Estimation of Degree of Connectivity to Predict Quality of Software Design

S.D. Thilothame, M. Mashetty Chitralekha, U.S. Poornima and V. Suma

Abstract The main goal of Object Oriented Methodology is to deliver software which is maintainable. *Post_Development_Quality_Requirements* such as software maintainability is depending on design quality. Coupling and Cohesion (C&C) are two design quality factors which are measurable. C&C are influenced by structure of a class which is a basic unit of Object Oriented design. Defining the class structure and their relationships measures the design quality which in turn is indicator for quality requirements such as maintainability, reusability and scalability. This paper explores how different dependency types between classes adds on to design complexity and hence software quality by proposing a model which calculates Degree of Connectivity (DC) between the classes and Coupling Index (CI) of overall software. Thus, it is now possible to infer that design quality not only depends on class structure, but also upon the level of relationships such as inheritance, aggregation, composition and association present in software.

Keywords Design quality · Class relationships · C&C · Degree of Connectivity (DC) · Coupling Index (CI) · Post_Development_Quality_Requirements

1 Introduction

Software Engineering domain provides a platform for programmers and researchers to develop quality software, tools and process methodologies. Object Oriented Programming is a popular methodology which facilitates to develop complex software. Complexity increases due to bad design. Researchers are working on

S.D. Thilothame · M.M. Chitralekha · U.S. Poornima (✉) · V. Suma
Dayananda Sagar College of Engineering, Bangalore, India
e-mail: uspaims@gmail.com

V. Suma
e-mail: sumadsce@gmail.com

U.S. Poornima
Raja Reddy Institute of Technology, Bangalore, India

assessing the design quality either at code level (Post Assessment) or during High Level Design (Pre Assessment) through UML diagrams. Research has been going on to measure internal binding between attributes, methods and attributes_methods [1, 2]. Cohesion reflects such binding at code level, namely Low Level Design (LLD). Author of [3] formed metric to measure the cohesion at High Level Design (HLD) using UML diagrams. Many such metrics are proposed and mathematically validated for both LLD and HLDs.

It is difficult to anticipate the complexity only on strength of internal dependency in a class. Design complexity also depends on external binding of classes. Class relationships such as inheritance, aggregation, composition and association have their own impact on design quality. Each class relationship coins at data and method level. This work considers (1) *Data Binding, when two classes binds each other through object data*, (2) *Method Binding, when two classes binds each other through methods,* (3) *Data_Method Binding, when two classes binds each other through both data and method. Further, the work considers that aggregation and composition relationships proposes Data Binding, Dependency relationship proposes Method Binding and Inheritance proposes Data_Method Binding.*

This paper proposes a model to measure the Degree of Connectivity (DC) of different relationships and their individual contribution to design complexity. The model then calculates Coupling Index (CI) of overall project which is an indicator of design complexity during maintenance. This part of the research has used a Java project with 6 classes and different coupling types to calculate DC and CI.

This paper is organised as below, Sect. 2 is on related work, Sect. 3 proposes a model, Sect. 4 is about the measuring degree of connectivity using matrix representation, Sect. 5 is on calculating Coupling Index and Sect. 6 concludes the paper.

2 Related Work

This section summarises the survey of related work on coupling and cohesion.

Author [4] discusses software coupling approach on Object Oriented dependencies between classes mainly concentrated on types of coupling that are unavailable until after the completion of implementation. The coupling measure from source code has the advantage of having quantitative and more specific measure, but information is not available before implementation.

Author [5] made survey on design pattern. The pattern contains the dynamic and static behaviour of different types of entities which can be traced as functional diagram with dependency between them. The coupling factor represents the degree of relationship which the industries perceive and measures for quality of software design. The main principle of design of pattern is to reduce the coupling index for minimizing the complexity of design. The entire quality of software design is thus based on complexity of relationship between modules.

Author [6] discusses the significance of cohesion and coupling on design quality. Objects and classes are the entities in solution space and software quality directly depends on design quality of such logical entities. C&C are the two prime factors in object oriented design, measuring them can become an indicator to reduce the complexity. In complicated software, architecture of design required to be flexible and maintainable.

Author of [7] discussed that there are many number of techniques and tools are available to perform metric analysis on such code or software. The entire software modularization is partitioned into three main components (1) Use of API (2) Use of non API (3) Use of shared variable. This study provides a conceptual and practical framework for measurement of various factors like polymorphism, inheritance, coupling and cohesion and depth of inheritance. They used "step-in out" technique to get their functioning therefore increasing the entire quality of software and productivity.

Author of [8] discusses that earlier coupling measures consider only the static coupling but they do not consider dynamic coupling because of polymorphism and may usually deprecate the software complexity and miscalculate the need for coding inspection, testing and debugging. The proposed method consists of three steps such as introspection procedure, post processing and coupling measure. Finally metrics of coupling are evaluated for dynamic coupling. The development result represent that propose system will accurately evaluate the metrics of coupling dynamically. Finally author recommended dynamic coupling evaluation techniques which contain introspection procedure, including trace events into functions of all classes and anticipating dynamic behaviour at the time of execution of source code.

Author of [9] suggested the basic metric of coupling for object-oriented systems. In that metric, they stated CBO (Coupling between Objects) metric as number of non-inheritance dependent couples with other remaining classes.

Author of [10] states that measure the dynamic coupling at phase of analysis only. They define that dynamic coupling depends on the frequency with which classes communicate at runtime. They suggested Dynamic or run time Clustering Mechanism (DCM) that performs by capturing the circumstances for dynamic coupling at analysis phase.

Author of [11] recommended a dynamic method to calculate coupling index of software systems. It gives final result that generally used analysis of the and gives the partial dynamic behaviour of the system.

3 Algorithmic Representation of Proposed Model

The proposed model works as below.

Step 1: Accepts a Java Project
Step 2: Calculates the DC which is overall connection between classes
Step 3: Calculates types of dependency of individual class
Step 4: Calculates Coupling Index of each relationship
Step 5: Reports severity of coupling using severity index table.

4 Measuring the Degree of Connectivity Using Matrix Representation of Coupling

To assess the complexity, it is better to have quantitative information on different types of connectivity between the classes. Hence, this part of the work dealt with designing a model which takes a moderate size Java project with 6 classes, namely, A, B, C, D, E and F. The intension is to get a quantitative measure on different types on connectivity between the classes and to calculate Coupling Index which is an indicator for project maintainability. The connectivity is represented in matrix form to identify the degree of connectivity as in Table 1.

In the above Table 1, the value 0 denotes connectivity present between two classes and 1 denotes no connectivity. The degree of connectivity is formulated as below.

$$Degree\ of\ Connectivity\ (DC) = \frac{Number\ of\ connectivity}{Total\ Number\ of\ Connectivity} \times 100 \qquad (1)$$

Result: DC for sample project = (13/36) × 100 = 36 %.

The sample project has 36 % of overall connectivity between the classes which is a first level indicator of representing design complexity for maintenance group.

After finding the degree of connectivity, the model determines which type of dependency exits between two classes. The dependency between classes can be

Table 1 Matrix representation of coupling

Name of classes	A	B	C	D	E	F
A	0	1	0	1	1	0
B	1	0	1	1	1	1
C	0	0	0	1	1	0
D	1	1	0	0	0	0
E	0	0	1	0	0	0
F	1	0	0	0	0	0

Table 2 Matrix representation of class relationships of a sample project

Name of classes	A	B	C	D	E	F
A	No dependency	Inheritance	No dependency	Aggregation	Association	No dependency
B	Composition	No dependency	Inheritance	Composition	Aggregation	Association
C	No dependency	No dependency	No dependency	Inheritance	Inheritance	No dependency
D	Inheritance	Aggregation	No dependency	No dependency	No dependency	No dependency
E	No dependency	No dependency	Aggregation	No dependency	No dependency	No dependency
F	Inheritance	No dependency	No dependency	No dependency	No dependency	No dependency

Table 3 Types of dependency for class A

Name of class	Inheritance	Aggregation	Association	Composition
A	1	1	1	0

Table 4 Types of dependency for class B

Name of class	Inheritance	Aggregation	Association	Composition
B	1	1	1	1

Data Binding (Aggregation, Composition), Method Binding (Dependency) or Data _Method Binding (Inheritance) as shown in Table 2.

Table 2 Presents relationships between various classes present in project which is an input to identify individual class relationships to find CI as in Tables 3 and 4.

Similarly, dependency types are calculated for classes C, D, E and F.

5 Calculating the Coupling Index

After finding types of dependency for each class relationship with each class is tabularised as in Table 5. This is used to calculate CI of a project.

Hence the CI represents strength of each relationship in a project which can be calculated using the formula as shown below.

$$Coupling\,Index\,(CI) = \sum \frac{number\,of\,relation\,present}{total\,number\,relations} \times 100 \tag{2}$$

Result: The following Table 6 lists CI of each relationship in a sample project.

Using CI, severity index of each class coupling relationship is calculated as shown in Table 6. The range of 1–4 is taken to fix the severity of each coupling type (Table 7).

Using the severity index table, the sensitivity of coupling for the sample is calculated which shows that Inheritance coupling in a project is extremely coupled,

Table 5 Class with total dependency

Name of classes	Inheritance	Aggregation	Association	Composition
A	1	1	1	0
B	1	1	1	2
C	2	0	0	0
D	1	1	0	0
E	0	1	0	0
F	1	0	0	0

Table 6 CI of each relationship

Name of relationship	Formula	CI in percentage (%)
Inheritance	6/14	42.85
Aggregation	4/14	28.57
Association	2/14	14.28
Composition	2/14	14.28

Table 7 Severity index of each coupling using CI

Coupling range	Percentage (%)	Severity of coupling type
More than 4	40	Extremely coupled
2–3	30	Tightly coupled
1–2	20	Moderately coupled
0–1	10	Loosely coupled

aggregation is tightly coupled and both association and composition are loosely coupled.

Thus, CI decides the severity of each relationship in software. Depending on CI value, the software maintainability can be identified as high risk, medium risk and low risk which mirrors the design complexity.

6 Conclusion

Software Engineering domain invites researchers and programmers to improve development process, Techniques and Tools to provide quality software. Improving design quality facilitates *Post_Development_Quality_Requirements* such as software maintainability, reusability at ease. Quality design is achieved through well defined class in Object Oriented Programming. Much research has been going on Coupling and Cohesion which are two design quality decisive factors represents dependency between classes and attributes within a class respectively. This paper proposes a model to calculate overall Degree of Connectivity (DC) and Coupling Index (CI) of a project to find the level of complexity which becomes an indicator for maintainability in future.

Data Binding (Aggregation, Composition), Method Binding (Dependency) or Data _Method Binding (Inheritance) are major coupling types exists between the classes. Since Coupling and Cohesion are closed knitted, the other features of Object Oriented Programming such as dynamic binding, polymorphism in coupling types would influences the cohesion and vice versa.

Acknowledgments The authors would like to sincerely acknowledge all the industry personnel for their valuable suggestions, help and guidance in carrying out this part of research. The complete work is undertaken under the framework of Non Disclosure Agreement.

References

1. Al Dallal, Jehad, L.C. Briand, A precise method-method interaction-based cohesion metric for object-oriented classes. ACM Trans. Softw. Eng. Methodol. (TOSEM) **21**(2) (2012)
2. Al Dallal, Jehad, Incorporating transitive relations in low-level design-based class cohesion measurement. Softw.: Pract. Exp. **43**(6), 685–704 (2013)
3. Al Dallal, Jehad, L.C. Briand, An object-oriented high-level design-based class cohesion metric. Inf. Softw. Technol. **52**(12), 1346–1361 (2010)
4. J. Offutt, A. Abdurazik, S.R. Schach, Quantitatively measuring object-oriented couplings. Softw. Qual. J. **16**(4), 489–512 (2008)
5. P. Wolfgang, *Design patterns for object-oriented software development* (Addison-Wesley, Reading, Mass, 1994)
6. U.S. Poornima, Factors modulating software design quality (2014). arXiv:1402.2374
7. Deepti Gupta, Coupling based structural metrics—an quality assessment of software modularization. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **3**(6) (2013). ISSN: 2277 128X
8. S. Babu, Dr. R.M.S. Parvathi, Development of dynamic coupling measurement of distributed object oriented software based on trace events. Int. J. Softw. Eng. Appl. (IJSEA) **3**(1) (2012)
9. S.R. Chidamber, C.F. Kemerer, Towards a metrics suite for object-oriented design, in *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications, (OOPSLA' 91)*, SIGPLAN Notices, vol. 26, no. 11 (1991), pp. 197–211
10. H. Paques, L. Delcambre, A mechanism for assessing class interactions using dynamic coupling during the analysis phase, in *Proceedings of XVIII Brazilian Symposium on Software Engineering—SBES'99*, Florianopolis, Santa Catarina, Brasil (1999)
11. E. Schikuta, *Dynamic Coupling Metrics* (1993)