Cohesion (computer science)

From Wikipedia, the free encyclopedia

In computer programming, **cohesion** is a measure of how well the lines of source code within a module work together to provide a specific piece of functionality. Cohesion is an ordinal type of measurement and is usually expressed as "high cohesion" or "low cohesion" when being discussed. Modules with high cohesion tend to be preferable because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability whereas low cohesion is associated with undesirable traits such as being difficult to maintain, difficult to test, difficult to reuse, and even difficult to understand.

Cohesion is usually contrasted with coupling. High cohesion often correlates with low coupling, and vice versa. The software quality metrics of coupling and cohesion were defined by Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine from source code analysis they conducted on several programming projects while at IBM, all in an effort to identify the characteristics of "good" programming practices.

Contents

High cohesion

In object-oriented programming, it is good to assign responsibilities to classes in a way that keeps cohesion high. The likelihood of reuse is increased and complexity kept manageable.

Cohesion is a measure of how strongly related and focused the responsibilities of a single class are.

Cohesion is decreased if:

- The responsibilities (methods) of a class have little in common.
- Methods carry out many varied activities, often using varied data.

Disadvantages of low cohesion (or "weak cohesion") are:

- Difficult to understand.
- Difficult to maintain, because it is constantly affected by changes.
- Difficult to reuse a class because most applications won't need a random set of operations attached to a class.

Types of cohesion

Cohesion is a qualitative measure meaning that the source code text to be measured is examined using a rubric to determine a cohesion classification. The types of cohesion, in order of lowest to highest, are as follows:

Coincidental cohesion (worst)

Coincidental cohesion
: Coincidental cohesion is when parts of a module are grouped arbitrarily; the parts have no significant relationship (e.g. a module of frequently used functions).

Logical cohesion
: Logical cohesion is when parts of a module are grouped because of a slight relation (e.g. using control coupling to decide which part of a module to use, such as how to operate on a bank account).

Temporal cohesion
: Temporal cohesion is when parts of a module are grouped by when they are processed - the parts are processed at a particular time in program execution (e.g. a function which is called after catching an exception which closes open files, creates an error log, and notifies the user).

Procedural cohesion
: Procedural cohesion is when parts of a module are grouped because they always follow a certain sequence of execution (e.g. a function which checks file permissions and then opens the file).

Communicational cohesion
: Communicational cohesion is when parts of a module are grouped because they operate on the same data (e.g. a method updateStudentRecord which operates on a student record, but the actions which the method performs are not clear).

Sequential cohesion
: Sequential cohesion is when parts of a module are grouped because the output from one part is the input to another part (e.g. a function which reads data from a file and processes the data).

Functional cohesion (best)
: Functional cohesion is when parts of a module are grouped because they all contribute to a single well-defined task of the module (a perfect module).

Since cohesion is a ranking type of scale, the ranks do not indicate a steady progression of improved cohesion. Studies by various people including Larry Constantine and Edward Yourdon as well as others indicate that the first two types of cohesion are much inferior to the others and that modules with communicational cohesion or better tend to be much superior than lower types of cohesion. The seventh type, functional cohesion, is considered the best type.

However, while functional cohesion is considered the most desirable type of cohesion for a software module, it may not actually be achievable. There are many cases where communicational cohesion is about the best that can be attained in the circumstances. However the emphasis of a software design should be to maintain module cohesion of communicational or better since these types of cohesion are associated with modules of lower lines of code per module with the source code focused on a particular functional objective with less extranous or unnecessary functionality, and tend to be reusable under a greater variety of conditions.

References

- Yourdon, E.; Constantine, L L. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, copyright 1979 by Prentice-Hall, Yourdon Press.

See also

- List of object-oriented programming terms

Category: Software architecture
