International Journal of Knowledge Based Computer Systems
6 (1), June 2018, 01-06
http://www.publishingindia.com/ijkbcs/

Evaluating the Dynamic Metrics for Object-Relational Modeling

Justus Selwyn[1*] and Meenakshi K. Sundaram[2]
[1]School of Computing Sciences and Engineering, VIT University, Chennai, Tamil Nadu, India.
Email: justusdr@gmail.com
[2]Botho University, Botswana, South Africa. Email: meenakshisk@gmail.com
*Corresponding Author

Abstract: Runtime cohesion and coupling metrics are used to estimate the togetherness and the relatedness of objects in object oriented (OO) systems. The advancement of OO systems have combined the relational traits of database systems to form a powerful architecture called object-relational data models. The runtime cohesion and coupling metrics of an object help determine the behavioural complexity in terms of fetching and pre-fetching of objects. In this work we are proposing two algorithms for calculating the runtime cohesion and runtime coupling metric values for the objects in object-relational data models and have presented the metric results evaluated from the implementation of the algorithm. Also the runtime behavior of objects based on the runtime metrics values is also validated.

Keywords: Algorithms, Dynamic behavior of objects, Object relational data models, Runtime metrics.

I. Introduction

Object-relational data modelling have formed a new framework in combining the object nature of information systems and the relational nature of the databases. In the context of working on the dynamism of databases and its functionality, we attempted to propose that cohesion and coupling, when able to be measured in runtime, will help assess the performance of the Object-relational data based systems. Mitchel and Power [1, 2] have given thoughts on measuring the runtime cohesion and coupling for object-oriented systems. Motivated by their work, we attempt here to apply the definitions of the runtime cohesion and coupling in the object and relational runtime environment.

An Object Oriented Design for an object-relational data model is conceptualized as object oriented system built upon the traditional relation database concepts. The components associated with the relational system are records, triggers/SQL procedures, tables and the persistent objects itself. However, object-relational database consists of classes, objects, all the ACID properties of a relaional system, inheritance, encaptulation and methods acting on the object elements [3]. Hence metrics have become mandatory for such a complex database, and not ample research is done on this. The most common size measures for OR systems are table size (TS) a and Object Size (OS). The complexity measures for structure of the OR systems are depth in the relational tree (DRT), number of weighted methods per class (NWC), number of shared classes (NSC), referential degree (RD) and number of involved classes (NIC), [4, 5]. In addition to the structural complexity of OO systems, works in [1, 6, 7] states that cohesion and coupling are measures for assessing the dynamic behavior of objects. In [8, 9], the authors have made an advanced measurement of cohesion in packages in OO systems. In [10-13], the authors have given a survey on the existing runtime metrics. However, none of the works pertaining to object behavior were discussed.

II. Defining The Runtime Metrics

A. Cohesiveness of an Object

Cohesiveness of an object is the togetherness of the instance variables and member functions of an object. It is the ratio of common instance variables to total number of instance variables [14]. If the instance variables Iv_i and Iv_j are accessed by the member functions Md_i, Md_j respectively then the cohesiveness is defined as:

$$\mu(Md_i, Md_j) = \frac{Iv_i \cap Iv_j}{Iv_i + Iv_j} \quad (1)$$

Where $Iv_i \cap Iv_j \neq \varphi$

Cohesion between methods is not only static [6, 14] and hence its dynamism is considered as an important trait. The runtime cohesion of an object is the number of times the instance variables are accessed by its member variables. So, the ratio of $N(Iv_i \cap Iv_j)$ to the total number of instance variables gives the runtime cohesion metric value. Consider an object with n member methods, Md_1. , . , . , . Md_n and let $\{Iv_i\}$ be the set of instance variables referenced by method Md_i. Hence $N(Iv_i \cap Iv_j)$ is the number of times method Md_i and Md_j dynamically accesses instance variables from the sets $\{Iv_i\}$ and $\{Iv_j\}$. Hence the runtime calculation of cohesion for two methods can be derived as:

$$\mu(Md_i, Md_j)_R = \sum_{i=1,j=1}^{n,m} \frac{N(Iv_i \cap Iv_j)}{N(Iv_i + Iv_j)} \quad (2)$$

The summation of all these pair of members methods will define the runtime cohesiveness of an object:

$$\mu(Cohesiveness)_R = \sum_{i=1,j=1}^{n,m} (\mu(Md_i, Md_j)_R) \quad (3)$$

The runtime cohesive of an object indicates how rigid is the class design of that object. Higher the value, the object will be too rigid, and lower the value, the object will be too flexible without internal communication. Hence an optimal range of value is needed.

B. Coupling between Objects

The design-time coupling metric which is static was defined as 'the number of links to other objects and links from other objects to this object' [15]. Links are by means of method invocation and/or instance variables invocation. The runtime coupling is different from the design-time static coupling, in the sense, number of object invocation increases during runtime as and when new objects are instantiated. Even objects of same class_type, that have same characteristics, their interactions are dynamic by the several runtime factors that dictates the object-relational system [16].

The definition for the dynamic runtime coupling is given as the number of times thesum count of instance variables and member methods are invoked at runtime by an object of another class_type.

$$Count_{cp} = N\left(\sum_{i=0}^{n} I_i + \sum_{j=0}^{m} M_{INVj}\right) \quad (4)$$

$$M_{INV} = \sum_{i=0}^{m} M_i \times [1 + Arg_m \times 0.046]$$

Invoking the member methods and the instance variables involves the argument list that is passed or refered by the calling object. The number of arguments is determined by polymorphism, inheritance or overloading/overridding. The $Count_{cp}$ values ranges from 0 to 1; values tending to 1 indicates the object is tightly embedded in the schema, or if tending to 0 incidates the object is loosely coupled with the schema.

$$Coupling_R = \frac{k}{Count_{cp}} \quad (5)$$

Where k is a proportionality constant which is determined by the experimental factors [5]. This implies that $Count_{cp}$ is inversely proportional of the runtime $Coupling_R$. To have a increase in the runtime coupling value as the interactions increase, the reverse coupling approach in adopted.

The metric is defined as:

$$\mu(CBO_R) = 1 - Coupling_R \quad (6)$$

where the degree of coupling varies non-linearly between 0 and 1.

III. Runtime Evaluation of the Metrics

In this section we have derived and proposed two algorithms for calculating the runtime cohesion and coupling metric values. The experimental results obtained from implementing these algorithms are also analysed in this section.

A. The Algorithms

The algorithms for finding the runtime cohesion and coupling metric values are given in this section. The instances of the objects that are captured during runtime in the object-relational data model are the input for the algorithm in estimating the runtime cohesion metric ($Cohesiveness_R$).

The algorithm given in Fig. 1 uses functions like *get Method Count (), get Inst Var Accessed (), num Of Invocs (), get Total Inst Var (),* which are defined based on the platform of execution.

The inputs for the runtime coupling algorithm are the runtime instances of the objects and their connectivity, during runtime. The real-time generation and access of the objects vary from the design-time schema.

The coupling metric value calculated during design time gives the structural complexity of the schema whereas the coupling metric value calculated during runtime gives the functional and behavioral complexity of the schema.

```
Runtime_Cohesion_Metrics
Input: Or: Runtime instance of Object r
Output: µRT (Or): cohesion metric value
Declaration
 methodCount: number of methods in a class;
 metricCount: Initialized to zero;
 Ii: ith Instance variable;
 Mi: ith Method of a class;
1: if((methodCount=getMethodCount(Or))>1)
2:   for(i=1;i<=methodCount; increment(i) by 1)
3:     Ii = getInstVarAccessed(Mi);
4:     for(j=1;j<=methodCount; increment(j) by 1)
5:       Ij = getInstVarAccessed(Mj);
6:       commonInstVar = numOfInvocs(getCommonVar(Ii,Ij));
7:       sumInstVar = sum(Ii,Ij);
8:       unitCohesionCount = Divide(commonInstVar,sumInstVar);
9:     endfor /*end of j for loop*/
10:    metricCount = Sum(metricCount, unitCohesionCount);
11:  endfor /*end of i for loop*/
12:  µRT (Or)= metricCount; /* Cohesion metric value */
13: elseif (methodCount == 1)
14:    I = getInstVarAccessed(M);
15:    ITOT = getTotalInstVar(M);
16:    µRT (Or)= divide(I, ITOT); /*metric with instance variables*/
17: elseif (methodCount == 0)
18:    µRT (Or) = 0; /*Null Cohesion*/
19: endif
20: return µRT (Or)
```

Fig. 1: Runtime Cohesion Algorithm

```
Runtime_Coupling_Metrics
Input:   OS: Runtime instance of Object S
         Or: Runtime instance of Object r
Output: µRT (Or): coupling metric value
Declaration
 methodCount: number of methods in a class
 methodsInvoked: Number of methods invoked
 invComplexity: Invocation complexity initialized to zero
 Mi: ith method of a class
 Mxj: jth method of an external Object
1: if((methodCount=getMethodCount(Or)) > = 1)
2:   for(i=1;i<=methodCount; increment (i) by 1)
3:   if((objectsCreated(Mi)) and (not(thisObject(Mi))))
4:     instVarInvoked = getInstVarAccessed(Or,OS);
5:     methodsInvoked = getMethodsInvoked(Or,OS);
6:     for(j=1;j<=methodsInvoked; increment (j) by 1)
7:       numOfArgs = getArguments(Mxj, Mi);
8:       Sum(invComplexity, Sum(1, Multiply(numOfArgs, 0.046)));
9:     endfor /* end of j for loop*/
10:    Sum(couplingCount, Sum(instVarInvoked, invComplexity));
11:    initialize(invComplexity) to zero;
12:  endif
13:  endfor /*end of i for loop*/
14: elseif (methodCount==0)
15:    couplingCount = Sum(1, Multiply(numOfInstVar(Or),0.023));
16: endif
17: objectCoupling = Divide(1,couplingCount);
18: µRT (Or) = Difference(1, objectCoupling); /*Coupling metric */
19: return µRT (Or);
```

Fig. 2: Runtime Coupling Algorithm

The runtime coupling of the given database schema can be estimated during the pilot run of the schema design and also during the execution of the entire information systems for which the database acts as the back-end. The runtime coupling metric value keeps changing for every instances of schema-capture. The algorithm given in Fig. 2 uses functions like *get Method Count (), get Inst Var Accessed (), get Methods Invoked (), get Arguments ()* which are defined based on the platform of execution and implementation.

B. Preliminary Experimental Results

In our preliminary experimental study, the runtime cohesion among methods for objects of our own software Music-Expert-System (MES). The main object taken for consideration is staff_t type, which represents the five lines in the music software. Each musical score has one or more staff. In our experiments, we took one staff for one music score. We took three music scores, *Cradle, Father's Garden, Iam Late*, which are the music scores mentioned in the Grade exams of Trinity College London. Each song has different key signature and time signature, so that the data set forms the variety in the experimental setup. The runtime cohesion metric values are given in Table I. The cohesiveness of two methods are calculated iteratively which is finally summed up to the final runtime cohesion metric value.

The *staff_t* objects for the three different music scores exhibit different metric values at runtime. Higher the metric values, together are the objects, that is, the objects are cohesive.

TABLE I: COM_R FOR STAFF_F OBJECTS

Objects	Md_1,Md_2	Md_2,Md_3	Md_3,Md_1	$\mu(Coh)_R$
$Staff1_{Cradle}$	0.478	0.776	0.745	1.999
$Staff2_{Iam\text{-}Late}$	0.418	0.432	0.469	1.319
$Staff3_{Garden}$	0.612	0.627	0.522	1.761

TABLE II: CBO_R FOR STAFF_F OBJECTS

Objects	Bar_t	Chord_t	Accomp_t
$Staff1_{Cradle}$	0.735	0.728	0.716
$Staff2_{Iam\text{-}Late}$	0.618	0.789	0.721
$Staff3_{Garden}$	0.487	0.572	0.677

TABLE III: RUNTIME METRIC VALUES FROM ALGORITHM IMPLEMENTATION

Objects	COM_R	*Std Dev*	*Max*	CBO_R	*Std Dev*	*Max*
Staff_t	1.523	0.4769	1.622	0.824	0.4575	0.899
Note_t	0.242	0.0203	0.341	0.274	0.0345	0.347
Key_ Sig_t	1.355	0.4024	1.427	0.961	0.3953	0.978
Time_ Sig_t	1.475	0.1738	1.499	0.927	0.2149	0.956
Bar_t	1.274	0.3149	1.374	0.342	0.1783	0.421
Chord_t	1.682	0.2316	1.721	0.877	0.2465	0.917

As the objects shows more cohesion the number of clusters identified will be low, depicting an inverse proportion with the runtime cohesion metric. Hence, it is assessed that the runtime cohesion exhibits the behavior of either being together or not, at the object-level. Three objects for *staff_t* is chosen and its runtime coupling value is calculated using equation (5). The object level coupling of the objects derived from a single class show different behavior in its invocation of methods of objects of other classes (Note_t, Key_Sig_t, Time_Sig_t, Bar_t, Chord_t).

C. Behavioral Complexity of Objects

These runtime cohesion and runtime coupling metrics values are able to project the runtime dynamism exhibited in the considered object-relational systems.

The runtime cohesion and coupling metric values and its interpretation shows us that the objects exhibits dynamic behavior in the aspects of interacting with own instance variables (runtime cohesion) and invoking of other objects (runtime coupling).

The validation of these runtime metrics is needed to test whether the metrics measures what it intends to measure, the runtime behavior of the objects. The validation is carried out by testing the following hypothesis.

H0: The dynamic metrics does not measure the behavior of the objects at runtime.

H1: The dynamic metrics do measure the behavior of the objects at runtime.

Since the study is on the behavior of the object-relational databases, a set of objects are collected from the object-relational schema's runtime environment.

The runtime measures of the coupling and cohesion determines the objects' runtime behaviour based on this calculated coefficient of variance (*Cv*). *Cv* measures the relative scatter in data with respect to the mean and is calculated as given in the formula Cv = (σ / μ) * 100.

The *Cv* is directly proportional to the degree of data scatter. If the Cv value for all the classes is zero then it leads to accept the null hypothesis. Else if there exists a class with Cv>0, then it is inferred that the objects show different behavior during runtime and leads to the rejection of H0 and accept H1.

D. Analysis of the Behavior

Behavior of objects is assessed by the clusters (NC) formed for all the objects created is determined using correlation cluster analysis [17]. The correlation between two metric values is calculated. If the correlation is high between them, then these two objects from the first cluster. Consequently, those objects with high correlation are added to this cluster. Additional clusters are also formed as explained above. Karl Pearson's coefficient of correlation (Correl) is used to find the correlation between two values.

If the Correl value is greater than 0, and tends to 1, then those pair of objects can be accommodated in any of the clusters, cohesion cluster or coupling cluster. If the Correl value is zero, then that object is said to stand alone, because it cannot be paired with any of the object clusters. This approach helps to find the number of clusters in runtime in the collection set of objects metric values, cohesion and coupling.

E. Experimental Results

The experiments we conducted on the runtime cohesion and coupling metrics showed some interesting results which are shown in Table IV and Table V. Obviously, the experimental results could be interpreted to exhibit dynamic behavior of the objects during runtime. The higher runtime cohesion and coupling values are taken, the mean of all the metric values are given. The scatter of the metrics values is calculated from the Cv value.

Table IV: Clusters From $Cohesion_R$

Objects	$Cohesion_R$	μ	σ	Cv (%)	NC
staff_t	1.242	1.247	0.4211	33.77	1
note_t	1.268	1.215	0.7653	62.99	7
Key_ sig_t	0.427	0.6523	0.3212	49.24	3
Time_ sig_t	0.311	0.365	0.1328	36.38	2
Bar_t	1.623	1.676	0.3623	21.62	1
Chord_t	1.755	1.711	0.2276	13.30	4
Total	-	-	-	-	18

Table V: Clusters From $Coupling_R$

Objects	$Coupling_R$	μ	σ	Cv (%)	NC
staff_t	0.365	0.313	0.212	67.73	2
note_t	0.652	0.674	0.211	31.31	6
Key_ sig_t	0.862	0.722	0.372	51.52	3
Time_ sig_t	0.814	0.862	0.492	57.08	2
Bar_t	0.632	0.619	0.263	42.49	4
Chord_t	0.684	0.627	0.541	86.28	6
Total	-	-	-	-	23

Also the number of clusters formed for each of the objects of the class_types are calculated, which ultimately shows that the objects show dynamic behavior. The Cv and the NC values gives an indication that the objects exhibits behavioral complexity in runtime, leading to accept the hypothesis H1.

The runtime behavior of the objects in terms of object fetcing and pre-fetching has shown significant results. The objects of *staff_t, bar_t*, and *chord_t* are considered in this study. The objects fetching from the read operations are shown in Fig. 3 and Fig. 4.

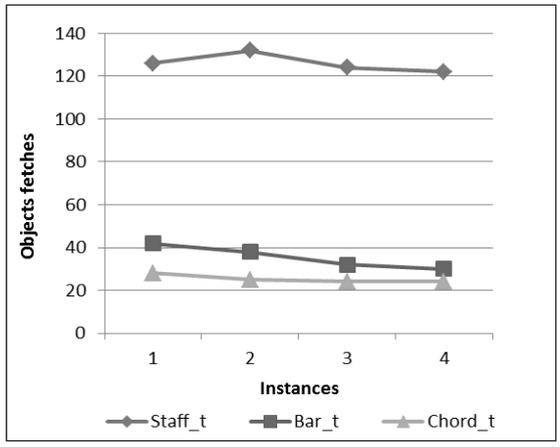

Fig. 3: Object Behavior in Fetches Due to Cohesionr

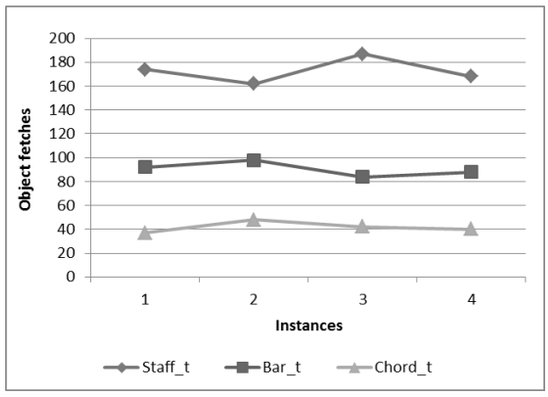

Fig. 4: Object Behavior in Fetches Due to Couplingr

The same music score is executed for four instances, and objects produced in these instances are pre-fetched from the disk. Initially, the object fetches for the objects were high, and there is a slow down in the number of object fetches. This is due to the togetherness (cohesiveness of the objects) and the connectedness (coupling between objects). Pre-fetching of relevant objects with minimal read operation is due to optimal cohesion and coupling that is exihibited during runtime. When the disk read operation becomes minimal, then the performance of the system will improve in terms of response time

IV. Conclusion

In this paper, we have derived and proposed algorithms for estimating the runtime cohesion and coupling metric values in object-relational data modelling. The runtime instances of the objects and the object-relational schemas were considered as the implementation data-set for the two proposed algorithms. The experiments conducted using these algorithms were helpful in assessing the behavior of objects in runtime. The sample music data-set used was simple, but covered a wide range of objects that were quite well answered our research hypothesis with significant results. The behavior of these objects is advantageous in evaluating and monitoring the performance of the object-relational system [18]. We are working towards the implementation of an object based storage mechanism in finding better solution in optimizing the performance of the objects. The two algorithms for evaluating the runtime metrics cohesion and coupling, proposed in this paper, can well be beneficial in studying the object behavior and performance of the systems.

References

1. A. Mitchell, and J. F. Power, "Toward a definition of run-time object-oriented metrics," *In 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, Darmstadt, Germany, 2003.
2. A. Mitchell, and J. F. Power, *"Run-time cohesion metrics: An empirical investigation,"* Technical Report, Department of Computer Science, NUI Maynooth, Ireland, 2004.
3. S. Justus, and K. Iyakutti, "An empirical validation of the suite of metrics for object-relational data modeling," *International Journal of Intelligent Information and Database Systems*, vol. 5, no. 1, pp. 49-80, 2011.
4. A. L. Baroni, Formal definition of object-oriented design metrics, Master Thesis, Vrije Universiteit Brussel, Belgium, 2002.
5. R. S. Pressman, *"Software Engineering: A Practitioners' Approach,"* (6thed.). McGraw-Hill International Edition, 2010.
6. Amandeep et. al., "Class cohesion metrics in object oriented system," *International Journal of Software and Web Sciences,* vol 3, no. 2, pp. 78-82, December, 2013.
7. A. Mitchell, and J. F. Power, "An approach to quantifying the run-time behaviour of java gui applications," *International Symposium on Information and Communication Technologies*, Cancun, Mexico, 2004.
8. V. Gupta, and J. K. Chhabra, "Package level cohesion measurement in object-oriented software," *Journal of Brazilian Computer Society,* vol. 18, no. 3, pp. 251-266, 2012.
9. S. A. Ebad, and M. A. Ahmed, "Review and evaluation of cohesion and coupling metrics at package and subsystem level," *Journal of Software*, vol. 11, no. 6, pp. 598-605, June 2016.
10. J. K. Chhabra, and V. Gupta, "A survey of dynamic software metrics," *Journal of Computer Science Technology,* vol. 25, no. 5, pp. 1016-1029, 2010.
11. G. Rani, and P. Singh, "Dynamic coupling metrics for object oriented software systems- A survey," *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 2, pp. 1-8, March 2014.
12. S. Yadav, S. Sikka, and U. Shrivastava, "A review of object-oriented coupling and cohesion metrics," *International Journal of Computer Science Trends and Technology*, vol. 2, no. 5, pp. 101-108, 2014.
13. N. Rajkumar, C. Viji, and S. Duraisamy, "Measuring cohesion and coupling in object oriented system using java reflection," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 7, pp. 3096-3101, 2015.
14. M. Ahmed, A. Abubakar, & J. AlGhamdi, "A study on the uncertainty inherent in class cohesion measurements," *Journal of Systems Architecture Embedded Systems Design*, vol. 57, no. 4, 474-484, 2011.
15. S. R. Chidamber, and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.
16. B. Újházi, R. Ferenc, D. Poshyvanyk, and T. Gyimothy, "New conceptual coupling and cohesion metrics for object-oriented systems," *10th IEEE International Working Conference on Source Code Analysis and Manipulation*, pp. 33-42, 2010.
17. D. Arora, P. Khanna, A. Tripathi, S. Sharma, and S. Shukla, "Software quality estimation through object oriented design metrics," *International Journal of Computer Science and Network Security*, vol. 11, no. 4, pp. 100-104, 2011.
18. S. Justus, and K. Iyakutti, "Object relational database metrics: Classified and evaluated," *International*

Workshop on Software Engineering, Potsdam, Germany, pp. 119-131, 2006.

19. A. K. Jakhar, and K. Rajnish, Measurement of complexity and comprehension of a program through a cognitive approach," *International Journal of Engineering Transaction B: Applications*, vol. 28, no. 11, pp. 1579-1588, November 2015.
20. T. R. Reddy, B. V. Vardhan, and P. V. Reddy, A document weighted approach for gender and age prediction based on term weight measure, *International Journal of Engineering, Transactions B: Applications,* vol. 30, no. 5, pp. 643-651, May 2017.