International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 12 Issue 11, November, 2023, Impact Factor: 7.751

Analysis of Object-Oriented Metrics Based on Dynamic Coupling and Cohesion Metrics

Supriya[1], Dr. Brij Mohan Goel[2]

[1]Research Scholar, Computer Science & Application Department, Baba MastNath University, Asthal Bohar, Rohtak
[2]Professor, Department of Computer Science and Engineering. , Baba Mastnath University Asthal Bohar, Rohtak

ABSTRACT

This study examines dynamic metrics related to coupling and cohesion. specifically Dynamic Coupling RCD, DMCC, OIF, DIC, RCI and Dynamic Cohesion RCI, DMIC, OLC, DAUC and their influence on software quality attributes like maintainability, modularity, and scalability. The results emphasize the importance of these metrics in practical applications and offer recommendations for utilising them to enhance software design and performance. Dynamic object-oriented metrics are crucial for assessing the quality of object-oriented software systems, emphasising runtime behaviour rather than static structure.

Keywords: Dynamic Metrics, RCD, DMCC, OIF, DIC, RCI, DMIC, OLC, DAUC Object-Oriented Design, Software Quality, Coupling.

INTRODUCTION

Dynamic coupling metrics measure the runtime interactions and dependencies between classes in an object-oriented system[1]. These metrics provide insights into how tightly coupled different parts of the system are during actual execution, which can reveal potential bottlenecks and areas for improvement. Schikuta(1993) was the first to propose a dynamic approach to measure coupling of software systems[2]. He defined static measures as those that are derived from the source code. Briand et al.(1997) studied a vast amount of available literature on coupling in object-oriented systems and concluded that all the metrics at that time measured coupling at the class level from static code[3]. No measures of runtime object-level coupling had been proposed till that time. Singh and Singh proposed four class-level dynamic coupling metrics to assess the quality of object oriented software systems[4]. They proposed Dynamic Clustering Mechanism (DCM) that works by tracking hot spots for dynamic coupling at analysis phase. Yacoub et al.(1999) defined a set of object-level dynamic coupling metrics designed to evaluate the change-proneness of a design[5]. The metrics were applied at an early development phase to determine design quality. Arisholm(2004) proposed dynamic Import Coupling (IC) and Export Coupling (EC) measures quantifying message communication among objects at runtime[6]. These measures are further defined at object-level and class level with which metrics based upon either number of messages sent/received or methods invoked or classes accessed. Mitchell and Power(2003a,2003b,2003c,2004a,2004b&2005)conducted a number of studies on the quantification of many runtime class-level coupling metrics for object-oriented (Java) programs[7,8,9,10]. Zaidman and Demeyer(2004)exposed the usage of dynamic coupling metrics in analysing the event traces of large scale industrial applications[11]. Hassoun et al.(2004&2005) studied object-level coupling as it evolves during program execution and proposed a dynamic coupling measure that takes object interactions into consideration[12,13]

Dynamic coupling metrics are mainly focused on the Runtime Coupling Degree (RCD), Dynamic Method Call Coupling (DMCC), Object Interaction Frequency (OIF), and Dynamic Inheritance Coupling (DIC).

Runtime Coupling Degree (RCD)
Definition: Measures the number of different classes that a particular class interacts with during execution.

Formula:

$$\mathrm{RCD}(C) = \text{Number of distinct classes } C \text{ communicates with at runtime}$$

Example: Consider a class A that interacts with classes B, C, and D during execution. The RCD for class A is 3.

Implications: High RCD values indicate a class with high dependency on other classes, suggesting tight coupling and potential difficulties in maintenance and testing.

Dynamic Method Call Coupling (DMCC)
Definition: Measures the number of method calls made by a class to methods in other classes during runtime.

Formula:

$$\text{DMCC}(C) = \sum_{m \in M(C)} \text{Number of external method calls made by m}$$

Example: If class A has methods that make 10 calls to methods in class B and 5 calls to methods in class C, the DMCC for class A is 15.

Implications: High DMCC values suggest that a class heavily relies on methods from other classes, indicating high coupling and potential design issues.

Object Interaction Frequency (OIF)
Definition: Measures the frequency of interactions between objects of different classes at runtime.

Formula:

$$\text{OIF}(C_1, C_2) = \text{Number of interactions between objects of } C_1 \text{ and } C_2 \text{ during execution}$$

Example: If objects of class A interact with objects of class B 20 times during execution, the OIF between A and B is 20.

Implications: High OIF values between two classes indicate a strong coupling, which may necessitate examining the relationship between these classes to ensure it aligns with design principles.

Dynamic Inheritance Coupling (DIC)
Definition: **Measures the number of times subclasses access methods or attributes of their parent classes at runtime.**

Formula:

$$\text{DIC}(C) = \sum_{m \in M(C)} \text{Number of superclass method/attribute accesses by m}$$

Example: If a subclass B accesses its super class methods or attributes 12 times during execution, the DIC for class B is 12.

Implications: High DIC values suggest that subclasses are highly dependent on their parent classes, potentially indicating a need to refactor the inheritance hierarchy.

Dynamic cohesion metrics measure the degree of relatedness and interdependence among elements within a class based on their runtime behavior. High cohesion indicates that the class has a well-defined purpose and its elements work together effectively. Dynamic cohesion metrics are defined as Runtime Cohesion Index (RCI), Dynamic Method Interaction Cohesion (DMIC), Object Lifetime Cohesion (OLC), and Dynamic Attribute Usage Cohesion (DAUC).

Runtime Cohesion Index (RCI)
Definition: Measures the degree of relatedness of methods within a class based on shared runtime data access.

Formula:

$$\text{RCI}(C) = \frac{\sum_{m_i m_j \in M(C)} \text{Number of shared attribute accesses by } m_i \text{ and } m_j}{|M(C)| \times (|M(C)| - 1)}$$

Example: If methods within class A share access to attributes 30 times, with |M(A)|=6 methods, the RCI for class A is:

$$\text{RCI}(A) = \frac{30}{6 \times (6-1)} = \frac{30}{30} = 1$$

Implications: High RCI values indicate high cohesion, suggesting that methods within the class frequently access shared data, which is a sign of a well-designed class.

Dynamic Method Interaction Cohesion (DMIC)
Definition: Measures the extent to which methods within a class interact with each other during execution.

Formula:

$$\mathrm{DMIC(C)} = \frac{\sum_{m_i m_j \in M(C)} \text{Number of calls from } m_i \text{ to } m_j}{|M(C)| \times (|M(C)| - 1)}$$

Example: If methods in class A call each other 15 times and there are 5 methods, the DMIC for class A is:

$$\mathrm{DMIC(A)} = \frac{15}{5 \times (5-1)} = \frac{15}{20} = 0.75$$

Implications: High DMIC values indicate high cohesion, as methods within the class frequently invoke each other, suggesting they work together closely to achieve the class's functionality.

Object Lifetime Cohesion (OLC)
Definition: Measures the cohesion of a class based on the lifetime of its objects and their interactions.

Formula:

$$\mathrm{OLC(C)} = \frac{\text{Total interaction time between objects of C}}{\text{Total lifetime of objects of C}}$$

Example: If the total interaction time between objects of class A is 1200 ms and the total lifetime of these objects is 2000 ms, the OLC for class A is:

$$\mathrm{OLC(A)} = \frac{1200}{2000} = 0.6$$

Implications: High OLC values suggest that objects of the class interact consistently throughout their lifetimes, indicating high cohesion and a well-defined role within the system.

Dynamic Attribute Usage Cohesion (DAUC)
Definition: Measures the extent to which methods within a class use the same set of attributes during execution.

Formula:

$$\mathrm{DAUC(C)} = \frac{\text{Number of shared attributes accessed by methods}}{|A(C)|}$$

Example: If methods in class A frequently use 8 out of 10 attributes, the DAUC for class A is:

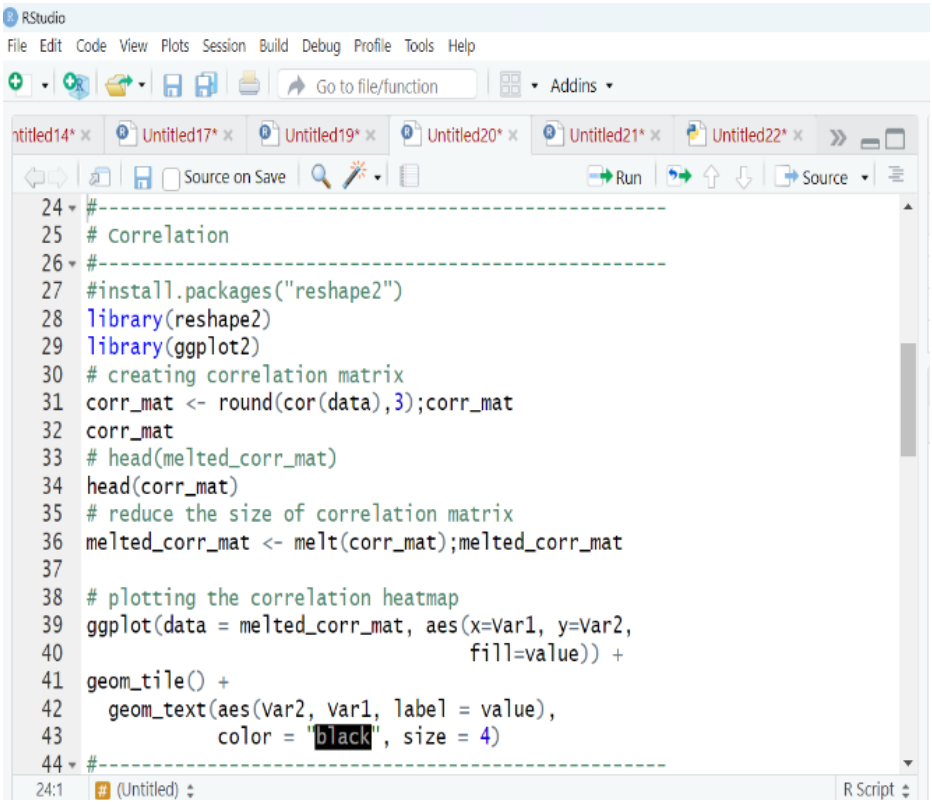
$$\mathrm{DAUC(A)} = \frac{8}{10} = 0.8$$

Implications: High DAUC values indicate high cohesion, as methods within the class frequently use the same attributes, suggesting they are related in terms of functionality.

Correlation Analysis
We perform a correlation analysis to understand the relationships between the coupling and cohesion metrics.
The correlation coefficient is denoted by (r) and its range lies between -1 to 1.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 (y_i - \bar{y})^2}},$$

Where, r is correlation coefficient, x_i is the values of x variable in a matrix component, y_i is the values of y variable in a matrix component, and $\bar{x}$ and $\bar{y}$ is the mean value. In the R software, we are using the corr prompt for correlation matrix.



Result: After compiling these commands then got the result, correlation coefficients given below

	RCD	DMCC	OIF	DIC	RCI	DMIC	OLC	DAUC
RCD	1	0.1	0.028	0.086	0.061	0.141	-0.166	0.114
DMCC	0.1	1	-0.072	-0.038	0.149	0.131	0.014	0.084
OIF	0.028	-0.072	1	-0.162	-0.065	0.001	0.263	-0.062
DIC	0.086	-0.038	-0.162	1	0.076	0.095	-0.133	0.031
RCI	0.061	0.149	-0.065	0.076	1	-0.038	-0.161	-0.077
DMIC	0.141	0.131	0.001	0.095	-0.038	1	0.005	0.146
OLC	-0.166	0.014	0.263	-0.133	-0.161	0.005	1	-0.108
DAUC	0.114	0.084	-0.062	0.031	-0.077	0.146	-0.108	1

DISCUSSION

Coupling Metrics: RCD, DMCC, OIF, and DIC provide different perspectives on class dependencies and interactions. High values in these metrics may indicate potential areas for reducing coupling by refactoring code to minimize inter-class dependencies.

Cohesion Metrics: RCI, DMIC, OLC, and DAUC focus on the internal consistency and relatedness of methods and attributes within a class. High values in these metrics suggest that the class has a single, well-defined purpose and that its methods and attributes are highly interrelated.

Runtime Coupling Degree (RCD): This metric highlights the extent of a class's dependencies on other classes. High RCD values indicate tight coupling, which can hinder modularity and make the system harder to maintain and test.

Dynamic Method Call Coupling (DMCC): DMCC provides a detailed view of method-level dependencies. High values suggest that a class relies heavily on external methods, pointing to potential design improvements to reduce dependency.

Object Interaction Frequency (OIF): OIF captures the frequency of interactions between objects, offering insights into class relationships and potential areas for decoupling.

Dynamic Inheritance Coupling (DIC): DIC measures subclass dependencies on parent classes. High values can indicate a need to re-evaluate the inheritance hierarchy to reduce coupling.

Runtime Cohesion Index (RCI): RCI measures how methods within a class access shared data. High RCI values indicate strong

Dynamic Method Interaction Cohesion (DMIC): DMIC reflects the extent of method interactions within a class. High values indicate that methods work closely together, enhancing class cohesion.

Object Lifetime Cohesion (OLC): OLC measures the consistency of object interactions throughout their lifetimes. High OLC values suggest that objects fulfil cohesive roles within the system.

Dynamic Attribute Usage Cohesion (DAUC): DAUC indicates how methods within a class use shared attributes. High values suggest that methods are related in terms of functionality, enhancing class cohesion.

Implications for Software Design: Developers should strive for low dynamic coupling and high dynamic cohesion to enhance modularity, maintainability, and scalability. Tools and practices that monitor and manage dynamic interactions can significantly improve software design and performance.

Limitations and Future Work: This study focused on specific metrics and may not capture all aspects of dynamic behavior. Future research should explore additional dynamic metrics and their impact on different types of software systems.

REFERENCES

[1]. Gupta, V. & Chhabra, J. K. Dynamic cohesion measures for object-oriented software. J. Syst. Archit.**57**, 452–462 (2011).
[2]. Hitz, M., Montazeri, B. Measuring Coupling and Cohesion in Object-Oriented Systems. International Symposium on Applied Corporate Computing, Monterrey, Mexico, pp. 25-27, 1995.
[3]. Briand, L. C., Morasca, S., Basili, V. Measuring and Assessing Maintainability at the End of High-Level Design. In Proc. of International Conference on Software Maintenance, Montreal, Canada, pp. 88-97, 1993.
[4]. P. Singh, H. Singh, Class-level Dynamic Coupling Metrics for Static and Dynamic Analysis of Object-Oriented Systems, International Journal of Information and Telecommunication Technology, 1(1): 16-28, 2010.
[5]. Yacoub, S., Ammar, H., Robinson, T. Dynamic Metrics for Object-Oriented Designs. In Proc. of the 5th International Software Metrics Symposium, Boca Raton, USA, pp. 50-61, 1999.
[6]. Arisholm, E., Briand, L. C. & Foyen, A. Dynamic coupling measurement for object-oriented software. IEEE Trans. Softw. Eng.**30**, 491–506 (2004).
[7]. Mitchell, A., Power, J. F. Toward A Definition of Run-Time Object-Oriented Metrics. In Proc. of 7thECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, Darmstadt, Germany, pp. 1-7, 2003.
[8]. .8. Mitchell, A., Power J. F. An Empirical Investigation into the Dimensions of Run-Time Coupling in Java Programs. In Proc. of 3rd Conference on the Principles and Practice of Programming in Java, Las Vegas, USA, pp. 9-14, 2004.
[9]. Mitchell, A., Power, J. F. A Study of the Influence of Coverage on The Relationship Between Static and Dynamic Coupling Metrics. Science of Computer Programming Elsevier, vol. 59, issue 1-2, pp. 4-25, 2005.
[10]. Mitchell, A., Power, J. F. A Study of The Influence of Coverage on The Relationship Between Static and Dynamic Coupling Metrics. Science of Computer Programming, vol. 59, issue 1-2, pp. 4-25, 2006.
[11]. Zaidman, A., Demeyer, S. Analyzing Large Event Traces with The Help of Coupling Metrics. In Proc. of International Workshop on Object- Oriented Reengineering, Antwerp, Belgium, 2004.
[12]. Hassoun, Y., Johnson, R., Counsell, S. Empirical Validation of a Dynamic Coupling Metric Technical Report, BBKCS-04-03. School of Computer Science and Information Systems, Birkbeck College, University of London, UK, 2004.
[13]. Hassoun, Y., Johnson, R., Counsell, S. Dynamic Coupling Metric-Proof of Concept. IEE Proc.-Software, vol. 152, issue 6, pp. 273-279, 2005, doi: 10.1049/ip-sen:20045067.