AN EVALUATION OF THREE FUNCTION POINT MODELS FOR ESTIMATION OF SOFTWARE EFFORT

Daniel V. Ferens
and
Robert B. Gurner

Air Force Institute of Technology
Wright-Patterson AFB, Ohio

ABSTRACT

Function point analysis has become well-established as a method for estimating software size and, sometimes, software effort. However, the accuracy of estimation models using function points to estimate size or effort has not been ascertained significantly. This paper summarizes a study performed at the Air Force Institute of Technology (AFIT) in 1991 to assess the expected accuracy of three function point-based models: the Tecelote Software Program Acquisition Network Simulation (SPANS) model, the Checkpoint model, and the Costar model.

INTRODUCTION

This paper is presented in three segments. The first segment of the paper surveys the theory of function points and the history of their usage from Albrecht's research in the early 1980's to the present time. A general comparison of function point models to other software size and effort estimation models and techniques is included, as is a brief description of the three models used in the AFIT study.

The second segment of the paper describes the AFIT study. The three models selected were compared to a historical data base of thir-

ty-six points using three statistical techniques; the Least Squares Best Fit, Wilcoxon "T", and Percentage Error techniques. The results were used to assess estimating reliability of the models, estimation bias present in the models, and a comparison of model estimates to actual amounts of effort at the twenty and thirty percent levels. The models were then compared both on an absolute basis of how well they performed on the statistical tests, and a relative basis of how they compared to each other.

The third segment of the paper briefly examines the future directions for function points as an estimation technique. Current efforts of the International Function Points Working Group are discussed, as well as endeavors to adapt function points to the scientific and real time environments.

HISTORY OF FUNCTION POINTS

Function points originated with the work of Allan Albrecht [1] as a methodology for estimating the size of a program in source lines of code (SLOC). Based on his research on large data processing programs written in the COBOL and PL/1 languages, Albrecht further hypothesized that function points may be an alternative to using SLOC to estimate the cost or effort re-

quired for software development. Based on earlier research by Maurice Halstead, Albrecht's function points are computed from five attributes of a program: the number of external input types (I), the number of external output types (O), the number of logical internal file types (F), the number of external interface types (N), and the number of external inquiry types (Q). Albrecht's original basic equation for function points, according to Behrens [2], was:

$$BFP = 4I + 5O + 10F + 7N + 4Q \quad (1)$$

where BFP is "basic function points", before any complexity adjustments are made.

While the BFP equation may be suitable for preliminary estimates, two complexity adjustments can provide a more refined estimate. First, Albrecht recommends that each item for each attribute be classified as "simple", "average", or "complex", and weighted accordingly. (The BFP equation assumes all items are "average".) For the logical internal files attribute, for example, a simple item would contain 1 logical record format, an average item would contain 2-5 formats, and a complex item would contain 6 or more formats [3]. Table 1 shows the numerical values for the weights assigned to the complexity ratings for each attribute [1]. The revised basic equation using these attribute complexity ratings results in "unadjusted function points" (UFP).

Table 1: Complexity Ratings

Attribute	Rating		
	Simple	Average	Complex
Inputs (I)	3	4	6
Outputs (O)	4	5	7
Data Files (F)	7	10	15
Interfaces (N)	5	7	10
Inquiries (Q)	3	4	6

A second adjustment to the basic equation is a processing complexity adjustment which accounts for the degree of influence of fourteen characteristics of the physical system on which the software resides. The fourteen characteristics, as described by Dreger [3], are as follows:

1. Data Communications
2. Distributed Data/Processing
3. Performance Objectives
4. Heavily-Used Configuration
5. Transaction Rate
6. On-Line Data Entry
7. End-Use Efficiency
8. On-Line Update
9. Complex Processing
10. Reusability
11. Conversion/Installation Ease
12. Operational Ease
13. Multiple Site Use
14. Facilitate Change

Each characteristic is rated, preferably by a system user, from 0 to 5 based on the "degree of influence" for that characteristic, with a rating of 3 is usually "average". For example, for Characteristic 6 above, on-line data entry, a rating of 0-2 indicates that 0-15% of the transactions are interactive data entry; a rating of 3-4 indicates that 15-30% are interactive; and a rating of 5 indicates that 30-50% are interactive [3]. The ratings of all fourteen factors are summed to compute a total degree of influence (TDI), and adjusted function points (FP) are computed using the equation:

$$FP = UFP (0.65 + .01 \times TDI) \quad (2)$$

From equation (2) and the information above, it can be seen that the TDI can alter the UFP by up to 35% in either direction.

Several independent studies have verified that, for data processing applications, function points as computed in equation (2) are indeed

superior to SLOC for estimating cost or effort. A study performed by Kemerer [4] compared two function point cost estimation models with two SLOC-based models by comparing model estimates with actual levels of effort for fifteen programs, most of which were written in COBOL. The two function point models, ESTIMACS and an Albrecht-derived model, were much more accurate than the two SLOC-based models, PRICE-S and SLIM. However, for even the most accurate model, ESTIMACS, the estimates averaged 85% higher than actual levels of effort. Therefore, the expected accuracy of the models is uncertain, although it appears that accuracy may be improved by calibration. Another study by Low and Jeffery [5], performed for COBOL and PL/1 business programs, concluded that function points were a more consistent estimator of software size than was SLOC. The AFIT study, which will now be discussed, also adds credibility to function points as an effort estimator.

THE AFIT FUNCTION POINTS STUDY

In 1991, a thesis study [6] was performed at AFIT to assess the expected reliability and accuracy of three function point-based cost estimating models; SPANS, Checkpoint, and Costar; by comparing model effort estimates with historical results. The study was sponsored in part by the Air Force Standard Systems Center (SSC) at Gunter Air Force Base, Alabama. The details of this study regarding the database used, three models used, the results of statistical tests, and study conclusions will now be discussed.

The Database

The database used for this study, summarized in Table 2, contains 36 business programs, most of which

were written in COBOL, for which function point information and historical effort results were available. The first 22 entries were from a database developed by Albrecht, and the last 14 were from Kemerer's data set [4]. The combined data set is considered sufficiently large for the statistical tests run on the models. In Table 2, function counts are UFP discussed previously, function points are FP computed in Equation (2), and complexity is $(0.65 \times TDI)$ in Equation (2).

The Three Models

The three cost models selected for the study were models which allowed function points as an input and which were readily available to AFIT at the time of the study. Each is now briefly described.

SPANS was developed by Tecelote Research, Inc. primarily for use by the SSC. It incorporates two function point algorithms; the Albrecht equations discussed previously, and an algorithm derived from historical data at the SSC. SPANS then uses algorithms based on Dr. Boehm's [7] Constructive Cost Model (COCOMO) to compute effort. It was developed for use in military information systems, although it can be used in other environments.

Checkpoint is a commercial model developed by Software Productivity Research [8]; it is based on the work of Capers Jones. It uses a variant of the Albrecht equations to compute function points for business programs, and uses feature points, to be discussed later, for real time programs. Portions of the effort estimating equations are company-proprietary.

Costar is a commercial software model marketed by Softstar Systems [9] which uses the COCOMO algorithms for effort and allows

TABLE 2.
The Research Data Set

PROJECT	LANGUAGE	KSLOC	FUNCTION COUNT	FUNCTION POINTS	COMPLEXITY	ACTUAL EFFORT
1	COBOL	130	1750	1750	1.00	673.7
2	COBOL	318	1902	1902	1.00	692.1
3	COBOL	20	522	428	0.82	73.0
4	PL/1	54	660	759	1.15	138.8
5	COBOL	62	479	431	0.90	189.5
6	COBOL	28	377	283	0.75	65.8
7	COBOL	35	256	205	0.80	52.6
8	COBOL	30	263	289	1.10	32.2
9	COBOL	48	716	680	0.95	84.9
10	COBOL	93	690	794	1.15	125.0
11	COBOL	57	465	512	1.10	71.1
12	COBOL	22	299	224	0.75	19.1
13	COBOL	24	491	417	0.85	49.3
14	PL/1	42	802	682	0.85	78.9
15	COBOL	40	220	209	0.95	27.0
16	COBOL	96	488	512	1.05	103.9
17	PL/1	40	551	606	1.10	120.4
18	COBOL	52	364	400	1.10	58.6
19	COBOL	94	1074	1235	1.15	250.7
20	PL/1	110	1310	1572	1.20	402.6
21	DMS	24	694	694	1.00	77.6
22	COBOL	29	263	260	0.99	40.1
23	COBOL	254	1010	1217	1.20	287.0
24	COBOL	214	881	788	0.89	86.9
25	COBOL	254	1603	1611	1.00	258.7
26	COBOL	41	457	507	1.11	95.5
27	COBOL	450	2284	2307	1.01	1107.3
28	COBOL	450	1583	1338	0.85	336.3
29	BLISS	50	411	421	1.02	84.0
30	COBOL	43	97	100	1.03	23.2
31	COBOL	200	998	993	0.99	130.3
32	COBOL	39	250	240	0.96	72.0
33	COBOL	129	724	789	1.00	230.7
34	COBOL	289	1554	1593	1.09	116.0
35	COBOL	161	705	691	0.98	157.0
36	COBOL	165	1375	1348	0.98	246.9

function points as a size input. Unlike the other models, Costar converts function points to SLOC before computing effort required. For COBOL programs, the model assumes 91 SLOC per function point.

Statistical Test Results

Table 3 shows the results of the three statistical tests run on the models. Each test and results are now summarized.

Table 3
Statistical Test Results

<u>Statistical Test</u>	<u>Model</u>		
	<u>SPANS</u>	<u>Checkpoint</u>	<u>COSTAR</u>
Least Squares Best Fit			
R^2	.81	.79	.80
F_{Calc}	145.26	129.19	138.93
Wilcoxon T_{Stat}	-2.44	-1.78	-4.52
Percentages:			
Raw Error: Mean	.51	.27	1.02
Raw Error: Standard Deviation	.63	.65	.87
MRE Error: Mean	.62	.46	1.05
MRE Error: Standard Deviation	.53	.53	.84
Prescribed: +/- 30%	.39	.47	.19
Prescribed: +/- 20%	.25	.36	.14
Adjusted Raw Error: Mean	.00	.09	.00
Adjusted MRE: Mean	.34	.32	.34
Adjusted: +/- 30%	.50	.56	.50
Adjusted: +/- 20%	.33	.39	.31

a. Least Squares Best Fit (LSBF): The well-known Statistical Analysis System (SAS) was run to compute two LSBF-related values for all three models, the coefficient of determination (R^2) and the F-ratio (F_{Calc}). The coefficient of determination is a measure of the amount of variation in actuals explained by the estimates, or the reliability of the estimates. For this study, an R^2 value of .80 was considered "reliable". The F-Ratio measures the level of significance between the estimates and actuals. If F_{Calc} is above a critical value (7.48, in this case), significance is established. For the LSBF analyses, the log-log equation was used:

$$\log(Y) = \log(A) + b_1 * \log(X) \quad (3)$$

where X are model estimates, Y are actual values, and A and b_1 are constants.

As shown in Table 3, all values easily passed the F-ratio test, and all except Checkpoint have an R^2 of .80 or better. However, since Checkpoint's value is so close to .80, and the three models show no significant differences, they all are of about equal reliability.

b. Wilcoxon "T" Statistic: The "T" statistic (T_{Stat}) is a measure of bias in estimating; a large T_{Stat} indicates a non-zero mean of errors, and the sign indicates the direction of bias. The computed T_{Stat} is compared to a reference value. In this case, the reference value for 36 estimates are 2.41 at the 99% confidence level and 1.70 at the 95% confidence level.

Table 3 shows that all models' T_{Stat} values exceed the 95% confidence value, and only Checkpoint does not exceed the 99% confidence level. This indicates that the models gen-

erally are biased, and the negative signs indicate that the models tend to overestimate effort. An implication here is that models may benefit from calibration to offset the bias present.

c. Percentage Error Tests: Percentage error tests show the fractional or percentage differences between the estimated and actual results. For this study, three percentage tests were run: raw error, magnitude of relative error (MRE), and predescribed range.

The raw error test sums the percentage errors for individual data points, then computes a mean and standard deviation. The individual errors are computed by the formula:

$$\text{Error} = (X - Y) / Y \quad (4)$$

where X and Y are as defined in Equation (3). Table 3 shows the means and standard deviations of the raw errors for the three models, expressed as a decimal percentage. All means were positive; a finding consistent with the Wilcoxon test that all models tend to overestimate. Checkpoint had the lowest overall error, although the relatively high standard deviations for all three models indicate inconsistency.

The MRE test sums the absolute values for the errors computed in equation (4) above, and avoids the potential effects of positive and negative values canceling each other out. Therefore, it may be a more valid test of model accuracy. Table 3 shows that the differences among models were not as great for this test; however, the model accuracies were even worse.

The predescribed percentage test shows the fraction of estimates within a predescribed range of actual values; 20% and 30% were ranges selected for this study. Check-

point showed the most accuracy; however, no model was accurate even to within 30% half of the time.

A final test was accomplished by adjusting the model's estimates to eliminate bias. This adjustment was done by dividing each estimated value by a factor of 1 plus the mean of the raw error. The three percentage tests were run for the new actuals. Table 3 shows that the raw and MRE errors decreased significantly, and that the models are more accurate. Furthermore, the three models are nearly equal in predictive capability. However, the accuracy is still not overwhelming; the models only have half a chance of estimating within 30% of the actuals, even using the same database for which they were adjusted.

Study Results and Conclusions

The study shows that the three function point models can be useful in effort estimation, especially if they can be adjusted for bias. Based on the percentage tests, the unadjusted Checkpoint model was more accurate than the other two models. However, when models were adjusted for bias, the results were very similar. This is consistent with the LSBF test results, where the three models' R^2 values were nearly identical. Calibration of models, or adjusting models to a particular environment, appears to be a worthwhile endeavor if greater accuracy is sought.

Still, the models' performances are not scintillating; the adjusted models were accurate to within 30% only about half of the time. This finding is diminished in that the same database was used for both adjustment and comparison; a practice not advocated in actual model calibration efforts. It may be concluded from this study that function points can indeed be use-

ful for effort estimation, but spectacular accuracy can not be expected.

FUTURE DIRECTIONS

Function points are currently in good repute, especially for business and data processing applications. The International Function Points User's Group (IFPUG) has been formed to continually improve function points theory and practice. The IFPUG is currently studying and revising some of Albrecht's equations.

Function points do have disadvantages, however. One problem is that Albrecht's five attributes are sometimes hard to define and count. Symons [10] has proposed the concept of Mark II function points with only three attributes: inputs, outputs, and entities. For the Mark II concept, external interfaces and inquiries are treated as inputs or outputs, and internal files are "replaced" by a measure of entity types referenced by transactions. This concept has the advantage of requiring fewer attributes to be counted, but has not yet received extensive analysis.

Another disadvantage of function points, according to Symons [10] and others, is that they are not readily adaptable to the real-time or scientific environments. Several attempts have been made to adapt the concept to these environments. Some of these will now be briefly discussed.

Gaffney and Werling [11] have researched the potential use of external function points in estimating SLOC for military systems. Using a database of nineteen aerospace programs, the authors established relatively high correlations (R^2 values of .89 and .87 respectively) for equations relating the

counts of four external attributes (the five Albrecht attributes minus internal files) and three external attributes (four attributes minus external interfaces) to SLOC. The researchers concluded that SLOC may indeed be estimated from function points for real-time programs. This may enhance the cost estimation capability of primarily SLOC-based models such as COCOMO.

Capers Jones, the developer of the Checkpoint model, has proposed feature points as an adaption of function points to the real-time environment [12]. He uses Albrecht's five attributes, assigning a different weight to internal files, and adds algorithms (A) as a sixth attribute. Algorithms are bounded computational problems which are included within a specific computer program. The basic feature points (BFEA) equation is:

$$BFEA = 3A + 4I + 5O + 7F + 7N + 4Q \quad (5)$$

This equation is similar to Equation (1) except that algorithms are added and the coefficient for F is reduced from 10 to 7.

Reifer has proposed a different method for real-time applications in the ASSET-R sizing model marketed by his company, Reifer Consultants, Inc. [13]. In addition to Albrecht's five attributes, Reifer adds three more attributes: operating modes, stimulus/response relationships, and rendezvous. (For scientific programs, only operating modes are added.) Operating modes are time-dependent end-to-end processing flows to which software performance can be related; stimulus/response relationships measure the amount of sequencing and control in real-time systems; and rendezvous are a measure of the degree of concurrency in real time systems [13]. Unlike Equations (1) and (5), all attributes are initially

weighted with a value of "1", and other factors are used to adjust function point totals.

At the current time, little independent research has been done on any of the real-time variations of function points; therefore, it is difficult to ascertain whether function points are indeed useful outside of the data processing environment. The amount of attention devoted to function points and variations seems to remain high, however, and function points promise to be a fascinating area of study for future software sizing and effort estimating efforts.

REFERENCES

1. Albrecht, Allan J., and John E. Gaffney, Jr., "Software Function; Source Lines of Code, and Development Effort Prediction", IEEE Transactions On Software Engineering, Volume SE-9, Number 6, November, 1983, pp. 639-648.
2. Behrens, Charles A., "Measuring the Productivity of Computer Systems Development Activities With Function Points", IEEE Transactions On Software Engineering, Volume SE-9, Number 6, November, 1993, pp. 648-652.
3. Dreger, J. Brian, Function Point Analysis, Englewood Cliffs, NJ, Prentice-Hall: 1989.
4. Kemerer, Chris F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, Volume 30, Number 5, May, 1987, pp. 416-429.
5. Low, Graham C., and D. Ross Jeffery, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Transactions on Software Engineering, Volume 16, Number 1, January, 1990, pp. 64-71.
6. Gurner, Robert B., A Comparative Study of the Reliability of Function Point Analysis in Software Development Effort Estimation Models (AFIT Thesis GCA/LSY/91S-2), Dayton, OH, Air Force Institute of Technology: 1991.
7. Boehm, Barry W., Software Engineering Economics, Englewood Cliffs, NJ, Prentice-Hall: 1981.
8. Checkpoint Version 2.0 User's Guide, Burlington, MA, Software Productivity Research: 1991.
9. Costar Version 3.00 User's Manual, Amherst, NH, Softstar Systems: 1989.
10. Symons, Charles R., "Function Point Analysis: Difficulties and Improvements", IEEE Transactions On Software Engineering, Volume 14, Number 1, January, 1988, pp. 2-11.
11. Gaffney, John E., Jr., and Richard Werling, "Estimating Software Size From Counts of Externals: A Generalization of Function Points", International Society of Parametric Analysts 1991 Conference Proceedings, New Orleans, LA, 1991, pp. SW62-SW72.
12. Jones, Capers, Applied Software Measurement, New York, McGraw-Hill: 1991.
13. Reifer, Donald J., ASSET-R Version 2.0 Manual, Torrance, CA, Reifer Consultants, Inc.: 1989.