Advances in Engineering Software 54 (2012) 30–37

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Information flow metrics analysis in object oriented programming and metrics validation process by RAA algorithm

Abdul Jabbar *, Subramani Sarala

Department of Information Technology, Bharathiar University, Coimbatore, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 26 April 2012
Received in revised form 2 July 2012
Accepted 12 August 2012
Available online 14 September 2012

Keywords:
Information flow metrics
Object oriented programming
Rule induction
Ant colony algorithm
Rule accuracy algorithm
Object oriented information flow

ABSTRACT

Transparent data flow metrics and control flow metrics had no main concern which to be handled by a compiler. Nowadays similar hardware and multithreaded coding is increased. Consequently, both data flow and control flow become more important in analyses the reusability and maintainability. The present analysis of source code and the ability of metrics are incompetent to predict the actual amount of information flow complexity in the modules. In this work, object oriented metric IF-C focuses on the improved information flow complexity estimation method, which is used to evaluate the data flows in object oriented source code and decrease the effort of maintainability and reusability. The object oriented information flow complexity incorporates various internal and external flows in object orientation. The adequacy of software metrics is validated by the rule accuracy algorithm which is based on rule induction technique. The technique applied in the software metrics dataset that has been selected using fitness proportionate selection algorithm. The competence and efficacy of the software metrics have verified by the predefined rules. The rules have if and then clause which hold metrics adequacy standards.

1. Introduction

As computer systems become gradually more interlinked with aspects of human life, promotion of software failures rise, thereby the need of software assurance research and practice are enhanced. With the assistance of metrics the developers recognize various resources and efforts needed. To sustain the software quality, software metrics have a major responsibility that includes maintainability, effort and productivity. Prognostic ability of software metrics is attractive, thus it is an important factor of the maintainability, productivity, quality and effort for cost estimation [2,4]. Information flow is the course of many information groups that are significant to one module or whole program. Information groups include input and output information. Fenton and Pfleeger [28] described the information flow contain both inter modular and intra modular attribute in the system. According to object oriented programming paradigm the concept is similar but it differs in structure. To quantify the information flow metrics from the object oriented programming source code is complex task. However in this work attempt to resolve this issue using improved information flow metrics [3] concept.

Object oriented paradigm the flows initiate in different ways. Primarily, principles of object oriented programming offer enhanced methodology to manage information. It contains various types of data blocks which contain statements that are allocating and managing information in the execution. Basically, the blocks commence several flows in the source code which is predictable from the source code designing. The fundamental concept of information flow occurs changes on information during execution. Accordingly, in this work focuses various forms of information accessed and retrieved between various blocks and within the blocks. The movement of information defines in the fan-in and fan-out framework. It counts the various flows initiate and end from a block. Programming paradigm information is managed in a storage units called variable. Variable reading, writing and executing in the code are called information manipulation. OOP paradigm object provide enhanced information processing scheme to the source code. When an object, A, sends a message to object B, by means of the information flows from A to B and it is called forward flow. Correspondingly, when B replies to it mean by there is a flow of information from B to A and it is called backward flow [32]. Proposed method it counted as fan-in and fan-out based on their direction. Further, it has discussed the calls between blocks are defined by object oriented principles.

A number of software metrics proposed to analyze object oriented programming. In [33,34] explained various metrics that analyze the object oriented programming which linearly count the features in the source code. The proposed method widely considers the linearity issues of earlier metrics. Object oriented programming contains various blocks which developed based on predefined methodology. Information moves within the blocks

* Corresponding author.
E-mail addresses: Jabbar123p@gmail.com (A. Jabbar), sriohmau@yahoo.co.in (S. Sarala).

0965-9978/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2012.08.004

and between blocks. The proposed metrics count the each motion in the source code. The movements are forced or initiated by any other blocks considered as object oriented method/procedural call. Basically stored procedure calls (SC), method calls inside the class (IC), method calls outside the class (OC) and constructor calls (CC) are the calls in the OOP concept. The object oriented features such as association, aggregation, abstraction, generalization, interface and inheritance are considered under the fan-in and fan-out or object oriented call when it initiate any transfer within the code. More over the proposed method reflects the important of the lines of code (LOC) metrics [3].

The validation of software metrics is very problematic as well as it has complex process. From the previous works, a number of approaches have been proposed for validation of software metrics adequacy. Limitation of existing validation technique in every perspective the Rule Accuracy Algorithm (RAA) has been proposed [1]. RAA includes number of optimized rule which induced best data according to the predefined rule from set of data. Rule induction technique is the common form of knowledge discovery from huge data in unsupervised learning systems [6,18–20]. RAA allows for a more complicated evaluation of software metrics. From the given software metrics data set, the best metrics data are induced by RAA and find the average. A database generates using software metrics where the instances and the set of variable referred from a number of programming units given by github; it is open-source free public repositories, code review, graphs and much more.

2. Mechanism in existing work

In the last few years, a number of approaches under the name of information flow complexity metrics have been developed, contributing interesting outcome. The author [3] criticize the Henry and Kafura measure [30] of Information Flow Complexity (IF-C) which as,

$$\text{Henry and Kafura IF-C } (\mathrm{M}) = \text{length } (\mathrm{M}) * (\text{fan-in } (\mathrm{M}) * \text{fan-out } (\mathrm{M}))^2 \tag{1}$$

As of the optimistic elevation of the work, product of fan-in fan-out in their measure IF-C, the modules with both a low fan-in low fan-out are, inaccessible from the system and hence have low "complexity" [28]. Fan-in shows how many modules reliably control a specified module; fan-out is number of modules that are reliably controlled by another module [29]. Specifically fan-in of a module M is explained, the number of local flows that discharge at M, and sum of the number of data structures from which information is retrieve by M. Also, the fan-out of a module M is the count of local flows that originate from M, and sum of the number of data structures that are reorganized by M.

Henry and Kafura measure criticized fractional view on a comprehensive information flow aspects. For capture specific view of information flow structure include global and local information and avoid length factor and refinement to the Henry and Kafura measure of information flow complexity for a module by Martin Shepperd [28] is

$$\text{Shepperd complexity } (\mathrm{M}) = (\text{fan-in } (\mathrm{M}) \cdot \text{fan-out } (\mathrm{M}))^2 \tag{2}$$

Shepperd's refinements challenge to confine an exact vision of information flow configuration and reliable with analyzing hypothesis. The experiential validation studies scrutinize how closely the counts associate with a definite procedure measure, which is improvement instance. The correlation between development time and the Henry–Kafura measure was not significant for Shepperd's data but pure-information flow-structure is considerably related. Thus the level of information flow is directly interrelated with development time [28]. Multiplication in (1) and (2), either fan-in or fan-out is zero, complexity measure shows zero. The inadequacy of (1) and (2) in various code levels, thereby a new technique introduced that involves the information flow and its complexity. Flow of information and its complexity are attempt to measured [3] using metrics such as fan-in (F_in), fan-out (F_out), sum of fan-in and fan-out (F_(I + O)), procedure called in a program (PC) and code length (CL) [3].

The both (1) and (2) information flow complexity metrics have identified major deficiencies in numerous aspects. Multiplication leads some serious issues as well as the framework not organized in all information representation in the code. Accordingly a new information flow metrics IF-C has been proposed [3], which considers structural programming source code features. The proposed work formulates based on the concept of new information flow metrics IF-C. The work input information was classified different ways. Primarily, local variable LR, it will access and use the data of within the procedure. The data access is possible both static and dynamic mode. The limitation of the LR is access only in the procedure. Fig. 1 shows the LR and its flow in procedure. There are n local variable $\mathrm{LR}_1\ \mathrm{LR}_2\ \mathrm{LR}_3 \ldots \mathrm{LR}_n$ and its value $x_1, x_2, \ldots \mathrm{x}_n$, it is using in the various process in the source code.

Secondly, global variable reading GR, it will access only once in the source code and use everywhere in the source code as static and dynamic mode. Fig. 2 depicts GR and its flow of the source code. There are n global variable $\mathrm{GR}_1\ \mathrm{GR}_2\ \mathrm{GR}_3 \ldots \mathrm{GR}_n$ Re used in the source code.

Fig. 2 refers the global variable using dynamically in the process. Local variable used as static method that also declared before the execution of the source code. Procedural concept has an important feature that data can share within the procedure. For this parameter data PR can read statically or dynamically in the execution. Fig. 3 express the dynamic reading data for the process.

IF-C explained the input information from the source code as local information, global information and passing parameter. These are defined as in fan-in method, which is used as input information of a module. Fan-in is the number of information admittance by a module. Following system process the output information is produced. Those are classified as local variable writing, global variable writing and parameter writing. After the process change the input variable in to information and stored as variables. The output information has marked in Fig. 4. The input variable has explained as local, global and parameter variable also used to represent the output information.

2.1. Framework for information flow metrics

In [3] discussed about Information flow that can be measure in competent to quantify the information for an entire operation of data within the execution of source code. Flow of information and its complexity are measured using metrics such as fan-in (F_in), fan-out (F_out), sum of fan-in and fan-out (F_(I + O)), procedure called in a program (PC) and code length (CL). These metrics measure the information flow and complexity of executable code within the procedures. Henry Kafura and Shepperd IF-C, either

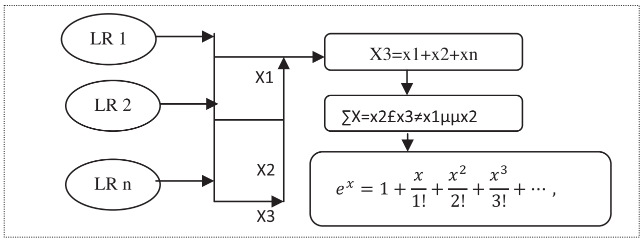
LR 1
LR 2
LR n
X1
X2
X3
X3=x1+x2+xn
∑X=x2£x3≠x1µµx2
$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots,$

Fig. 1. Local variable reading and usage in the source code.