[CHI98] Chidamber, S. R., D. P. Darcy, and C. F. Kemerer, "Management Use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Trans. Software Engineering,* vol. SE-24, no. 8, August 1998, pp. 629–639.

[CHU95] Churcher, N. I., and M. J. Shepperd, "Towards a Conceptual Framework for Object-Oriented Metrics," *ACM Software Engineering Notes,* vol. 20, no. 2, April 1995, pp. 69–76.

[CUR80] Curtis, W., "Management and Experimentation in Software Engineering," *Proc. IEEE,* vol. 68, no. 9, September 1980.

[DAV93] Davis, A., et al., "Identifying and Measuring Quality in a Software Requirements Specification," *Proc. First Intl. Software Metrics Symposium,* IEEE, Baltimore, MD, May 1993, pp. 141–152.

[DEM81] DeMillo, R. A., and R. J. Lipton, "Software Project Forecasting," in *Software Metrics* (A. J. Perlis, F. G. Sayward, and M. Shaw, eds.), MIT Press, 1981, pp. 77–89.

[DEM82] DeMarco, T., Controlling Software Projects, Yourdon Press, 1982.

[DHA95] Dhama, H., "Quantitative Models of Cohesion and Coupling in Software," *Journal of Systems and Software,* vol. 29, no. 4, April 1995.

[EJI91] Ejiogu, L., *Software Engineering with Formal Metrics,* QED Publishing, 1991.

[FEL89] Felican, L., and G. Zalateu, "Validating Halstead's Theory for Pascal Programs," *IEEE Trans. Software Engineering,* vol. SE-15, no. 2, December 1989, pp. 1630–1632.

[FEN91] Fenton, N., *Software Metrics,* Chapman and Hall, 1991.

[FEN94] Fenton, N., "Software Measurement: A Necessary Scientific Basis," *IEEE Trans. Software Engineering,* vol. SE-20, no. 3, March 1994, pp. 199–206.

[GRA87] Grady, R. B., and D. L. Caswell, *Software Metrics: Establishing a Company-Wide Program,* Prentice-Hall, 1987.

[HAL77] Halstead, M., *Elements of Software Science,* North-Holland, 1977.

[HAR98] Harrison, R., S. J. Counsell, and R. V. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEEE Trans. Software Engineering,* vol. SE-24, no. 6, June 1998, pp. 491–496.

[HET93] Hetzel, B., *Making Software Measurement Work,* QED Publishing, 1993.

[IEE93] *IEEE Standard Glossary of Software Engineering Terminology,* IEEE, 1993.

[IEE94] *Software Engineering Standards,* 1994 edition, IEEE, 1994.

[IFP01] *Function Point Counting Practices Manual,* Release 4.1.1, International Function Point Users Group, 2001, available from http://www.ifpug.org/publications/ manual.htm.

[IFP03] Function Point Bibliography/Reference Library, International Function Point Users Group, 2003, available from http://www.ifpug.org/about/bibliography. htm

[KOK95] Kokol, P., I. Rozman, and V. Venuti, "User Interface Metrics," *ACM SIGPLAN Notices,* vol. 30, no. 4, April 1995, can be downloaded from: http://portal.acm.org/.

[KYB84] Kyburg, H. E., *Theory and Measurement,* Cambridge University Press, 1984.

[LET03] Lethbridge, T., private communication of software metrics, June, 2003.

[LON02] Longstreet, D., "Fundamental of Function Point Analysis," Longstreet Consulting, Inc, 2002, available at http://www.ifpug.com/fpafund.htm.

[LOR94] Lorenz, M., and J. Kidd, *Object-Oriented Software Metrics,* Prentice-Hall, 1994.

[MCC76] McCabe, T. J., "A Software Complexity Measure," *IEEE Trans. Software Engineering,* vol. SE-2, December 1976, pp. 308–320.

[MCC77] McCall, J., P. Richards, and G. Walters, "Factors in Software Quality," three volumes, NTIS AD-A049-014, 015, 055, November 1977.

[MCC89] McCabe, T. J., and C. W. Butler, "Design Complexity Measurement and Testing," *CACM,* vol. 32, no. 12, December 1989, pp. 1415–1425.

[MCC94] McCabe, T. J., and A. H. Watson, "Software Complexity," *Crosstalk,* vol. 7, no. 12, December 1994, pp. 5–9.

[NIE94] Nielsen, J., and J. Levy, "Measuring Usability: Preference vs. Performance," *CACM,* vol. 37, no. 4, April 1994, pp. 65–75.

[ROC94] Roche, J. M., "Software Metrics and Measurement Principles," *Software Engineering Notes,* ACM, vol. 19, no. 1, January 1994, pp. 76–85.

[SEA93] Sears, A., "Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout, *IEEE Trans. Software Engineering,* vol. SE-19, no. 7, July 1993, pp. 707–719.

[SHE98] Sheppard, M., *Goal, Question, Metric,* 1998, available at http://dec.bournemouth.ac.uk/ESERG/mshepperd/SEMGQM.html.

[SOL99] van Solingen, R., and E. Berghout, *The Goal/Question/Metric Method,* McGraw-Hill, 1999.
[UEM99] Uemura, T., S. Kusumoto, and K. Inoue, "A Function Point Measurement Tool for UML Design Specifications," *Proc. of Sixth International Symposium on Software Metrics,* IEEE, November 1999, pp. 62–69.
[USA87] *Management Quality Insight,* AFCSP 800-14 (U.S. Air Force), January 20, 1987.
[WHI97] Whitmire, S., *Object-Oriented Design Measurement,* Wiley, 1997.
[WIL93] Wilde, N., and R. Huitt, "Maintaining Object-Oriented Software," *IEEE Software,* January 1993, pp. 75–80.
[ZUS90] Zuse, H., *Software Complexity: Measures and Methods,* DeGruyter, 1990.
[ZUS97] Zuse, H., *A Framework of Software Measurement,* DeGruyter, 1997.

## Problems and Points to Ponder

**15.1.** Develop a software tool that will compute cyclomatic complexity for a programming language module. You may choose the language.

**15.2.** McCall's quality factors were developed during the 1970s. Almost every aspect of computing has changed dramatically since the time that they were developed, and yet, McCall's factors continue to apply to modern software. Can you draw any conclusions based on this fact?

**15.3.** Try to come up with a measure or metric from everyday life that violates the attributes of effective software metrics defined in Section 15.2.5.

**15.4.** A class, **X**, has 12 operations. Cyclomatic complexity has been computed for all operations in the OO system, and the average value of module complexity is 4. For class **X**, the complexity for operations 1 to 12 is 5, 4, 3, 3, 6, 8, 2, 2, 5, 5, 4, 4, respectively. Compute the weighted methods per class.

**15.5.** A system has 12 external inputs, 24 external outputs, fields 30 different external queries, manages 4 internal logical files, and interfaces with 6 different legacy systems (6 EIFs). All of these data are of average complexity, and the overall system is relatively simple. Compute FP for the system.

**15.6.** Measurement theory is an advanced topic that has a strong bearing on software metrics. Using [ZUS97], [FEN91], [ZUS90], [KYB84] or some other source, write a brief paper that outlines the main tenets of measurement theory. Individual project: Develop a presentation on the subject and present it to your class.

**15.7.** Why is it that a single, all-encompassing metric cannot be developed for program complexity or program quality?

**15.8.** A major information system has 1140 modules. There are 96 modules that perform control and coordination functions and 490 modules whose function depends on prior processing. The system processes approximately 220 data objects that each have an average of three attributes. There are 140 unique data base items and 90 different database segments. Finally, 600 modules have single entry and exit points. Compute the DSQI for this system.

**15.9.** A legacy system has 940 modules. The latest release required that 90 of these modules be changed. In addition, 40 new modules were added and 12 old modules were removed. Compute the software maturity index for the system.

**15.10.** Develop a small software tool that will perform a Halstead analysis on programming language source code of your choosing.

**15.11.** Software for System *X* has 24 individual functional requirements and 14 nonfunctional requirements. What is the specificity of the requirements? The completeness?

## Further Readings and Information Sources

There is a surprisingly large number of books that are dedicated to software metrics, although the majority focus on process and project metrics to the exclusion of product metrics. Kan (*Metrics and Models in Software Quality Engineering,* Addison-Wesley, second edition, 2002), Fenton and Pfleeger (*Software Metrics: A Rigourous and Practical Approach,* Brooks-Cole Publishing, 1998), and Zuse [ZUS97] have written thorough treatments of product metrics.

Books by Card and Glass [CAR90], Zuse [ZUS90], Fenton [FEN91], Ejiogu [EJI91], Moeller and Paulish (*Software Metrics,* Chapman and Hall, 1993), and Hetzel [HET93] all address product metrics in some detail. Oman and Pfleeger (*Applying Software Metrics,* IEEE Computer Society Press, 1997) have edited an anthology of important papers on software metrics. In addition, the following books are worth examining:

Conte, S. D., H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models,* Benjamin-Cummings, 1984.

Grady, R. B., *Practical Software Metrics for Project Management and Process Improvement,* Prentice-Hall, 1992.

Sheppard, M., *Software Engineering Metrics,* McGraw-Hill, 1992.

The theory of software measurement is presented by Denvir, Herman, and Whitty in an edited collection of papers (*Proceedings of the International BCS-FACS Workshop: Formal Aspects of Measurement,* Springer-Verlag, 1992). Shepperd (*Foundations of Software Measurement,* Prentice-Hall, 1996) also addresses measurement theory in some detail. Current research is presented in the *Proceedings of the Symposium on Software Metrics* (IEEE, published annually).

A comprehensive summary of dozens of useful software metrics is presented in [IEE94]. In general, a discussion of each metric has been distilled to the essential "primitives" (measures) required to compute the metric and the appropriate relationships to effect the computation. An appendix provides discussion and many references.

Whitmire [WHI97] presents the most comprehensive and mathematically sophisticated treatment of OO metrics published to date. Lorenz and Kidd [LOR94] and Hendersen-Sellers (*Object-Oriented Metrics: Measures of Complexity,* Prentice-Hall, 1996) offer the only other books dedicated to OO metrics. Hutcheson (*Software Testing Fundamentals: Methods and Metrics,* Wiley, 2003) presents useful guidance in the application and use of metrics for software testing.

A wide variety of information sources on software metrics are available on the Internet. An up-to-date list of World Wide Web references that are relevant to software metrics can be found at the SEPA Web site:
**http://www.mhhe.com/pressman.**

# Three

# APPLYING WEB ENGINEERING

In this part of *Software Engineering: A Practitioner's Approach* you'll learn about the principles, concepts, and methods that are used to create high-quality Web applications. These questions are addressed in the chapters that follow:

- Are Web applications (WebApps) different from other types of software?
- What is Web engineering, and what elements of software engineering practice can it adopt?
- What are the elements of a Web engineering process?
- How does one formulate and plan a Web engineering project?
- How are requirements for WebApps analyzed and modeled?
- What concepts and principles guide a practitioner in the design of WebApps?
- How does one conduct architecture, interface, and navigation design for WebApps?
- What construction techniques can be applied to implement the design model?
- What testing concepts, principles, and methods are applicable to Web engineering?

Once these questions are answered you'll be better prepared to engineer high-quality Web applications.

CHAPTER

# 16 WEB ENGINEERING

**KEY CONCEPTS**

- basic questions
- best practices
- process framework
- quality criteria
- WebApps
  - attributes
  - categories
- Web engineering
  - methods
  - process
  - tools

The World Wide Web and the Internet that empowers it are arguably the most important developments in the history of computing. These technologies have drawn us all (with billions more who will eventually follow) into the information age. They have become integral to daily life in the first decade of the twenty-first century.

For those of us who can remember a world without the Web, the chaotic growth of the technology harkens back to another era—the early days of software. It was a time of little discipline, but enormous enthusiasm and creativity. It was a time when programmers often hacked together systems—some good, some bad. The prevailing attitude seemed to be "Get it done fast, and get it into the field; we'll clean it up (and better understand what we really need to build) as we go." Sound familiar?

In a virtual round table published in *IEEE Software* [PRE98], I staked out my position with regard to Web engineering:

> It seems to me that just about any important product or system is worth engineering. Before you start building it, you'd better understand the problem, design a workable solution, implement it in a solid way, and test it thoroughly. You should probably also control changes to it as you work and have some mechanism for ensuring the end result's quality. Many Web developers don't argue with this; they just think their world is really different and that conventional software engineering approaches simply don't apply.

**QUICK LOOK**

**What is it?** Web-based systems and applications (WebApps) deliver a complex array of content and functionality to a broad population of end-users. Web engineering (WebE) is the process that is used to create high-quality WebApps. WebE is not a perfect clone of software engineering, but it borrows many of software engineering's fundamental concepts and principles. In addition, the WebE process emphasizes similar technical and management activities. There are subtle differences in the way these activities are conducted, but the overriding philosophy dictates a disciplined approach to the development of a computer-based system.

**Who does it?** Web engineers and nontechnical content developers create the WebApp.

**Why is it important?** As WebApps become increasingly integrated in business strategies for small and large companies (e.g., e-commerce), the need to build reliable, usable, and adaptable systems grows in importance. That's why a disciplined approach to WebApp development is necessary.

**What are the steps?** Like any engineering discipline, WebE applies a generic approach that is tempered with specialized strategies, tactics, and methods. The WebE process begins with a formulation of the problem to be solved by the WebApp. The WebE project is planned, and the requirements and design of the WebApp are