worth of SNAP sizes with corresponding work effort in person-hours. This is to hopefully ensure a statistically large sample size.

If corresponding function point and work effort data can also be collected, then so much the better. This permits additional research. However, such function point counting data is considered optional.

Collect application descriptive data such as types of applications, types of industry, types of software, etc. This data may be used to help improve correlations. However, maintain source confidentiality.

Conduct the beta test throughout August and early September 2012. During the beta test, after counters finish with individual application SNAP sizings, they are to email their data collection spreadsheets to IFPUG. These data sheets will be then "cleaned" of any source information to maintain confidentiality, and then will be forwarded to one of several members of the SNAP team who will perform a "quality control" of the data collection.

As the SNAP data pass "quality control," they will be then forwarded on for statistical analysis.

The beta test analytics will consist of trying to determine the degree of statistical significance using the following tests. First, we will test the data plotting the SNAP sizes of the applications on the x-axis as the independent variables, and the effort expended on the y-axis as the dependent variables. We will use simple linear regression, and especially look at the r^2, what Excel calls "Significance F" (which is the p-value of the corresponding F test), and the p-values of the coefficients of the regression line. We will check for the appropriateness of testing for regression using regression through the origin. We will conduct the Runs test and Spearman test, and also test for convergence to Benford's Law. We will also experiment with changing weighting factors and other aspects of SNAP to try to both improve correlation and its degree of realism.

Presentation and Analysis of Data

We collected data from a wide variety of applications. This ensured that the sample was as close to random as reasonably possible. We had SNAP sizes for 58 applications usable for the part of the test correlating SNAP sizes with work effort, and an additional 14 SNAP sizes usable for the Benford's Law test (but did not have work effort data).

Data was collected from the following countries: Brazil, China, France, India, Italy, Mexico, Poland, Spain, UK, and the USA. We collected data from the following industries: Aerospace, Automotive, Banking, Government, Fast Moving Consumer Goods, Financial Services, Insurance, Manufacturing, Systems Integrators and Consulting, Telecommunication, and Utilities.

After reviewing the data, 58 data points (representing 58 software applications) had sufficient SNAP size and work effort data for further analysis. The first statistical test was a simple linear regression analysis for 58 applications with the SNAP sizes on the x-axis, and the corresponding work efforts in person-hours on the y-axis. The graph below shows the results of this regression. NOTE: the actual work effort hours are not shown on the y-axis of the forthcoming graphs; we do not want to imply that the productivity rate found in this beta test should necessarily be used as a benchmark—we feel that this is premature at this point.

The r^2 for this analysis is .33, which basically means that 33% of the reason for the work effort was due to the SNAP size.

A closer analysis of the graph (and Excel regression tables) shows that the trendline crosses the effort axis at about 100

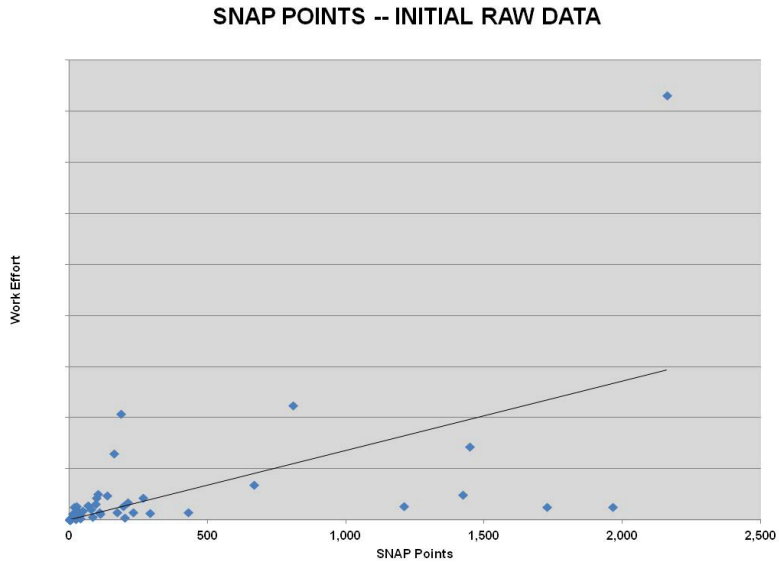

Figure 2

person hours. In theory, this means that if there were zero SNAP points, then the corresponding work effort should be about 100 person hours. This is not reasonable—if there are zero SNAP points then the work effort should also be zero. Therefore, we upgrade the analysis and use a standard technique called "regression through the origin." This forces the trendline through (0,0). This improves the common sense test and increases the r^2 to .41.

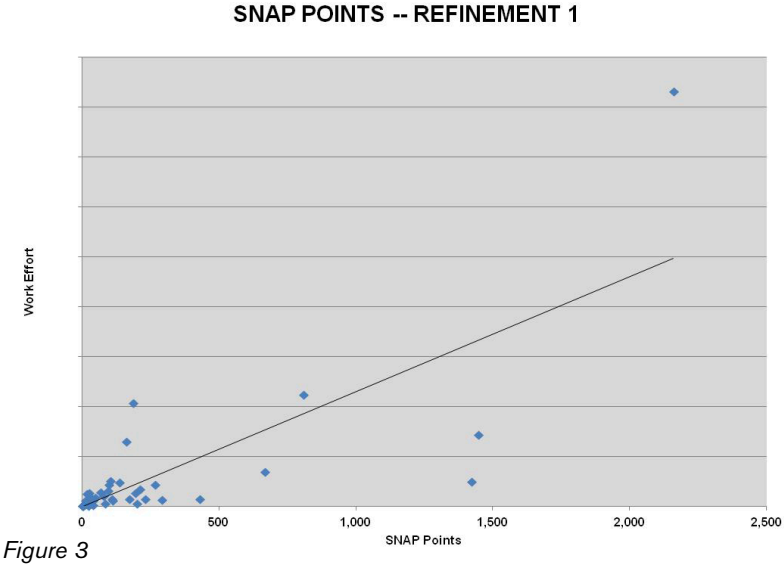

Figure 3

In reviewing the raw data, three applications contained large quantities of Help features. These applications had productivity rates, according to the current version of the model, that were roughly 10 times higher than the other 55 applications. This led us to believe that we may need to reformulate the Help Methods (subcategory 2.2) portion of the SNAP manual. This is an area for future research, so we removed these three applications from the data set. This improved the r^2 from .41 to .66. We later removed seven other applications that counted some Help features, to maintain consistency.

Also, we changed the weighing factors for subcategory 1.5 "Delivering Value Added to Users through Data Configuration"

by changing the weights for low, average, and high from 3-4-6 to 6-8-12. This improved the model's r^2 to .89, with a corresponding Significance F of $1.7 * 10^{-23}$.

To test the requirement that the data points in this model must be randomly scattered about the regression line, we conducted the Runs test. There were 19 runs in the data, which compares favorably with the theoretically optimal 19.96 runs.

We ran the Spearman test for rank correlation. This test produced a rank correlation of .85, with an associated confidence of statistical significance of greater than 99% (p-value <.0001).

The final results of this analysis are on the following viewgraph (refer to Figure 4).

We tested the final version of the results for compliance with Benford's Law. In terms of software development, Benford's Law says that the leading digits in a large portfolio of SNAP sizes should be distributed as in Table 3, repeated below. For example, in a large number of SNAP sizes, about 30.10% of the SNAP sizes should have a leading digit of "1," such as sizes of 15, 139, or 1,728.

First Digit	Percentage of Occurrences
1	31.10%
2	17.60%
3	12.50%
4	9.70%
5	7.90%
6	6.70%
7	5.80%
8	5.10%
9	4.60%

Table 3

Figure 5 shows the SNAP leading digit distribution from the beta test. We used 65 SNAP sizes for this analysis. In general, Benford's Law seems to converge rather slowly, i.e., it requires a very large sample size to "pure out." This SNAP sample size is much smaller than the ISBSG sample size, so the degree of compliance is markedly less; however, we appear to be converging nicely.

Conclusions

We believe that the SNAP Assessment Practices Manual 2.0 has passed the beta test.

a. The test was based on very good sampling techniques
b. The data points are randomly scattered about the regression line, as shown by the Runs test
c. The regression r^2 for 48 projects was .89
d. The Spearman test correlation was .85
e. We are over 99% sure that both tests are statistically significant
f. The distribution of the first digits of 65 SNAP sizes is converging nicely towards Benford's Law

We recommend that the SNAP procedure (with the exception of Help Methods subcategory 2.2) is ready for use by the industry, and is ready for further research.

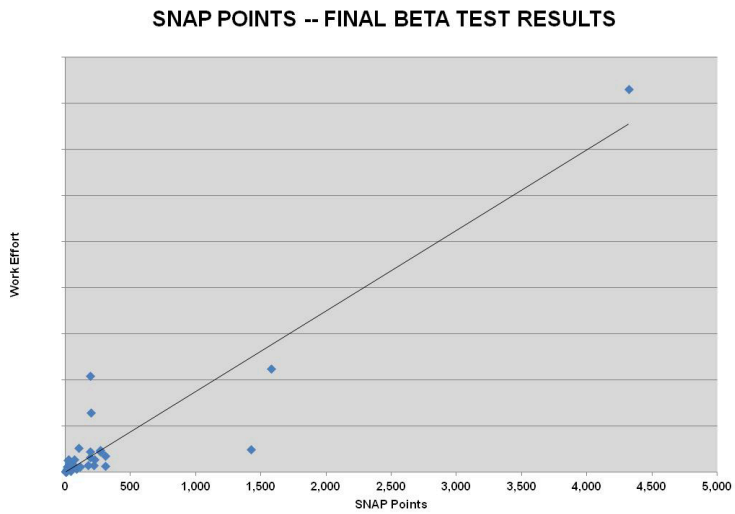

Figure 4:
n = 48 r^2 = .89 Significance F = $1.7 * 10^{-23}$ Spearman = .85 Runs = pass

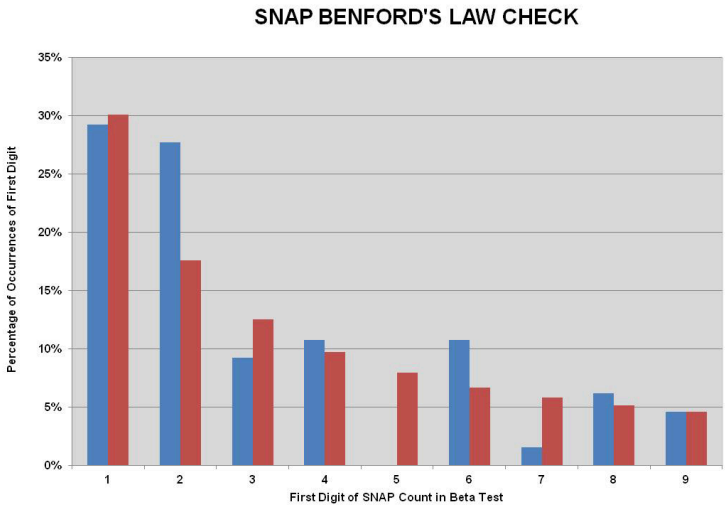

Figure 5

IFPUG has formed a Non-functional Sizing Standards Committee, similar to the Functional Sizing Standards Committee. This committee will continue to develop the SNAP process, encourage SNAP research, develop SNAP training, and maintain the SNAP Assessment Practices Manual.

Areas For Future Research

One possible source of data collection error during the beta test was the experience of the SNAP counters. This was their first use of the SNAP Assessment Practices Manual 2.0. Consistency has been tested for function point counters with very favorable results. Repeat similar consistency tests for SNAP counters after there is much SNAP counting experience in the field.

Continue to experiment with reasonably varying the values of the factors for each subcategory's low, average, and high complexity weights to improve the correlation between SNAP sizes and work effort.

Continue to research the Help Methods, subcategory 2.2.

After a statistically large number of applications have been counted for both function points and SNAP points, conduct research to determine if function points and SNAP points can be combined into a single metric, which correlates to the combined work effort to develop both. Try to combine them like real numbers can be combined with imaginary numbers to produce the complex numbers; try other ideas.

Using a large sample from the ISBSG database, function point counts were tested for compliance with Benford's Law. This almost perfect compliance gave good statistical indication for the soundness of the underlying mathematical structure of function points. After completing a larger number of SNAP sizings (probably over 100), continue repeating this research by testing SNAP sizes for compliance with Benford's Law.

Comments:

This paper is written on behalf of the IFPUG SNAP team. The team developed the SNAP process and published the 130 page "Software Non-functional Assessment Process (SNAP) Assessment Practices Manual," now in version 2.1. The team conducted the version 2.0 beta test to include its research design, the call for SNAP assessors, their training, and analysis of the test results. The team also developed a two-day workshop to introduce the Assessment Practices Manual at the seventh International Software Measurement & Analysis conference in Phoenix, AZ in October 2012.

The SNAP Project Manager and IFPUG Board Member is Christine Green. The IFPUG Non-functional Sizing Standards Committee Chair is Talmon Ben-Cnaan. Other SNAP team members were Wendy Bloomfield, Steve Chizar, Peter R. Hill, Kathy Lamoureaux, Abinash Sahoo, Joanna Soles, Roopali Thapar, Luc Vangrunderbeeck, Jalaja Venkat, and Charlene Zhao.

ABOUT THE AUTHOR

Charley Tichenor is the newest member of the SNAP team, joining in the Fall of 2011 and serving primarily as the team's Statistician. He has been a member of IFPUG since 1991, and was certified as a Certified Function Point Specialist in 1994 and 1997. He has a Bachelor of Science Degree in Business Administration from the Ohio State University, a Master of Business Administration degree from Virginia Tech, and a Ph.D. in Business from Berne University.

Phone: 703-901-3033
E-mail: charles.tichenor@dsca.mil

REFERENCES

1. Jones, Capers, "Sizing Up Software," Scientific American, a division of Nature America, Inc., December 1998.
2. International Function Point Users Group (IFPUG), Software Non-functional Assessment Process Manual, (now in version 2.1), Princeton Junction, New Jersey, USA 08550, 2012.
3. Jones, Capers, "Software Sizing During Requirements Analysis," Modern Analyst, retrieved November 5, 2012 from <http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/512/Software-Sizing-During-Requirements-Analysis.aspx>, copyright 2008 by Capers Jones & Associates LLC; all rights reserved.
4. International Function Point Users Group (IFPUG), Counting Practices Manual (now in version 4.3), Princeton Junction, New Jersey, USA 08550, 2009.
5. ISO. "ISO/IEC 20926:2009 Software and Systems Engineering -- Software Measurement -- IFPUG Functional Size Measurement Method 2009," retrieved November 5, 2012 from <http://www.iso.org/iso/fr/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51717>.
6. Dekkers, Carol, "Musings About Software Development," retrieved November 5, 2012 from <http://caroldekkers.blogspot.com/>, 2008.
7. Albrecht, Allan, "Measuring Application Development Productivity," IBM, 1977.
8. Walpole, R. E., & Myers, R., Probability and Statistics for Engineers and Scientists Third Edition, New York, New York, Macmillan Publishing Company, a division of Macmillan, Inc., 1985.
9. Davis, Bobby, & Tichenor, Charley, "The Applicability of Benford's Law to the Buying Behavior of Foreign Military Sales Customers," Global Journal of Business Research, The Institute for Business and Finance Research, (volume 2, 2008).
10. Tichenor, Charley, "Why Function Point Counts Comply with Benford's Law," presented at the Fourth International Software Measurement & Analysis conference, Chicago, IL, 2009.