ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Study of Software Metrices Coupling and Cohesion

Ashrami Shukla*, Aditya Bisen
Acropolis Institute of Technology & Research, Indore, India

Abstract

Software quality in context to software engineering refers to two different notions which are namely Software Functional Quality and Software Structural Quality. Software functional quality could be defined as something that reflects how well a function complies with a given design where as on the other hand software structural quality refers to how well a function meets to non functional requirements. Software crisis refers to the difficulties faced in writing useful and efficient computer programs. These crises are mainly due to the rapid increase in computer power and complexity of the problems that could be tackled. With the increase in complexity of the software, many problems arise because methods are neither sufficient nor efficient enough. In this research paper we are going to present object oriented metrics coupling and cohesion. They can be use as a powerful tool in not only gaining knowledge about the quality of the software but also help us to check the complexity of the system.

Keyword: Software metrics, cohesion, coupling.

Introduction

Software quality is a measure that helps one know how well software is designed and how well confirms to that design. Software quality may be defined as conformance to unambiguously stated functional and performance requirements. The key points of software quality are

- Software requirements are the foundations for the measurement of quality
- If given set of criteria is not followed then the result is lack of quality
- If the software confirms to meet some specific requirements and it fails to meet those requirements then in such a case the quality of the software is suspected.

Coupling and cohesion are considered as the two most important aspects in quality of a software. The ever increasing need for software quality measurement has led to a lot of research in software metrics and the development of software metrics tools like coupling and cohesion. As object oriented analysis and design is considered at front spot of the software engineering cohesion metrics have been developed. For the ideal choice of a good quality software it is desired to have low coupled and high cohesive design. Coupling is the degree t which each program module relies in each other. Coupling measures the strength of all relationships between functional units where as cohesion measures the semantic strength between components within a function, cohesion is closely bound to the idea of abstraction. Software metrics are basically of two types namely internal quality and external quality. Internal quality measured is performed on the software itself and are measurable both during and after the creation of software. External quality measures are evaluated with respect to how a product relates to its environment. Even though coupling and cohesion deal with software quality they are entirely different concepts. In order to have the best quality of a software coupling and cohesion should reach the two opposite ends. Which means loose coupling and strong cohesion provides the best out of software. Loose coupling means having private fields, non public classes and private methods while making all members visible having package as default visibility comes under high cohesion. Coupling is affected by the object-oriented features like inheritance and dynamic binding dynamically; interaction-based coupling is bound to show the impact of such features in the dynamic analysis results. Cohesion is related to robustness, reliability and reusability. Cohesion is the concept that captures intra module. Cohesion is in form of several levels.

Coupling

Since object oriented technology uses objects and not algorithms as its fundamental building blocks, the approach to software metrics for object oriented programs must be different from the standard metrics set. Thus there is a need for a good software quality. In order improve the quality of the software on must first define the aspects of quality and then decide how he is going to measure them. Some of the seminal

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

methods of evaluating an object-oriented design are through the use of measures for coupling and cohesion. Coupling or dependency is the degree to which each module unit relies on other unit. Coupling and cohesion are two contrasted things. Low coupling correlates with high cohesion and vice versa.

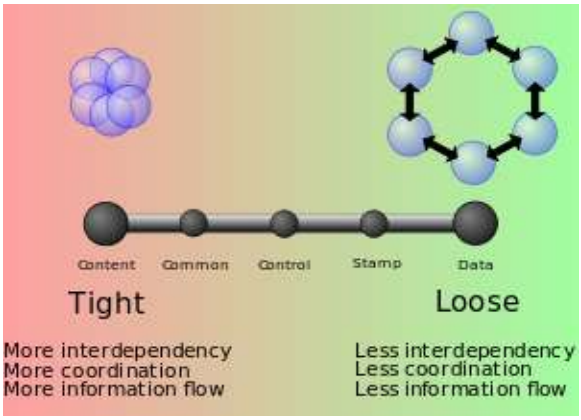

Figure 1: Types of coupling

Types of coupling:

It is also known as data structured coupling. It occurs when modules share composite data structures and use only same part of it. This may lead to changing of the way a module reads a record because the field which is not required by the module has now been modified.

F. Data Coupling:

Data coupling occurs when modules share data through parameters. Each datum is an elementary piece also these are the only data that is actually shared.

G. Message Coupling:

This type of coupling is the loosest type of coupling. Message coupling could be achieved by state decentralization and component communication done via parameters or message passing.

H. No Coupling:

As the name suggest in this type of coupling modules do not communicate with each other at all.

Limitations of coupling:

Tightly coupled systems tend to exhibit the following the following developmental characteristics, which are often seen as limitations to coupling:

- Any change in one module usually forces a ripple effect of changes in the other module connected to it.
- Assembly of modules may require more efforts and time due to increased inter-module dependency.

Coupling can be high (tight) coupling or low (loose) coupling. Some other types are discussed below

A. Content Coupling:

It is also known as pathological coupling. It occurs when one module depends on the internal working of another module. And thus making any change in the second module will produce data which will automatically lead to the changes in first module.

B. Common Coupling:

It is also known as global coupling .it usually occurs when two modules share the same global data. Making any change in one of the shared module causes changes in all the other modules using it.

C. External Coupling:

As the name suggests this coupling occurs when two modules share an externally imposed data format, communication protocol or device interfaces. This coupling is basically related to external tools and devices.

D. Control Coupling:

This coupling could be defined as one module controlling the flow of another module, by passing it information on what to do and how to do.

E. Stamp Coupling :

A particular module might be harder to reuse because dependent modules must be included with the given module.

Cohesion

As discussed above cohesion plays an important role in the quality of software it could be defined as the following definition given below. Cohesion refers to the degree to which the elements of a module belong to one another in other words it is the measure of how strongly related each piece of functionality expressed by the source code of a software module. Cohesion is dependent on the two given criteria if these help the cohesion to increase. If the functionalities embedded in the class are common to the functionalities accessed through the methods. Methods carry out a small number of related activities. Cohesion is thus dependent on both the factors that are coupling and complexity.

Types of cohesion:

Cohesion being a qualitative measure which means that the source code to be measured is examined using a rubric to determine a classification. Types of cohesion which starts from worst to best are as follows:

A. Coincidental Cohesion:

Coincidental cohesion may be defined as a cohesion in which parts of module are arbitrarily grouped. The relationship

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

between the parts is that they are grouped together.

B. Logical Cohesion:
As the name suggests logical cohesion may be defined as a cohesion in which the parts of module are logically categorized to do the same thing even if they are different in nature.

C. Temporal Cohesion:
When the parts of module are grouped during the process they fall under the category of temporal cohesion. The processing is done at a particular time during the execution.

D. Procedural Cohesion:
When the grouping of parts of module is done on a criteria that they follow certain sequence of execution it is called procedural cohesion.

E. Communicational / Informational Cohesion:
When the parts of a module are operated on the same data they fall under the category of communicational cohesion. This is also known as informational cohesion.

F. Sequential Cohesion:
When the output of one part is the input for other part of the module this type of cohesion is known as sequential cohesion.

G. Functional Cohesion:
Functional cohesion may be defined as a cohesion in which parts of module are grouped as they all participate in a single well defined task of the module.

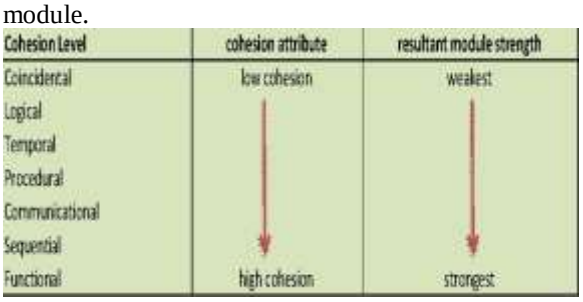

Figure 2: Types of Cohesion

Conclusion

The software metrics is the way that helps us to take the corrective steps to develop the good quality software's. In this paper we have only discussed the software metrics like coupling and cohesion and how they are helpful for developing the good quality software's. The types of cohesion and coupling are very much useful when we design the application and it also helps us to check the complexity of the application at designing stage.

References

1. M.C. Carlisle and A. Rogers. Software caching and computation migration in olden. In ACM Symposium on Principles and Practice of Parallel Programming, pages 29–38, Santa Barbara, California, USA, July 1995.
2. I.M. Chakravarti, R.G. Laha, and J. Roy. Handbook of Methods of Applied Statistics, volume 1. John Wiley and Sons, New York, USA, 1967.
3. S.R. Chidamber and C.F. Kemerer. Towards a metrics suite for object-oriented design. In Object Oriented Programming Systems Languages and Applications, pages 197–211, Phoenix, Arizona, USA, November 1991.
4. S.R. Chidamber and C.F. Kemerer. A metrics suite for object-oriented design. IEEE Transactions on Software Engineering, 20(6):467–493, June 1994.
5. E.J. Chikofsky and J.H. Cross II. Reverse engineering and design recovery: A taxonomy. IEEE Software, 7(1):13–17, 1990.
6. Choi, M. Gupta, M.J. Serrano, V.C. Sreedhar, and S.P. Midkiff. Stack allocation and synchronization optimizations for Java using escape analysis. ACM Transactions on Programming Languages and Systems, 25(6):876 – 910, November 2003.
7. L.L Constantine and E. Yourdon. Structured Design. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1979.
8. M. Dahm. Byte Code Engineering Library (BCEL), version 5.1, April 25 2004. http://jakarta.apache.org/bcel/.
9. D.P. Darcy, C.F. Kemerer, S.A. Slaughter, and T.A. Tomayko. The structural complexity of software: An experimental test. TOSE, 31(11):982–995, 2005.
10. S. Demeyer, S. Ducasse, and O. Nierstrasz. Finding refactorings via change metrics. In 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pages 166–178, Minneapolis, Minnesota, USA, 2000.
11. B. Dufour, K. Driesen, L. J. Hendren, and C. Verbrugge. Dynamic metrics for Java. In Conference on Object-Oriented Programming Systems, Languages and Applications, pages 149–168, Anaheim, California, USA, October 26-30 2003.