Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Dynamic Coupling Based Performance Analysis of Object Oriented Systems

Brij Mohan Goel
Assistant Professor,
Vaish College of Engineering, Rohtak, India

Satinder Bal Gupta
Associate Professor,
Indira Gandhi University, Meerpur, Rewari, India

Abstract: This work is based on dynamic performance study. The DCBO (Dynamic Coupling Between Objects) that is known to be run time version of CBO (Coupling Between Objects) is selected for the study. DCBO is a dynamic coupling metric which can be used to measure coupling between objects at run time. This work focuses on empirically investigating DCBO at class, method and message levels using a set of real world applications. A scenario based approach in combination with appropriate statistical techniques has been used in an attempt to capture the correlation between static and dynamic coupling metrics and, in turn, the structural and performance aspects of object oriented software system.

Keywords: Dynamic Coupling, Object Oriented Metrics, Performance, Use Case

1. INTRODUCTION

Theoretically every system is a stand-alone unit, but in reality systems may depend on each other as either they require services form other systems or provide services to other systems. Thus, coupling between systems cannot be completely avoided but can only be controlled. Coupling between systems is an important factor that affects external quality attributes of software e.g. understandability, maintainability, reusability etc. But still, only a few quantitative studies of the actual use of coupling have been conducted at the system level. Coupling is one of the internal attributes which is considered important due to its profound repercussions for external software quality attributes. In the past decade, a vast number of object oriented coupling metrics have been proposed and validated. But most of them are static in nature and do not take into account the runtime behavior of object oriented software systems. A lesser emphasis has been placed on designing dynamic metrics. Of those proposed so far, only few have been validated using real world applications inhibiting their use in the software industry.

2. OBJECTIVES

- To assess the dynamic performance of object oriented software system by carrying out an empirical validation using real world applications.
- To exploit use case based approach to study the run time performance of real world applications.

3. REVIEW OF LITERATURE

Mitchell and Power [10] proposed a set of metrics that are designed to be applied to a software application at run time. These new metrics seek to quantify coupling at different layers of granularity that is at class-class and object-class level. The first three metrics (DCBO, Degree of dynamic coupling between two class and degree of dynamic coupling within a given set of classes) are based on CK's CBO metric. These metrics are designed to evaluate run time class coupling. The rest four metrics (RI, RE,RDI, RDE) measure degree of import and export coupling at object level. Mitchell and Power conducted a number of studies on the quantification of a variety of run-time class-level coupling metrics for object-oriented programs. They used statistical analysis to investigate the differences in the underlying dimensions of coupling captured by static versus the run-time coupling metrics. The results indicated that the run-time metrics did capture different properties than the static metrics alone. So, it is worthwhile to continue the investigation into run-time coupling metrics and their relationship with external quality, as extra information can be provided by the runtime metrics to complement that obtainable from a static analysis. The study was carried out on a set of java programs from JOlden Benchmark Suite, SPECjvm98Benchmark Suite and some real world programs.

Yourdon and Constantine [1] defined coupling as an abstract concept which is a measure of degree of interdependence among modules. They defined it as a measure of strength of interconnections. Myers [2] in his work defined six distinct levels of coupling to measure interdependence among modules. Page-Jones [3] further defined a set of principles which he adapted from Yourdon and Constantine's structured design. These principles defined the kind of connections that are desired in modules. He refined coupling levels defined by Myers by ordering them from high to low level of coupling based on maintainability and reusability of the coupled modules. Offutt et al. [4] extended the coupling levels of Yourdon et al. and Myersto reflect the features of modern programming languages. These coupling levels offer a very fine grain resolution measures. They also described algorithms to automatically measure the coupling level between each pair of units in a program. The parameters are classified by the way they are used for each coupling level. Uses are classified into computation uses (C-uses), predicate uses (P-uses) and indirect uses (I-uses). A C-use occurs when a variable is used in an assignment statement, in an output statement, or a procedure call. A P-use occurs when a variable is used in a predicate statement. An I-use occurs whenever a variable is a C-use that affects some predicates in the module. The I-use is considered to be in the predicate rather than in the assignment.

Hassoun et al. [5] proposed Dynamic Coupling Metric (DCM) for object level coupling which takes program execution into account. DCM can be used to measure the coupling of particular objects and/or the entire system at runtime. The DCM can also be used to predict the runtime complexity of the system. The measure can be used to compare systems built on meta level architectures with systems having no reflective features yet, at the same time, exhibiting the same interface. Their results state that classes with high object couplings need more attention and care and consequently induce higher maintenance cost.

Zaidman and Demeyer [6] proposed a metric Class Request for Service (CQFS) metric to analyze event traces of large scale industrial application. CQFS registers every message that the instantiations of a certain class sends during the execution of the program. This metric can abstract duplicate instantiations from the same class but a downside is that different objects that react differently due to the polymorphic behavior also get abstracted. They based their results on the fact that classes which have more than average responsibilities have a greater coupling compared to other classes.

4. METHODOLOGY

4.1 Scenario based Applications

Scenarios represent a series of actions that are associated together. Scenarios are like short stories about users describing how they use the system [7]. Scenarios may be defined as path of interaction which may occur among use cases. A use case defines a goal-oriented set of interactions between external users and the system under consideration or development. Use cases are used to capture functional requirements in the object-oriented software design [8]. A Use Case Scenario is a description that illustrates, step by step, how a user is intending to use a system, essentially capturing the system performance from the user's point of view. A use case scenario can include stories, examples, and drawings. Use cases are extremely useful for describing the problem domain in unambiguous terms and for communicating with the potential users of a system [8]. It describes an interaction between a user and a system that produces some useful outcome. For example: If Use case = Do assignment then Scenario=My own way to do assignment [9]. Each interaction usually starts with an initial stimulus from a user (also called an actor) and proceeds through a series of stimuli and responses between the actor, the system, and possibly other actors, until the interaction reaches its logical conclusion.

A list of scenario based applications used in this work is given below:

i. Library Management System
ii. Vehicle Management System
iii. Student Management System
iv. Anti Chess
v. Task Blocks
vi. Ekit

4.1.1 Scenario based Metric Selection

Once the sample applications were selected, the next step was to collect metric data. As selected sample application included scenario based applications, a set of major use case scenarios were considered for each application so as to provide effective coverage of each application. Dynamic CBO value was evaluated for each use case scenario in an application. To test various dimensions of dynamic CBO, min, max and average values for DCBO was taken for every class over all scenarios.

4.2 Non-Scenario based Applications

Non–scenario based applications include applications which do not require any user input. These applications may include Non-GUI applications which operate from command line only. This category of applications are considered in this work to investigate how different an input based application behave from non-input based application and does this difference is reflected on the coupling behaviour of the classes of that application. Non-scenario based application used for experimentation included the following:

- JavaCalendarPanel
- SPECjvm2008
- DragonConsole

4.3 Tools Used

4.3.1 DynaMetrics

In order to collect static and dynamic measures a value, a tool was required which can fetch sample java application and execute them to provide both static and dynamic metric values. For this purpose, we chose DynaMetrics [11] as metric collection tool. DynaMetrics is a metric evaluation tool that calculates the run time metric values for Java programs and also relates them to external quality attributes. It supports only java based applications. DynaMetrics works over data collected at runtime to evaluate the static and dynamic metric values. DynaMetrics operates on event log files created at runtime to get the runtime information about particular java program. DynaMetrics sits between the JVM (Java Virtual Machine) and the Java code to intercept and collect the useful information. The advantage of DynaMetrics is that it has user friendly GUI. It is compatible with all the platforms that support Java 2 or higher runtime environment (Windows 9x/2000/XP/7, UNIX, Linux). It can evaluate the class-level, object-level and method-level static and runtime metrics for Java programs. DynaMetrics has been implemented using Eclipse 3.2 and NetBeans IDE5.5. All the components except GUI has been implemented in Eclipse 3.2 while GUI is implemented using Net Beans IDE 5.5.

4.3.2 Statistical and Presentational System Software (SPSS)

SPSS Statistics [53] is a software package used for statistical analysis. SPSS is an acronym for Statistical and Presentational System Software. (Previously it stood for Statistical Package for Social Scientists). It is now officially named "IBM SPSS Statistics“. SPSS is a powerful, feature-rich, statistical analysis program. SPSS Statistics can take data from almost any type of file and use them to generate tabulated reports, charts, and plots of distributions and trends, descriptive statistics, and complex statistical analyses due to which SPSS is a powerful, feature-rich, statistical analysis program. It is used in a wide variety of disciplines.

It is a fairly comprehensive program and a researcher is able to perform 95%+ of his analyses with it. It encompasses a wide range of statistical techniques from basic descriptive statistics through to regression analysis and graphics. Statistics analysis tasks that can be performed with base package include Descriptive statistics, Bi-variate statistics, Prediction for numerical outcomes and prediction for identifying groups: Cross tabulation, Frequencies, Descriptive, Explore, and Descriptive Ratio Statistics. The main advantage of SPSS Statistics is that many important features are accessible directly via pull-down menus. The graphical user interface of SPSS included two views: Data View and Variable View. The Data View is used for viewing and editing the raw data of a dataset. This takes the same form as standard spreadsheet software, like Microsoft Excel, where each column represents a variable and each row represents a case. The Variable View is for viewing the metadata for a dataset's variables. The Variable View for a dataset looks very similar to its Data View however the columns (variables) are transposed so that each row contains the metadata for the specified variable. Different versions of SPSS are available. Early versions were designed for batch processing. SPSS statistics 16.0 and later can run on Windows, Mac and Linux. The GUI is written in java. In addition, beneath the menus and dialog boxes, SPSS Statistics uses a command language. Some extended features of the system can be accessed only via command syntax. In our analysis, we have used SPSS Statistics 19.0 version base options for descriptive statistics.

5. ANALYSIS TECHNIQUES

5.1 Descriptive Statistics

For each sample application, mean and standard deviation of static CBO and DCBO is calculated. The fundamental reason behind determining descriptive statistics was to determine whether these measures show enough variance to qualify for further analysis.

5.2 Correlation

To further investigate how the static CBO and DCBO are related to each other, a correlation study was conducted using Pearson Product Moment Correlation Test. Pearson's coefficient r is used to determine the strength and direction of correlation. It can take value between -1 to +1 through 0. A positive value of r indicates that value of one variable increases with another. The statistical results obtained suffer from a limitation that they may occur by chance. So, each statistical result is associated with a significance level called p-value. p-value is the probability which is used to measure significance of empirical analysis. The correlation coefficient is considered statistically significant with p-value < 0.05.The sample applications for which p-value was found to be greater than 0.05 were discarded as they do not qualify for further investigation. All the static and dynamic CBO metric values were collected at class, method and message levels for each project. Dynamic CBO for scenario based application is evaluated by exercising three variants of DCBO which are defined below:

- DCBOmax: represents maximum value of each class for all use case scenarios.
- DCBOmin: represents minimum value of each class for all use case scenarios.
- DCBOavg: represents average value of each class for all use case scenarios

The correlation among static CBO and DCBO was also carried out for each project at class, method and message level where dynamic CBO has been presented using its three variants DCBOmax, DCBOmin and DCBOavg.

5.3 Principal Component Analysis (PCA)

PCA test was conducted to analyse the covariate structure of selected and to identify the structural dimensions captured by these metrics. PCA can tell if all the metrics are likely to be capturing the same class property. A number of principal components are generated; however, the number of principal components to be retained is decided using Kaiser criteria**.** The main aim behind conducting this test was to identify whether all the three representative used for measuring dynamic CBO were measuring distinct information or are they simply redundant. The second reason was to affirm that dynamic coupling metric capture additional information as compared to their static counterparts.

6. RESULT SUMMARY

From the correlation study of static and dynamic CBO for sample projects, only the projects which have significance level<=0.05 have been considered valid. The following correlation results have been drawn at various levels for all projects.

At class level

- Two projects exhibit strong correlation between static CBO and DCBOmax.
- Four projects exhibit moderate correlation between CBO and DCBOmax.
- One project exhibit weak correlation between static CBO and DCBOmax.
- Two projects exhibit strong correlation between static CBO and DCBOmin.
- Two projects exhibit moderate correlation between CBO and DCBOmin.
- One project exhibit weak correlation between static CBO and DCBOmin.
- Three projects exhibit strong correlation between static CBO and DCBOavg.
- Two projects exhibit moderate correlation a between static CBO and DCBOavg.
- One project exhibit weak correlation between static CBO and DCBOavg.

At method level

- Two projects exhibit very strong correlation between static CBO and DCBOmax.
- Two projects exhibit strong correlation between static CBO and DCBOmax.
- One project exhibit moderate correlation between static CBO and DCBOmax.
- Two projects exhibit strong correlation between static CBO and DCBOmin.

- Two project exhibit moderate correlation between static CBO and DCBOmin.
- Two projects exhibit very strong correlation between static CBO and DCBOavg.
- Two projects exhibit strong correlation between static CBO and DCBOavg.
- One project exhibit moderate correlation between static CBO and DCBOavg.

At message level

- Two projects exhibit strong correlation between static CBO and DCBOmax.
- Three projects exhibit moderate correlation between static CBO and DCBOmax.
- Two projects exhibit strong correlation between static CBO and DCBOmin.
- Two projects exhibit moderate correlation between static CBO and DCBOmin.
- One project exhibit weak correlation between static CBO and DCBOmin.
- Two projects exhibit strong correlation between static CBO and DCBOavg.
- Two projects exhibit moderate correlation between static CBO and DCBOavg.
- One project exhibit weak correlation between static CBO and DCBOavg.

7. CONCLUSIONS

This work focuses on empirically investigating Dynamic Coupling Between Objects at class, method and message levels using a set of real world applications. A scenario based approach in combination with appropriate statistical techniques has been used in an attempt to capture the correlation between static and dynamic coupling metrics. PCA test was conducted to analyse the covariate structure of selected and to identify the structural dimensions captured by these metrics. At class level, four projects, at method level, two projects and at message level three projects exhibit moderate, strong and moderate correlation respectively between CBO and DCBOmax. Similarly, other combinations of correlation is shown by the analysis at the three levels.

8. REFERENCES

[1] L.L Constantine and E. Yourdon, "Structured Design", Prentice-Hall,Englewood Cliffs, New Jersey, USA, 1979.
[2] G. Myers, "Reliable Software Through Composite Design", Mason andLipscomb Publishers, New York, USA, 1974.
[3] M. Page-Jones, "The Practical Guide to Structured Systems Design", YourdonPress, New York, NY, 1980.
[4] A.J. Offutt, M.J. Harrold, and P. Kolte, "A software metrics system formodule coupling",The Journal of Systems and Software, Vol. 20, No.3, pp.295-308, 1993.
[5] Y. Hassoun, R. Johnson and S. Counsell, "A Dynamic Runtime CouplingMetric for Meta Level Architectures", In Proceedings of Eighth EuromicroWorking Conference on Software Maintenance and Reengineering(CSMR '04), pp. 339, 2004.
[6] A. Zaidman and S. Demeyer, "Analyzing large event traces with the help ofcoupling metrics", In Proceedings of the 4th International Workshop on OOReengineering, University Antwerpen, 2004.
[7] Scenarios:http://undergraduate.csse.uwa.edu.au/units/CITS2220/lecturenotes/lec02.usecases.pdf
[8] G. Schneider, J.P. Winters, I. Jacobson, "Applying Use Cases: A practicalGuide", Pearson Education, pp 272, 2001.
[9] Use case vs Scenario: http://www.cs.cityu.edu.hk/~wkchan/content/use-casevs-scenario.
[10] A. Mitchell and J.F. Power, "Runtime Coupling Metrics for the Analysis ofJava Programs – preliminary results from SPEC and Grand suites", Dept. ofComputer Science, National University of Ireland, Maynooth, Co. Kildare,Ireland, Technical Report NUIMCS- TR2003-07, 2003.
[11] P. Singh and H. Singh, "DynaMetrics: A Runtime Metric-Based Analysis Tool for Object-Oriented Software Systems", SIGSOFT Software Engineering Notes, Volume 33, Number 6, 2008.

