**Journal of Theoretical and Applied Information Technology**
28th February 2019. Vol.97. No 4
© 2005 – ongoing JATIT & LLS

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

# A SOFT COMPUTING APPROACH TO OPTIMIZE COMPONENT BASED SOFTWARE COMPLEXITY METRICS

**[1]POOJA RANA, [2]RAJENDER SINGH**

[1]Research scholar, Department of Computer Science and Applications, M.D. University, Rohtak, India

2 Professor, Department of Computer science and Applications, M.D. University, Rohtak, India

[1]poojakaul24@gmail.com, [2]chhillar02@gmail.com

**ABSTRACT**

Identification of software components is found to be a crucial task and remains as a challenging area in the software domain to extract optimal components from the component repository[18]. Several methods are deployed to identify the software components and observed that clustering based technique is frequently used and offer a solution with certain limitations such as prior specification of a set of clusters, overlapping and difficulty in selecting the correct distance metric. In this research, an optimization technique is applied on some components to get the best result. The genetic algorithm using the concept of number of chromosomes is applied on software complexity metrics such as Cohesion of Variables within a Component (COVC), Cohesion of Methods within a Component (COMC) and Total Cohesion Complexity of a Component (TCCC) which are proposed by Rana and Singh et.al [14]. Fitness function metrics is proposed for finding out the fitness value. Rana and Singh also empirically evaluate cohesion metrics(COVC, COMC, TCCC) and perform comparative analysis with existing metrics in another research paper [15]. In this paper genetic algorithm is applied on these metrics for optimization of results. After application of genetic algorithm, SPSS a statistic tool is applied on the result to find out the significance of result. The result obtained shows that better optimization of software metrics is obtained through application of genetic algorithm compared to without usage of genetic algorithm.

**Keywords:** *Software Component Metrics, Software Cohesion, Reusability, Maintainability, Genetic Algorithm.*

## 1. INTRODUCTION

The software development process is found to be development paradigms which are being promoted as a primary mean for reducing the time to market, enhancing the rate of productivity, capturing the effectiveness in cost and the overall efficiency during the development through the efforts of software [11]. The incorrectness and partial requirements has given major concern since they are found to be key aspect for the failure of software projects [20]. Therefore, there is a huge requirement for an effective method to extract and document the user requirements which will enhance the success rate of the entire project. Moreover, the development of the software depends on the basis of prioritization requirement in view of client's responsibility [3]. The above mentioned are the reasons where metrics should be considered to measure the success of the project. Several factors that should be, measured for the development of the project are the effort, estimation of the cost and quality and performance of the final product and rate of success is determined after the completion and validation of the product. But the usage of metric provides a way to determine and predict the success rate before the product is being validated or completed.

The traditional based software projects developed are found to be inadequate and unreliable, so there is a requirement of sophisticated metrics to provide a reliable solution [10]. Several metrics used by number of authors is found to provide a solution but it is not effective to predict the performance ability of the system. The different techniques of software development are agile based software development process, clustering based software development approach and component based software development approach. From the above mentioned framework, component based framework is found to be generally used because of the advantages such as time shrinkage, production cost and reuse capability.

The development of Component Based Software System (CBSS) is a promising solution in the complex and large scale industries. The traditional existing techniques for software development are having high cost for development, low productivity rate, the quality of software is unmanageable compared to component based software which leads to a requirement of new technology for the analysis of software. The CBSS is found to provide an efficient mechanism to reduce the development cost, and increase the maintainability and quality of the software system [6]. The implementation has provided remarkable interest in the developing field especially in software and academia and the use of modularity concept provided more controllable, comprehensible and flexibility options during software integration [21]. In recent years, very few works is done in the modular software development by considering the criteria of reducing the rate of coupling and enhancing the software module cohesion [5] [8]. High rate of cohesive module provides more reusability property and better maintenance is achieved with loosely coupled systems. Through optimization and optimal selection of software components it exhibits high reliability and reduction in the cost or few modules in which budget has given more concentration [4] [17] [22].

The main theme of this paper is to develop an effective component based software metrics using genetic algorithms. CBSS comprises with several reusable components which makes it advantageous compared to other existing techniques, but limitations arises with the complexity in designing process, selection of irrelevant components and high computational time. A novel approach is developing which comprising genetic algorithms to solve this issues and complexity metrics such as cohesion is considered for optimization and reducing the complexity which will be explained in further stages.

## 2. LITERATURE REVIEW

A multimode amount of research is available in the field of software development analysis and different researchers have developed much advancement to reduce certain factors and improve the process of the system. The different research is as follows

Agile software development process is found to be effective and comprises of several agile values to represent methodologies such as extreme programming and scrum. The combination of Capability Maturity Model Integration (CMMI) along with agile software technique to synthesize and evaluate software is developed by author [9]. Agile technique is found to have high common place during the software development. CMMI model is generic software used by numerous organizations and its hybridization along with agile development is found to be advantageous through as per the report submitted by CMMI institute [7]. The advancement has led to the implementation of scaling factors of agility in various software development large industries. The advancement of a hybrid methodology through a reference qualitative constructive empirical approach has been developed by author [2].

Several factors such as business policy, legal, context, business value, agility, abstraction, and components or facility elements that is not been explicitly modeled is discussed in terms of ISO/IEC 24744 meta model standards. A context aware hybrid architecture strategy is used as reference architectural elements for product enhancement and a developed model with case studies and constraints is found to be effective in designing the software. Furthermore, an agile learning designing approach is used in the design and development of a system for adaptive education [1].

Adaptive Educational Hypermedia systems (AEHS) provide a path to access the data and information and enhance the process of learning with the same. AEHS based system is found to be advantageous in several software engineering aspects and with the combination of agile based learning technique allows the designers to modify, evaluate the design as per the user requirement. From the study, it is observed that agile learning aspects show an average successful rate in terms of design and development of user component model. The machine learning based Bayesian network model has been developed by author [19] to estimate the effort of task effort during agile software development. The author has designed Bayesian network in correspondence to the prediction of method in any agile based method. The developed model doesn't have any practical impact in terms of agility and the author described several strategies to access the precision rate of the model and the results are evaluated in terms of relative error mean, prediction and accuracy level of predicted instances, mean absolute error, root relative squared error. The result obtained from the study provides a high accuracy level of 99.37% with 100% prediction accuracy for 160 tasks.

A novel based software development approach comprising component based software analysis to solve the issues raised in multitude embedded domains [12]. The specific features of target platform are found to be hidden during the language design and can be extracted by deploying specific mapping tools. The author has considered "C" code of a populated DOM model to define the embedded system and wrappers is generated to access external ports through the implemented code. From the study on results, it is observed that the component development through interrupt handlers and vector tables can be used to represent the typical aspects of the embedded system and by using an appropriate library with open source tool chain can be useful to enhance the availability of the developed approach. Furthermore, an optimal approach on the basis of coupling and cohesion through component selection based build or buy scheme is developed by author [16].

The author is concerned about hybridization and integration of components to build modular software systems. The terms namely coupling and cohesion (C&C) plays a critical role in developing a software system in terms of maintainability, reliability and availability. The author defined that high cohesion range and the rate of low coupling is a major criteria for defining the good software and selection of optimization model namely fuzzy bi-criteria is proved to be effective in build-or-buy scheme. The results obtained from the study specify that the developed model increases the rate of intra-modular coupling density and functionality of the system within the limited plan of budget. Furthermore, the delivery time is also reduced with increase in the reliability of the system. The study is summarized by stating the performance of the developed model can be improved by including the features of compatibility amongst the model components. The component based approach for designing the flexible software integration model is developed by author [13].

The thorough analysis of the software plan, the requirement of the modification plan and runtime execution can act as an integrated part of the software development. Developing the integration model is found to be a challenging task because of the existence of numerous modeling languages and their supportive comprehensive runtime. The author has considered model driven and component based orthogonal development approaches to overcome the above mentioned challenges. Model driven approach is found to solve the issues regarding different modeling languages and component based approach facilitates modular systems through divide and conquer approach with their capability of reuse. The author has considered the assumption that model deployment and development of program should be considered as a first class entity for the software development life cycle and result obtained from the study shows that the developed model is found to be a promising approach to analyze the behavior and feasibility of the system.

## 3. RESEARCH METHODOLOGY

Genetic algorithms are suitable for modifying software architectures as they too have certain constants which can be implemented in various ways. Architecture is based on the requirements as to what the software system is supposed to do. The basic architecture deals with the question of how the operations related to the requirements are divided into components. When further developing architectures, mechanisms such as interfaces and inheritance can also be added to the design. Thus, the set of requirements and their positioning in the system represents the basic individual, which too then evolves as positions of requirements are changed and mechanisms are added. As there is theoretically an exponential amount of possible designs for a system, the use of genetic algorithms to solve the problem is justified. The architecture is defined as a group of plan resolutions and architecture texting minimizes the documentation through the collection of plan resolutions. However, difficulties and complexities are more during the practical implementation. During the deployment of software modules, the logic considered during the development inaccessible and in some cases the outcome of the project plan will be hidden or known only by the developer.

In this research, architectural knowledge containing the advantages and limitations of the software structure design is considered and is found to provide major changes in the community of software architecture. A software system is constructed to serve a specific purpose. In order to achieve the desired outcome, the software needs to complete several tasks leading to the final solution. Component metrics such as COVC( Cohesion of variables within a component), COMC (Cohesion of methods within a component), TCCC (Total Cohesion Complexity of a Component) proposed by Rana and Singh [14] are considered for experiment analysis and evaluated in terms of cohesion and Fitness Function metrics is proposed to find out the fitness value for each chromosomes. The experimentation is conducted for number of generations. In later stages, genetic algorithm is applied for the same input metrics and the number of generations and the results tabulated is compared with the values without genetic algorithm which will be explained in the following stages. Two type of comparison is performed firstly by graphical representation and secondly by using paired sample t-test.

Maintainability and reusability are the key components considered for the identification of components. Genetic algorithm is found to be effective in the following key components and it employs the COVC, COMC, TCCC will be explained below

## 4. MEASUREMENTS OF SOFTWARE DESIGN

The two main factors for the selection of component are maintainability and reusability. Cohesion metrics such as COVC, COMC and TCCC are to be used to find out the best component. These metrics are employed on GA Algorithm for selection of optimized component.

### 4.1 COVC(Cohesion of variables within a component) [ 14 ]

COVC refers to the frequency of variables usage by the component. In the component, when the associated variables focuses on a accomplishing a single task then a component is said to be cohesive [14]. Critical, moderate and standard are the three categories for instance variables. The instance variables are classified on the basis of data types. Critical includes class type, user defined component, pointers and references. Moderate includes string, arrays, vector, list and standard includes integer, float, double, Boolean etc. [14] Suppose a component C such as a class has a set of methods M(C)={m c1 , m c2 ,m c3 , ...........,m cn ) and a set of instance variables v in V(C) ={v c1 ,v c2 ,v c3 , .......... V cn }. Fv(C) is the set of pairs (v c ,m c ) for each instance variables v in V(C) that is used by methods m in M(C). Fv(C) is further divided into three i.e. a set of pairs (vsi ,mci ) and a set of pairs (vmi ,mci ) and a set of pairs(vci ,mci ) for each instance variable v in V(C) that is used by methods m in M(C). [14]

$$\text{COVC}=\sum_{i=0}^{n}\frac{FIV}{TV} \qquad (1)$$

$$\text{FIV}= \sum_{i=0}^{n}\{[f(vsi)*Ws]+[f(\,vmi)*Wm]+[f(vci)*Wc]\} \qquad (2)$$

Here

FIV = frequency of the instance variables within a component

TV= total no of Instance Variable in a component

$F(v_{si})$= Frequency of standard variables

$F(v_{mi})$= Frequency of moderate variables

$F(v_{ci})$= Frequency of critical variables

Ws, Wm, Wc are the weight factor of the standard, moderate and critical type of variables respectively.

### 4.2 COMC (Cohesion of Methods within a component) [14]

Cohesion of Methods in a component refers to the relatedness of methods and instance variables of a component. This metrics considers the interaction between the methods with in a component. Here, the sum of methods is found that uses the same type of variables i.e standard, moderate, critical. [14]

$$\text{COMC}= \sum_{i=0}^{n}\frac{COM}{TM} \qquad (3)$$

$$\text{COM}= \sum_{i=0}^{n}\{(Msi*Ws)+(Mmi*Wm)+(MCi*Wc)\} \qquad (4)$$

COM = count of methods that use same type of variables

TM= total no of methods

$Ms_i$= sum of methods that use Standard type of variables.

$Mm_i$= sum of methods that use Moderate type of variables.

$Mc_i$= sum of methods that use critical type of variables.

Ws, Wm, Wc are weight factor for standard, moderate and critical type of variables.

### 4.3 TCCC(Total Cohesion Complexity of a Component) [ 14 ]

Total cohesion complexity of a component is the combination of cohesion of variables in a component and cohesion of methods in a component. As the metric name suggests, this is the combination of both cohesion of methods and cohesion of variables. [14]

$$TCCC = COVC + COMC \qquad (5)$$

COVC= cohesion of variables within a component matrices

COMC= cohesion of methods within a component matrices

## 5. GENETIC ALGORITHM

Genetic algorithm is found to be an effective way of using the evolution techniques in computer science. In technical terms, genetic algorithm is used to obtain the optimal solution from a large and complex input set. To apply genetic algorithm, the solution should be encoded in terms of solutions namely chromosomes, an initial population cross over mutation, fitness function and an optimal factor for the next generation. For fitness function evaluation a fitness function metrics is proposed to get the fitness value. The output of this metric is input for the fitness equation.

In genetic algorithm the initial population which is also called as chromosomes are the input metrics which are generated randomly. Then fitness value is evaluated for each chromosome according to the fitness function. After fitness value the best chromosomes are selected for reproduction using random or roulette wheel method. GA operators such as crossover and mutation are applied on selected chromosomes. Crossover has three methods one point, two point and three points. One of these methods is applied on selected chromosomes. After crossover mutation operator is applied. Mutation has two methods, method of elimination and method of addition.

After applying all genetic algorithm operators each offspring is evaluated. Low fitness valued offspring is to be eliminated and a new generation of population is generated. This process is repeated until condition is met and the fittest chromosomes found.

The overall process of the genetic algorithm is as follows,

```
Initialize(input population)
Analysis (Population) // Pre-processing
If (Stopping criteria is not satisfied)
{
Do
{
Selection (Population set)
Crossover (Population set)
Mutation (Population set)
Analyze (Population set)
}
}
```

The above algorithm process continues until an optimal solution is obtained for the input problem criteria or to a maximum number of iterations till the condition is met. The steps involved in deploying genetic algorithm are as follows,

### 5.1 Fitness function evaluation

Fitness Function Metrics

The fitness function metric approach provides several advantages to the metric research through effective metric set and provides additional mechanism support for software analysis. Equation 6 is considered to evaluate the fitness function. The value from this function is considered as input to find the fitness value and it is given by

$$\text{Predict Fitness (a)} = c_x + \sum_y c_y, f_{a,y} \qquad (6)$$

Where, a is defined with number of chromosomes in the software component feature set $f_a$

A training set comprising example variables is considered in learning global co-efficient $C_{y.}$, $C_x$ is the global intercept used to compute the fitness function of each individual chromosome a.

The individual string in the initial population is assigned with a fitness value on the basis of objective function. The function maximization is obtained through the fitness value which is equal to the objective function value of the string. The equation for the optimal minimum fitness is given by the equation 7, f(x) in the fitness equation is the fitness function.

$$\text{FA} = \frac{1}{(1+f(x))} \qquad (7)$$

### 5.2 Selection and Reproduction

Reproduction is used to extract the selective strings from the population and processed to calculate the fitness function. Two chromosomes are selected randomly and crossover and mutation operators are applied on selected strings.

#### 5.2.1 Crossover Operator

Some chromosomes from population are randomly selected for crossover to produce new offspring. Crossover has three methods. Out of these methods two point method is applied on chromosome. In two point crossover two component from one metric and two component from another metric is exchanged.

#### 5.2.2 Mutation Operator

After crossover mutation operation is performed on newly produced chromosome. Mutation operation is performed to eliminate invalid component which produces bad result and add new component with best fitness value is to be kept.[18] The main theme is to extract the strings which are above the average fitness value in the current population and apply genetic functions to new strings to obtain the successive population. The chances of strings being selected to the reproduction phase are proportional to its fitness parameter. Furthermore, on the basis of the evaluation criteria, individual genomes are selected from a set of chromosomes.

## 5. EXPERIMENTAL EVALUATION AND RESULTS

The experiments are done through the utilization of with and without genetic algorithm for software architecture. The implementation is carried out by considering input component metrics namely FF, COVC, COMC and TCCC which is proposed by Rana and Chhiller [14] and the same is modeled in JAVA based platform. This experiment is performed on a JAVA based project on online hospital management system. This software contains ten components on which theses metrics are applied The test cases are extracted from input component metrics and the same is optimized through number of iterations and the derived classes. The analysis is more concerned with factors namely reusability and computational time.

The input parameter component metrics is selected on the basis of fitness value and processed to calculate the required and optimized metric parameter. For effective processing, the parameter with high fitness value is selected and processed through the number of chromosomes. The evaluation is carried with and without genetic algorithm to calculate the effectiveness of the system. The iteration value is increased for every step and process is repeated to obtain the fitness value. The results obtained will be tabulated and it is shown below,

*Table 1 Component Complexity Metrics without Genetic Algorithm*

| Generations | FF Metric | COVC | COMC | TCCC |
|---|---|---|---|---|
| 10 | 0.45 | 0.23 | 0.13 | 0.46 |
| 20 | 0.47 | 0.42 | 0.21 | 0.52 |
| 40 | 0.56 | 0.15 | 0.32 | 0.4 |
| 60 | 0.59 | 0.35 | 0.17 | 0.37 |
| 80 | 0.67 | 0.41 | 0.24 | 0.48 |
| 100 | 0.71 | 0.19 | 0.31 | 0.51 |

Table 1 and 2 represents the components software metrics obtained from several software's with and without genetic algorithm and the same is processed through number of chromosomes to validate the performance analysis of the developed system. Table 1 shows metrics values of different metrics on different iterations without applying GA. Table 2 shows application of GA on different complexity metrics. A comparison can be drawn from the result of the different metrics.

*Table 2 Component Complexity Metrics with Genetic Algorithm*

| Generations | FF Metric | COVC | COMC | TCCC |
|---|---|---|---|---|
| 10 | 0.71 | 0.32 | 0.25 | 0.54 |
| 20 | 0.74 | 0.41 | 0.31 | 0.61 |
| 40 | 0.76 | 0.4 | 0.49 | 0.35 |
| 60 | 0.73 | 0.37 | 0.29 | 0.42 |
| 80 | 0.82 | 0.48 | 0.34 | 0.57 |
| 100 | 0.87 | 0.32 | 0.43 | 0.68 |

Table 1 and table 2 contain four metrics FF, COVC, COMC and TCC which are cohesion metrics proposed by Rana and Singh et.al [14]. These metrics are basically extension of cohesion metrics the author categorize the metrics according to data types. These metrics help the software developer to make the component independent and also showcase which part of the program should be more focused.

For FF metrics in table 1(without GA) the value at 100th generation is 0.71 and in table 2 (with GA) the value at 100th generation is 0.87 which is comparative high the fitness function value should be high to get the optimal result which can be achieved by application of GA. COVC (cohesion of variables within a component) in table 1 at 100th iteration is showing 0.19 metrics value which is without applying GA and in table 2 at 100th iteration is showing 0.32 which is with applying GA. From this result a conclusion can be drawn that by application of GA an effective and optimal solution can be reached. In table 1 at 100th iteration the value of COMC(Cohesion of methods within a component) without GA is 0.31 and 0.43 is with GA and for TCC at 100th iteration without GA the metrics value is 0.51 and with GA the value of metrics is 0.68. From the results, it is evident that the proposed optimized approach delivers improved metric value when compared to the existing approach and it can be seen in the following graph.

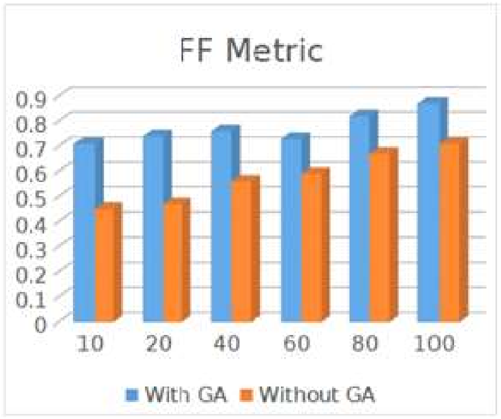

*Figure 1 Representation of FF metric*

Figure 1 is the graph plotted with metric values obtained from FF metric with and without genetic algorithm. It is observed from the graph that better result is achieved with genetic algorithm compared to without genetic algorithm.

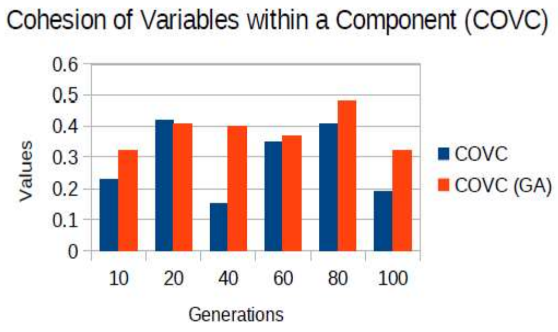

*Figure 2 Representation of COVC metric*

Figure 2 represent the values obtained from cohesion of variables within a component (COVC) metric through number of generations and with and without genetic algorithm. X- axis represents the number of generations and Y axis represents the metric value. From the graph, it is observed that better metric co-efficient is obtained through the application of genetic algorithm.

Cohesion of variables within a component (COVC) metric also forms a sub metric to cohesion and from the obtained results it is clear that the proposed approach achieves improved metrics utilizing GA.

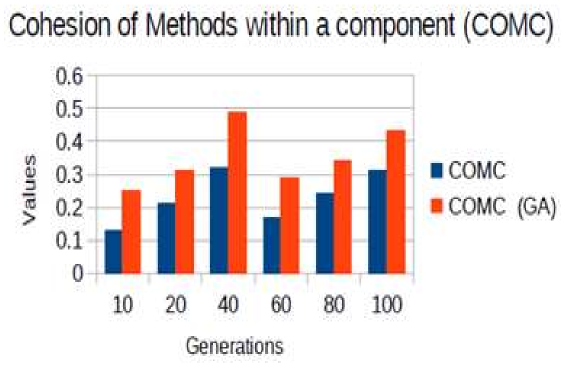

*Figure 3 Representation of Cohesion of Methods within a component*

Figure 3 represents the graph plotted with metric values obtained from cohesion of methods within a component (COMC) with and without genetic algorithm. It is observed from the graph that better result is achieved with genetic algorithm compared to without genetic algorithm.

As per the results, enhanced metric values are achieved using the proposed approach.

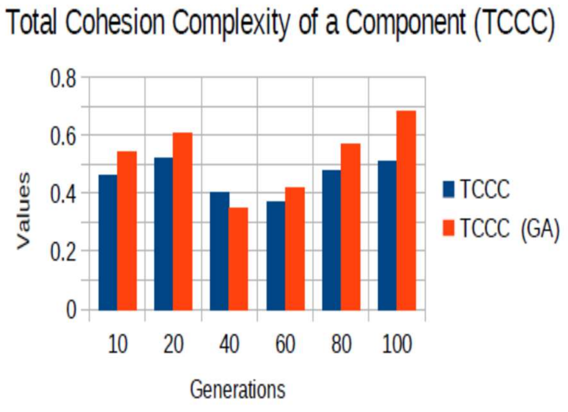

*Figure 4 Represent Total Cohesion Complexity of a component*

Figure 4 represent the graph with metric values total cohesion complexity metric (TCCC) with or without genetic algorithm. The graph shows that result with genetic algorithm is better than the result without genetic algorithm

Cohesion is defined as the degree to which all elements of a module, class, or component work together as a functional unit. High cohesion is good, and low cohesion is bad. The ideal situation is one where a module, class, or component provides only one function or, at most, a very closely related set of functions. Coupling is defined as the degree of interdependence between two or more classes, modules, or components. Tight coupling is bad, and loose coupling is good. Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa. Low coupling is often a sign of a well structured computer system and a good design, and when combined with high cohesion, supports the general goals of high readability and maintainability

*Table 3 Represent comparative chart of different cohesion metrics*

| Generations | FF Metric | | COVC | | COMC | | TCCC | |
|---|---|---|---|---|---|---|---|---|
| | Without GA | With GA | Without GA | With GA | Without GA | With GA | Without GA | With GA |
| 10 | 0.45 | 0.71 | 0.23 | 0.32 | 0.13 | 0.25 | 0.46 | 0.54 |
| 20 | 0.47 | 0.74 | 0.42 | 0.41 | 0.21 | 0.31 | 0.52 | 0.61 |
| 40 | 0.56 | 0.76 | 0.15 | 0.4 | 0.32 | 0.49 | 0.4 | 0.35 |
| 60 | 0.59 | 0.73 | 0.35 | 0.37 | 0.17 | 0.29 | 0.37 | 0.42 |
| 80 | 0.67 | 0.82 | 0.41 | 0.48 | 0.24 | 0.34 | 0.48 | 0.57 |
| 100 | 0.71 | 0.87 | 0.19 | 0.32 | 0.31 | 0.43 | 0.51 | 0.68 |

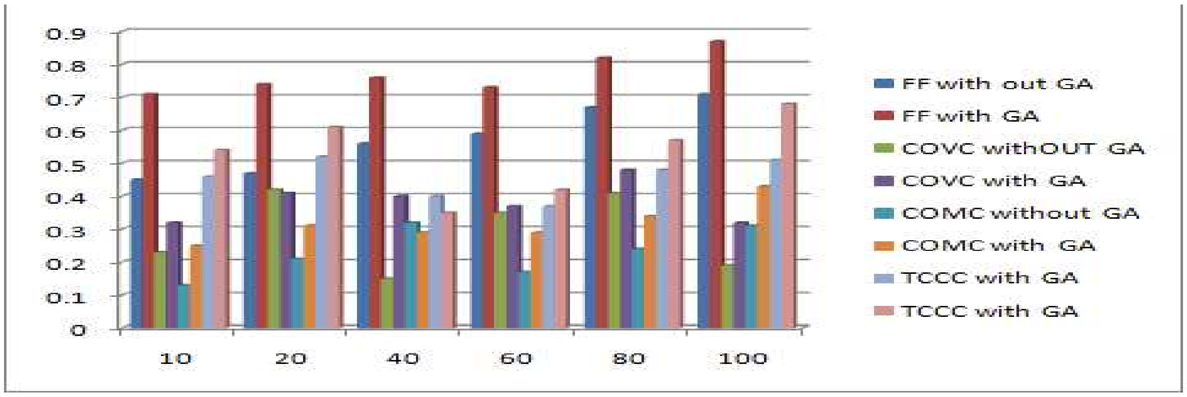

*Figure 5 Represent Comparative Analysis Of Different Metrics.*

Table 3 contains four metrics FF, COVC, COMC and TCC which are cohesion metrics proposed by Rana and Chhiller et.al [14]. This table shows the comparison between the metrics with GA and without GA. From the Table 3, Figure 4 is drawn, that compares graphically at each iteration the difference between the results. These metrics are basically extension of cohesion metrics the author categorize the metrics according to data types. These metrics helps software developer to make the component independent and reusable and also reflect which part of the program should be more focused.

Figure 4 represents the values obtained from Fitness Function (FF) metric, cohesion of variables within a component(COVC) metric, cohesion of methods within a component(COMC) metric and total cohesion complexity metric through number of generations and with and without application of genetic algorithm. X- axis represents the number of generations and Y axis represents the metric value. For each generation six bars represented for each metrics with and without genetic algorithm. From the graph, it is observed that better metric co-efficient is obtained through the application of genetic algorithm.

After comparing table 1 and table 2 results with graphical method by using bar graph which shows that FF, COVC, COMC and TCCC proposed by Rana and Singh et.el. [14] give an optimal and best result. To check the significance of the results of COVC with GA and COVC without GA , COMC with GA and COMC without GA, TCCC with GA and TCCC without GA (Table3) a paired sample t test is to be applied. Paired sample t-test is suitable test for judging the significance of the difference between the means of two samples which are related to each other. Paired sample t-test is applied in case of small sample size.

Paired sample t-test is applied on variables COVC without GA and COVC with GA data are shown in table 5. The result shows that the mean value of COVC with GA (0.3833) (Table 4) is higher than the mean value of COVC without GA. The paired sample t-test value is -2.436 (Table 5) and same is significant at 90% level of confidence which means that the COVC with the application of GA is higher and significant(10%).

*Table 4 Mean And Std Deviation Of COVC With And Without GA*

| | Mean | N | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|
| COVC without GA | 0.2917 | 6 | .11669 | 0.04764 |
| COVC with GA | 0.3833 | 6 | .06088 | 0.02486 |

*Table 5 Paired Sample T-Test*

| | T | DF | Sig.(2-tailed) |
|---|---|---|---|
| Pair 1 COVC without GA And COVC with GA | -2.436 | 5 | 0.059 |

Paired sample t-test is applied on variables COMC without GA and COMC with GA data are shown in table 3. The result shows that the mean value of COMC with GA (0.3517) (Table 6) is higher than the mean value of COMC without GA. The paired sample t-test value is -11.630 (Table 7) and same is significant at 99% level of confidence which means that by applying GA on COMC will give higher and significant value and provide best and optimal solution.

*Table 6 Mean And Std Deviation Of COMC With And Without GA*

| | Mean | N | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|
| COMC without GA | 0.2300 | 6 | .07563 | 0.03088 |
| COMC with GA | 0.3517 | 6 | .09087 | 0.03710 |

*Table 7 Paired Sample T-Test*

| | T | DF | Sig.(2-tailed) |
|---|---|---|---|
| Pair 2 COMC without GA And COMC with GA | -11.630 | 5 | .000 |

Paired sample t-test is applied on variables TCCC without GA and TCCC with GA data are shown in table 3. The result shows that the mean value of TCCC with GA (0.5283) (Table 8) is higher than the mean value of COMC without GA. The paired sample t-test value is -2.449 (Table 9) and same is significant at 90% level of confidence which means that by applying GA on TCCC will give higher and significant value and provide best and optimal solution.

*Table 8 Mean And Std Deviation Of TCCC With And Without GA*

| | Mean | N | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|
| TCCC without GA | 0.4567 | 6 | .06022 | 0.02459 |
| TCCC with GA | 0.5283 | 6 | .12254 | 0.05003 |

*Table 9 Paired Sample T-Test*

| | T | DF | Sig.(2-tailed) |
|---|---|---|---|
| Pair 3 TCCC without GA And TCCC with GA | -2.449 | 5 | .058 |

This analysis concludes that the application of Genetic Algorithm on cohesion metrics COVC, COMC and TCCC proposed by Rana and Singh et.al [14] are significant and GA provides the optimized result. The change in the result by applying GA is significant and acceptable.

## 6. RELATED WORK

The selection of component depends on many factors. One of the main factors is reusability and independent component. There are many authors which worked on selection of component. Various type of techniques are used for selection of component like graph partitioning, clustering, FCA methods. Here soft computing is to be applied for selection of a component.

**Graph based Approach** Albani et.al. [23] had taken process, objects and actors as vertices and their relationship as edges of a graph. In the graph partitioning approach weights are assigned to each edge to show the relationship among elements of domain model. After assigning the weights, a graph is partitioned into components by applying heuristic technique of graph theory. Authors by using graph partitioning approach identify component which is high in cohesion and low in coupling. One disadvantage of this approach is weight assignment is done manually.

**Clustering Technique** Lee et.al [24] used the technique of clustering classes for identifying logical component. In this process key classes are identified and some more classes also identified which are fully dependent on the key classes. Key class identification is performed manually. No automatic process is used to identify Key classes. This part of the process is most critical. Different authors used different type of clustering technique like Jain et. al [25] used agglomerative clustering technique, Shahmohammadi et.al [26] used feature based clustering method to identify logical component. There are various clustering techniques like k-means, Hierarchical, Graph based, Fuzzy C-mean etc.

**FCA-Based approach** Hamza et.al [27] used FCA-based technique to identify classes and component. FCA is formal concept analysis technique which is used for identification of class. The author proposed a frame work by taking FCA base theory to partition a class diagram into logical component. CAI et.al.[28] also develop a novel approach to identify component. They used fuzzy FCA technique. They also used dispersion and distance concept to measure cohesion and coupling. The performance of their methods can be measured manually. They have many limitations because they used simple clustering method.

**Evolutionary Method** now a day evolutionary method is applied on software for optimization of the result. A new concept in software engineering had come i.e. search based software engineering. In this field the design of the software take module clustering and also take into consideration that cohesion of a module will be high and coupling among module should be low. In this paper GA is applied on components to optimized the result the aim is to identify a component with high degree of cohesion and low degree of coupling.

Various search based clustering techniques are used like Hill Climbing, Simulating annealing, Meta Heuristic Genetic algorithm etc. if the proposed technique is compared with other authors approach like Doval et.al [29] used same objective Fitness Function for coupling and cohesion whereas Praditwong et.al [30] use Pareto optimality by using multi objective approach and Simons et.al [31] use iterative multi objective genetic algorithm to identify classes at design level. The difference between proposed genetic algorithm and existing search based algorithm is that there aim is to optimized clustering criteria and proposed GA and fitness function aim is to maximize cohesion and minimize coupling.

## 7. CONCLUSION

The advancement in the application of machine learning algorithms and evolution in the field of software engineering has led to automatically design and identify the logical and effective software components through an effective search optimization algorithm. In this paper, a novel approach has been presented to determine the software component metric using a genetic algorithm. The genetic algorithm based component metric approach is found to be beneficial in optimising the software metrics and provides more accurate results compared to other existing techniques.

Rana and Singh et.at. [15] already compared there proposed metrics with existing metrics and found that the proposed metrics are more significant than existing cohesion metrics. In that research paper authors Rana and Singh et.at [15] conclude that the complexity of a software or component depends on the frequency of the variables and also conclude that by using the moderate type of variables, the strength of the component will increased.

In this paper author take the field into new direction by using Genetic algorithm. Cohesion metrics which helps the developer to find a component which is suitable for reuse and easy to maintain is taken for analysis which is further divided into three metrics COVC, COMC and TCC. The components obtained are processed through a system of chromosomes. The evaluation is done by varying the number of chromosomes and fitness value is obtained with and without a genetic algorithm.

GA takes components as chromosomes and these chromosomes are encoded. The efficiency of GA was evaluated by applying GA on online software online Hospital Management System. This software contains ten components on which theses metrics are applied. The test cases are extracted from input component metrics and the same is optimized through number of iterations and the derived classes.

Findings from this research is that by applying genetic algorithm on cohesion of variables within component(COVC), cohesion of methods within a component (COMC) and total cohesion complexity metrics (TCCC) metrics provides an optimized result. Evaluation results revealed that the developed approach provides better fitness value with optimization compared to without genetic algorithm.

The comparative analysis also performed between cohesion metrics(COVC, COMC, TCCC) without GA with Cohesion Metrics (COVC, COMC, TCCC) with GA. Paired sample t-test is applied on the data and the finding of this result is that by applying genetic algorithm on cohesion metrics will always give a significant result and the same is acceptable. It's a good change which helps to provide optimized and efficient results.

In future, different machine learning algorithms can be deployed to improve the optimization and enhance the performance of the software cohesion metrics.

## REFERENCES

[1] A. Battou, O. Baz, & D. Mammass (2017). Toward a Framework for Designing Adaptive Educational Hypermedia System Based on Agile Learning Design Approach, *Europe and MENA Cooperation Advances in Information and Communication Technologies* (pp. 113-123). Springer, Cham.

[2] A.Q. Gill, B. Henderson-Sellers, & M. Niazi(2018). Scaling for agility: A reference model for hybrid traditional-agile software development methodologies, *Information Systems Frontiers*, *20*(2), 315-341.

[3] A. Sillitti & G. Succi, (2005). Requirements engineering for agile methods, *Engineering and Managing Software Requirements* (pp. 309-326). Springer, Berlin, Heidelberg.

[4] B. Zachariah & R.N. Rattihalli (2007). A multi criteria optimization model for quality of modular software systems, *Asia-Pacific Journal of Operational Research*, *24*(06), 797-811.

[5] C. Ghezzi, M. Jazayeri & D. Mandrioli(2002). *Fundamentals of software engineering*, Prentice Hall PTR.

[6] C.K. Kwong, L.F. Mu, L. J.F. Tang & X.G Luo(2010). Optimization of software components selection for component-based software system development, *Computers & Industrial Engineering*, *58*(4), 618-624.

[7] CMMI Institute, Maturity Profile Reports, March 2013. http://cmmiinstitute.com/assets/presentations/2013MarCMMI.pdf (accessed March 2014).

[8] F.B. Abreu & M. Goulão(2001). Coupling and cohesion as modularization drivers: Are we being over-persuaded?, *Software Maintenance and Reengineering, 2001. Fifth European Conference on* (pp. 47-57). IEEE.

[9] F.S. Silva, F.S.F Soares, A.L. Peres, I.M. de Azevedo, A.P. L. Vasconcelos, F.K. Kamei & S.R. de LemosMeira, (2015). Using CMMI together with agile software development: A systematic review, *Information and Software Technology*, *58*, 20-43

[10] J.G Borade & V.R. Khalkar (2013). Software project effort and cost estimation techniques, *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(8).

[11] K. Tian & K.Cooper (2006, August). Agile and software product line methods: are they so different , *1st international workshop on agile product line engineering*.

[12] M. Dixon(2018). Generating Embedded Systems Software using a Component Based Development Approach. *GSTF Journal on Computing (JoC)*, *2*(3).

[13] M. Derakhshanmanesh, J. Ebert, M. Grieger & G. Engels(2018). Model-integrating development of software systems: a flexible component-based approach, *Software & Systems Modeling*, 1-30.

[14] Pooja Rana, Rajender Singh (2016), "A Design of Cohesion and Coupling Metrics for Component based Software Systems", International Journal of Computer Applications, (0975 – 8887) Volume 146 – No.4, July 2016, PP- 23-27

[15] Pooja Rana, Rajender Singh (2018), "Comparative Analysis of Cohesion

Metrics For Component Based Software Systems", Journal of Theoretical and Applied Information Technology , (1992 – 8645) Volume 96 – No.14, July 2018, PP- 4369-4378

[16] P.C. Jha, V. Bali, S. Narula, & M. Kalra(2014). Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme, *Journal of Computational Science*, *5*(2), 233-242.

[17] R. Seker, Der Merwe Van, A. J., P. Kotze, M.M, Tanik, & R. Paul(2004). Assessment of coupling and cohesion for component-based software by using Shannon languages, *Journal of Integrated Design and Process Science*, *8*(4), 33-43.

[18] Seyed Mohammad Hossein Hasheminejad, Saeed Jalili, Software Component Identification using Genetic Algorithm, Journal of Obect Technology, Vol 12, No 2, 2013, pages 3: 1-34.

[19] S. Dragicevic, S. Celar, & M. Turic(2017). Bayesian network model for task effort estimation in agile software development, *Journal of Systems and Software*, *127*, 109-119.

[20] Standish Group, 2009. Project Smart, http://www.projectsmart.co.uk/thecurious-case-of-the-chaos-report-2009.html, (viewed October 2014).

[21] S. Parsa & O. Bushehrian(2004, June). A framework to investigate and evaluate genetic clustering algorithms for automatic modularization of software system, *International conference on computational science* (pp. 699-702). Springer, Berlin, Heidelberg.

[22] W. Zhiqiao, C.K. Kwong, J. Tang, & J.W.K. Chan(2012). Integrated model for software component selection with simultaneous consideration of implementation and verification, *Computers & Operations Research*, *39*(12), 3376-3393..

[23] A Albani, S Overhage, and D Birkmeier. Towards a systematic method for identifying business components. In *Proceedings of CBSE, LNCS 5282*, pages 262–277, 2008. doi: 10.1007/978-3-540-87891-9_17.

[24] JK Lee, SJ Jung, SD Kim, WH Jang, and DH Ham. Component Identification Method with Coupling and Cohesion. In *Proceedings of the 8th Asia-Pacific Software Engineering Conference*, pages 79-86, 2001. doi: 10.1109/APSEC.2001.991462.

[25] H Jain, N Chalimeda, N Ivaturi, and B Reddy. Business Component Identification a Formal Approach. In *Proceedings of the 5th IEEE Int. Enterprise Distributed Object Computing Conf.*, pages 183-187, 2001. doi: 10.1109/EDOC.2001.950437.

[26] GR Shahmohammadi, S Jalili, and SMH Hasheminejad: Identification of System Software Components Using Clustering Approach. *Journal of Object Technology (JOT).* 9(6):77-98, 2010. doi: 10.5381/jot.2010.9.6.a4.

[27] HS Hamza. A Framework for Identifying Reusable Software Components Using Formal Concept Analysis. In *Proceedings of the 6th International Conference on Information Technology: New Generations*, pages 813-818, 2009. doi: 10.1109/ITNG.2009.276.

[28] Z-g Cai, X-h Yang, X-y Wang, and A Kavs: A Fuzzy-based Approach for Business Component identification. *Journal of Zhejiang University-SCIENCE C (Computers & Electronics).* 12(9):707-720, 2011. doi: 10.1631/jzus.C1000337.

[29] D Doval, S Mancoridis, and BS Mitchell: Automatic Clustering of Software Systems Using a Genetic Algorithm. In *Proceedings of Int'l Conf. Software Tools and Eng. Practice.* 1999. doi: 10.1109/STEP.1999.798481.

[30] K Praditwong, M Harman, and X Yao: Software Module Clustering as a Multi-Objective Search Problem. *IEEE Trans. Software Eng.* 37(2):264-282, 2011. doi: 10.1109/TSE.2010.26.

[31] CL Simons, IC Parmee, and R Gwynllyw: Interactive, Evolutionary Search in Upstream Object-Oriented Class Design. *IEEE Transactions on Software Engineering.* 36(6):798-816, 2010. doi: 10.1109/TSE.2010.34.