A METAHEURISTICS ALGORITHM APPROACH TO TRACK THE REUSABILITY LEVEL OF A COMPONENT IN DESIGN PHASE

R. Kamalraj[1] | Dr. A. Rajivkannan[2] | K. Kavya[3] | K.R. Nishanth[4]
[1](Computer science and Engineering, SNSCT, Coimbatore, India, mailtokamalraj@yahoo.com)
[2](Computer science and Engineering, KSRCE, Coimbatore, India, rajivkannan@ksrce.ac.in)
[3](Computer science and Engineering, SNSCT, Coimbatore, India, kavya.kmk97@gmail.com)
[4](Computer science and Engineering, SNSCT, Coimbatore, India, itsmenishanthkr@gmail.com)

Abstract— *Reusability is one of the main characteristic required in the component side to reduce the development cost in further project works. So checking the 'Reusability' is suggested in the design phase itself of a software system to identify that whether the system has any 'Reusable' elements for future use. To identify that characteristic many methods and approaches are available. But compared to existing methods the 'FireFly' algorithm can give high optimized result to specify the 'Reusability' Level of the component. To derive the solution for that the different design metrics along with their threshold values and relationships among those metrics are analyzed perfectly. This approach also can be used as a 'Classification Technique' on software components.*

Keywords—FireFly Algorithm, Reusability, Meta-Heuristic algorithm, Package Scalability, Stability

1. INTRODUCTION

Developing a Software System with minimal effort is a challenging activity in Software engineering field. To reduce development cost and effort to get better benefit the development team use various approaches like 'Reusing Components', 'Take Well-Trained People' and 'Advanced Tools' [2]. Among them 'Reusing Existing Components' play an important role to minimize the cost and effort and also it increase productivity of development team. The 3 phases 'Domain analyses', 'Package Analysis' and 'System Analysis' will be followed by Project Management Team to find similar elements from the already completed project source [4]. To reuse an element it should satisfy some criteria to give good functionality without producing any side effects. While designing a system, the development team can track that whether an element or a component to act as 'Reusable Component' in future [3]. In a dynamic fashion the different design metrics of all components should be analyzed to find the fitness of a component in the design phase itself.

A. Requirements for Components Reuse

From the Software Development Life Cycle, after recording the user needs in Software Requirement Specification (SRS) document the design phase begins to give to solve the problem by applying different techniques. In design phase design patterns and 'System Architecture' are created for developing the system. The Designing of a software system states that providing solutions to the give problem as in 'Pictorial Documents'. During that for giving simplicity in the business goal the same or similar element will be grouped together to be component. While making a component it has to be verified to check the quality of 'Reusability'. The reusable components can reduce the stress on software development team, gives efficient 'Resource Management' and it reduces conflict in 'Time Management'. Reusing the existing elements can give good quality of systems to satisfy the customer needs. The metrics are 'Fan-in' and 'Fan-out', 'Cohesion', 'Coupling', 'Component Stability', 'Complexity' and 'Package Size'. Those are discussed below [5].

B. Relationship among Fan-in, Fan-out, Coupling, Stability and Complexity

Design phase has the responsibility of creating 'High-Level' patterns and 'Low-Level' patterns. The 'High-Level' pattern of a system describes the message passing between the different component or packages in the design. Each component of a system may be depending upon other component for any of the reasons.

The different reasons are

a) To get Control to execute its tasks

b) To get Data to give result and

c) To get Control to pass control to other elements in the system.

One component may have full control to call other components in the entire system. So the number of control comes out of a component is known as 'Fan-out'. The number of incoming control to a component is called 'Fan-in'. So total number of connections is (TOC) can be represented like in below mathematical notation.

TOC= (no. of Fan-in) + (no. of Fan-out) - (1)

If a component has only 'Fan-out' characteristic then it may be 'Independent' one and it has only 'Export Coupling' [1]. So it can act as 'Server Component' of other components in the system. But if a component has one and only 'Fan-in' then it may be depending on other component(s) and it reaches the metric 'Import Coupling' type. So this type of component can exist without help of suitable parent components. This is called as 'Pure Client

Component'. By comparing the dependency of a component the 'Stability' metric can be measured. The server component can run without help of other components. So it has high 'Stability' value or 1. But the Client component may not have high stability value due to its dependency on other components. When a Component has very high 'Fan-in' and 'Fan-out' it may have an 'Excessive Complexity'. It cannot be preferred for further use due to required repairing mechanism effort may be high. Excessive complexity component is very difficult to understand its elements communication to produce the result. That kind of component elements may have high cross communication and it leads to difficult to redesign or improve the quality of the component.

C. Relation between Cohesion and Scalability

The Logical relationship among component elements is known as 'Cohesion' metric. An element in a component may be related with another element(s) in the same package through different relationships like 'Generalization', 'Aggregation' or 'Association'. Thorough analysis on 'Cohesion' of different elements may reduce or increase the identified reuse component size. If elements are selected only for specified requirements in the new domain then remaining elements may be removed. That may reduce the original size of the reuse component. This can be stated below.

Selected Component Size – SCS

Original Size – OS

SE – No. of Selected Elements for Particular Function

$$SCS = OS - SE \quad (2)$$

Suppose, selected elements from the same reuse component and other elements from different components may be required for fulfilling the current needs. Hence the original size of the same reuse component may be increased if all elements of that component selected. So the above equation can be changed as

SEC- No. Of elements from other components

$$SCS = OS – ((SE) + (SEC)) \quad (3)$$

2. FIREFLY ALGORITHM

The 'FireFly' Algorithm is one of the 'meta-heuristic' algorithm to produce optimized result in an application domain [9].

The concept of this algorithm can be represented as follows.

i) One FireFly can attract other FireFly

ii) Attractiveness is direct proportional to their brightness

iii) if brightness increases and reached its limit then high-attractiveness is possible.

To derive the optimal result the search space should be identified with its possible results and constraints. The populated constraints and cases have to be collected from the search space. Different parts of the search space have to monitored to collect the required cases and conditions. The identified case with a condition can be considered as a 'FireFly'. That created case may attract another case when it is satisfied condition perfectly. The 'Attracted Firefly' can come to its peak level when its conditions are matched perfectly. If all the cases are reached its peak level than expected solution is derived. If any one of the case

'Firefly' not in the peak level then it leads to that expected result cannot be reached at that moment.

3. FIREFLY ALGORITHM IN COMPONENT DESINGING

In every software development new components are created and existing components are reused to solve the given problem domain [7]. So searching similar components is a challenging task due to perfection of that activity. If identified components not satisfied the development requirements then it may lead wastage of time and resources. To avoid those problems, during a system development the identified new components can have 'Reusability' characteristic or not. It is an additional activity of a 'Designer' to check the different components and their metrics value in each construction phase.

A. Search Space for Checking Reusability

The 'Search Space' for Reusability specifies the different design metrics and their values or threshold values of new components in the proposed system. Assume that the different design metrics are different 'FireFly' of a particular component. When each 'Firefly' attract each other then expected result from the search space is obtained.

B. FireFlies in the Search Space

As discussed above, the design metrics such as

a) Coupling

b) Stability

c) Complexity

are treated as 'FireFly'.

C. Coupling FireFly

This Firefly's case is whether the component has only Export Coupling. It means that the component provides services to other component in the system. While analyzing the coupling value and its type, if designer found that the component has only this type of coupling then this 'Firefly' can fly and it lightness increases to attract other metrics or 'Firefly' such as 'Stability'.

D. Stability FireFly

The 'Stability Firefly can have the constraint like the import coupling is not occurred in the system at any cost. So the component is not having dependency with other modules and it can survive its own stuff. When a component is independent and doing service other components in the system it has high stability value like 1. When 'Coupling FireFly' fly, the 'Stability Firefly' is attracted to check its level. If it is reached 1 then it can meet with 'Coupling FireFly'.

E. Complexity FireFly

It has a conditional statement like whether the component elements are in 'Excessive Complexity' or not. By analyzing 'Structural Complexity' of a Component the complexity level can be identified. If a component has

'Excessive Complexity' then this firefly cannot able to fly to meet other fireflies like 'Coupling FireFly' and 'Stability Firefly'. When this 'Firefly' has low complexity value then it can able to fly and attract other 'FireFlies'. When these 'Fireflies' are in flying state and attracted each other then that corresponding component is reached the solution 'Reusability' one. s

So consider the intensity value of different design metrics of a component like

Coupling Intensity value = Ic

Stability Intensity value = Is

Complexity Intensity value = Ix

The firefly algorithm for defining the 'Reusability' level of package can be represented as follows.

Begin

1) Objective function: C(x), x =(Ic, Is,Ix);

2) Generate an initial population of fireflies

3) Formulate light intensity I so that it is associated with C(x)

4) if C(Coupling) == 'Export Coupling'

Then

Ic = High Attractiveness

Ic move Towards Is

if C(Stability) == '1'

Then

Is = High Attractiveness

Is move towards Ic.

If

C(Complexity) == Low Complexity

Then

Ix = 'High Attractiveness

Ix move towards Ic and Is.

5) If

(Ic && Is && Ix = 'High Attractiveness)

Then

Component = 'Reusable One'

else if(Ic or Is or Ix = 'Not Attractive')

Then

Component = Not Reusable one

End.

When any of the design metric not satisfied then they may not be in the 'Flying State'. That's why the final statement of the 5th step is the component cannot be reusable one. By implementing this algorithm as a tool it will help to check the package characteristic dynamically to know the level of 'Component Reusability'.

4. RESULTS AND DISCUSSIONS

The proposed algorithm can be used to trace the level of package reusability by checking the design metrics values. When a design metric of a component lies between its ranges then it indicates that some amount of modification required in future to make the component as 'Reusable' one. The intermediate values specifying the effort required for improving 'Reusability' characteristic. For an example, Assume that the Instability metric of a component value = 0.75. Then that corresponding component may require only 25% of effort from total effort for implemented that appropriate component. This proposed methodology will give appropriate values of those design metric and those will be recorded for further analysis.

In that same way, suppose a component has combination of 'Fan-in' and 'Fan-out' then Coupling value will be in the intermediate level. If a component 'Highly-Coupled' Type and no 'Fan-in' type then there modification may not be needed. Suppose the same component 'Highly-Coupled' type with 'Fan-in' type then it has dependency relationship with other modules in the same system. So it requires high development cost to remove the dependency. That dependency can be removed by parent elements from other component will be identified and grouped in the child component. That scales the size of the child package. The proposed approach classifies the components as 'Reusable' one or not.

5. CONCLUSION

Firefly Algorithm can lead the development team with higher confidence on tracing the reusable elements in the proposed system. And also it shows the results for further modification required or not for making that component as 'Reusable' one or not. This 'Meta-heuristic' algorithm can give optimized results to categorize components as 'Reusable' one. After categorizing the components then they are accessed in different approaches to find the suitable component in further projects [8]. This proposed methodology takes only less effort from the design phase to track the components type. So it may improve 'Resource Management', 'Time Management' and 'Quality Management' to produce the system with high quality within the given time period [6].

6. REFERENCES

[1] Andrea Capiluppi and Cornelia Boldyreff (2007) 'Coupling Patterns in the Effective Reuse of Open Source Software' IEEE Computer Society.

[2] Anthony Finkelstein, Spanoudakis G. and Ryan .M (1996) 'Software Package Requirements & Procurement' Software Specification and Design, 1996, Proceedings of the 8th International Workshop.

[3] Merijn de Jonge (2003) 'Package-Based Software Development'Euromicro, p. 76, 29th Euromicro Conference (EUROMICRO'03).

[4] Mili A., Mili R. and Mittermeir R.T (1998) 'A survey of software reuse libraries', Annals of Software Engineering, vol.5 1998.

[5] Wang A.J.A (2002), "Reuse Metrics and Assessment in Component-Based Development",Proceedings of Software Engineering and Applications, Vol. 47, pp. 693- 707.

[6] Nancy Bazilchuk; Parastoo Mohagheghi, (2005) "The Advantages of Reused Software Components". R&D and Technology Transfer.

[7] Parvinder S. Sandhu; Hardeep Singh , (2006) "Automatic Reusability Appraisal of Software Components using Neurofuzzy Approach", International Journal of Information Technology Vol 3, No. 3.

[8] Rajesh Bhatia; Mayank Dave; R. C. Joshi (2006) "Retrieval of Most Relevant Reusable Component Using Genetic Algorithms ", software Engineering Research and Practice -SERP, pp. 151-155, 2006.

[9] X. S. Yang,(2009) "Firefly algorithms for multimodal optimization, in:Stochastic Algorithms: Foundations and Applications, SAGA 2009, Computer Sciences, Vol. 5792, pp. 169-178.