7.2 Design at the Routine Level

Code Complete

Cohesion:

- The idea of cohesion was introduced in a paper by Wayne Stevens, Glenford Myers, and Larry Constantine (1974).
  Other more modern concepts, including abstraction and encapsulation, tend to yield more insight at the class level.

- For routines, cohesion refers to how closely the operations in a routine are related.

Understanding the concepts is more important than remembering specific terms. Use the concepts as aids in thinking about how to make routines as cohesive as possible.

1. Functional cohesion is the strongest and best kind of cohesion, occurring when a routine performs one and only one operation. Examples of highly cohesive routines include sin(), GetCustomerName(),

    - The evaluation of their cohesion assumes that the routines do what their names say they do—if they do anything else, they are less cohesive and poorly named.

Several other kinds of cohesion are normally considered to be less than ideal:

2. Sequential cohesion exists when a routine contains operations that must be performed in a specific order, that share data from step to step, and that don't make up a complete function when done together.
   How would you make the routine functionally cohesive? You'd create separate routines.

3. Communicational cohesion occurs when operations in a routine make use of the same data and aren't related in any other way.

    - If a routine prints a summary report and then reinitializes the summary data passed into it, the routine has communicational cohesion: the two operations are related only by the fact that they use the same data.

- Split the operations into individual routines. The first prints the report. The second reinitializes the data, close to the code that creates or modifies the data. Call both routines from the higher-level routine that originally called the communicationally cohesive routine.

4. Temporal cohesion occurs when operations are combined into a routine because they are all done at the same time.
    - Typical examples would be Startup(), CompleteNewEmployee(), and Shutdown(). Some programmers consider temporal cohesion to be unacceptable.

    - It will be clear that the point of the routine is to orchestrate activities rather than to do them directly.

The remaining kinds of cohesion are generally unacceptable: they result in code that's poorly organized, hard to debug, and hard to modify.

5. Procedural cohesion occurs when operations in a routine are done in a specified order.
    - An example is a routine that gets an employee name, then an address, and then a phone number. The order of these operations is important only because it matches the order in which the user is asked for the data on the input screen.
    - To achieve better cohesion, put the separate operations into their own routines.

6. Logical cohesion occurs when several operations are stuffed into the same routine and one of the operations is selected by a control flag that's passed in. It's called logical cohesion because the control flow or "logic" of the routine is the only thing that ties the operations together—they're all in a big if statement or case statement together.

    - the operations are unrelated, a better name might "illogical cohesion."

    One example would be an InputAll() routine that inputs customer names, employee timecard information, or inventory data depending on a flag passed to the routine.

    - If the operations use some of the same code or share data, the code should be moved into a lower-level routine and the routines should be packaged into a class.

    - It's usually all right, however, to create a logically cohesive routine if its code consists solely of a series of if or case statements and calls to other routines. In

such a case, if the routine's only function is to dispatch commands and it doesn't do any of the processing itself, that's usually a good design. The technical term for this kind of routine is "event handler." An event handler is often used in interactive environments such as the Apple Macintosh, Microsoft Windows, and other GUI environments.

Cross-Reference

Although the routine might have better cohesion, a higher-level design issue is whether the system should be using a case statement instead of polymorphism. For more on this issue, see "Replace conditionals with polymorphism (especially repeated case statements)" in Specific Refactorings

7. Coincidental cohesion occurs when the operations in a routine have no discernible relationship to each other. Other good names are "no cohesion" or "chaotic cohesion."

Resumen:

- None of these terms are magical or sacred.
- Learn the ideas rather than the terminology.
- It's nearly always possible to write routines with functional cohesion, so focus your attention on functional cohesion for maximum benefit.