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

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...
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 form 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,

\[
\text{Henry and Kafura IF-C (M) = length (M) \times \left[ \frac{\text{fan-in (M)}^2}{\text{fan-out (M)}} \right]}
\]

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

\[
\text{Shepperd complexity (M) = (fan-in (M) \times fan-out (M))^2}
\]

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 configuration 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 \(LR_1, LR_2, LR_3, \ldots LR_n\) and its value \(x_1, x_2, \ldots x_n\). 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 GR1, GR2, GR3, \ldots GRn 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 reading 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

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

\[
\text{IF-C} = \frac{\text{fan-in (M)} \times \text{fan-out (M)}}{\text{fan-in (M)}^2}
\]

Fig. 1. Local variable reading and usage in the source code.
fan-in or fan-out is zero, complexity measure shows zero. Towards the evaluation, limitation is detected; thereby a new technique has introduced that involves the information flow and its complexity. The various types of information initiated in storage gadget and the changes are occurring during execution of the program, information flow complexity metrics is evaluated. Structural frame work of information flow complexity in Fig. 5 addresses the follow of information in the program, PC and CL. The length factor considers the total information and procedural call in the program.

2.1.1. Code length (CL)

The work in [3] the source code length is measured as statement count, which includes data definition, data read, data write and control statement. There are many differences from lines of code measure (LOC). Lines of code and effective lines of code (ELOC) traditional method of software size measure [28].

2.1.2. Fan-in (F_in)

The work in [3] Fan-in has explained as input information in a module, consequently metrics contains sum of variable reading that include local variable reading (LR), global variable reading (GR) and parameter reading (PR). Fan-in (F_in) is,

\[
F_{\text{in}} = \sum_{i=0}^{n} LR_i + \sum_{i=0}^{n} GR_i + \sum_{i=0}^{n} PR_i
\]

2.1.3. Fan-out (F_out)

The work in [3] Fan-out has marked the sum of output information from the modules that are reference parameter written (RPW), local variable written (LW) and global variable written (GW). Fan-out is,

\[
F_{\text{out}} = \int_0^{LR} dLR + \int_0^{GR} dGR + \int_0^{PR} dPR
\]

2.1.4. Metrics of total information usage

As per the work [3] total information in a code is sum of the fan-in and fan-out that are defined in the above cases. The metrics defined as total information of the code is,

\[
F_{\text{I+O}} = \left( \int_0^{LR} dLR + \int_0^{GR} dGR + \int_0^{PR} dPR \right) + \left( \int_0^{RPW} dRPW + \int_0^{GW} dGW + \int_0^{PW} dPW \right)
\]
Fig. 4. Output information and its flow of the source code.

Fig. 5. Framework of information flow in source code.
2.1.5. Procedural call (PC)

The procedure concept is used to reuse and simplify which extract procedure calls of the program. When procedural calls are checked a number of procedures are accessed in the process.

2.1.6. Information Flow Complexity (IF-C)

Information flow complexity of every unit is compute by information flow (IF), procedural call (PC) and code length (CL). The expression measures information flow and its complexity. More precisely, the work expresses the details study on information flow traffic in a source code unit. The complexity analyzes the quality of code, thereby guarantee the enhancement of the code quality.

\[
\text{IF-C} = \frac{\sum \left( R \cdot dLR + \sum R \cdot dGR + \sum R \cdot dPR + \sum R \cdot dPW \right)}{CL}
\]

3. Object oriented programming

Object-oriented systems are collected as objects. Objects can be defined as a secure form of data state, and methods for manipulating that data. Classes are prototypes of objects. An object is a physical implementation of a class, or an instance, of a class. A class is defined to be a set of attributes and methods and may have many instance objects. Objects interact and communicate by passing messages. This work consider the flow which initiated by any type of objects from data blocks.

4. Object Oriented Information Flow Metrics (OIF-C)

The IF-C generalized measurement strategy which can apply in object oriented programming with efficient outcome. The work attempts to measure the flow between data blocks and inside the blocks as it is a difficult task. Information flow between data blocks within the data blocks are confusing and has complex architecture when it measure with particular criterion.

Information flow measurement considers a collection of set of attributes. In this work analyze the blocks that applied various object oriented principles initiate any flows or end any flow in to it. Proposed metrics the flow counted and addressed as object oriented fan-in and object oriented fan-out. The object oriented concept information handled in different form like Instance Variables (Non-Static Fields), Class Variables (Static Fields), Local Variables, global variable and object passing. Various object oriented calls analyzed such as stored procedure calls (SC), method calls inside the class (IC), method calls outside the class (OC) and constructor calls (CC). Limitation of the work is not to address independent details of fan-in and fan-out measure from each call. Finally the metrics addresses the source code size CL interrelates with the information flow. Object oriented information flow metrics is providing the flexibility of understanding the data manipulation and maintenance. In addition, code reusability process is enhanced.

4.1. Object Oriented Fan-in (OOF_in)

Fan-in specifies the input information in a module; therefore metrics contains sum of local variable reading (LR), global variable reading (GR), Object reading (PR) instance variable (IR, non-static variable) reading and class variable (CR, static variable) reading. Fan-in is defined as,

\[
\text{OOF}_{\text{in}} = \sum \left( LR \cdot dLR + \sum R \cdot dGR + \sum R \cdot dPR + \sum R \cdot dIR + \sum R \cdot dCR \right)
\]

4.2. Object Oriented Fan-out (OOF_out)

Fan-out metrics consist of sum of object written (RPW), local variable written (LW), instance variable (IW, non-static variable) written, class variable (CW, static variable) written and global variable written (GW). The object written is one of the Java features. Fan-out is,

\[
\text{OOF}_{\text{out}} = \sum \left( LW \cdot dLW + \sum W \cdot dGW + \sum W \cdot dPW + \sum W \cdot dIW + \sum W \cdot dGW \right)
\]

4.3. Calls in object oriented programming (COOP)

Most of the data flows initiate in a class or method which calling a method or instance. According to java there are many calls like stored procedure calls (SC), method calls inside the class (IC), method calls outside the class (OC) and constructor calls (CC). COOP can compute the sum of total call instigate in a class. Thus, the COOP is defined as,

\[
\text{COOP} = \sum (SC \cdot dSC + IC \cdot dIC + OC \cdot dOC + CC \cdot dCC)
\]

4.4. Object oriented Information Flow Complexity (OOIF-C)

Information flow complexity of each unit is measured using information flow (OIF), number of calls in object oriented programming (CO) and code length (CL). The information flow complexity is,

\[
\text{OOIF-C} = \left( \sum R \cdot dLR + \sum R \cdot dGR + \sum R \cdot dPR + \sum R \cdot dIR + \sum R \cdot dCR \right) + \frac{\text{OOF}_{\text{out}}}{\text{CL}}
\]

5. Software metrics validation

Software metrics has important conscientiousness in software development life cycle as it show incredible ability in maintainance, readability, restructuring and reusability of source code. Consequently the adequacy of software metrics is a considerable issue to the stakeholders. Therefore, very short software metrics validation approaches are developed by researchers. The validation of software metrics has been performed in various methods, analytical validation and empirical validation [8,26]. In analytical validation the theoretical foundation of metrics and the features associated with the parameters is also analyzed. Empirical validation is to classify precision and accuracy of software metrics which has associated from the different aspects of different trade.

In [5] the authors have specified that the software metrics validity has been measured by defined properties. Obviously, the most related work [26] has been found in the research. Software metrics validation is the described number of properties to validate the software metrics measures which has depend only in the context of syntactic attributes of a program. Taking properties as the relation between syntactical variations during the result formation, exactly nine properties have been proposed [27]. Then, the reaction in validation, the properties are relevant for various software complexity metrics. In [26], the authors validate cohesion and information theoretic metrics using properties. In the valida-
tion which is abortive to gratify the properties but most of the metrics being complex. Metrics validation is a difficult process, as the properties must confirm to the mathematical transparency in all perception. Since it is a difficult practice, it would be consider for few measurements. The other side, this validation approach has been offered hopeful results but the results are not applicable in all perspective. Hence, it required appropriate alternative to realize the absolute effortless validation of metrics in all perspective. Therefore a new methodology has been practiced to validate metrics adequacy [1].

6. Rule accuracy algorithms

Object oriented software metrics validation is performed by rule accuracy algorithm. Rule accuracy algorithm, rule induction method induced best data from data set to validate the metrics adequacy [1]. The predefined rules are inducing the adequacy information from metrics data set which confirms the effectiveness of the metrics. The rule induction is an important element of expert system as well as familiar classification method in the research [7,13–17]. In this work the proposed RAA classification algorithm verifies the object oriented metrics validity, to perform the validation in advance. Given set of metrics data set, RAA validate whether the metrics is sufficient or not. The fundamental issue is to commence the validation as the declaration of a rule optimization to find best rules. For rule formulation, rule explained in [31] the work has involved as linguistic. The sample rule is,

\[ R_1: \text{IF the rule1 is YES or NO AND THEN adequate} \]

The if-then rules are defined as, rule 1, rule 2, ..., rule \( n \), has the structural attributes of metrics, and thus it varied according to the characteristics of metrics. The rules optimization performed by classification method basis on the improved ant colony optimization algorithm [9]. Towards ACO, the researchers believe how the almost blind ants establish the shortest path to their feeding sources and rear. Basically, the phenomenon is that ants use pheromone for communication. The path of an ant lays a few of pheromone, which are evident by other ants, along its path. Accordingly, the ants can establish the shortest path to the feeding sources and back. Like this biological features use most difficult classification issues in research. For example, ACO finds the solution for various classification issues [10–12,21–25]. Based on rule induction techniques a number of approaches have been used to discover the information from data base with good accuracy. The various if-then-rules are recognized based on adequacy criterion [13]. The if-then rules are defined as, rule 1, rule 2, ..., rule \( n \), containing attributes of metrics. It can be varied based on the metrics behavior. Thus RAA algorithm is reliable to validate the metrics. Exactly, the work [1] concentrates on structural metrics, and thus it varied according to the characteristics of available source code.

The important issue of RAA is how to choose the finest rules from the rule set. The validation with number of rules is big hazard; therefore it is optimized by ACO classification method. Metrics adequacy factors are identified by the rules, the adequacy factors which reflect on the source code behavior and measuring aspects. The value exposure may vary significantly on the rules precision. In order to avoid misleading result definite policy are fixed for all rule definition. RAA algorithm is defined as the instances that are selected according to the rules, which induced the data set and finds the accuracy [1]. To define RAA algorithms for metric adequacy, the rules, induce the metrics data set which generate using metrics from source code. In this work data set generate proposed OIF-C from github it is an open-source free public repositories. Let individual from a metrics set \( m = (m_1, m_2, m_3, \ldots, m_n) \) of a metrics set \( M = (M_1, M_2, \ldots, M_n) \) and rule set \( R = (R_1, R_2, \ldots, R_n) \). Given rules of the best selection, \( R \), and the favorite metrics, \( M \), signify a data set, \( m \). RAA estimate the metrics adequacy by each rule metrics values and parameters are examined. Fig. 6 shows the architecture of RAA algorithm.

7. Experimental results

Object oriented information flow metrics have been implemented. The study on the adequacy of information flow metrics is to evaluate rule accuracy algorithm. The evaluation process is required dataset of system information. Accordingly, in the proposed work object oriented information flow metrics and Henry Kafura information flow metrics are used to generate dataset. Evaluators were creating dataset; exactly five program source codes from github open repository were analyzed. The experiment is limited only for information flow attributes. Table 1 indicates the dataset of proposed object oriented information flow metrics and Table 2 expresses the dataset of Henry Kafura information flow metrics. Henry and Kafura measure has identified major issues to measure source code. More over it not copiously addressed the various flow initiated in the block. Hence, adequate metrics framework has been developed. Fitness Proportionate Selection (FPS) algorithm is employed [35] for optimize the process. Fitness Proportionate Selection (FPS) is a genetic operator used in genetic algorithms for selecting potentially useful solution for recombination. If \( f_i \) is the fitness of individual \( i \) in the population, its probability of being selected is

\[ P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \]

where \( N \) is the number of individuals in the population. FPS selects the values for metrics using predefined fitness function. Assume, with the whole system input and output information, that a metric takes size values to represent increasing system compression. If a relation between total information and a metric CL existed, example, a linear grouping or direct proportion, subsequently values in Fig. 7 both to be increased.

Salem et al. and Kumar et al. [33,34] are discussed about the object-oriented metrics. Both object oriented metrics linearly addressed, consequently, a comprehensive study is limited. Object oriented information flow metrics provide an extensive vision on the drawbacks of these metrics. The accuracy of famous existing method Henry Kafura is unreliable in various possible inputs and outputs [3]. OIF-C never fails to measure the complexity in low fan-in and fan-out metrics. In this work extend the metrics in to object oriented concept as well. Using all these equations number of source code are evaluated. Code contains a high fan-in, fan-out and higher calls; the result as high complexity. High complexity is a sign of increased data transfer, caused collision in continuous execution, hence maintenance is required. In addition this metrics

![Fig. 6. Architecture of rule accuracy algorithm.](image-url)
would assist and enhance the maintenance process. After preservation of the program is evaluated the complexity and the values are tested. By this approach, high error prone with sample code is observed.

Software metrics adequacy evaluated by RAA algorithm with rule induction technique is implementing with given set of data. For the validation process data set is formulated using OIF-C metrics, which originate by the analysis of the source code evaluation. The comparison of Henry et al. metrics and proposed OO metrics are displayed in Tables 1 and 2. RAA analyze the datasets which is preferred earlier by best rules from the given a set of rules. RAA algorithm, the best Rule set R analysis each row in Table 1, i.e. $M_1, M_2, \ldots, M_n$. The rule is defined to embrace the reach of competence intention. Metric set $M_i$ validate by rule set $R_i$, i.e. $M_i$ validate by each rule in Rule set $R_1, R_2, R_3, \ldots, R_n$. Rules are framed to consider the entire system behaviors. Hence a number of rules are obtained. There are nine rules are formed after the ACO optimization. Table 3 shows the result of the RAA. First column shows optimized rules; the second is result of rule analysis of OOIF-C and third comparison Henry and Kafura information flow metrics respectively. The study controlled number of rules by ACO optimization to guarantee the consistency and accuracy. The metrics M convince the rule R means yes condition or no condition will displayed. Compute the average of satisfied rules, which is representing adequacy criterion.

The OOIF-C metrics provide to predict error prone in the code and the flexibility source code maintenance in all perspective. The OOIF-C metrics shows the system framework as well as the artifacts information which be included in the code. Thus it can easily relate to maintainability and reuse activity. Fig. 8 represents the maintenance ability of source code is increasing with OOIF-C metrics. Maintenance and code refactoring are effortless as system information and the flexibility source code maintenance in all perspective. The OOIF-C is expecting most accepted measurement technique in industry.

### 8. Conclusion

In this work attempt a new direction for object oriented metrics to analyze the source code it has been formulated and validated the adequacy of OOIF-C. OOIF-C validates using predefined rules. Induced the metrics data as satisfying the best rules set are finding the best metrics. The above experiment has shown the adequacy of OOIF-C measure in various inputs. It shows OOIF-C is much more consistent to measure the information flow complexity of object oriented source code. To recognize the existing metrics for information flow is inadequate to analysis the OO languages. OOIF-C addresses the appropriate and quantitative measure of information flow for OO languages and has proved with the calculation of information complexity in the source code what the user intends. The advanced concept of information flow metrics proposed in OO concept by direct proportionality of total information and LOC. The various OO features strappingly incorporate for the information flow analysis framework will improve the OOIF-C metrics. Thereby, OOIF-C is expecting most accepted measurement technique in industry.

### References


