

BARBARA  
KITCHENHAM  
Keele University

# counterpoint

## The Problem with Function Points

**T**he concept of function points goes a long way toward addressing the genuine need for front-end measures of product size. My concern is that specific types of function points—such as Albrecht’s version,<sup>1</sup> which is popular in the US, and Symons Mark II version,<sup>2</sup> which is popular in the UK—are not the straightforward, simple measures some people imagine.

Function points have fundamental flaws in their construction that prevent them from being valid measures.<sup>3</sup> This means there is a danger they will behave in unexpected ways, for example asserting that one program is larger than another when it is not. In most cases, if function points are used with caution within a specific organization, you will probably have few problems. But if you want to use them for cross-company benchmarking, as the basis for development or support contracts between different companies, or to develop generic estimation models, there is a non-negligible risk that you will encounter problems.

**Construction Problems.** Albrecht function point construction involves classifying inputs, outputs, logical master files, system interfaces, and queries as simple, average, and complex. This means that the absolute scale counts are reduced to ordinal scale measures, and you can no longer apply other transformations to make them interval or ratio scale. Formally ordinal scale measures cannot be added together, because it is nonsensical to add the label “simple” to the label “complex” even if you use the label 3 as a synonym for “simple” and the label 5 as a synonym for “complex.” In addition, you cannot multiply or divide ordinal *s*, so you cannot convert them to productivity measures. Nor can you define the difference between values: you cannot possibly say the value 4 is twice the amount of the value 2. Measuring size on an ordinal scale can tell you that one system is bigger than another, but not by how much.

Symons Mark II function point construction involves adding weighted counts. There is no problem with the individual counts, but the additive model causes problems. It assumes that the weights turn counts of input data elements, output data elements, and accesses into equivalent measures

in the same way that multiplying miles by 1.069 converts miles to kilometers. Unfortunately, there are no standard conversion factors to equate inputs, outputs, and entity accesses. Using industry-average weights does not really solve the problem; instead, it raises questions as to the inherent variability of the averages, how representative the systems contributing to the average are, and how stable the averages are over time. In addition, it is unclear that such weights are appropriate for new systems as well as system enhancements. Is it as difficult to create a data model as it is to enhance a data model? Does enhancement difficulty depend on the type of technology you are using?

**Unstable Predictive Models.** The additive model is a problem if you try to correlate its function point elements *and* then use the final measure predictively. In such cases the model may be unstable because the same underlying phenomenon is captured more than once. Ross Jeffery<sup>4</sup> and I<sup>5</sup> both found correlations between Albrecht function point elements, but we found *different* correlations. This shows that any predictive model based on the sum of the elements will not be stable for different datasets.

A further problem with function points is the technology adjustment factor. This is based on a subjective assessment of 14 project factors on a six-point ordinal scale. The ordinal measures are added together and used to weight the raw function point count. Not only are the measures treated incorrectly, but there is no evidence that they improve any predictive models involving function points. In various empirical studies, effort predictions based on raw function points have been just as accurate as predictions based on adjusted function points.<sup>6,7</sup>

Finally, function point counting involves judgment on the part of the counter. Chris Kemerer reported a 12-percent difference for the same product by people in the same organization.<sup>8</sup> Graham Low and Ross Jeffery reported worse figures: a 30-percent variance *within* an organization, which rose to more than 30 percent *across* organizations.<sup>9</sup> Thus, even if the construction of function points leads to valid

*Continued on page 31*

# counterpoint

measures, you could still make significant mistakes comparing the size of different products.

**Improving Function Points.** So what should we do? First, we should not ignore the problem and spend the next 10 years refining function point counting rules like medieval monks arguing about the number of angels on the head of a pin. Nor should we reject the concept of function points and return to the equally problematic lines-of-code measure. The need for early-phase size measures will not go away. Basing such measures on the flow of data across the human-computer system boundary and the amount of data that needs to be held long-term within the computer system is eminently reasonable for information systems. What we should do is understand the limitation of our measures and how to use them safely.

We should also consider simplifying the concept of function points such that basic counts can be collected automatically from early system representations. Furthermore, we should not add the resulting counts together. We should use the basic (unweighted) counts as a vector of measures that describe the system. After all, we don't mind using three measures—chest size, waist size, and hip size—to identify a person's clothing size.

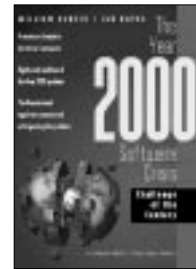
If we want to predict effort using early-phase measures, we should develop such models on a local basis, that is, using our own effort data and our own system size counts. We can then use stepwise multivariate analysis to develop predictive models that include only the counts that contribute significantly to the accuracy of the predictions. We should not include new development projects, enhancement projects, and small-product corrections in the same model unless our dataset indicates the predictive model is the same for all project types. ◆

## REFERENCES

1. A.J. Albrecht and J.R. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: a Software Science Validation," *IEEE Trans. Software Eng.*, Vol. 9, No. 6, 1983, pp. 639-648.
2. C.R. Symons, *Software Sizing and Estimating, Mk II FPA*, John Wiley, Chichester, U.K., 1991.
3. B.A. Kitchenham, S.L. Pflieger, and N.E. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Trans. Software Eng.*, Vol. 21, No. 12, Dec. 1995, pp. 929-944.
4. B.A. Kitchenham and K. Känsälä, "Inter-Item Correlations Among Function Points," *1st Int'l Software Metrics Symp.*, Baltimore, IEEE Computer Society Press, Los Alamitos, Calif., 1993, pp. 11-14.
5. D.R. Jeffery and J. Stathis, "Specification-Based Software Sizing: an Empirical Investigation of Function Metrics," *Proc. NASA Goddard Software Eng. Workshop*, Greenbelt, Md., 1993.
6. C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM*, Vol. 30, No. 5, 1987, pp 416-429.
7. G.C. Low and D.R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process," *IEEE Trans. Software Eng.*, Vol. 16, No. 1, 1990, pp. 64-71.
8. C. Kemerer, "Reliability of Function Points Measurement: a Field Experiment," *Comm. ACM*, Vol. 36, No. 2, 1993, pp. 85-97.
9. J.R. Jeffery, G.C. Low, and M.A. Barnes, "Comparison of Function Point Counting Techniques," *IEEE Trans. Software Eng.*, Vol. 19, No. 5, 1993, pp. 529-532.

Barbara Kitchenham is a principal researcher in software engineering at Keele University and a member of *IEEE Software's* editorial board. She can be reached at [barbara@cs.keele.ac.uk](mailto:barbara@cs.keele.ac.uk).

**IEEE COMPUTER**



Now In Stock!

## The Year 2000 Software Crisis Challenge of the Century

by William Ulrich and Ian Hayes

If you are an IS manager or professional, you must understand the coming Year 2000 software crisis now — while there's still time to turn it into an opportunity. In this book, two leading consultants provide a realistic overview of the Year 2000 challenge, and a comprehensive guide to every aspect of solving the problem. Understand the critical issues of asset management, strategic planning, budgeting, and support technologies.

340 pages. 1997. ISBN 0-13-655664-7.

Catalog # RS00134

\$37.95 Members / \$39.95 List



## Handbook of Software Reliability Engineering

by Michael R. Lyu

Features contributions from the world's leading reliability experts, this book/CD-ROM package offers you the most comprehensive and up-to-date resource on software reliability engineering available today. The Handbook takes you step by step through software reliability measurement and prediction the attributes and metrics of product design, development process, software operational environment, and their effects on reliability — and the application of this information in software development, acquisition, use, and maintenance.

875 pages. 1996. ISBN 0-07-039400-8.

Catalog # RS00030

\$59.00 Members / \$72.50 List

Order from the Online Catalog

<http://computer.org/cspres>

Call toll-free +1-800-CS-BOOKS