Journal of King Saud University – Computer and Information Sciences (2017) **29**, 93–102

King Saud University

**Journal of King Saud University – Computer and Information Sciences**

www.ksu.edu.sa
www.sciencedirect.com

# Fuzzy inferencing to identify degree of interaction in the development of fault prediction models

**Rinkaj Goyal** *, **Pravin Chandra, Yogesh Singh**

*University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, Sector 16 C, Dwarka, Delhi 78, India*

Received 8 October 2014; revised 24 December 2014; accepted 24 December 2014
Available online 3 November 2015

**KEYWORDS**

Software fault prediction;
Fuzzy inference system;
Influential metrics;
Object oriented metrics

**Abstract** The software fault prediction models, based on different modeling techniques have been extensively researched to improve software quality for the last three decades. Out of the analytical techniques used by the researchers, fuzzy modeling and its variants are bringing out a major share of the attention of research communities. In this work, we demonstrate the models developed through data driven fuzzy inference system. A comprehensive set of rules induced by such an inference system, followed by a simplification process provides deeper insight into the linguistically identified level of interaction. This work makes use of a publicly available data repository for four software modules, advocating the consideration of compound effects in the model development, especially in the area of software measurement.

One related objective is the identification of influential metrics in the development of fault prediction models. A fuzzy rule intrinsically represents a form of interaction between fuzzified inputs. Analysis of these rules establishes that Low and NOT (High) level of inheritance based metrics significantly contributes to the F-measure estimate of the model. Further, the Lack of Cohesion of Methods (LCOM) metric was found insignificant in this empirical study.


## 1. Introduction

Fault-proneness of a software module is an external metric and a fault prediction model applies a modeling technique to the internal software metrics to predict fault-proneness (Catal and Diri, 2007; Fenton and Neil, 1999). Due to the increased practice of object oriented technology in industry, an extensive usage of object oriented metrics has been proposed, and efforts have concentrated on building models that predict defective modules (Arisholm et al., 2010). These metrics not only indicate the complexity of an object and its association (interaction) with other objects, but also measure different characteristics of a quality model. The widely used Chidamber and Kemerer (CK) metrics along with a metric, Lines of Code (LOC) are used in this paper to conduct empirical study.

New fault prediction models may be developed from statistically validated improved models reported earlier or one may

* Corresponding author. Tel.: +91 8826020315.
E-mail addresses: rinkajgoyal@gmail.com (R. Goyal), chandra.pravin@gmail.com (P. Chandra), ys66@rediffmail.com (Y. Singh).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.12.008

develop their own customized models. The growth of such customized models reflects the necessities of a software project and applies the expertise of people affected in the operation (Weyuker et al., 2007).

Fuzzy based models translate the subjective understanding and expertise of the processes into mathematically exposable figures and rules to generate systems with some degree of uncertainty. The use of fuzzy logic in experimental software engineering to model various aspects of software evolution process is increasingly attaining recognition of the research community (Ahmed and Muzaffar, 2009; Aljahdali and Sheta, 2011; Bouktif et al., 2010; Chiu, 2011; Engel and Last, 2007; Gray and MacDonell, 1997; Khoshgoftaar and Seliya, 2003; MacDonell, 2003; Meneely et al., 2008; Ozcan et al., 2009; Pandey and Goyal, 2009; So et al., 2002; Verma and Sharma, 2010). The present study discusses the use of fuzzy inference mechanism to recognize the most notable rules in the development of fault prediction model using GUAJE framework (Alonso and Magdalena, 2011a,b; Alonso et al., 2012).

This study makes use of the data set of four software modules available in the NASA data repository for experimental usage (Jureczko and Madeyski, 2010).

Though in this paper, we derive knowledge from data itself. Nevertheless, fuzzy inference mechanism permits adding of experts experience in the formulation of fuzzy rules.

In this paper, interaction between variables is not conceptualized as a multiplicative term (generally used in regression analysis), rather the type of interaction analyzed here is moderated by FIS operators. The rest of this paper is organized as follows: Section 2 presents the core ingredients of a fuzzy inferencing system and semantics of a typical fingram; Section 3 describes the data set and FIS simulation parameters;Section 4 presents the results derived, and further elaborates the accuracy measures the developed FIS. Our conclusion is presented in Section 5.

## 2. Fuzzy inference system and semantics of a fingram

The core elements of a data driven fuzzy inference systems are:

1. Fuzzification module: alters crisp inputs into fuzzy values using membership function. In the construction of data driven fuzzy inferencing system, fuzzy partitioning of crisp values of the variables of data set carries out this fuzzification process.
2. Fuzzy inference systems (FIS): accomplish input–output mapping of linguistically communicated information in the form of rules. Different potential building algorithms induce these rules by learning from data.
3. Simplification module: holds the number of rules small and keeps consistent rule base. Though this module is optional in fuzzy inferencing process, GUAJE framework provides improved readability of generated fuzzy rules using this module.
4. Defuzzification: transcribes fuzzy outputs back into crisp values.

Fig. 1 elucidates the core elements of data driven fuzzy inference system discussed above (Guillaume and Charnomordic, 2011). Admitting that the expert knowledge

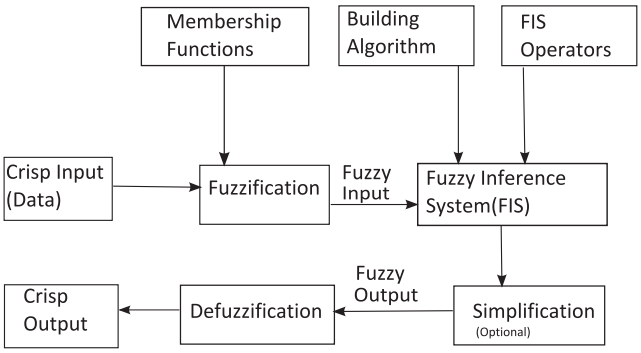


**Figure 1** Core elements of a fuzzy inference system.

is not accessible, the diagram does not explicitly confer the feedback perspective from the user and rule induction takes place through available information simply.

The simulation environment utilized in this study creates a number of fuzzy partitions of various sizes exercising three mechanisms namely regular partition, k-means algorithm with different numbers of groups and hierarchical fuzzy partitioning (hfp). Incorporation of criteria like partition coefficient (PC), partition entropy (PE) and Chen index (CI) determines suitable partition. A reliable partition minimizes the entropy and maximizes the partition coefficient and the Chen index (Guillaume and Charnomordic, 2011).

To induce rules from the fuzzified input data, the experimentation puts to use the Fast Prototyping Algorithm (FPA). FPA generates rules based on the basis of the rule matching degree and the number of records in the data set.

The patterns generated by FPA are quite large in number. Thus, simplification module protects the strongest interactions and rules out other variables, which are seeming in some rules only. Derived FIS uses normalized entropy as a criterion for classification problems to hold the balance between precision and interpretability.

Results obtained from the aforementioned fuzzy inference mechanism make use of fingrams for visual analysis of strong interactions and other useful outcomes.

In fingram, a circular node enacts a rule with its size as the comparative magnitude of the covered data samples. Classified colors of circles describe different classes of data set. For example, in Fig. 2. There are two classes in the data set depicted by the legend in Fig. 2.

*Classes 1.0 and 2.0 represent non-faulty and faulty data sample, respectively, and same terminology and legend is used in the interpretation of all fingrams constructed in this paper.*

Each circle represents the values of cov, G, Ci which are explained in as follows (Pancho et al., 2013a):

1. *Coverage of a node (cov)*: Ratio of covered data samples to the total number of samples (Relevant in interpreting fingram).
2. *Goodness (G)*: Indicates the goodness of a rule to classify data samples. Its value varies from −1 to 1. A negative value signifies the lower number of data samples correctly classified.
3. *Relative coverage of a node (Ci)*: This refers to the coverage of a rule corresponding to an output class. It is the proportion of the number of data samples covered by that rule to the total number of data samples in that class.

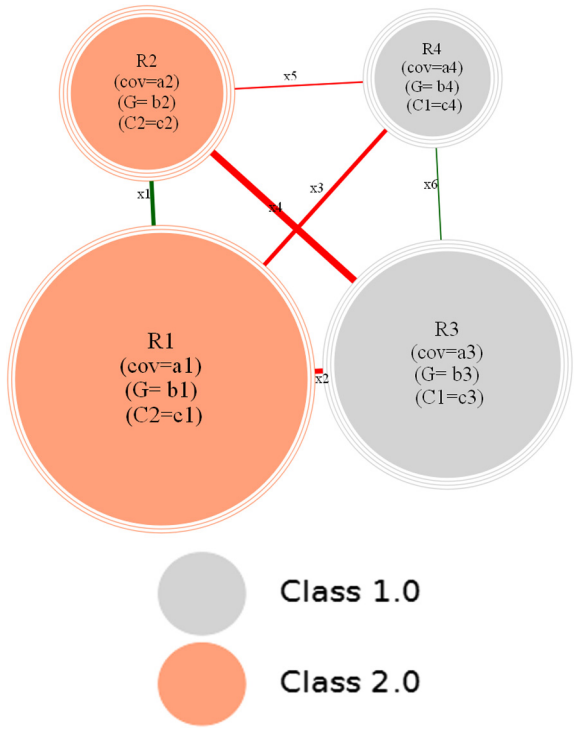


**Figure 2** Structure of a fingram.

Level, color and thickness of the link have significant intent. Label measures the degree of co-firing, visually rendered by the thickness of the link. Green color of the link aims to the node of a same class. Whereas, red color link points to the node of another class. As illustrated in Fig. 2, Rule R1 includes the maximum data samples of one class and the Rule R3 maximally covers instances of another class. Such visualizations will mostly help in analyzing coverage perspective of rules, while also demonstrating the rules fired at the same time in the FIS. The classification of such significant rules co-fired at the same time in the knowledge base design will further assist in the cataloging of influential metrics.

The results and discussion section of this paper explains and evaluates the comprehended significance of different metrics stored alongside the design of generated fingrams.

## 3. Dataset and FIS simulation parameters

In this work, we use small-medium size data set with reasonable defectiveness from publicly available promise data repository (Jureczko and Madeyski, 2010). This empirical study makes use of four software systems, namely Lucene, Poi, Synapse, Velocity and Xalan.

Table 1 shows the descriptive statistics of the data set used and Table 2 provides the concise summary of widely used Chidamber and Kemerer (CK) metrics used in this study to quantify different features of object oriented system development (Chidamber and Kemerer, 1994; Chidamber et al., 1998; Basili et al., 1996). We utilize 6 class level source code metrics and Lines of Code (LOC) metric among additional metrics available.

Table 3 depicts the correlation matrix between different CK metrics of individual software module observed in this study.

**Table 1** Empirical aspects of the data set used to generate FIS.

| Data set | No. of instances | % Defectiveness |
|---|---|---|
| Lucene | 340 | 59.71 |
| Poi | 442 | 63.37 |
| Velocity | 229 | 34.06 |
| Xalan | 741 | 48.119 |

For all four software modules, following parameters are used to generate FIS:

1. FIS operators: Conjunction is minimum and disjunction is maximized.
2. Defuzzification: Max-crisp.
3. Rule induction: Fast prototype algorithm.
4. Fingrams generation: Goodness threshold high 0.5 and low = 0.1

Fingrams visually represent the overall behavior of a FIS by adopting analysis techniques used so far in distinct domains like Scientometrics and social network analysis (Pancho et al., 2013a).

In Scientometrics, a scientogram visual represents co-citations, i.e. repetition of the simultaneous citation of two documents by others (Pancho et al., 2013b). Social network analysis (SNA) uses network theory to see social relationships between nodes and edges. Scaling algorithms take different proximity measures between nodes to draw abridged underlying organization, which promote natural reasoning of relevant patterns in the data (Pancho et al., 2013b).

The scaling algorithm used in this study is a Pathfinder algorithm (PFA). This algorithm exhibits an excellent capability to model symmetric, asymmetric relationships and notable relationships present in the data. The result of implementing Pathfinder to a network is a pruned network called PFNET (Alonso et al., 2011; Guillaume and Charnomordic, 2011).

## 4. Results and discussion

Experimental results derived from the developed fuzzy inference system (FIS) for four software modules are presented here.

### *4.1. Experimentation with Lucene module*

Table 4 below shows the outcome of fuzzification process. The fuzzy modeling of crisp data makes use of fuzzy partitioning as discussed in Section 3. BUG variable is the classifier here.

*Its values 1.0 and 2.0 represent non-faulty and faulty data sample respectively.*

The rules are induced from data using Fast prototype algorithm as briefly highlighted in Section 1. Table 5 shows the outcome of simplification module of a typical FIS i.e. reduced number of rules and Table 6 lists the reduced number of metrics, corresponding to these reduced numbers of rules; Fig. 3 illustrates the generated fingram.

It is evident (Section 4.5) from the fingram that Rule R1 and R2 fire simultaneously and Rule 1 covers maximum instances of faulty class with significant level of goodness.

**Table 2** Description of class level source code metrics.

| Metric name | Description | Influenced characteristics of quality model |
|---|---|---|
| Weighted Methods per Class (WMC) | This metric relates to the methods implemented in a class and determined by summing up the Cyclomatic complexity of all the methods implemented in a class | Maintainability, reusability and understandability |
| Lack of Cohesion of Methods (LCOM) | This metric indicates internal unity within components of a class design. It is estimated by counting the method pairs in a class with zero similarity, which are not sharing same class instance variables | Efficiency and reusability |
| Response for a Class (RFC) | This metric pertains to the request (message), an object makes to other objects and, calculated as the count of methods in the set of all methods implemented in the classes that can be called remotely in response to a message sent. This infer to the sum of the number of local methods and methods that can be called remotely | Understandability, maintainability, and testability |
| Coupling between Object Classes (CBO) | This metric reveals the dependence of one class over other classes in the design. Such dependence may occur due to the message passing mechanism or inheritance. It is estimated by summing the number of distinct non-inheritance related classes coupled with other classes | Reusability and efficiency |
| Depth of Inheritance Tree (DIT) | This metric exhibits the level of inheritance in the class design and, denotes the length of the longest path from a given class to the root class in the inheritance hierarchy. Though, inheritance fosters reusability of a class, but makes the maintenance and debugging more complex | Efficiency, reusability, understandability and testability |
| Number of Children (NOC) | This metric relates to the inheritance feature, similar to DIT, and computed by counting the number of immediate child classes inherited from a given class. A large value of NOC though enhances reusability, but makes a class difficult to test | Reusability, efficiency, and testability |

**Rule R1:** *IF* **dit** *is low AND* **noc** *is low AND* **cbo** *is NOT (high) THEN bug is 2.0.*

**Rule R2:** *IF* **wmc** *is NOT(high) AND* **dit** *is more or less (average) AND* **noc** *is low AND* **cbo** *is average THEN bug is 2.0.*

Thus RFC and LCOM metrics do not appear in the prominent rules identified above in the fault prediction. This can be justified on account of the fact that RFC metric is significantly correlated with the CBO (Table 3) and CBO is occurring towards the higher side of linguistic terms in the Rule 1 and Rule 2 both.

Similarly, LCOM is highly correlated with RFC and effect of RFC has been already captured by CBO. Consequences derived from Lucene module support lower side NOC and DIT linguistic terms, whereas, CBO and WMC towards higher side (linguistic terms).

### 4.2. Experimentation with Poi module

Observational study of the Poi software module (Jureczko and Madeyski, 2010) replicates the considerations made for the Lucene software module. Realizing the fuzzification of the dataset, the use of FPA brings about 19 rules and simplification process decreases the number of rules to 4 (Tables 7–9).

Fingram study of these lessened rules establishes ensuing results (Fig. 4). By using taxonomy of the constitution of a fingram, Rule R1 and R2 are the famed ones in the classification of faulty cases. Moreover, Rule 1 has a preponderance over Rule 2 on account of the imbibed goodness factor in the design.

**Rule 1:** *IF* **wmc** *is low AND* **dit** *is low AND* **noc** *is low AND* **cbo** *is NOT(high) AND* **rfc** *is NOT(high) THEN bug is 2.0.*

**Rule 2:** *IF* **wmc** *is NOT(high) AND* **dit** *is NOT(high) AND* **noc** *is low AND* **cbo** *is NOT(high) AND* **rfc** *is average THEN bug is 2.0.*

LCOM and LOC are not emerging in the rules here. Adverting to the correlation matrix for Poi module, the highest correlation values are seen between RFC and LCOM, LOC both. A review of the prominent rules exhibits the occurrence of RFC towards the higher side in linguistic terms, thereby carrying the predictive ability of LCOM and LOC.

### 4.3. Experimentation with Velocity module

To highlight the robustness of the road map presented here, experiments and discussions, alike to the previous two modules are performed for the Velocity module (Jureczko and Madeyski, 2010). Tables 10–12 report the result of fuzzification process and different rules induced thereafter. Summary of the generated fingram (Fig. 5) establishes the superiority of Rule 1 and Rule 3.

**Rule 1:** *IF* **wmc** *is low AND* **dit** *is low AND* **rfc** *is NOT(high) AND* **loc** *is low THEN bug is 2.0.*

**Rule 3:** *IF* **wmc** *is more or less (average) AND* **dit** *is low AND* **rfc** *is low AND* **loc** *is low THEN bug is 2.0.*

Referring to the correlation matrix (Table 3) for velocity module LCOM, CBO and NOC metrics have the highest correlation coefficient values with WMC metric. Therefore, a significant reduction in the resolute set of metrics is perceived, and LCOM, CBO and NOC metrics are not rising in the list of prevailing rules.

### 4.4. Experimental analysis of Xalan module

Empirical study similar to previous modules is carried out for a relatively larger dataset of Xalan. Tables 13–15 show the decreased rules (metrics). Fingram interpretation (Fig. 6) establishes the supremacy of Rule 1, Rule 2 and Rule 3.

**Table 3** Correlation matrix for four software modules of data set.

| | WMC | DIT | NOC | CBO | RFC | LCOM | LOC | FAULT |
|---|---|---|---|---|---|---|---|---|
| *LUCENE* | | | | | | | | |
| WMC | 1 | | | | | | | |
| DIT | −0.0973 | 1 | | | | | | |
| NOC | 0.0123 | −0.1817 | 1 | | | | | |
| CBO | 0.4921 | −0.1852 | 0.35 | 1 | | | | |
| RFC | 0.9264 | −0.0367 | −0.0507 | 0.4462 | 1 | | | |
| LCOM | 0.8488 | −0.0809 | 0.0268 | 0.4376 | 0.7999 | 1 | | |
| LOC | 0.8447 | −0.0976 | −0.0723 | 0.2699 | 0.8518 | 0.7812 | 1 | |
| FAULT | 0.2473 | −0.079 | −0.175 | 0.0932 | 0.2756 | 0.1111 | 0.2654 | 1 |
| *POI* | | | | | | | | |
| WMC | 1 | | | | | | | |
| DIT | −0.0317 | 1 | | | | | | |
| NOC | −0.0136 | −0.0687 | 1 | | | | | |
| CBO | 0.2477 | −0.0876 | 0.3919 | 1 | | | | |
| RFC | 0.8593 | −0.0988 | 0.0065 | 0.3741 | 1 | | | |
| LCOM | 0.7189 | −0.0147 | −0.0101 | 0.2221 | 0.6621 | 1 | | |
| LOC | 0.5338 | −0.0831 | 0.0076 | 0.2036 | 0.5857 | 0.409 | 1 | |
| FAULT | 0.1831 | −0.322 | −0.0487 | 0.1395 | 0.2691 | 0.1355 | 0.2277 | 1 |
| *VELOCITY* | | | | | | | | |
| WMC | 1 | | | | | | | |
| DIT | −0.1391 | 1 | | | | | | |
| NOC | 0.1083 | −0.0621 | 1 | | | | | |
| CBO | 0.5017 | 0.0728 | 0.3329 | 1 | | | | |
| RFC | 0.8656 | −0.0694 | 0.0703 | 0.4379 | 1 | | | |
| LCOM | 0.8086 | −0.0848 | 0.0314 | 0.4335 | 0.6651 | 1 | | |
| LOC | 0.736 | −0.0705 | −0.0015 | 0.1741 | 0.6255 | 0.5603 | 1 | |
| FAULT | 0.171 | −0.3634 | 0.0387 | 0.0163 | 0.2157 | 0.0959 | 0.1314 | 1 |
| *XALAN* | | | | | | | | |
| WMC | 1 | | | | | | | |
| DIT | −0.1599 | 1 | | | | | | |
| NOC | 0.1357 | −0.0178 | 1 | | | | | |
| CBO | 0.4237 | −0.0079 | 0.3142 | 1 | | | | |
| RFC | 0.8645 | −0.0621 | 0.0932 | 0.4788 | 1 | | | |
| LCOM | 0.7989 | −0.1051 | 0.1024 | 0.3355 | 0.6192 | 1 | | |
| LOC | 0.4702 | 0.0302 | 0.0786 | 0.1839 | 0.5432 | 0.3308 | 1 | |
| FAULT | 0.3181 | −0.2174 | 0.0437 | 0.2853 | 0.3835 | 0.1319 | 0.1074 | 1 |

**Table 4** Fuzzification of the data set of Lucene module.

| Linguistic variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [1, 166] |
| DIT | Low, average, high | [1, 5] |
| NOC | Low, average, high | [0, 17] |
| CBO | Low, average, high | [0, 128] |
| RFC | Low, average, high | [1, 392] |
| LCOM | Low, average, high | [0, 6747] |
| LOC | Low, average, high | [1, 8474] |
| BUG | 1.0, 2.0 | [1, 2] |

**Table 5** List of rules after applying simplification process for the Lucene.

| Rules | Linguistic rules |
|---|---|
| R1 | IF dit is low AND noc is low AND cbo is NOT(high) THEN bug is 2.0 |
| R2 | IF wmc is NOT(high) AND dit is more or less (average) AND noc is low AND cbo is average THEN bug is 2.0 |
| R3 | IF wmc is low AND dit is more or less (average) AND noc is NOT(high) AND cbo is low THEN bug is 1.0 |
| R4 | IF wmc is low AND dit is low AND noc is average AND cbo is average THEN bug is 1.0 |

**Rule 1:** *IF* **dit** *is low AND* **noc** *is low AND* **rfc** *is NOT (high) THEN bug is 2.0.*
**Rule 2:** *IF* **wmc** *is NOT (high) AND* **noc** *is NOT (high) AND* **cbo** *is more or less (average) AND* **rfc** *is NOT(high) THEN bug is 2.0.*
**Rule 3:** *IF* **wmc** *is NOT (high) AND* **dit** *is NOT (high) AND* **noc** *is low AND* **cbo** *is low AND* **rfc** *is average THEN bug is 2.0.*

LCOM and LOC metrics do not appear in the list of the strongest rules. This occurs because LCOM and LOC are highly correlated with RFC, which is going forth towards the higher side of linguistic terms.

The rules generated by FIS signify interaction between variables, thereby providing favorable modeling of complex nonlinear functions. Rules (influential rules) obtained after the

**Table 6** List of variables after applying simplification process for the Lucene.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | low, average, high | [1, 166] |
| DIT | low, more or less (average) | [1, 5] |
| NOC | low, average, high | [0, 17] |
| CBO | low, average, high | [0, 128] |
| BUG | 1.0, 2.0 | [1, 2] |

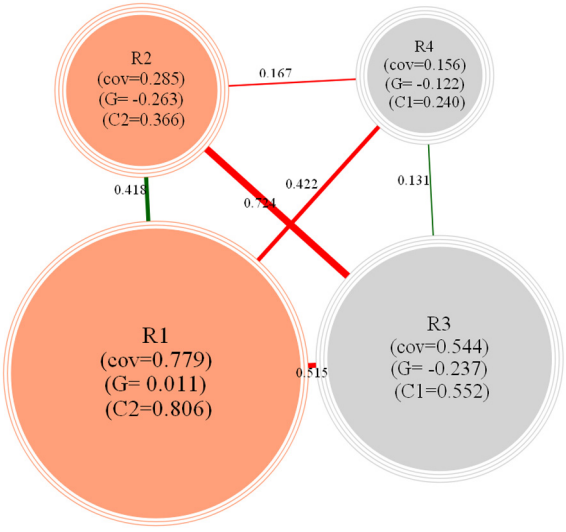


**Figure 3** Fingram corresponds to the reduced set of rules for Lucene.

**Table 7** Fuzzification of the data set of Poi.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [0, 134] |
| DIT | Low, average, high | [1, 6] |
| NOC | Low, average, high | [0, 134] |
| CBO | Low, average, high | [0, 214] |
| RFC | Low, average, high | [0, 390] |
| LCOM | Low, average, high | [0, 7059] |
| LOC | Low, average, high | [0, 9886] |
| BUG | 1.0, 2.0 | [1, 2] |

process of simplification formulate effective interactions between variables.

Fig. 7 depicts the frequency of variables under varying level of fuzzification in the influential rules for all four software modules. Since FIS created in this study also takes co-firing of rules as a significant element in the measurement of the impact of a rule, it is likely that the same variable may get a room at multiple cells of the table.

Nevertheless, in such scenarios, additional count assigned to the variable will not alter the interpretability of results. It is evident from Fig. 7 that Low degree (membership) of Inheritance based metrics significantly contributes to identify faulty classes.

**Table 8** List of rules after applying simplification process for Poi.

| Rule | Linguistic rules |
|---|---|
| R1 | IF wmc is low AND dit is low AND noc is low AND cbo is NOT(high) AND rfc is NOT(high) THEN bug is 2.0 |
| R2 | IF wmc is NOT(high) AND dit is NOT(high) AND noc is low AND cbo is NOT(high) AND rfc is average THEN bug is 2.0 |
| R3 | IF wmc is low AND dit is NOT(low) AND noc is low AND cbo is low AND rfc is low THEN bug is 1.0 |
| R4 | IF wmc is average AND dit is low AND noc is low AND cbo is low AND rfc is low THEN bug is 1.0 |

**Table 9** List of variables after applying simplification process for Poi.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [0, 134] |
| DIT | Low, average, high | [1, 6] |
| NOC | Low, more or less (average) | [0, 134] |
| CBO | Low, average, high | [0, 214] |
| RFC | Low, average, high | [0, 390] |
| BUG | 1.0, 2.0 | [1, 2] |

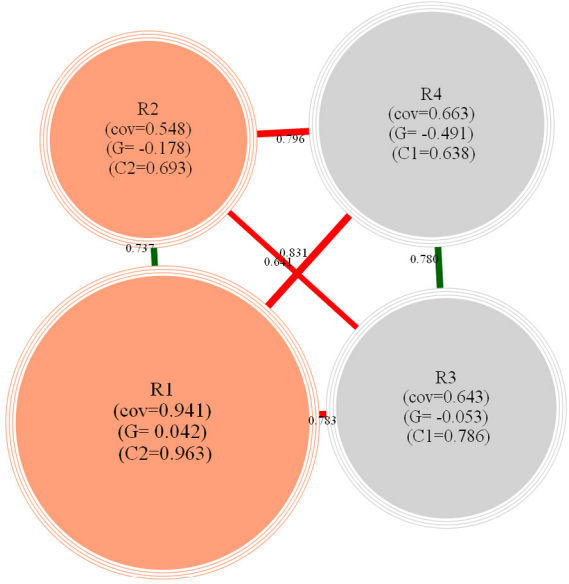


**Figure 4** Fingram corresponds to the reduced set of rules for Poi module.

Frequency count of the DIT and NOC is six and five respectively, which is more eminent than the tally of other variables. As is obvious, the most significant enrichments come from the Low and NOT (High) level of interaction of variables. In the literature, cohesive measure (LCOM) has been reported a key metric to classify fault prone classes. But, in this study LCOM is irrelevant with a frequency count of null. Since metrics are often correlated, the interaction effect of LCOM

**Table 10** Fuzzification of the dataset of velocity module.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [0, 153] |
| DIT | Low, average, high | [1, 5] |
| NOC | Low, average, high | [0, 39] |
| CBO | Low, average, high | [0, 80] |
| RFC | Low, average, high | [0, 250] |
| LCOM | Low, average, high | [0, 8092] |
| LOC | Low, average, high | [0, 13175] |
| BUG | 1.0, 2.0 | [1, 2] |

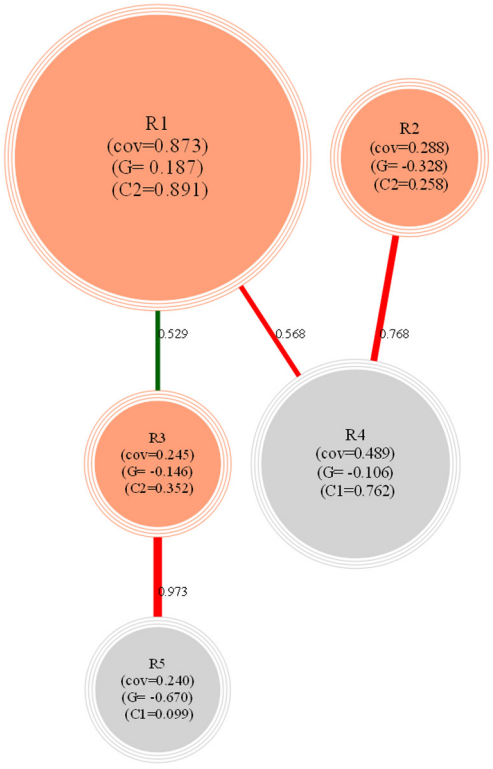


**Figure 5** Fingram corresponds to the reduced set of rules for Velocity module.

**Table 11** List of rules after applying simplification process for velocity module.

| Rules | Linguistic rules |
|---|---|
| R1 | IF wmc is low AND dit is low AND rfc is NOT(high) AND loc is low THEN bug is 2.0 |
| R2 | IF wmc is low AND dit is more or less (average) AND rfc is average AND loc is low THEN bug is 2.0 |
| R3 | IF wmc is more or less (average) AND dit is low AND rfc is low AND loc is low THEN bug is 2.0 |
| R4 | IF wmc is low AND dit is more or less (average) AND rfc is low AND loc is low THEN bug is 1.0 |
| R5 | IF wmc is more or less (average) AND dit is low AND rfc is average AND loc is low THEN bug is 1.0 |

**Table 12** List of variables after applying simplification process for velocity module.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, more or less (average) | [0, 153] |
| DIT | Low, more or less (average) | [1, 5] |
| RFC | Low, average, high | [0, 250] |
| LOC | Low, more or less (average) | [0, 13175] |
| BUG | 1.0, 2.0 | [1, 2] |

**Table 13** Fuzzification of the dataset of Xalan module.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [0, 130] |
| DIT | Low, average, high | [1, 8] |
| NOC | Low, average, high | [0, 29] |
| CBO | Low, average, high | [0, 173] |
| RFC | Low, average, high | [0, 391] |
| LCOM | Low, average, high | [0, 7393] |
| LOC | Low, average, high | [0, 4275] |
| BUG | 1.0, 2.0 | [1, 2] |

**Table 14** List of rules after applying simplification process for Xalan module.

| Rules | Linguistic rules |
|---|---|
| R1 | IF dit is low AND noc is low AND rfc is NOT(high) THEN bug is 2.0 |
| R2 | IF wmc is NOT(high) AND noc is NOT(high) AND cbo is more or less (average) AND rfc is NOT(high) THEN bug is 2.0 |
| R3 | IF wmc is NOT(high) AND dit is NOT(high) AND noc is low AND cbo is low AND rfc is average THEN bug is 2.0 |
| R4 | IF wmc is average AND dit is average AND noc is low AND cbo is low AND rfc is low THEN bug is 2.0 |
| R5 | IF wmc is low AND dit is high AND noc is low AND cbo is low AND rfc is low THEN bug is 2.0 |
| R6 | IF wmc is low AND dit is average AND noc is low AND cbo is low AND rfc is low THEN bug is 1.0 |
| R7 | IF wmc is low AND dit is low AND noc is average AND cbo is low AND rfc is low THEN bug is 1.0 |

**Table 15** List of variables after applying simplification process for Xalan module.

| Variable | Linguistic terms | Range |
|---|---|---|
| WMC | Low, average, high | [0, 130] |
| DIT | Low, average, high | [1, 8] |
| NOC | Low, average, high | [0, 29] |
| CBO | Low, more or less (average) | [0, 173] |
| RFC | Low, average, high | [0, 391] |
| BUG | 1.0, 2.0 | [1, 2] |

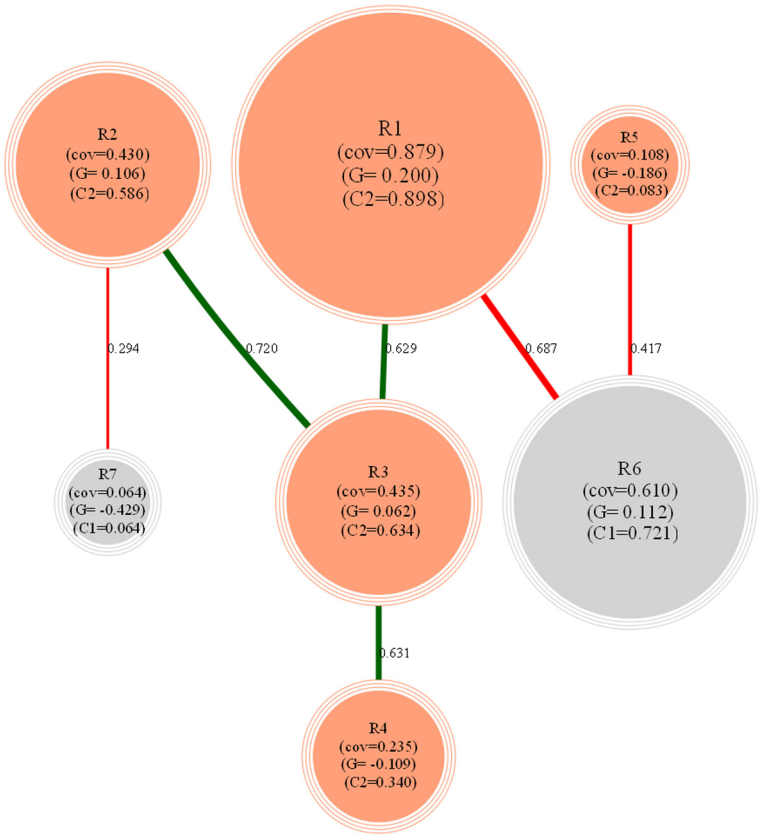


**Figure 6** Fingram corresponds to the reduced set of rules for Xalan module.

| | Low | Average | NOT (High) | High |
|---|---|---|---|---|
| WMC | ▲▲▲ | | ▲▲▲▲ | |
| NOC | ▲▲▲▲▲▲ | | ▲ | |
| RFC | ▲ | ▲▲ | ▲▲▲▲ | |
| LCOM | | | | |
| DIT | ▲▲▲▲▲ | ▲ | ▲▲ | |
| CBO | ▲ | ▲▲ | ▲▲▲ | |
| LOC | ▲▲ | | | |

**Figure 7** Frequency count of fuzzified variables.

might be taken up by other metrics. Earlier sections have demonstrated this fact with a corresponding entry to the correlation matrix.

This study explores the degree of interaction between metrics promoting the notion of compound effects in the model development, primarily in the software engineering domain. One associated objective is the identification of prominent metrics.

### 4.5. Accuracy of developed models

Following cases arise, while examining the execution of a FIS in classification problems (Alonso and Magdalena, 2011b)

1. *Unclassified cases*: Data samples that do not fire at least one rule beyond a threshold of activation degree of a rule.
2. *Ambiguity causes*: Data samples where more than one rule gets fired with varying level of activation.
3. *Error cases*: Data samples, where observed and inferred output differ.
4. *Ambiguity cases*: Can further be categorized into a separate category, where the ambiguity heads to the error cases.

*Based on the aforementioned cases, following statistical measures are used to ascertain the performance of the model in this study (Alonso et al., 2012);*

1. *Precision*: Ratio of the number of data samples correctly classified to the total figure of data samples (correctly classified and mistakenly classified as correct). Probabilistically, this is the average probability of the relevance of a class.It is expressed as a ratioPrecision = True positive/(True positive + False positive)
2. *Recall*: Ratio of the number of data samples correctly classified to the total figure of data samples (correctly classified and mistakenly classified as faulty). Probabilistically, this is the average probability of proper classification. It is expressed as a ratioRecall = True positive/(True positive + False negative)
3. *F-measure*: Harmonic mean of precision and recall and expressed as followsF-measure = 2 * ((Precision * Recall)/(Precision + Recall))

The table below displays the accuracy of FIS for all four software modules.

FIS further performs reasonably well to the accuracy part. The results presented in Table 16 are comparable to the estimates reported in published studies (Okutan and Yildiz, 2012). The effectiveness of a particular metric may vary with the internal procedural implementations linked to the project, and inconsistent conclusions are usually reported in this domain. This further strengthens reasoning made in the publications using regression analysis (Goyal et al., 2013a,b, 2014, 2015) that the issue of interactions plays an important role.

**Table 16** Accuracy measures of FIS.

| Module | Precision | Recall (Sensitivity) | F-measure |
|---|---|---|---|
| Lucene | 0.519 | 0.505 | 0.508 |
| Poi | 0.629 | 0.563 | 0.517 |
| Velocity | 0.766 | 0.755 | 0.759 |
| Xalan | 0.653 | 0.648 | 0.650 |

## 5. Conclusions

Software development is a human activity, and fuzzy modeling not only effectually manages uncertainty in human-centric systems analysis, but also simulates the development of the customized predictive models, even with the limited availability of historical data. This study presented the FIS based mechanism to encompass a different character of interaction while also making a fairly reliable classifier of faulty classes. FIS maintains progressive addition of expert experience to express effective interactions.

## Conflict of interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

Corresponding author would take this opportunity to thank Dr. Jose M, Alonso, Associate Researcher (European Centre for Soft Computing, Spain) for providing valued support in making GUAJE Framework operative.

## References

Ahmed, M.A., Muzaffar, Z., 2009. Handling imprecision and uncertainty in software development effort prediction: a type-2 fuzzy logic based framework. Inf. Softw. Technol. 51 (3), 640–654.

Aljahdali, S., Sheta, A.F., 2011. Predicting the reliability of software systems using fuzzy logic. In: 2011 Eighth International Conference on Information Technology: New Generations (ITNG). IEEE, pp. 36–40.

Alonso, J.M., Magdalena, L., 2011a. Generating understandable and accurate fuzzy rule-based systems in a java environment. In: Fuzzy Logic and Applications, vol. 15. Springer, pp. 212–219.

Alonso, J.M., Magdalena, L., 2011b. HILK + +: an interpretability-guided fuzzy modeling methodology for learning readable and comprehensible fuzzy rule-based classifiers. Soft. Comput. 15 (10), 1959–1980.

Alonso, J.M., Cordon, O., Quirin, A., Magdalena, L., 2011. Analyzing interpretability of fuzzy rule-based systems by means of fuzzy inference-grams. In: World Congress on Soft Computing.

Alonso, J.M., Pancho, D.P., Magdalena, L., 2012. Enhancing the fuzzy modeling tool GUAJE with a new module for fingrams-based analysis of fuzzy rule bases. In: 2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, pp. 1–8.

Arisholm, E., Briand, L.C., Johannessen, E.B., 2010. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J. Syst. Softw. 83 (1), 2–17.

Basili, V.R., Briand, L.C., Melo, W.L., 1996. A validation of object-oriented design metrics as quality indicators. IEEE Trans. Softw. Eng. 22 (10), 751–761.

Bouktif, S., Ahmed, F., Khalil, I., Antoniol, G., 2010. A novel composite model approach to improve software quality prediction. Inf. Softw. Technol. 52 (12), 1298–1311.

Catal, C., Diri, B., 2007. Software fault prediction with object-oriented metrics based artificial immune recognition system. Product-Focus. Softw. Process Improv., 300–314.

Chidamber, S.R., Kemerer, C.F., 1994. A metrics suite for object oriented design. IEEE Trans. Softw. Eng. 20 (6), 476–493.

Chidamber, S.R., Darcy, D.P., Kemerer, C.F., 1998. Managerial use of metrics for object-oriented software: an exploratory analysis. IEEE Trans. Softw. Eng. 24 (8), 629–639.

Chiu, N., 2011. Combining techniques for software quality classification: an integrated decision network approach. Expert Syst. Appl. 38 (4), 4618–4625.

Engel, A., Last, M., 2007. Modeling software testing costs and risks using fuzzy logic paradigm. J. Syst. Softw. 80 (6), 817–835.

Fenton, N.E., Neil, M., 1999. A critique of software defect prediction models. IEEE Trans. Softw. Eng. 25 (5), 675–689.

Goyal, R., Chandra, P., Singh, Y., 2013a. Impact of interaction in the combined metrics approach of fault prediction. Softw. Qual. Prof. (ASQ) 15 (3), 15–23.

Goyal, R., Chandra, P., Singh, Y., 2013b. Identifying influential metrics in the combined metrics approach of fault prediction. SpringerPlus 2 (1), 627.

Goyal, R., Chandra, P., Singh, Y., 2014. Suitability of KNN regression in the development of interaction based software fault prediction models. IERI Procedia 6, 15–21.

Goyal, R., Chandra, P., Singh, Y., 2015. Comparison of M5'Model Tree with MLR in the development of fault prediction models involving interaction between metrics. In: New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering, Springer International Publishing, pp. 149–155.

Gray, A., MacDonell, S., 1997. Applications of fuzzy logic to software metric models for development effort estimation. In: Fuzzy Information Processing Society. IEEE, pp. 394–399.

Guillaume, S., Charnomordic, B., 2011. Learning interpretable fuzzy inference systems with FisPro. Inf. Sci. 181 (20), 4409–4427.

Jureczko, M., Madeyski, L., 2010. Towards identifying software project clusters with regard to defect prediction. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, vol. 9. ACM.

Khoshgoftaar, T.M., Seliya, N., 2003. Analogy-based practical classification rules for software quality estimation. Empir. Softw. Eng. 8 (4), 325–350.

MacDonell, S.G., 2003. Software source code sizing using fuzzy logic modeling. Inf. Softw. Technol. 45 (7), 389–404.

Meneely, A., Williams, L., Snipes, W., Osborne, J., 2008. Predicting failures with developer networks and social network analysis. In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, pp. 13–23.

Okutan, A., Yildiz, O.T., 2012. Software defect prediction using Bayesian networks. Empir. Softw. Eng., 1–28

Ozcan, F., Atbackslashcs, C.D., Karahan, 2009. Comparison of artificial neural network and fuzzy logic models for prediction of long-term compressive strength of silica fume concrete. Adv. Eng. Softw. 40 (9), 856–863.

Pancho, D.P., Alonso, J.M., Magdalena, L., 2013a. Quest for interpretability-accuracy trade-off supported by fingrams into the fuzzy modeling tool GUAJE. Int. J. Comput. Intell. Syst. 6 (sup1), 46–60.

Pancho, D.P., Alonso, J.M., Alcalá-Fdez, J., 2013b. A new fingram-based software tool for visual representation and analysis of fuzzy association rules. In: 2013 IEEE International Conference on Fuzzy Systems (FUZZ). IEEE, pp. 1–7.

Pandey, A.K., Goyal, N.K., 2009. A fuzzy model for early software fault prediction using process maturity and software metrics. Int. J. Electron. Eng. 1 (2), 239–245.

So, S.S., Cha, S.D., Kwon, Y.R., 2002. Empirical evaluation of a fuzzy logic-based software quality prediction model. Fuzzy Sets Syst. 127 (2), 199–208.

Verma, H.K., Sharma, V., 2010. Handling imprecision in inputs using fuzzy logic to predict effort in software development. In: 2010 IEEE 2nd International Advance Computing Conference (IACC). IEEE, pp. 436–442.

Weyuker, E.J., Ostrand, T.J., Bell, R.M., 2007. Using developer information as a factor for fault prediction. In: Proceedings of the Third International Workshop on Predictor Models in Software Engineering, vol. 8. IEEE Computer Society.