the points we raise, and it may well be that the information we seek has become a casualty of the pressure on space in academic journals. Nevertheless, it is vitally important to precisely specify the mapping from a language-independent set of metrics to specific programming languages and sets of observations. Such precision is particularly important since the source code used in published work is generally not publicly available. The usefulness of the proposed metrics (and others) will be limited until their application to specific languages is clearly specified. Failure to resolve such issues in the near future may impede the development and validation of effective OO software metrics. We believe that this is as important as the establishment of a sound theoretical basis for the metrics.

REFERENCES

[1] E. V. Berard, "Object coupling & object cohesion," in *Essays on Object-Oriented Software Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[2] S. R. Chidamber and C. F. Kemerer, "Toward a metric suite for object oriented design," in *Proc. OOPLSA '91*, ACM, 1991.

[3] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Trans. Software Eng.*, vol. 20, pp. 476–493, 1994.

[4] M. H. Halstead, *Elements of Software Science*. Amsterdam, The Netherlands: Elsevier North-Holland, 1977.

[5] J.-L. Lassez *et al.*, "A critical examination of software science," *J. Syst. Software*, vol. 2, no. 2, pp. 105–112, 1981.

[6] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *J. Syst. Software*, vol. 23, pp. 111–122, 1993.

[7] V. Y. Shen, S. D. Conte, and H. E. Dunsmore, "Software science revisited: A critical analysis of the theory and its empirical support," *IEEE Trans. Software Eng.*, vol. 9, pp. 155–165, 1983.

Authors' Reply[2]

Shyam Chidamber and Chris F. Kemerer

The main thesis of the Churcher and Shepperd comment is that it is important that the software metrics be clearly defined in order that other researchers can replicate the results, a point that we, of course, completely agree with. Therefore, we view this response to their comment not as a rebuttal, but merely as an opportunity to provide additional detail and insight to our thinking about how we developed the metrics for the readership of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING who may have interest in object-oriented metrics.

Churcher and Shepperd focus on the calculation of the number of methods per class, and suggest approximately half a dozen binary questions about what methods should either be included or not in the count. They then note that a potentially large number of different possible answers can be generated by combining all the possible combinations of answers to these questions

In our view, the entire list of questions can be answered by reference to a simple principle that methods which required additional design effort and are defined in the class should be counted, and those that do not should not. Therefore, we would count all the distinct methods and operators in the class even when some share a name identifier or even when they are not interface methods, and we would count these prior to preprocessing, consistent with the role of WMC as a design metric. We would not count indirect methods available through ancestors, friends, (a C++ construct), or any inherited methods, as these are defined outside the class. (See the portion of our definition of WMC on p. 482 of the original article, as follows: "...Consider a Class C_1 with methods $M_1 \cdots, M_N$ that are defined in the class. Let...") We have provided other metrics in the suite which directly relate to notions of complexity arising from inheritance.

We find this principle to be relatively straightforward in its application and consistent with designers' intuitions about the complexity of a class. Additional evidence for this may be found in the fact that our value for the Churcher and Shepperd example is 37 methods, which is the same value that they determine before they begin discussing some hypothetical options.

We are encouraged that Churcher and Shepperd are exploring the use of some of our metrics in their own research, and greatly appreciate their effort in considering alternative formulations for object-oriented metrics. In fact, we would like to conclude by emphasizing a point made in our original article: while we propose a particular suite of six metrics, there is no reason to believe that these six will ultimately be found to be comprehensive. Further work by ourselves or others may result in additions, changes, or even possible deletions from this suite, particularly as at the current time the suite has been subject to only limited empirical observation. We welcome questions and comments from the software engineering community that will enable the further explication and refinement of our metrics suite.

[2]Manuscript received July 1994. Recommended by S. H. Zweben.

The authors are with the Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02142 USA.

IEEE Log Number 9409035.

Correction to "A Practical Approach to Programming with Assertions"

D. S. Rosenblum

In the above paper,[1] a printing error resulted in the incorrect publishing of line 7 of Fig. 3 on p. 22.

The incorrect line read:

```
&& all (int i=0; i < in size-1; i=i+1) S[i] <= S[i+1] // S is ordered.
```

The correct line should read:

```
&& all (int i=0; i < in size-1; i=i+1) S[i] < S[i+1] // S is ordered.
```

Manuscript received February 6, 1995.

The author is with AT&T Bell Laboratories, Murray Hill, NJ 07974 USA (e-mail: dsr@research.att.com).

IEEE Log Number 9410122.

[1]D. S. Rosenblum, *IEEE Trans. Software Eng.*, vol. 21, pp. 19–31, Jan. 1995.

0098–5589/95$04.00 © 1995 IEEE