Chapter 1

INTRODUCTION

1.1 Background

Measuring the quality of object-oriented software aimed at developing a high quality software product has been one of the most interesting areas of research in the field of software engineering. Software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving software quality [Basili et al.(1996), Briand(1999)]. Traditional metrics used for measuring the quality of software, such as Lines of Code (LOC) [Conte et al.(1986)], have been found to be inadequate for analyzing the quality of object-oriented software systems [Fenton and Neil(1999)].


Most of the available metrics are evaluated from the static source code-based analysis and are thus called static metrics. But static metrics are insufficient in evaluating the dynamic behavior of a software application [Mitchell and Power(2003c)]. This is because the behavior of a software application is not only influenced by the complexity but also by the operational or runtime environment of the source code [Mitchell and Power(2003c)]. With the introduction of dynamic analysis for measuring the quality of
Design and Validation of Dynamic Metrics for Object-Oriented Software Systems

object oriented software, the focus has shifted to dynamic/runtime metrics because of their ability to record the actual behavior of a software because these metrics are evaluated from the data collected at runtime. A latest research [Yacoub et al.(1999)] has indicated that useful information may be obtained from a measure of quantifying the dynamic complexity of software in its operational environment. Object-oriented software have been shown to behave differently in static and dynamic analysis [Mitchell and Power(2003b)], using CK Metrics like CBO, LCOM [Chidamber and Kemerer(1991&1994)].

There are a number of internal quality attributes, such as coupling, cohesion, size, complexity etc. of a software, that are used to predict the external quality attributes of a software. Coupling has been defined as one of the most basic qualitative measures for measuring the performance of a software at design or implementation phase [Briand et al.(1997)]. Coupling was originally defined as the degree of interdependency between modules [Yourdon and Constantine(1979)]. It characterizes the usage dependencies among classes in object-oriented systems [Briand et al.(1999b)]. It has been found to have a direct impact on the external software quality attributes [Basili et al.(1996), Briand et al.(1997)]. A quality object-oriented program must have a low-level of coupling present among the classes. Coad and Yourdan(1991) classified coupling into following two types:

- Interaction Coupling: It results from the exchange of messages between two classes.
- Inheritance Coupling: It results from the interconnection between generalizations and specializations.

A large number of metrics have been developed for measuring static interaction-based coupling or simply coupling [Chidamber and Kemerer(1991&1994), Churcher and Shepperd(1995), Li and Henry(1993), Abreu et al.(1995), Briand et al.(1997), Lee et al.(1995)] such as Coupling Between Objects (CBO), Afferent Coupling (Ca), Coupling Factor (CF) etc.

Over the past few years, a number of dynamic metrics [Mitchell and Power(2003c &2004b), Zaidman and Demeyer(2004), Arisholm(2002), Yacoub et al.(1999), Hassoun et al.(2004)] have been introduced to the software engineering field that measure the object-oriented attributes such as coupling, cohesion, complexity etc. Most of the dynamic metrics proposed till date measure coupling in one way or the other way. Still in their early phases, most of these metrics need to be empirically validated to be able to become a reliable mean of software performance measurement. Also their comparison and
collaboration with the static metrics is a vital and promising area to be explored for accurately measuring the quality of object-oriented software.

Inheritance is a key feature of the OO paradigm. It is a mechanism whereby one object acquires characteristics from one or more other objects [Pressman(1997)]. The software engineering community has been preoccupied with inheritance and its effect on the software quality [Emam(2001)]. The effect of inheritance on the various external quality attributes such as fault-proneness, understandability, maintainability etc has been empirically proven [Daly et al.(1996), Basili et al.(1996), Chidamber and Kemerer(1991)]. For example, the reuse of software through inheritance is claimed to produce more maintainable, understandable and reliable software [Basili et al.(1996)].

One of the problems faced by runtime coupling is measuring the impact of inheritance on coupling [Mitchell and Power (2003b)]. Inheritance has always been covered as a separate concept with its own metrics. Metrics such as Number of Children (NOC) [Chidamber and Kemerer(1991)], Depth of Inheritance Tree (DIT) [Chidamber and Kemerer(1991)], Method Inheritance Factor (MIF) [Harrison et al.(1998)], IH-ICP [Lee et al.(1995)], Depth of Inheritance Tree of a Class (DITC) [Rajnish and Bhattacherjee(2006a)] etc. measure static inheritance-based coupling. However, most metrics for size, coupling and cohesion within the object oriented area ignore inheritance [Beyer et al.(2001)]. Research has also been conducted regarding class inheritance metrics by Rajnish and Bhattacherjee in [Rajnish and Bhattacherjee(2005,2006a,2006b&2007)]. But there have been no metrics known till date that measure the impact of inheritance on coupling at runtime.

1.2 Motivations

Motivations for this research have been derived from a number of sources. We list some of the important and broadly viewed motivational factors below:

a) According to the previous research [Mitchell and Power(2003c), Arisholm(2002)], dynamic analysis provides the actual behavioral information about an object-oriented software as it is carried out at runtime. Thus the dynamic metrics (using dynamic analysis) are able to evaluate the actual behavioral metric values for a software as compared to the expected values evaluated by the static metrics (using static analysis). But on the other hand, static analysis has its own advantages such as lesser evaluation effort required than the dynamic analysis. So an important
motivation is to analyze whether static and dynamic metrics could work together to understand the behavioral aspects of software like never before and hence to improve upon those aspects.

b) There have been just a numbered dynamic metrics available to stand with/against the much famous and dense jungle of static metrics. There is definitely a huge need to design and develop new dynamic metrics in the years to come to measure the quality of object-oriented software accurately. This need in itself acts as another motivation for this work.

c) A lot of already proposed dynamic metrics have been applied to just a few sample application programs. There is a need of more metric validation work to be carried out using real-world industrial applications to make them capable for commercial use in future.

d) There have been no dynamic metrics available that measures the impact of inheritance on coupling at runtime [Mitchell and Power(2003b)]. Also there are quite a limited number of dynamic metrics available for features such as polymorphism and effect of constructors.

e) Most of dynamic metrics available till date measure coupling. But all of these metrics are either class-level or object-level. There is a need to measure coupling at method-level to get a clearer behavioral picture at runtime as a method is the core of a software’s functionality.

f) There is a need to develop a complete set of dynamic metrics related to specific external quality attributes.

1.2.1 Related Research-based Motivations

The above mentioned motivations were drawn from the conclusions and future recommendations quoted by various researchers found from a thorough literature survey conducted for this research. This section lists such important conclusions and future recommendations from the past research.

1.2.1.1 Motivating Conclusions from Related Work

a) Dynamic metrics can be used to measure the actual runtime properties of an application as compared to the expected properties measured by static metrics [Yacoub et al.(1999)].
b) Two phases are important for establishing a metric for dynamic quality. The first is defining metrics differentiating them from existing metrics, and identifying the need for these metrics. The second phase is using empirical studies to validate the metric [Yacoub et al. (1999)].

c) A software design can be evaluated in terms of both its internal and external complexity and previous research has shown that the static coupling metrics provide a good indication of external complexity of a design. For this reason a number of dynamic metrics were proposed which may provide an important supplement to the existing static metrics [Mitchell and Power (2003b)].

d) Both static and dynamic metrics can give different indications of the levels of coupling present in a class. The reason for the different results obtained from the static and dynamic analysis may arise from the fact that static metrics are concerned with statically coupled and complex design elements whereas dynamic metrics are concerned with frequently invoked and frequently executed elements [Mitchell and Power (2003b), Yacoub et al. (1999)].

e) Dynamic class-level metrics could be used to evaluate external quality aspects of a design by measuring the actual runtime properties of a class [Mitchell and Power (2003b)].

f) Some sort of correlation may exist between the information obtained from static and dynamic analysis as static metrics evaluate the quality of a class at the code level whereas dynamic metrics quantify the situation when these classes are executed at runtime [Mitchell and Power (2003b)].

g) It is worthwhile to continue the investigation into runtime coupling metrics and their relationship with the external attributes [Mitchell and Power (2004a)].

h) The objects from the same class can behave differently at the runtime from the point of view of coupling. Such behavior is not identifiable from a simple static analysis of the source code giving merit to the further investigation into runtime metrics and their applications [Mitchell and Power (2005)].

i) It is better to abstract the metric up to the class-level instead of remaining at the object-level [Zaidman and Demeyer (2004)].

1.2.1.2 Motivating Future Recommendations from Related Research

a) Empirically validate the proposed metrics and their correlation with design quality attributes [Yacoub et al. (1999)].
b) Explore the dependencies/correlation between static and dynamic metrics [Yacoub et al. (1999)].

c) Investigate the prediction ability of hybrid models combining static and dynamic measures [Arisholm(2002), Mitchell and Power(2005)].

d) Investigate the impact of inheritance on dynamic coupling metrics [Mitchell and Power(2003b)].

e) Investigate the usefulness of dynamic measures in testing [Mitchell and Power(2003b)].

f) Developing a comprehensive set of runtime object-oriented metrics that can intuitively quantify some vital aspects of object-oriented applications such as inheritance, dynamic binding and polymorphism [Mitchell and Power(2004b)].

g) Develop a method to verify whether the classes which were considered important (/active) at design time really come out to be as important at runtime [Zaidman and Demeyer(2004)].

h) Investigate the possibility of designing dynamic versions of the classic static software metrics such as coupling and cohesion metrics [Dufour et al. (2003b)].

i) Apply dynamic metrics to the real world applications written in different object-oriented programming languages such as Java, C++ etc [Hassoun et al. (2004)].

1.3 Problem Description - Path to the Problem

Software metrics play a major role in measuring the performance of object-oriented software. Most of the software metrics available today analyze a software’s performance before runtime i.e. statically. Static software metrics can only measure the expected behavior of software in comparison to the dynamic metrics that can measure the actual behavior. This is due to the fact that static metrics are evaluated from the information collected from design code prior to program execution, whereas dynamic metrics are evaluated from the information collected at runtime. Coupling is one of the most important internal quality attributes that has been widely worked upon to get a measure of the external quality attributes of object-oriented software systems. Such external quality attributes are maintainability, reusability, testability, portability etc. A large set of static coupling metrics have been devised over the years to measure this ever evolving design attribute. But there are just a few dynamic metrics that are capable of studying the actual coupling behavior of a software at runtime. As the dynamic metrics can measure the actual
Figure 1: Problem description – Path to the problem
behavior of a software, it is important to trace and develop such metrics that can help to interpret dynamic analysis results for adjudging the quality of object-oriented software. There are two broad categories of coupling from the static context i.e. interaction coupling and inheritance coupling. There are no dynamic metrics available till date that can study the impact of inheritance on dynamic/runtime coupling, and just a few metrics for the dynamic interaction-based coupling. These few dynamic coupling metrics, already proposed, are still to be empirically validated. The study of any correlation existing between static and dynamic metrics also needs to be studied. Figure 1 shows the problem path diagrammatically.

1.4 Objectives

This research aims:

- To design and validate new dynamic metrics for object-oriented software systems, with a goal of more accurate software quality measurement.
- To empirically analyze the correlation, if any, between the static and dynamic metrics.
- To find the impact of inheritance on coupling at runtime.
- To study and statistically analyze already proposed dynamic metrics.

1.5 Thesis Outline

An outline of this thesis is presented below in the form of summary of all the chapters:

- **Chapter 1** describes the background, motivations and objectives of the thesis. It summarizes the problem addressed in this research with the help of a problem description diagram.

- **Chapter 2** covers the literature survey about software metrics. It provides an overview of the concept of software metric and its usage in measuring the performance of a software product. It also describes various measurement scales that are used for metrics. The relation of internal quality attributes to the external quality attributes is also studied in brief. It lists the types of software metrics. It is followed by the methodology to define and validate a software metric. Next it summarizes the related work on object-oriented software metrics that has been done over the years. It includes various object-oriented design concepts such as
coupling, cohesion, inheritance etc. The related work done in interaction-based coupling metrics and inheritance-based coupling metrics is also discussed. It further reviews the concept and the related work done on dynamic/runtime object-oriented software metrics. It includes the concept, types and desired properties of dynamic metrics. It was found that most of the dynamic metrics proposed till date measure coupling as coupling is most affected by method-method interaction resulting at runtime. The chapter also talks through the dynamic metric suites proposed by various researchers since the evolution of the concept of dynamic/runtime metrics for object-oriented systems. It was found that no metrics are devised to study the impact of inheritance on coupling at runtime.

- **Chapter 3** contains the description of proposed dynamic metric suite for object-oriented systems introduced in this research. The proposed metrics are mainly divided into two categories: dynamic coupling and dynamic inheritance metrics. Each metric description has four parts namely definition, literary basis, need and metric effect on quality attributes. Four external quality attributes were chosen to analyze the metric effects. These four attributes are maintainability, reusability, portability, testability. There are four dynamic class-level coupling metrics and five dynamic class-level inheritance metrics proposed in this work. There are three additional dynamic metrics that contain two method-level dynamic coupling metrics and one constructor-based automatic initialization metric. The chapter ends with the introduction of three additional proposed metrics. A table showing the effect of each metric on various external quality attributes concludes the chapter.

- **Chapter 4** explains the methodology followed to solve the research problem. This chapter initially focuses on the test cases considered for metric validation. Various modules of the test case application software Educational Institution Project (EIP) are discussed. EIP consists of four application programs that are used as test cases for this work. The test case description is followed by the stepwise methodology followed to validate the metrics. A new runtime metric-based analysis tool, called DynaMetrics [Singh and Singh(2008)], that was designed and developed in this research for static and dynamic metric evaluation, is briefly explained next. The methodology implementation and process flow is described using various modules of DynaMetrics. DynaMetrics was written in java using Eclipse 3.2 [EI] and NetBeans 5.5 [NB]. Finally the description of SPSS Statistics 17.0 tool [SPSS] that
is used for statistical analysis of metric data concludes the chapter. Next chapter contains the analysis of metric evaluation results, obtained from the test case execution, for validating the proposed metrics.

- **Chapter 5** covers the statistical validation of the proposed dynamic metric suite. The chapter is divided into two major sections: dynamic coupling metric results and dynamic inheritance metric results. Statistical analysis is divided into two parts: descriptive analysis and correlation study. Descriptive analysis is discussed for each test-case. Both the class-level and system-level metric values are analyzed for all the metrics. Normality of data is checked using P-P normality plots.

  Correlation study is conducted using *Pearson’s Correlation* or *Product Moment Correlation* for finding out any existing correlation between static and dynamic metric values. Correlation between each of the proposed dynamic coupling metrics and the Dynamic CBO (DCBO) coupling metric is also conducted. Similarly a correlation study is conducted among proposed dynamic inheritance metrics along with the Method Inheritance Factor (MIF) metric. System-level values of these inheritance metrics for each of the four test cases are also compared. The most important analysis in this chapter is the correlation study among the DCa metric and various dynamic inheritance metrics. This is done to trace out any existing relationship between coupling and inheritance at runtime. System-level metric values of each of the proposed dynamic inheritance metric are analyzed to assess the amount of inheritance utilized by the system classes as well as the effect of inheritance on coupling at runtime. Pie charts are used to clearly separate the percentage used inherited methods, unused inherited methods, used non-inherited methods and unused non-inherited methods from each other in order to analyze the impact of inheritance on dynamic coupling. Additional metrics were also statistically analyzed. The chapter concludes with a summary of the metric analysis-based conclusions. These conclusions are also divided into three categories: coupling, inheritance and coupling-inheritance.

- **Chapter 6** concludes the thesis work listing all the conclusions drawn from the research work. These conclusions are followed by carefully compiled future recommendations. The important contributions made by this research work have also been listed.
• **Appendix A** contains the data tables containing static and dynamic metric data results along with DynaMetrics metric-output snapshots for all the four software application test cases.

• **Appendix B** defines some additional proposed metrics that this work introduces as future recommendations. These metrics are in addition to the twelve proposed dynamic metrics evolved from this research.