10. How are table-based designs used to decrease code's complexity and increase both readability and maintainability?
11. In decision tables, what are missing policies and redundant actions? How does identifying these help make more efficient tables?
12. What is programming design language, and what is the main benefit that it provides over other construction design methods?
13. How are construction styles used to decrease code's complexity and increase both readability and maintainability?
14. What are the main methods used for evaluating quality of construction designs?
15. List and explain an efficient method for enforcing styles during construction.
16. What is cyclomatic complexity? Name three different ways to compute the cyclomatic complexity of a function during construction design.
17. What is the relationship between cyclomatic complexity and software quality? Explain.

CHAPTER EXERCISES

1. Select a project of choice, and identify three different functions that can be designed using flow-based, table-based, and state-based design. Create the designs for these functions, and prepare a 5- to 10-minute presentation including designs, code, and justification.
2. Compute the cyclomatic complexity of the following code using all three methods:

```
switch( state ) {

  case ONE:

    // Perform activity.
    if( /*some condition*/ ) {
      // Perform activity.
    }
    else if( /*some condition*/ ) {
      // Perform activity.
    }
    else {
      // Perform activity.
    }
    break;

  case TWO:

    // Perform activity FIVE.
    break;
```

```
        case THREE:
          // Perform activity FOUR.
          break;
        default:
          // Intentionally left blank.
          break;
    }
```

REFERENCES

Abran, Alain, James W. Moore, Pierre Bourque, and Robert Dupuis. *Guide to the Software Engineering Body of Knowledge—2004 Version—SWEBOK.* Los Angeles, CA: IEEE Computer Society Press, 2005.

Baldwin, Kenneth, Andrew Gray, and Trevor Misfeldt. *The Elements of C# Style.* Cambridge, UK: Cambridge University Press, 2006.

Booch, Grady, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*, 2d ed. Boston: Addison-Wesley, 2005.

Checkstyle 5.3. October 19, 2010. Available from: http://checkstyle.sourceforge.net/index.html (accessed March 11, 2011).

Collar, Emilio Jr. "An Investigation of Programming Code Textbase Readability Based on a Cognitive Readability Model." PhD thesis, University of Colorado at Boulder, 2005.

Fox, Christopher. *Introduction to Software Engineering Design: Processes, Principles, and Patterns with UML2.* Boston: Addison Wesley, 2006.

Galin, Daniel. *Software Quality Assurance: From Theory to Implementation.* Harlow, UK: Pearson Addison Wesley, 2003.

Hurley, Richard B. *Decision Tables in Software Engineering.* New York: Van Nostrand Reinhold, 1982.

IEEE. "IEEE Standard for Information Technology-Systems Design-Software Design Descriptions." 2009, p. 175.

Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*, 3d ed. New York: McGraw-Hill Osborne Media, 2008.

McCabe, Thomas J. "A Complexity Measure." *IEEE Transactions on Software Engineering* SE-2, no. 4 (1976): 308–320.

McConnell, Steve. *Code Complete*, 2d ed. Redmond, WA: Microsoft Press, 2004.

Meyer, Bertrand. *Object-Oriented Software Construction*, 2d ed. Upper Saddle River, NJ: Prentice Hall, 1997.

Mills, Harlan D. *Mathematical Foundations for Structured Programming.* Gaithersburg, MD: IBM Federal Systems Division, IBM Corporation, 1972.

Misfeldt, Trevor, Gregory Bumgardner, and Andrew Gray. *The Elements of C++ Style.* Cambridge, UK: Cambridge University Press, 2004.

Pressman, Roger S. *Software Engineering: A Practitioner's Approach*, 7th ed. Chicago: McGraw-Hill, 2010.

Vermeulen, Allan, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur, an Patrick Thompson. *The Elements of Java Style.* Cambridge, UK: Cambridge University Press, 2000.