Australian Journal of Science and Technology

ISSN Number (2208-6404)
Volume 3; Issue 2; June 2019

Original Article

Evaluating software components reusability using genetic-fuzzy soft computing approach

O. Ajayi Olusola*

Department of Computer Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria

ABSTRACT

The quest to develop software of great quality with timely delivery and tested components gave birth to reuse. Component reusability entails the use (reuse) of existing artifacts to improve the quality and functionalities of software. Many approaches have been used by different researchers and applied to different metrics to assess software component reusability level. In addition to the common quality factors used by many authors, such as customizability, interface complexity, portability, and understandability, this study introduces and justifies stability, in the context of volatility, as a factor that determines the reusability of software components. Sixty-nine software components were collected from the third-party software vendors, and data extracted from their features were used to compute the metric values of the five selected quality factors. Genetic-fuzzy system (GFS) was used to predict the level of the components' reusability. The GFS was implemented using MATLAB. The performance of the GFS was compared with that of adaptive neuro-fuzzy inference system (ANFIS) approach using their corresponding average root-mean-square error (RMSE), to ascertain the level of accuracy of the prediction. The results of the findings showed that GFS with an RMSE of 0.0019 provides better reusability prediction accuracy compare to ANFIS with an RMSE of 0.1480. The experiment conducted also proved Java components to be more reusable than the other two components used in the study. This work was able to establish a GFS for the evaluation of software component reusability, with the results proving the new system a better predictor than the most commonly used system (ANFIS).

Keywords: Adaptive neuro-fuzzy, agile development, genetic algorithm, genetic fuzzy, reusability, soft computing, software component

Submitted: 02-05-2019, **Accepted:** 23-05-2019, **Published:** 29-06-2019

BACKGROUND TO THE STUDY

Washizaki *et al*.[1] defined a software component as a unit of composition with contractually specified interface and explicit context dependencies only. He said a software component can be deployed independently and is subjected to composition by third party. A component can be a coherent package of software that can be independently developed and delivered as a unit and that offers interfaces by which it can be connected unchanged with other components to compose a larger system.[2] A software component is a reusable piece of code or software in binary form, which can be plugged into components from other vendors with relatively little efforts. They are black box entities that encapsulate services behind well-defined interfaces, which tend to be very restricted in nature, reflecting a particular model of plug compatibility supported by a component framework, rather than being very rich and reflecting real-world entities of the application domain.[3]

Component-based software development (CBSD) is a development approach in which systems are built from well-defined, independently produced pieces by combining the pieces with self-made components.[4] CBSD is a paradigm that aims at constructing and designing systems using a predefined set of software components explicitly created for reuse [Figure 1]. Component-based systems achieve flexibility by clearly separating the stable parts of the system (that is, the components) from the specification of their composition.[3]

Reusability is the degree to which a software component can be reused.[1,5] This consequently leads to reduced software development cost and less development time as it enables less writing but more of assembly. Reusability plays an important role in CBSD and also acts as the basic criterion for evaluating component [Figure 2]. Kumar *et al*.[6] asserted that reusability of a component is an important aspect,

Address for correspondence: O. Ajayi Olusola, Department of Computer Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria. E-mail: olusola.ajayi@aaua.edu.ng

which gives the assessment to reuse the existing developed component, thereby reducing the risk, cost, and time of software development. If a component is not reusable, then the whole concept of CBSD fails.[7] Reusability is one of the quality attributes of CBSD. It can measure the degree of features/components that are reused in building similar or different new software with minimal change.[8] To realize the reuse of components effectively, reusability estimation has to be carried out. For systematic reuse process, the use of metrics is very germane. Without metrics, evaluating the quality and qualification of the selected components for reuse becomes an uphill task.[8] Goel and Sharma[9] defined reusability as the quality of any software component to be used again with slight or no modification. Software reuse is the process of creating software systems from existing software assets rather than building them from scratch. Reusability was also viewed as the quality factor of software that qualifies it to be used again in another application, be it partially modified or completely modified. In other words, software reusability is a measure of the ease with which previously acquired concepts and objects can be used in new contexts. Kumar *et al*.[10] seen reusability of a component as an important aspect, which gives the assessment to reuse the existing developed component. Singh and Tomar[8] viewed reusability as a physical replaceable part of a system that adds functionality to the system, through the realization of a set of interfaces. The components having well-defined interfaces can be considered good for reuse. The interfaces have strong significance in context of reusability of components Figure 3.

Metrics, however, play an indispensable role in the successful evaluation of software component reusability. According to Washizaki *et al*.[1], it is necessary to measure reusability of software components to realize the effective reuse of such components. According to the author, metrics are used to determine quality factors that affect reusability. A component alone has certain characteristics that tend to affect its reusability. Quality factors are chosen to provide an analysis of the reusability of a component. The choice of factors affecting reusability is considered based on activities carried out while reusing the components.

Unlike in the past, where researchers employed statistical methods of predicting reusability[1,11], recent interdisciplinary techniques such as fuzzy logic, artificial neural network (ANN), and neuro-fuzzy have taken the lead due to their power of predictability.[6,9,12,13] This work investigates the works of Kumar *et al*. Sharma *et al*. and Sagar *et al*.[6,12,13] and Goel and Sharma,[9] who all adopted soft computing approach to predict reusability of software component, but with varying degree of accuracy. The problem of applying a method that yields the best accuracy level and the need to establish stability (in the

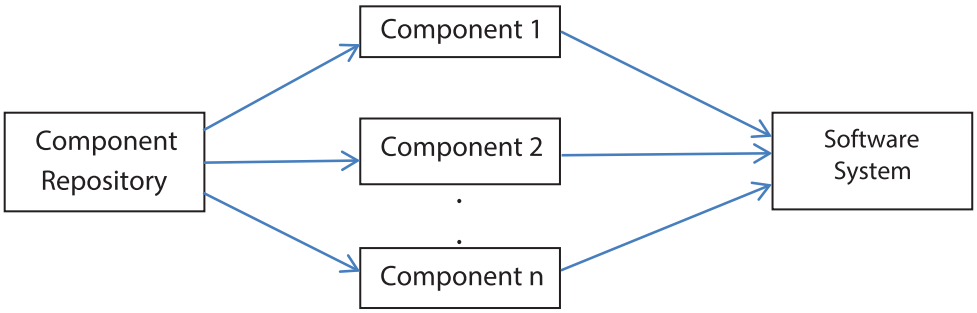

Figure 1: Schematic view of component-based software development Approach (Kaur and Singh, 2013)

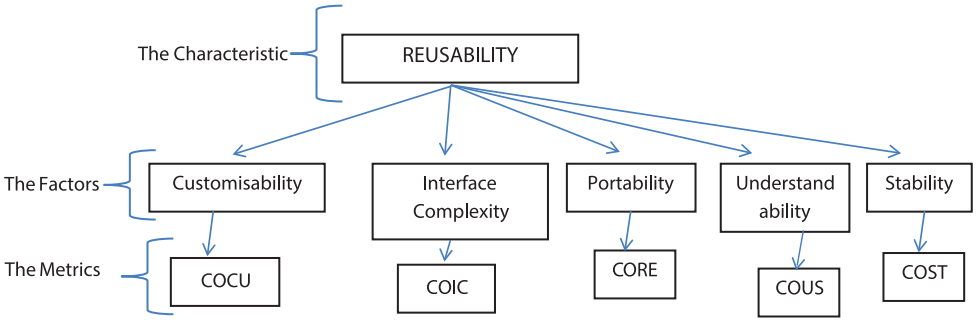

Figure 2: Component reusability tree – based on the study's identified metrics, where: COCU: Component customizability, COIC: Component interface complexity, CORE: Completeness of component return, COUS: Component understandability, COST: Component stability

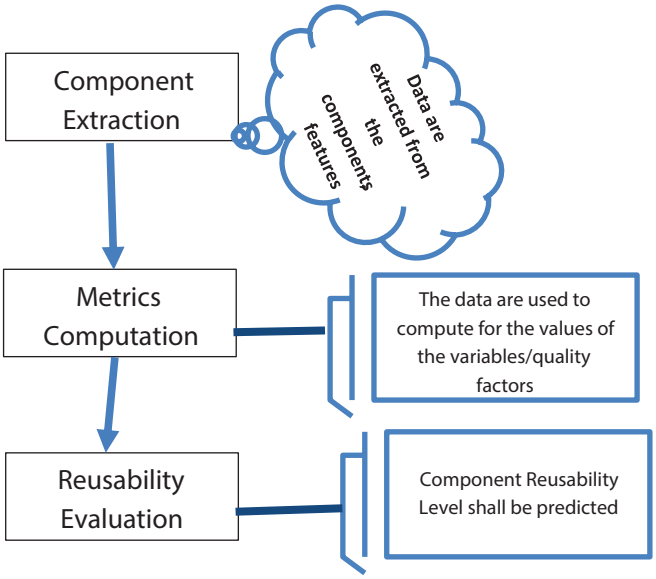

Figure 3: Study methodology

context of volatility) as a factor for determining component reusability motivated this study to lend a voice to the domain of component reusability. This research work presents a genetic-fuzzy system (GFS) with stability in the context of volatility. The result of the work will be compared with the result obtained using adaptive neuro-FIS (ANFIS) method since researches have shown that ANFIS predicts more accurately than ANN and FIS.[6,9]

RELATED WORK

Researchers have adopted the use of statistical approaches like correlation analysis, while some made use of soft computing techniques such as ANN and fuzzy logic to evaluate component reusability.

Washizaki *et al*.[1] applied statistical method to component reusability assessment issue. Metric suites for measuring reusability of software components were developed. In implementing the work, component overall reusability model was developed to assess and evaluate Java web components. The study proposed three quality factors as criteria for measuring reusability characteristic, while five metrics were deployed for the measurement. The factors are understandability, adaptability, and portability, while the metrics include existence of metainformation, Rate of Component Observability (observability) – for measuring understandability, Rate of Component Customizability (customizability) – for measuring adaptability, Self-completeness of Component's Return Value, and Self-completeness of Component's Parameter – for measuring portability. The result of the analysis conducted using 125 Java web components from www.jars.com shows that the proposed metrics were suitable. However, the empirical study was limited to evaluation with Java beans components; as other component technologies such as. Net and ActiveX were not explored for further validation.

Rotaru and Dobre[14] addressed reusability from the perspective of adaptability, composability, and complexity metrics. The work aimed to cover the main aspects of reusable software components, which in their opinion are composability and adaptability. Both factors were evaluated based on the complexity of the component interface. The major contribution of the work, which adopted qualitative approach, was the formulation of metrics and design of a mathematical model for practical assessment of the specified software component characteristics. The proposed model is, however, required to be validated by assessing several software components based on it.

Sharma *et al*.[12] contributed largely to software component reusability works by proposing an ANN soft computing-based approach to assess the reusability of software components. The work aimed at aiding developers to select the best component in terms of its reusability. In their research, four factors, on which reusability of components depends, were identified. These are customizability, interface complexity, portability, and understandability. The empirical work was carried out with 40 components collected from www.jars.com and www.elegantjbeans.com. Applying ANN soft computing approach, network is trained on training data by considering different number of hidden neurons for two training functions, namely *trainlm* and *trainbr*, to get the best results. This network was further validated by applying the proposed approach on test data. The adaptation learning function selected for the experiment was "*learngdm*." Performance function used was root-mean-square error (RMSE), with "*tan-sigmoid*" as the transfer functions in both layers. Results obtained showed that the network was able to predict the reusability of components with optimum performance and with an RMSE of 0.1348 using *trainlm* as the training function. The limitation of the work was in the limited number of data used to train the network. It was submitted that using more number of components may produce better results/accuracy for the training and testing.

Sagar *et al*.[13] discussed reusability in relation to component-based development (CBD) and proposed a reusability metrics for black-box components. It identifies factors affecting reusability as customizability, interface complexity, portability, and document quality. In the study, fuzzy logic-based approach was used to estimate the reusability of components using triangular membership functions. The authors used two classroom-based Java beans components, namely Calculator and Chart B for validation. Reusability values of 0.71 and 0.3124 were

arrived at proving that FIS is able to predict reusability of components with an acceptable level of accuracy. Further, it was submitted that the adopted approach can be validated against other approaches for estimated reusability of components.

Singh and Toora[15] applied neuro-fuzzy technique on a case study which they took from a reputed journal. The case study was concerned with the reusability of software components. The reusable components/attributes were coupling, complexity, volume regularity, and reuse frequency. They proved that neuro-fuzzy model yields less percentage average error as compared to standalone fuzzy logic and neural network. It also produces greater accuracy for software reusability as compared to FIS and ANN.

Kamalraj *et al*.[16] proposed concept of "stability-based clustering" method focusing on "stability" metric of component(s) and "clustering analysis" of data mining. Data mining technique that may help to maintain the reuse repository with quality reusable components was proposed. The data mining was used for analyzing bulk of data to extract the knowledge from them. Applying data mining on software engineering to simplify the data handling results in reduced efforts and cost in various aspects. Data mining was given very effective approach like "clustering analysis" to group the elements as per the required data item. Stability is an essential factor to represent the kind of dependency among components and communication among the components and their interior elements. Hence, by applying "stability-based reuse component repository," it can help the total system development with higher productivity in a very short period. In the research, stability was only introduced to track the type of dependency among components, communication among them and their interior elements; thus, stability was not used to determine the level of reusability of components.

In Jatain and Gaur,[17] emphasis was laid on the estimation of reusability of components by identifying some quality attributes of components which influence reusability. The five identified factors are customizability, configurability, interface complexity, portability, and compatibility. Fuzzy logic was employed as the soft computing approach adopted to test for reusability of four components. The approach was used to estimate reusability of some real-time projects. This result showed that to enhance the reusability factor of component, its customizability, configurability, compatibility, and portability should be high, whereas interface complexity should be low. However, the study's limitation hinges on further validation of the approach used.

Ravichandran *et al*.[18] developed an automated process of component selection using ANFIS-based technique using 14 reusable component's parameters. Neuro-fuzzy-based approach was adopted to select optimal reusable components efficiently. The developed approach was validated with three data sets for three proposed software architectures. The results showed that the proposed approach was able to predict the reusability of these components with an acceptable accuracy. However, stability was used as a fuzzy input with variables such as low, medium, and high in the ANFIS structure, without reference to porting of the components as suggested in their definition.

Christopher and Chandra[19] proposed a multicriteria fuzzy-based approach for predicting software requirement stability based on complexity point measurement and for finding out the complexity weight based on requirement complexity attributes such as functional requirement complexity, non-functional requirement complexity, input-output complexity, interface, and file complexity. The research paper discussed the importance of measuring the requirements changes for the lack of instability in the requirements. The prediction model for requirements stability approach provides the solution for measuring the requirements changes based on the complexity point measurement model. The work, however, did not justify nor demonstrate the applicability of the model for developing maintenance and transition projects based on different complexity attributes and different adjustment factors.

Aversano[20] provided the subset of the architectural components of the software project that could be actually reused. The paper presented an empirical study aimed at assessing software architecture stability and its evolution along the software project history. The study entailed the gathering and analysis of relevant information from several open source projects. The paper evaluated the stability of the core architecture during the development cycle of each software project, by adopting two metrics defined in the initial stage of the process. The analysis performed considered software systems developed using different paradigms, with different evolution trends and concerning different application domains. The work described in the paper was basically devoted to the study of the stability of the architectural core of a software project with the aim of understanding the potential reusability of their software component. The study only handled only stability measurement as it is related to architectural level of software leaving other aspects in which stability can be applied.

Kumar *et al*.,[6] however, adopted multidisciplinary technique of ANFIS in the assessment of component reusability. In the study, four dependent factors, namely customizability, interface complexity, understandability, and portability, were used to estimate the reusability of software components. The result obtained using ANN approach and using data from

Sharma *et al*. (2009) was an RMSE of 0.1852. Applying ANFIS approach to the same set of data yielded an RMSE of 0.1695, which shows that neuro-fuzzy gives a better and more accurate reusability result. The comparative analysis of the proposed ANFIS and the existing ANN was carried out on 48 Java components. It was, however, opined that accuracy of the used method is subject to availability of substantial number of data/components.

Goel and Sharma[9] taken into account three different factors for determining reusability of software components and then proposed a model for reusability assessment using the ANFIS. The quality factors used include coupling, complexity, and portability. The experiment used 338 records retrieved from open source produced an RMSE of 0.042482. It was suggested that new factors such as understandability, cohesion, clarity, and generality can also be added, and the cumulative effect of those factors can be seen on the future predictions. Furthermore, different techniques can be used other than ANFIS to predict reusability such as support vector machine. Finally, it was submitted that a much better generalized approach is expected if real-time data are considered.

Singh and Tomar[8] identified four attributes for estimating reusability of black-box components. The reusability metric was parameterized using component interface complexity, component understandability, component customizability, and component reliability. The project made use of file upload component of the Apache Commons project. The work proved that the proposed metrics were able to determine reusability. It was, however, submitted that the work requires further validation, suggesting that the weight values for the estimation of reusability be adjusted using neural networks.

Ekanem and Woherem[21] presented techniques for assessing the stability of components extracted from legacy applications using software maturity index. The research presents a technique for assessing the stability of components extracted from legacy applications using software maturity index. The practical demonstration of the approach was based on maintenance data generated with RANDBETWEEN function of spreadsheet package on three legacy applications used in the demonstration. The research work was designed as experimental research with the following processes: (i) Review of relevant documentation, (ii) randomization of the needed research data using RANDBETWEEN function in spreadsheet program, (iii) data coding and analysis, and (iv) results interpretation and discussions. The ranking scheme comprises the following ordered items, highly stable, fairly stable, stable, unstable, fairly unstable, and highly unstable. However, stability of legacy components was measured using maturity index but with no recourse to the reusability of the component.

The works of Kumar *et al*.[6] and Goel and Sharma[9] surpassed others in terms of result accuracy as a result of the approach used (ANFIS). Goel and Sharma,[9] however, call for the validations of the various results using different approaches and experimenting with components other than Java components. This work, therefore, researches into these noticeable gaps as a way of contributing to works on component reusability assessment.

METHODOLOGY FOR THE STUDY

This study adopts:

1. CBD approach: This methodology helps to build component analysis tool for accessing common software components;
2. Metric-based approach: This methodology aids to measure the degree to which a component is reusable Table 1;
3. Soft-computing approach: This methodology predicts the certainty for reusability.

The following procedures were followed in ensuring a successful implementation of the work:

1. Commercial off-the-shelf software components were extracted from the third-party software vendors. According to Sharma *et al*.,[4] the key to the success of CBSD is its ability to use software components that are often developed by and purchased from the third party.

Component Data Extraction

1. Sixty-nine software components were gotten from four different third-party component development organizations (www.elegantjbeans.com, www.jidesoft.com, www.math.hws.edu, and www.codeproject.com).

Table 1: Metrics and the quality factors they measure

S. No.	Metric	Quality factor to measure	
i.	COCU – Component customizability	Customizability (C)	CIPUS
ii.	COIC – Component interface complexity	Interface complexity (I)	
iii.	CORE – Completeness of component return	Portability (P)	
iv.	COUS – Component understandability	Understandability (U)	
v.	COST – Component stability	Stability (S)	

Table 2 shows the sources, nature, and numbers of the components.

2. Appropriate metrics for each quality factor that qualifies the characteristic, reusability, were applied. We consider the same quality factors as used by the duo of Sharma *et al*.[12] and Kumar *et al*.,[6] with stability (in the context of volatility) as an addendum Table 2.
3. Genetic-fuzzy soft computing approach was deployed for evaluating the level of reusability of the selected components. GFS is a system that exploits genetic algorithms to automatically generate or optimize the knowledge base of a fuzzy system since the fuzzy system is not able to learn on its own. Researches have shown that hybridized genetic algorithm gives a more accurate predictive result.[22-25]

DESIGN

Adapting Kumar *et al*. (2013) approach and establishing the need for stability as a factor for component reusability measurement,

$$\text{Let } R_{cn}=F_{cn}[X_n, Y_n, Z_n, J_n, K_n] \quad (1)$$

Where:

R_{cn} is the reusability of component.

F_{cn} is implemented using genetic fuzzy with X_n, Y_n, Z_n, J_n, and K_n as input-dependent variables, representing customizability (component customizability), interface complexity (component interface complexity), portability (completeness of component return), understandability (component understandability), and stability (component stability), respectively.

In the proposed model, GFS is developed, trained, and tested using MATLAB software. The steps involved in the development of the system [Figure 4] are as follows:

1. Extract component data
2. Compute the metric value of X_n, Y_n, Z_n, J_n, and K_n
3. Represent the variables in Fuzzy format
4. Load values of X_n, Y_n, Z_n, J_n, and K_n into fuzzy toolbox
5. Apply the Genetic Optimizer to tune the knowledge base
6. Compute the fitness value until the threshold/termination is reached.

The detailed model is presented in Figure 5

ORGANIZATIONAL STRUCTURE

The operational structure of the GFS for component reusability prediction was constructed using UML (use case, sequence, and activity diagrams) to describe the logical design that is implementation-independent design of the system. This shows the system's components and their relationships as it appears from user's inputs to processing of the tasks.

Use Case

Figure 6 shows the major users of the system, namely the software developers, the component developers, the component library administrator, and the component users. They all have possible access to six major operations, which are LOGIN, POPULATE DATA, RUN REUSABILITY TEST, GUIDE, FEEDBACK, and EXIT.

Sequence Diagram

Figure 7 shows the sequence diagram of the proposed system.

Activity Diagram

This is used to model the procedural flow of actions/events/activities that occurred in a system. It describes the use case and the sequence models. Figure 8 shows the system's activity diagram.

SYSTEM IMPLEMENTATION

System implementation refers a system life cycle phase in which the constructed system is tested and put into operation. It is the actualization of a specified designed and modeled system.

Table 2: Components used

S. No.	Component source	Nature of components	Number of components	Period of extraction
1.	www.elegantjbeans.com	Java components	48	March 2016
2.	www.jidesoft.com	Java components	4	April 2016
3.	www.math.hws.edu	Web components	13	October 2016
4.	www.codeproject.com	.Net components	4	November 2016

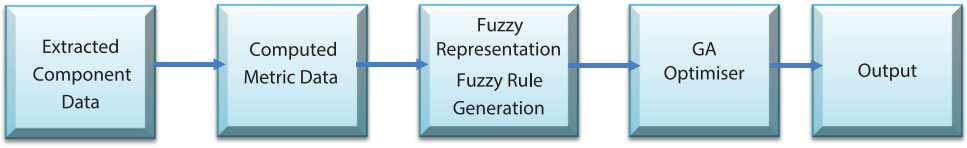

Figure 4: Component reusability prediction model

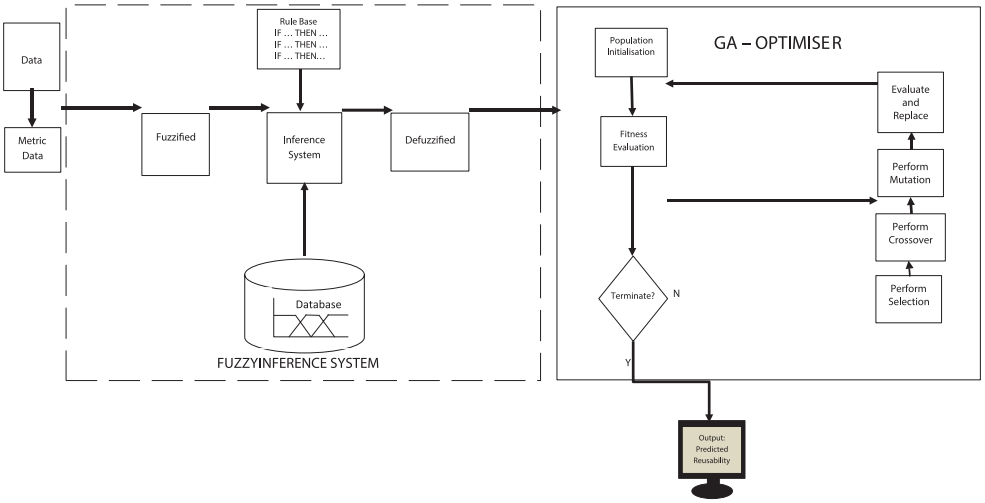

Figure 5: Detailed genetic-fuzzy model for component reusability prediction

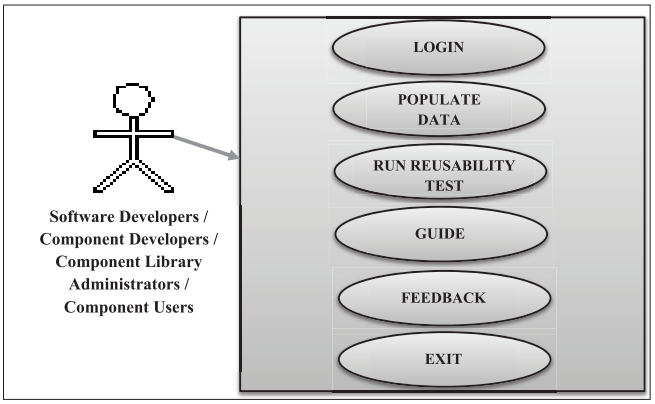

Figure 6: Use case diagram of the proposed system

Implementation Approach

Figure 9 is the adapted agile (feature driven) development model.

Implementation Flow

Figure 10 shows the flow diagram of the system implementation pattern (adaptive neuro-fuzzy inference system and genetic-fuzzy system).

Algorithm (ANFIS)

Select Loader
If loader = ANFIS, load cipus-run.m
browse to retrieve training data
load training data
if fileext = '*.csv', 'load successful'
else 'load unsuccessful', reload
endif
browse to retrieve testing data
load training data
if fileext = '*.csv', 'load successful'
else 'load unsuccessful', reload
endif
End Select
RUN Reusability R MSE
VIEW Reusability RMSE

Algorithm (GFS)

Select Loader
If loader = GFS, load myga.m
load fuzzy-excel formatted file (loaddata.m)
if fileext = '*.csv', 'load successful'
else 'load unsuccessful', reload
endif
load/call/invoke ga_fitfunc.m
if load_status = 'correct', proceed
else re-load/re-call/re-invoke fitness function
endif
End Select
RUN Reusability RMSE
VIEW Reusability RMSE

EXPERIMENTAL EVALUATION

The FIS Properties

Table 3 presents the details/structure of the FIS design properties.

The ANFIS Evaluation Parameters

Table 4 shows the specifications of the ANFIS evaluation parameters.

The GA Optimization Parameters and Algorithm

Table 5 shows the specifications of the parameters used for the GA.

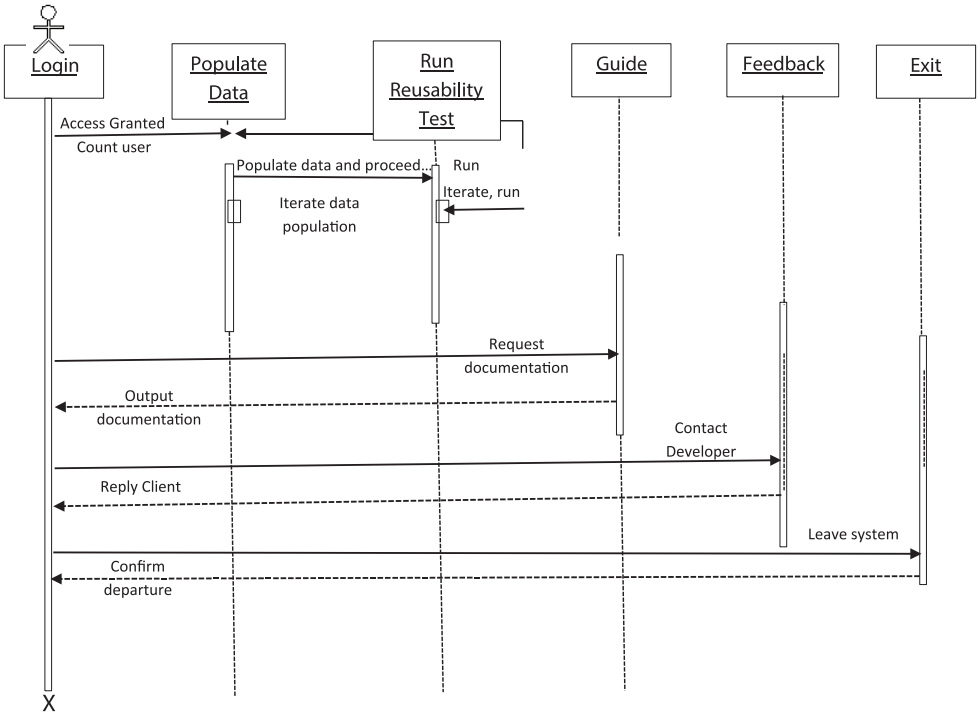

Figure 7: Sequence diagram of the proposed system

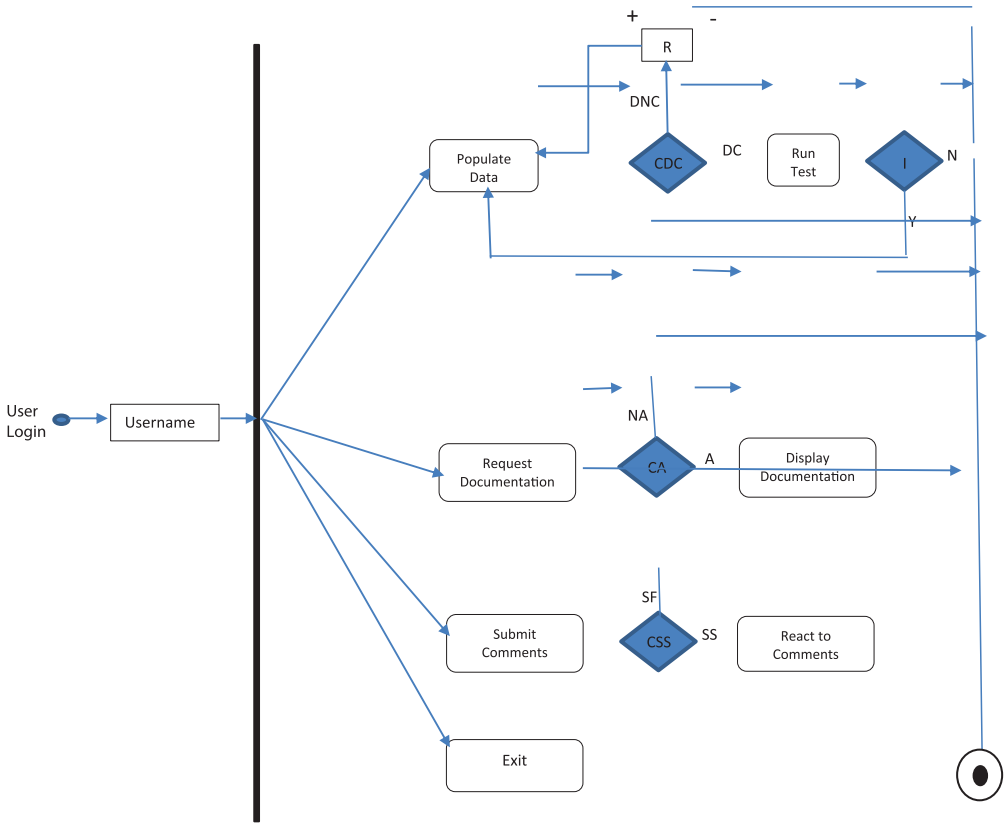

Figure 8: Activity diagram depicting the proposed system

Keys: CDC: Check data compatibility, DNC: Data not compatible, DC: Data compatible, R: Report on DNC, CA: Check availability, A: Available, NA: Not available, CSS: Check submission status, SS: Submission successful, SF: Submission failed, I: Iterate? Y: Yes, N: No

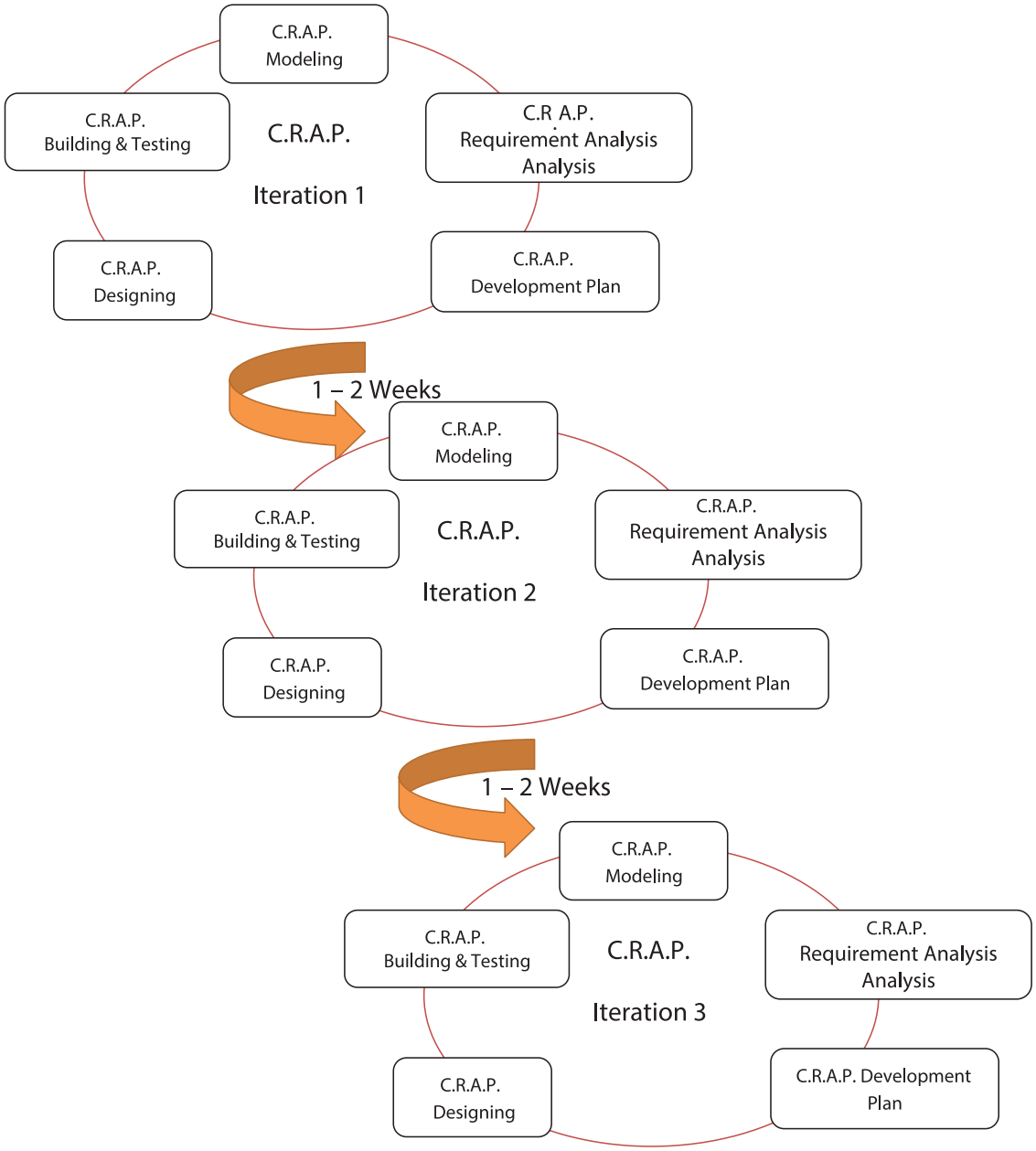

Figure 9: Adapted FDD model

Statistical Representation and Comparative Analysis

Table 6 shows the RMSE values of the two approaches (ANFIS and GFS) for the selected components.

Figure 11 represents the comparative chart for the ANFIS and GFSs RMSE in which GFS proved to have lower RMSEs (0.0019), implying better predictor.

Table 7 shows the aggregate values of Table 6 for the three components selected and for the five quality factors in use.

Analyzing with SPSS and using ANOVA (analysis of variance), the result is shown in Table 8.

From Table 8, java components proved more reusable as it recorded the least standard error (0.07935) compare to.Net component's 0.26680 and web component's 0.30975. Figure 12 shows the reusability prediction level of the various software components used.

FINDINGS

The followings are the findings from the study:

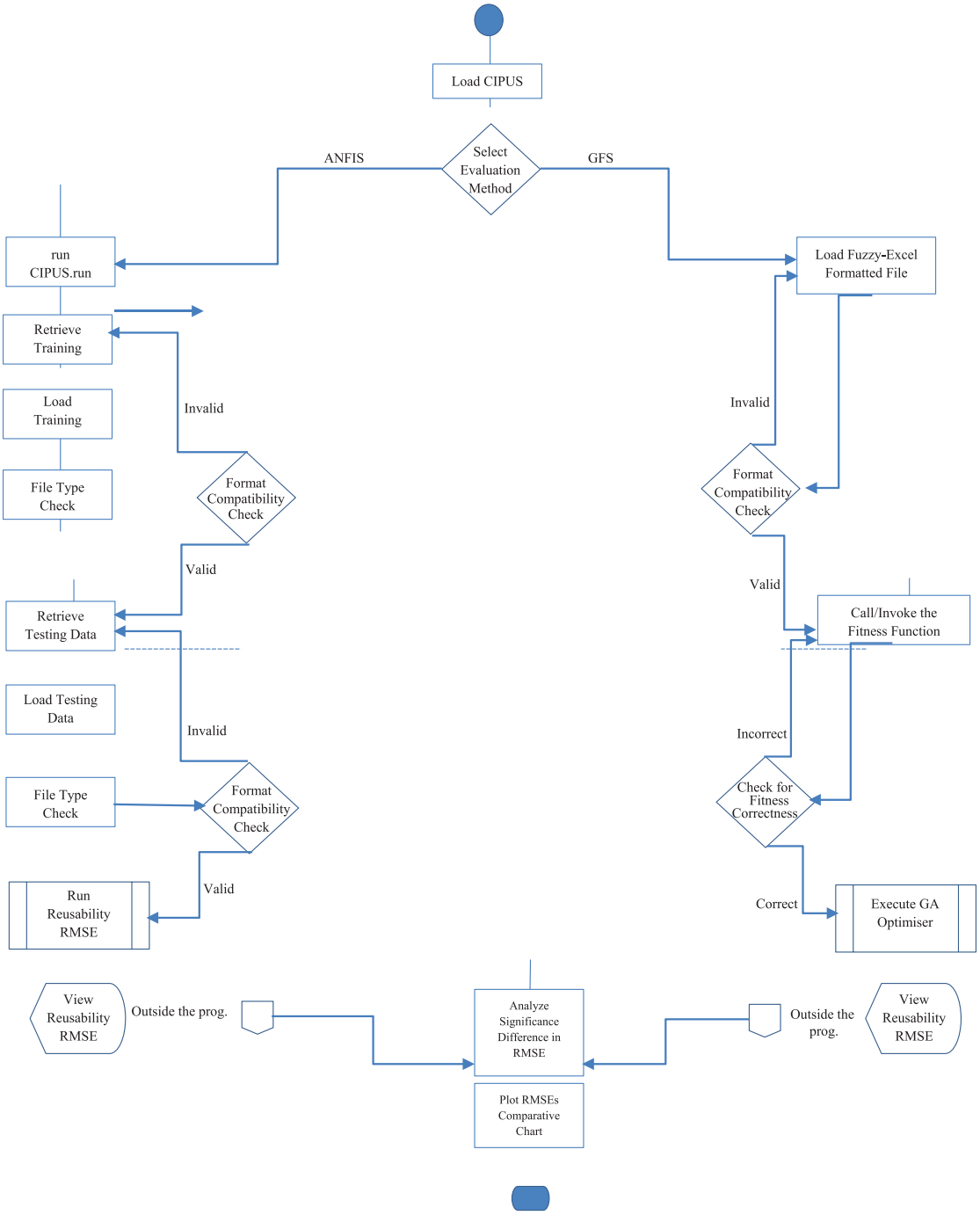

Figure 10: Implementation flow

1. The results of the findings show that GFS with an RMSE of 0.0019 provides better reusability prediction accuracy compare to ANFIS with an RMSE of 0.1480.
2. The experiments conducted showed that Java components, with an S.E. of 0.07935 proved more reusable compare to web component's S.E. of 0.30975 and.Net component's S.E. of 0.26680.

CONTRIBUTIONS TO KNOWLEDGE

The study established:

1. A GFS for the evaluation of software component reusability, with the results proving the new system a better predictor than the most commonly used system (ANFIS).
2. Stability (in the context of volatility) as a factor that also determines reusability. This study has been able to prove

Table 3: FIS structure/properties

Parameter	FIS names (s)	Property default/range value/ parameter range
Input parameter 1	COCU	[0 1]
Input parameter 2	COIC	[0 1]
Input parameter 3	CORE	[0 1]
Input parameter 4	COUS	[0 1]
Input parameter 5	COST	[0 1]
Input FIS type:		Sugeno
MF type:		Triangular
Output name:		Reusability
Output type:		Linear
Input parameters:		Low [0 0.25 0.5] Medium [0.25 0.5 0.75] High [0.5 0.75 1]
		Low [0 0.25 0.5] Medium [0.25 0.5 0.75] High [0.5 0.75 1]
		Low [0 0.25 0.5] Medium [0.25 0.5 0.75] High [0.5 0.75 1]
		Low [0 0.25 0.5] Medium [0.25 0.5 0.75 0] High [0.5 0.75 1 0]
		Low [0 0.25 0.5 0] Medium [0.25 0.5 0.75 0] High [0.5 0.75 1 0]
Output parameters:		Low [0 0 0 0 0 0] Medium [0.5 0.5 0.5 0.5 0.5 0.5] High [1 1 1 1 1 1]

FIS: Fuzzy inference system

that asides the commonly deployed attributes such as customizability, interface complexity, portability, and understandability (documentability), stability is a factor worthy of consideration while measuring reusability.

3. Software component assessment with other component types other than Java components. With researches showing that most studies on reusability of software components were done experimenting only with Java components, this study was able to carry out its assessment of component reusability using Java, web, and.Net components. The research took a leap to evaluate the level of reusability of each component, with Java components proving more reusable than the rest two component types. The study, therefore, contributed to the increasing body of knowledge that Java components are more reusable than other component types.

CONCLUSION

The essentiality of software component reusability no doubt aids software development cost and time, however, of greater necessity is the issue of measuring to ascertain the level of reusability of the selected software components for reusability. This, many researchers agreed with and deployed different evaluation techniques in assessing the level of reusability of software components.

Consequently, this work presented an evaluation of software components reusability using GFS. The study utilized five quality factors in measuring the reusability of 69 software components. The metric values for the selected five quality factors were computed using the data extracted from the components used. The design and detail analysis of the proposed system were elaborated upon. The system structure

Table 4: Adaptive neuro-fuzzy inference system specifications

S. No.	Parameters	Main attribute	Others
1.	Testing data	20 data	29% of the entire data used
2.	Training data	49 data	71% of the entire data used
3.	Number of epoch	50	
4.	Error tolerance	0	
5.	Rules	243	
6.	Logical operator	AND	
7.	Inputs	5	Customizability, interface complexity, portability, understandability, stability
8.	Input MF	3	Low, medium, and high
9.	Output	1	Reusability
10.	Output MF	3	Low, medium, and high
11.	Optimization method	Hybrid	

Table 5: GA specifications

Parameters	Main attribute	Others
Data	loaddata.m (matlab file)	x = csvread("data.csv")
Fitness function	ga_fitfunc (matlab function)	y = (x (1)+x (2)+x (3)+x (4)+x (5))/5
Population	Randomized	Constraint dependent
Bounds	Lower: [0 0 0 0 0] Upper: [1 1 1 1 1]	
Selection	Tournament	Size: 4
Mutation	Adaptive feasible	
Crossover	Two points (double)	
Stopping criteria	Generation	
Fitness scaling	Scaling function	Rank

Table 6: Components' RMSE values for ANFIS and GFS

S. No.	Component type	COCU	COIC	CORE	COUS	COST	RMSE (ANFIS)	RMSE (GFS)
1	Java components	1	1	0.92	0.9	1	0.1741	0.142
2	Java components	1	1	0.46	0.75	1	0.1727	0.142
3	Java components	0.91	1	0.46	0.75	0	0.1703	0.1367
4	Java components	0.81	1	0.37	1.05	1	0.1687	0.1367
5	Java components	0.5	1	0.25	0.45	1	0.1674	0.1367
6	Java components	0.75	0.5	0.68	0.75	0	0.1665	0.1367
7	Java components	0.75	0.5	0.68	0.7	1	0.1656	0.1367
8	Java components	0.93	1	0.82	1.05	1	0.1652	0.1367
9	Java components	1	1	0.5	0.45	0	0.1648	0.1305
10	Java components	0.73	0.68	0.44	1.05	0	0.1644	0.1305
11	Java components	0.78	1	0.57	1.2	1	0.1639	0.1305
12	Java components	1	1	0.65	1.2	1	0.1633	0.1302
13	Java components	0.75	0.5	1	1.05	1	0.1628	0.1273
14	Java components	0.74	0.48	1	1.05	0	0.1623	0.1263
15	Java components	0.98	1	0.77	1.2	1	0.1617	0.1263
16	Java components	0.79	1	0.63	1.2	1	0.1611	0.1263
17	Java components	1	1	0.61	1.2	1	0.1605	0.1177
18	Java components	0.89	1	0.65	1.2	1	0.1599	0.1052
19	Java components	0.82	0.74	0.37	1.05	1	0.1592	0.08672
20	Java components	0.8	0.68	0.86	1.05	1	0.1585	0.08672
21	Java components	0.91	1	0	0.9	1	0.1578	0.05859
22	Java components	0.92	0.86	0	0.9	1	0.1571	0.05859
23	Java components	0.92	0.92	0	0.9	1	0.1563	0.03984
24	Java components	0.97	0.93	0	0.9	1	0.1558	0.03672
25	Java components	0.94	0.96	0	0.9	1	0.1556	0.03672
26	Java components	0.92	0.94	0	0.9	1	0.1551	0.03672
27	Java Components	0.95	1	0.59	1.2	0	0.1549	0.03672

(Contd...)

Table 6: ***(Continued)***

S. No.	Component type	COCU	COIC	CORE	COUS	COST	RMSE (ANFIS)	RMSE (GFS)
28	Java components	0.86	1	0.55	1.2	0	0.1543	0.03672
29	Java components	0.96	1	0.69	1.2	1	0.154	0.03672
30	Java components	0.84	1	0.45	1.2	1	0.1535	0.03359
31	Java components	1	1	1	0.55	1	0.1525	0.03325
32	Java components	0.97	1	0.46	1.2	0	0.1515	0.02754
33	Java components	0.95	1	0.59	1.2	1	0.1523	0.02754
34	Java components	0.98	1	0.48	1.2	1	0.1513	0.02583
35	Java components	0.97	1	0.49	1.2	1	0.1513	0.02583
36	Java components	1	1	0.44	1.2	1	0.1515	0.01489
37	Java components	0.93	1	0.58	1.2	1	0.1512	0.01333
38	Java components	0.84	0.72	0	1.2	1	0.1515	0.01191
39	Java components	0.67	0.67	0	0.3	1	0.1511	0.01191
40	Java components	0.95	0.96	0	1.2	1	0.1514	0.01191
41	Java components	0.76	0.8	0.89	1.05	1	0.1507	0.01191
42	Java components	0.97	0.98	0.5	0.75	1	0.1513	0.01191
43	Java components	0.93	0.93	1	0.78	1	0.1505	0.01191
44	Java components	1	1	0.86	1.05	1	0.1508	0.01191
45	Java components	0.71	0.78	0.67	0.75	1	0.1499	0.01141
46	Java components	0.67	0.67	0.33	0.25	1	0.1507	0.01025
47	Java components	0.67	0.67	0.4	0.9	1	0.1497	0.007224
48	Java components	0.75	0.5	0	0.5	1	0.1506	0.007224
49	Java components	0.6	0.8	0.25	0.63	0	0.1494	0.007224
50	Java components	1	0.5	0	1	1	0.15	0.007224
51	Java components	1	0.5	0	0	0	0.1492	0.007224
52	Java components	1.15	0.44	0.8	0.45	0	0.1497	0.006442
53	Web components	0.5	0	4	0	0	0.1489	0.005661
54	Web components	0.5	0	2	0.58	1	0.1496	0.005661
55	Web components	1.11	0.45	0.74	0.47	1	0.1488	0.004099
56	Web components	0.5	0	1.5	0.5	1	0.1493	0.004099
57	Web components	0.86	0.58	2.25	0	1	0.1485	0.004099
58	Web components	1.07	0.47	0.85	1	1	0.1492	0.003925
59	Web components	1	0.5	0.83	0	1	0.1485	0.003925
60	Web components	0.5	0	2	0.88	1	0.149	0.003925
61	Web components	0.5	1	6	1	0	0.1484	0.003925
62	Web components	0	0	3	2	1	0.1487	0.003832
63	Web Components	0.99	0.5	0.88	2.67	1	0.1483	0.00368
64	Web components	0.5	0	1	0.3	0	0.1485	0.003362
65	Web components	0.5	0	2	0.15	0	0.1481	0.003362
66	.Net components	0.83	0.33	2.67	0.83	1	0.1483	0.003362
67	.Net components	0.84	0.31	2.45	0.84	0	0.148	0.00266

(Contd...)

Table 6: ***(Continued)***

S. No.	Component type	COCU	COIC	CORE	COUS	COST	RMSE (ANFIS)	RMSE (GFS)
68	.Net components	0.83	0.33	2.18	0.83	1	0.1482	0.001879
69	.Net components	0.65	0.9	0.71	0.65	1	0.1479	0.001879

RMSE: Root-mean-square error, ANFIS: Adaptive neuro-fuzzy inference system, GFS: Genetic-fuzzy system, COCU: Component customizability, COIC: Component interface complexity, CORE: Completeness of component return, COUS: Component understandability, COST: Component stability

Table 7: Computed aggregate values of component types

Component types	COCU	COIC	CORE	COUS	COST
Java	0.88	0.86	0.48	0.92	0.79
Web	0.66	0.27	2.08	0.73	0.69
.Net	0.79	0.47	2	0.79	0.75

COCU: Component customizability, COIC: Component interface complexity, CORE: Completeness of component return, COUS: Component understandability, COST: Component stability

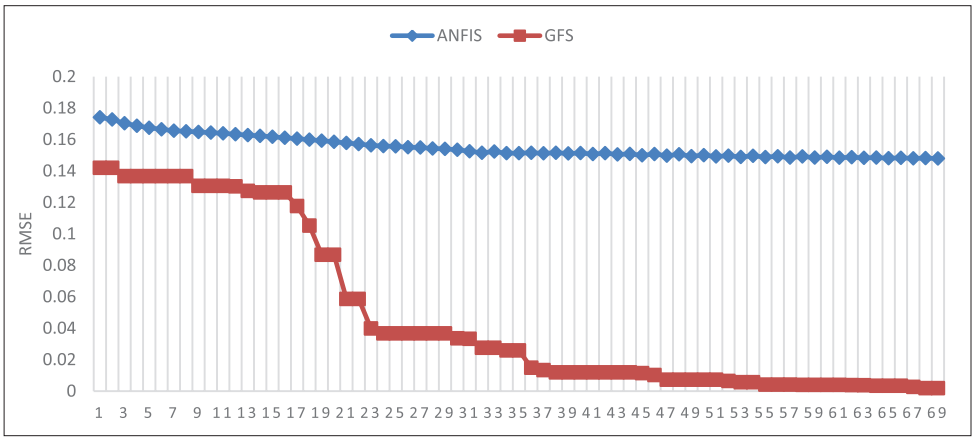

Figure 11: Adaptive neuro-fuzzy inference system and genetic-fuzzy system root-mean-square error

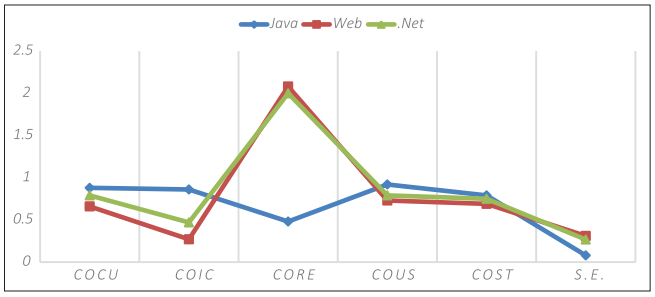

Figure 12: Components' reusability prediction level

and the visible activities that take place within the system were also presented using appropriate UML design. For the implementation, GFS was developed and deployed using MATLAB as the software tool.

The result of the evaluation shows that GFS predicts more accurately with an RMSE of 0.0019 as against the commonly used method, ANFIS, with an RMSE of 0.1480, adjudging GFS as a better predictor.

DIRECTION FOR FURTHER STUDIES

The designed architecture presented in this study is simplified such that it can easily be modified to enable adaptation and application to other research domains such as monitoring system, decision support system, data mining system, and control system. The hybridized power of the system can also be extended to solve other related and more advanced intelligent applications.

Five quality factors were used in the determination of the reusability of the selected components, other quality factors as related to software components (e.g., operability, statelessness, etc.) can also be considered in future research work in the prediction of software component reusability.

Table 8: ANOVA analysis of component types' aggregated values

Component types	*n*	Mean	Standard deviation	Standard error
Java	5	0.7860	0.17743	0.07935
Web	5	0.8860	0.69263	0.30975
.Net	5	0.9600	0.59657	0.26680
Total	15	0.8773	0.50318	0.12992

REFERENCES

1. Washizaki H, Yamamoto H, Fukazawa Y. A metrics Suite for Measuring Reusability of Software Components. Proceedings of the 9th International Symposium on Software Metrics. Sydney, Australia: 2003. p. 201-11.
2. Gill NS. Importance of software component characterization for better software reusability. ACM SIGSOFT Softw Eng Notes 2006;31:1-3.
3. Sharma A, Kumar R, Grover PS. Critical Survey of Reusability Aspects for Software Components. Proceedings of the World Academy of Science, Engineering and Technology, Bangkok, Thailand: 2007. p. 419-24.
4. Sharma A, Kumar R, Grover PS. Investigation of reusability, complexity and customisability for component-based systems. ICFAI J Inf Technol 2006;2:21-4.
5. Fazal-e-Amin, Mahmood, AK, Oxley A. A review of software component reusability assessment approaches. Res J Inf Technol 2011;3:1-11.
6. Kumar V, Kumar R, Sharma A. Applying neuro-fuzzy approach to build the reusability assessment framework across software component releases an empirical evaluation. Int J Comput Appl 2013;70:41-7.
7. Thakral S, Sagar SV, Vinay SE, Reusability in component based software development a review. World Appl Sci J 2014;31:2068-72.
8. Singh AP, Tomar P. Estimation of component reusability through reusability metrics. Int J Comput Control Quantum Inf Eng 2014;8:1865-72.
9. Goel S, Sharma A. Neuro-fuzzy based approach to predict component's reusability. Int J Comput Appl 2014;106(5):33-38.
10. Kumar A, Chaudhary D, Kumar A. Empirical evaluation of software component metrics. Int J Sci Eng Res 2014;5:814-20.
11. Aman H. A quantitative method of verifying metrics using principal component analysis and correlation analysis. J IEICE 2002;10:1000-2.
12. Sharma A, Kumar R, Grover PS. Reusability assessment for software components. ACM SIGSOFT Softw Eng Notes 2009;34:1-6.
13. Sagar S, Nerurkar NW, Sharma A. A soft computing based approach to estimate reusability of software components. ACM SIGSOFT Softw Eng Notes 2010;35:1-5.
14. Rotaru OP, Dobre M. Reusability Metrics for Software Components. AICCSA '05 Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications. Washington, USA: 2005. p. 24-31.
15. Singh H, Toora VK. Neuro-fuzzy logic model for component based software engineering. Int J Eng 2011;1:303-14.
16. Kamalraj R, Kannan AR, Ranjani P. Stability-based component clustering for designing software reuse repository. Int J Comput Appl 2011;27:33-6.
17. Jatain A, Gaur D. Estimation of Component Reusability by Identifying Quality Attributes of Component: A Fuzzy Approach. Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, Coimbatore, India: 2012. p. 738-42.
18. Ravichandran K, Suresh P, Sekr KR. ANFIS approach for optimal selection of reusable components. Res J Appl Sci Eng Technol 2012;4:5304-12.
19. Christopher D, Chandra E. Prediction of software requirements stability based on complexity point measurement using multi-criteria fuzzy approach. Int J Softw Eng Appl 2012;3:101-15.
20. Aversano L, Molfetta M, Tortorella M. Evaluating Architecture Stability of Software Projects. IEEE Conference; 2013. p. 417-24.
21. Ekanem BA, Woherem E. Legacy components stability assessment and ranking using software maturity index. Int J Comput Appl 2016;134:975-8887.
22. Sandhu PS, Dalwinder SS, Singh H. A comparative analysis of fuzzy, neuro-fuzzy and fuzzy-ga based approaches for software reusability evaluation. ProcWorld Acad Sci Eng Technol (WASET) 2008;2:292-5.
23. Hegazy O, Soliman OS, Toony AA. Hybrid of neuro-fuzzy inference system and quantum genetic algorithm for prediction in stock market. Issues Bus Manage Econ 2014;2:94-102. Available from: http://www.journalissues.org/IBME. [Last accessed on 2017 Oct 13].
24. Fazlic LB, Avdagic K, Omanovic S. GA-ANFIS expert system prototype forprediction of dermatological diseases. Eur Fed Med Inf 2015;2015:622-6.
25. Dhokley W, Ansari T, Fazlic N, Mohd.Hafeez H. New improved genetic algorithm for coronary heart disease prediction. Int J Comput Appl 2016;136:975-8887.