International Advance Journal of Engineering, Science and Management (IAJESM)
ISSN -2393-8048, January-June 2018, Submitted in May 2018, iajesm2014@gmail.com

A Quality Suite for Rule-based Object Oriented Systems

Prabhat Verma, Harcourt Butler Technological Institute, Kanpur. India, pvluk@yahoo.com
Vibhash Yadav, Institute of Technology and Management, Gurgaon, India, vibhashds10@yahoo.com
Prof. Raghuraj Singh, Harcourt Butler Technological Institute, Kanpur. India, rscse@rediffmail.com
Deepak Kumar Verma, University Institute of Engineering and Technology, Chhatrapati Shahu Ji Maharaj University, Kanpur, deepak300572@gmail.com

ABSTRACT

Rules play an important role in today's software as business logic or domain knowledge, which is prone to change much more frequently as compared to core functionality of the software. It is well established that code for rules should be well separated from the core functionality in order to change them easily as and when the need arises. Secondly, rules are best represented in declarative form with some inference engine mechanism.

The purpose of this paper is to investigate how design characteristics impact the ability of a software program to undergo changes, specifically in terms of its maintainability. To evaluate a program's adaptability to modifications, we have expanded upon the ripple effect metric for rule-based object-oriented systems. The proposed specialized metrics measure the quality of the software system in terms of its ability to configure itself with change in business rules or domain knowledge

KEYWORD: Ripple Effect, Change Impact Analysis, Rule variables, Spread of rule variables.

1. INTRODUCTION

The importance of smart software applications is increasing day by day. Intelligent Search Engines, Intelligent Help-Desk, Online Hospital Management are but a few examples which are knowledge intensive in nature. Rule-based logic plays a crucial role in such applications along with core functionality. Such applications are increasingly dominating the software Industry as an emergence of the Service Oriented Architecture. Business logic can be viewed as a part of rule based Knowledge.

In current software Engineering Practice, Object Oriented Methodology is used to design the core functionality of software application. Knowledge or business logic about the domain is often implicit and tangled with the core functionality. This is a bed design approach as it makes difficult to reconfigure the software for incorporating change in rules. As a good design, firstly, the code for rules should be well separated from the core functionality in order to change them easily as and when the
need arises. Hence, it is desirable to design and implement them as separate rule class. Secondly, the rules themselves be represented in declarative form with some inference engine mechanism. It ensures dynamic addition and deletion of rules. These two requirements, when fulfilled, ensure quality software in terms of its adaptability with frequently changing rules.

2. EXISTING METHODOLOGIES

Ripple effect has been used to measure the stability of procedural as well as object oriented software after going through a modification. The ripple effect metric in object-oriented programming measures the extent to which a local modification to a method or class may impact other methods or classes. In essence, this metric enables maintainers to anticipate the consequences of any changes they may wish to implement before actually carrying them out. [1].

To determine the extent of change caused by a modification, the ripple effect metric currently considers the impact of altering a single variable on the program as a whole. It should be noted that this effect may not be limited to the immediate vicinity of the change, and may instead propagate to other areas of the program. [2].

To calculate ripple effect values, two forms of change propagation are utilized. [3].

- One type of change propagation used in calculating ripple effect values is intra-module change propagation. This occurs when a modification to one variable within function1 affects other variables within that same function, as shown in Figure 1 with the propagation between b, a, and c..

- Another type of change propagation used in calculating ripple effect values is inter-module change propagation. This occurs when a modification to one function affects another function within the program. (Fig. 1) e.g. propagation of c from function1 to function2.

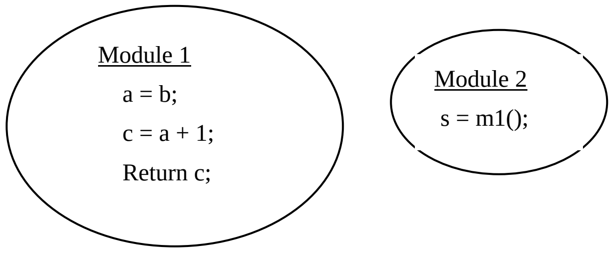

Fig. 1: Intra-modular and Inter-modular change propagations

The purpose of intra-module change propagation is to determine which variables within a module are impacted by the ripple effect resulting from a modification. This type of propagation involves analyzing assignment and definition use information. Assignment counts the propagation of changes from the right-hand side of an assignment to the left-hand side, while definition use counts the propagation of changes from the definition of a variable to its subsequent use. It is well understood from Fig. 2. The combination of information from assignment and definition-use pairings supply the required information for calculating intra-modular change propagation.

In inter-module change propagation, the flow of program changes is across module boundaries. All the affected modules as the consequence of the modified variable are to be considered.

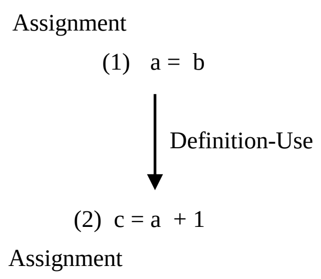

Fig. 2: Assignment and Definition-Use Pairing

3. RULE BASED OBJECT ORIENTED SYSTEMS

3.1 Separating the rule variables

Rule variables are those key variables in a program that participate in rules related to business logic or domain knowledge. They are more prone to be reconfigured than other variables as the business rules change. They may be scattered among several classes as per the definition of the rules. Thus, first task is to identify the rule variables and assign them weights on the basis of assessing the probability of their need to be reconfigured as change arises in business rules. These changes may be trivial as well as nontrivial.

3.2 Ripple effect measures for rule variables

To compute the ripple effect for rule based object oriented system, both intra-module and inter-module change propagations must be considered. Change is propagated from rule objects towards core functionality as rules change much more frequently. Hence, the origin of ripple is clearly defined in such systems. Intra-modular change is to be calculated within the rule module itself using the above described method. If the rules are represented in some declarative form with some inference mechanism in the rule module, they can be added/deleted/modified in runtime without any difficulty. This is the ideal case in which intra-module change propagation shall be zero for that rule object. If the rules are represented in imperative way, inter-module change propagation will carry some value. Hence, the ripple measure indicates how rules are represented in rule class (declarative or imperative).

3.3 Coupling Measure for rule objects

Next important issue is related to how the rule modules are coupled to the other modules that belong to core functionality. One can find three cases that are related to the type of coupling between rule based knowledge and object oriented core functionality. In the first case, rules are able to directly access the object and its attributes, which represents the most severe form of coupling, known as content coupling. The second case involves a form of tight coupling known as control coupling, where the activation of rule-based knowledge depends on the runtime properties of the object-oriented functionality. The most complex events involve conditional activation of rules, which are referred to as dynamic events. In the third case, a low level of coupling is present, specifically data coupling, where an object is passed as an argument for use in the rule object. [4].

A specialized metric, which is related to the Coupling measure of the rule modules with other affected core functionality modules in terms of its ability to reconfigure itself to the frequent changes in business rules, is proposed in this section. The proposed metric, '*spread of the rule variables*' measures how well separated are the rules from the core functionality in the object oriented software Lesser the value of the metric, more is the software configurable with respect to change in business rules.

By definition, the *spread*, for a single rule variable, is the ratio of the number of occurrences of the variable outside its own class to the total number of its occurrences in the program. The value of spread for a rule variable may lie between 0 and 1. In a bed design, if the rule variable is implemented inside the core functionality and not as a separate class, then the value of spread for this particular variable is 1 (case 1). On the other side of the spectrum, if a rule variable is well separated in its own class through some separation mechanism such as EJB (Enterprise Java Beans) or a lesser popular but more idealistic AOP(Aspect Oriented Programming), the value is 0 for that variable (Case 2). But, in most cases, the value of spread shall lie between 0 and 1 (Case 3). The value indicates how well separated is the rule object from the other objects comprising core functionality and by what mechanism. Lesser the value of the metric, better are the rules separated from core functionality and hence, indicates higher in software quality terms.

4. CONCLUSIONS AND FUTURE WORKS

The metrics mentioned above are specialized ones meant for Object Oriented Software which have plenty of rules that are prone to frequent changes e.g. with changing business scenario or as a result of competition in the market, one has to adapt rules because the rival has introduced a new offer in the market. Adaptability or changeability is the major indicator of quality in such software. Ripple effect measure and change Impact measure have been used for quality measure in procedural as well as Object oriented Software. The proposed specialized software are meant for rule intensive object oriented software and indicate the software quality in terms of its adaptability to changing rules in more specific and realistic way.

Most of the software metrics, available today, are based on the structural properties of the design or code, although they have been useful to a great extent, they do not consider the actual content or domain of the software system. The proposed metric is one step towards this direction. By identifying the frequently changing rule variables from other variables and assigning weights to them appropriately, we indirectly incorporate the domain knowledge in the metric for the quality measure purpose.

The authors of this paper intend to work on calibration and measuring process of the proposed metric in near future and apply the quality metric for some industry software.

REFERENCES

[1]. Nashat Mansour and Hani Salem, " Ripple effect in object oriented programs", Journal of Computational Methods in Science and Engineering, Volume 6, Supplement 1/ 2006.
[2]. Billal Haider, "Ripple effect : A Complexity Measure for Object Oriented Software", London South Bank University
[3]. Black, SE: Measuring Ripple Effect for Object Oriented Paradigm, IASTE International Conference on Software Engineering
[4]. M. D'Hondt. A survey of systems that integrate logic reasoning and object-oriented programming. Technical report, Vrije Universiteit Brussel, 2003.