International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-6, August 2012

Intelligent Classification and Retrieval of Software Components

N. Rajasekhar Reddy, R.Saraswati

Abstract: ***This work proposes a new methodology for smart classification and retrieval of software mechanism based onuser–defined necessities. The classification proposal utilizes a dedicated genetic algorithm which evolves a tiny number of classifiers by dividing the position of available components stored in a database into positive subsets (clusters). Each classifier consequently becomes the leader-representative of its cluster. When a customer wishes to trace a component he/she identifies the preferred characteristics (component profile) which are then compared with the distinctiveness of the available classifiers. The neighboring classifier matching the required distinctiveness over a user-defined threshold will effect in the "winning" set of components belong to its cluster that is accessible to the user in descending matching strength. We have validated our methodology over a artificial dataset of components and the consequences obtained were very encouraging. Last, we here the web application developed to bear the proposed intelligent categorization method.***

I. INTRODUCTION

Software components perhaps conceived as self-determining and autonomous units which are accessible by vendors and can be "glued" together, using their communal interfaces, to appearance a complete software scheme, which is more dependable than progress ex nihilo [10].To this end, users of software mechanism have to explore between the components obtainable in the market, evaluate their characteristics and purchase the most appropriate ones for integration according to the efficient requirements and constraints of their scheme. However, currently the obtainable schemes in the market for probing and retrieving components are lacking in terms of: a rich set of uniqueness that would agree to locating components using diversified components properties and features with the aptitude of combinations, a fast and robust rescue scheme, and competent mapping of user necessities on component characteristics. We suggest an innovative way to categorize components with the aim of contribution fast retrieval of the correct mechanism according to the preferences of the user. In our proposal, component classification is based on a predefined set of uniqueness, from which the user may choose those of interest (preferences). The user's preferences are utilized by an intelligent algorithm to return those software mechanism the characteristics of which contest the preferences up to a user distinct level. The rest of the paper is prepared as follows: Section 2presents a short literature review. Section 3 presents the proposed method. Section 4 presents a set of experiments, followed by a concise presentation of the sustaining software tool. Finally, Section 6 draws the conclusions and suggests future research steps.

Manuscript published on 30 August 2012.
***** Correspondence Author (s)
N. Rajasekhar Reddy*, Associate Professor MITS Engineering College Chittoor, Andhra Pradesh, India.
R.Saraswati, Associate Professor MITS Engineering College Chittoor, Andhra Pradesh, India.

II. RELEVANT WORK

A basic problem for software module providers is that of organizing a set of components for fast retrieval. Computational Intelligence (CI) technologies such as evolutionary calculation, neural networks, clustering algorithms [8], and fuzzy sets [2,9] have newly been used in component based software engineering [6].There are two admired representation schemes for software mechanism. The first is based on a prohibited vocabulary, which is organised in a firm hierarchical way. This schemehas received a lot of criticism as being inflexible and not producing a superior classification [4]. The second approach postulates a representation which is based on a succession of characteristics (attributes) that suppose values. This scheme is more flexible with regards to expansibility [4], which we have also adopted in the present work.

One of the earliest efforts in categorization of software reusable mechanism is based on a faceted method [3].Another influencing constituent selection system has been developed in the circumstance of the CdCE project [7]. Finally in the CLARiFi project a energetic classification schema has-been adopted [1].

III. AN INTELLIGENT METHOD FOR CLASSIFICATION AND RETRIEVAL OF SOFTWARE COMPONENTS

Our technique performs intelligent mechanism classification and fast retrieval of a requested component via a web available software tool. The classification/retrieval procedures are based on a dedicated genetic algorithm that processes a set of predefined component uniqueness. In what follows, we explain the basic concepts of our technique:

Encoding: An encoding scheme for the software module characteristics maps their initial appearance (whatever that may be, e.g. linguistic terms, numerical types, etc.) to a form that will be used by a genetic algorithm to determine component classifiers. In our case we used dual strings (series of 0/1) to encode the constituent characteristics, the length of which depends on the aggregation of the number of bits needed to symbolize all possible standards of each distinct attribute.

Classifier Discovery: The genetic algorithm attempts to determine several different classifiers, each of which classifies a number of software mechanisms into a homogenous set in terms of uniqueness.

The classifying sets may have frequent elements as the classification process is based on component characteristics, with which it attempts to find huge groups of components with frequent values. Typically, there will be a large number of mechanisms classified against a small number of classifiers (20 in our case). Thus, probing for a component will be performed by exploratory the user preferences against the classifiers rather than the real components, something which will result in a fast process. It is also possible that some components may not be confidential at all as this depends on the selection of a threshold limit that specifies the similarity of a component with a classifier (i.e. the number of perfectly coordinated characteristics).

Component Retrieval: This is the circumstances where a user tries to locate a explicit component. First, she/he determines the desired standards of the component characteristics, thus forming her/his preferences. Second, she/he sets the similar threshold value (obviously the lower the threshold value the more apparatus will be returned). The system will then instruct the user's request as a bit string and will compare it against all classifiers. The neighboring match will indicate the "winning" classifier and will activate the return of those components that communicate to this classifier. An overview of the rescue process is presented in Figure 1, where the sets of classified components are represented as ellipses. The classifier corresponding to each class of components is drawn next to its set. The classes can be overlapping as previously mentioned. Moreover, we can observe a pool of unclassified components that may possibly occur as a consequence of the threshold set.

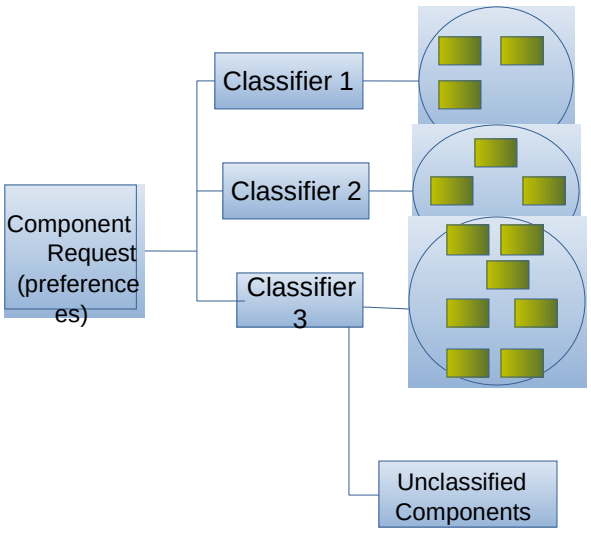


Figure 1: Overview of the system

3.1 Components characteristics and encoding

In the current untried setting each component is described with a set of 15 unique features, which were identified by examining a component from different perspectives (functional and non-functional). Our selected characteristics are: General functionality, Specific functionality, Platform, Implementation language, Operating system independence, Synchronization, Visibility of execution, Price, Processor utilization, Memory utilization, Disk Utilization, Binding, Data encryption, Data open format compatibility. Each of the aforementioned uniqueness is encoded as a string of 0/1. The encoding of a component is essentially the series association of its characteristics' bit strings. We calculated the required bits for all characteristics and accomplished to a string length equal to60 bits. Figure 2 shows an example of a component with explicit characteristics and its applicable bit encoding at the top.

111011001000011011100110001000001100010100010001001000101001
General functionality: Security **Specific functionality:** Secure Email **Operating System Independence:** Windows 98, 2000, Me, NT, XP, Unix, Linux **Implementation language:** C# **Platform:** Visual Studio .NET **Processor Utilization:** 100-200MHz **Memory Utilization:** 128MB **Disk Utilization:** 100-500KB **Data Encryption:** No **Data Open Format Compatibility:** No **Password Protected:** Yes **Visibility of implementation:** Glass box **Price:** $1-100 **Synchronization:** synchronous **Binding:** Static

Figure2: Example of component encoding

3.2 A Genetic Algorithm for identifying the classifiers

A enthusiastic GA [5] was developed to change candidate classifiers and select the optimal solution in terms of number of components in the equivalent classes, which works indiscrete steps as follows:

1. Create a random population of 100 chromosomes -potential classifiers
2. For every generation of the hereditary algorithm:
   - 2.1 Apply crossover to every duo of classifiers, where each pair is arbitrarily selected according to a crossover probability
   - 2.2 Apply mutation to a arbitrarily selected classifier according to a mutation prospect.
3. Perform module classification: for each of the100 classifiers:
   a. Compare each classifier's standards of characteristics with those of each constituent. If a component is close sufficient (determined by a threshold) to a classifier then allocate the component to the class represented by this classifier. Normally, a huge number of components will be assigned to each chromosome-classifier
   b. Select the top 20 classifiers (chromosomes) in terms of the number of assigned workings. Then find the average number of assigned workings of the 20 classifiers. This is the average fitness of the current generation.
   c. If the average fitness of the present generation is greater than that of the previous generation then create a new population by selecting chromosomes according to their fitness and repeat from step 3. Otherwise do not create anew population and repeat from step 2

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The above algorithm is repeated until a termination condition is reached. In our case the algorithm terminates if no improvement in the average fitness of the population is observed for 100 generations. A very important parameter is the value of the threshold, which determines whether a component belongs to a certain classifier. For example, a value of 40% means that at least 40% of the values of the classifier characteristics are identical to those of a component. This threshold essentially determines the "success" level of a classifier to gather a rich number of components in his class.

IV. EXPERIMENTS AND RESULTS

The first segment of the experiments was concerned with the categorization of a pool of components, whereas in the second segment we investigated the recovery of specific components. For the classification phase, we created arbitrarily 1000components, each comprising 60 bits. The results reported are averages over 100 runs. The categorization of the components is based on the 15 characteristics described in section 3.1. The threshold parameter is of paramount importance to our technique since it is a measure of similarity between the component characteristics and the classifier characteristics. We set the threshold value to suppose the values of 30%, 40%, 50%, 60%, 70% and 80% for comparison purposes. The value of 30% produced classifiers, where each classified almost all of the accessible software components. This denotes that the classifiers resulting cannot differentiate between the components. The threshold value of80% did not produce high-quality results either, because each classifier classified only between one and three mechanism, which is also undesirable as it leaves many components unspecified (recall that there are only 20 classifiers). Similarly, unsatisfactory were the consequences when the threshold was set to 70%.The results for the threshold values of 40%, 50% and 60%are scheduled in Table 2. The "Average" columns denote the average number of components confidential by each classifier, while the "Not confidential", denote the number of unspecified components. The scores for 50% are quite successful, since there are no unspecified components and each classifier includes almost half of the mechanism (47.5%). Thus, in the retrieval phase only half of the components need to be searched. Moving along the same appearance, the value of 60% is also acceptable since each class contains a small number of mechanism (58.3 on average), but there is a significant number of unspecified mechanism. The threshold of 40% is the worse since almost all of the components are confidential by each classifier. This threshold, however, was also included in the repossession phase for testing purposes. Furthermore, we have conducted experiments with5, 10 and 15 classifiers but the consequences were not satisfactory.

Table2. Experimental result of component classification by 20 classifiers

Threshold value:40%		Threshold value:50%		Threshold value:60%	
Average	Not classified	Average	Not classified	Average	Not classified
937.03	0	475.99	0	58.30	312.87

To test the retrieval segment we created 10 random user needs searching for software mechanism. Then we set the threshold from 40% to 70% at increments of 10% as shown in Table 3. We can view that the 40% threshold returned aricher number of mechanism, but not all of them were applicable to the users' query as expected. The 50% and 60%values retrieved less but additional relevant mechanism. The 70%threshold returned consequences for some of the queries only. However, the few cases for which we obtained consequences were highly relevant to the users' requests.

Table3: Retrieval Phase Results

40%	50%	60%	70%
Average number of components per classifier			
31,9	23	8,2	2,4

The experiments were conducted with a implement that exists together as standalone system and as a web application. The main functions of the tool are summarized to probing for software components, inserting a original component in the database and some administration functions. The tool differentiates between uncomplicated users (re users) who can only look for components and producers (component developers) who can insert original components. In Figure 3 we can see the uniqueness that may be selected when probing for a component.

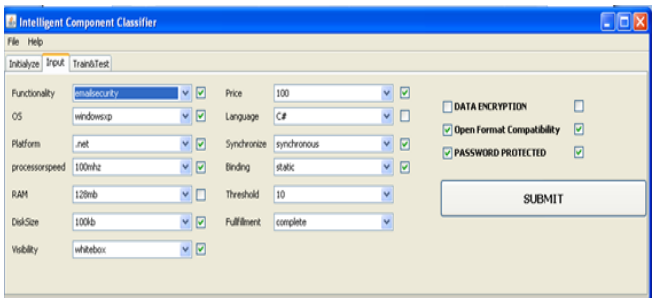
V.CONCLUSIONS AND FUTURE WORK

We have intended and implemented an intelligent system for software constituent classification and retrieval. Classification is based on a tiny set of classifiers which are evolved (formed) with the help of a dedicated genetic algorithm. Each classifier evolved by the GA attempts to classify the major possible number of software mechanism according to common individuality. Retrieval of the relevant mechanism may then be performed by comparing the developer's necessities with those of the classifiers. If the requirements are close enough to those of a classifier then the components that communicate to that classifier are returned. Thus, comparing a component's condition with only those of the classifiers instead of the entire set of accessible components saves a important amount of time and endeavor in the retrieval phase. A threshold is also used when developing the classifiers, which determines the degree (percentage) of comparison with a classifier that is necessary to classify a component in a convinced class. The threshold value has been originate to have a profound pressure in both the classifier's design phase (with the GAs) and the retrieval phase. The experiments exposed that the optimal threshold values for comparison were lying between 40% and 60%. In the future we intend to investigate the optimal achievement when a new component becomes obtainable. There are two possibilities: moreover to find the closest matching classifier and to insert it to the relevant class or to completely reorganize the mechanism by running the genetic algorithm formerly more. Also, ranking the mechanism that are returned by the system would be useful, (remember that the system returns all the mechanism that correspond to a classifier). Such a ranking method could be based on how secure a component is to the classifier or on user specific criteria which put bias on some characteristics

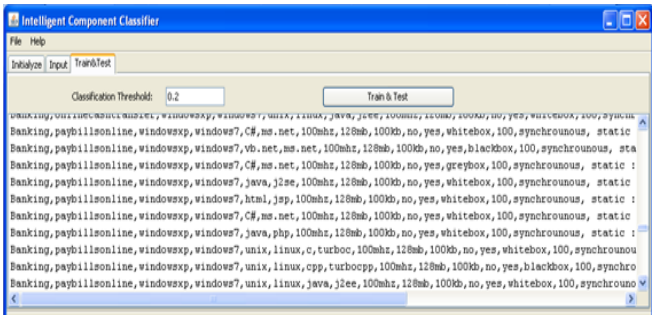
Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

(e.g. memory consumption is more important that value). Another route for enhancing our work will be based on a semantic interpretation of the user's/developer's queries beside a repository which is inspired beginning the work presented in [11].

a. Parameter initialization



b. Requirements input



c. Training and Testing

Figure 3. Searching for a component

REFERENCES

[1] CLARiFi, IST-1999-11631, http://clarify.eng.it

[2] E. Damiani and M. G. Fugini, "Automatic thesaurus construction supporting fuzzy retrieval of reusable components", Proceedings of the ACM Symposium on Applied Computing Tennessee, US, pp. 542-547, 1995

[3] R. P. Diaz, "Implementing Faceted Classification for Software Reuse", Communications of the ACM, Vol. 34, No. 5, pp.88-97, 1991

[4] W. B. Frakes and T. P. Pole "An Empirical Study Representation Methods for Reusable Software Components",IEEE Transactions on Software Engineering archive, vol. 20,no. 8, pp. 617-630, 1994

[5] D. E. Goldberg, "Genetic Algorithms", Addison-Wesley, 1989

[6] J. Lee, "Software Engineering with Computational Intelligence", Springer, 2003

[7] V. Maxville, J. Armarego, C.P. Lam, "Intelligent Component Selection", 28th Annual International Computer Software and Applications Conference, COMPSAC, pp 244-249, 2004

[8] S. Nakkrasae, P. Sophatsathit and W. R. Edwards, "Fuzzy Subtractive Clustering Based Indexing Approach for Software Components Classification", International Journal of Computer& Information Science, vol 5, no. 1, March 2005

[9] W. Pedrycz and J. Waletzky, "Fuzzy clustering in software reusability", Software – Practice and Experience, vol.27, no.3, pp.245-270, 1997

[10] C. Szyperski, D. Gruntz and S. Murer, "Component Software", Addison Weslsey, 2002

[11] H. Yao and L. Etzkorn, "Towards a semantic-based approach for software reusable component classification and retrieval", Proceedings of the 42nd annual Southeast regional conference,pp. 110-115, 2004

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)