Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Genetic optimized software component reuse model for reducing development and maintenance efforts

Amit Verma *, Gurpreet Singh Kamboj, Iqbaldeep Kaur

Department of Computer Science and Engineering, CGC Landran, Mohali-140307, Punjab, India

ARTICLE INFO

Article history:
Received 9 June 2017
Received in revised form
23 August 2017
Accepted 1 September 2017

Keywords:
Software engineering
Component based Image retrieval
Genetic algorithm and reusability

ABSTRACT

The proposed algorithm is a hybrid approach to find a software component. The hybrid approach is a combination of various modules called data extraction, Fact and rules, Optimization with genetic algorithm, etc. all these modules process raw data and provide the output as a component for reuse on the basis of various priorities matrixes. Proposed approach uses priority vector for the processing of all entries and define priorities on the basis of availability of various data factors along with issues in the software component. The entire component derived through the raking process with the help of genetic algorithm for the final output. Proposed approach provides average accuracy 99% for detection of software components. Various other parameters are also compared with the existing developed algorithms which provide comparative study and enhancement of the proposed method.

1. Introduction

Development of software systems are not only a structure of codes. It is used to fulfill the requirements of users with some calculation methods. These software systems are designed and implemented to humans to perform some specific task. Lots of modules are integrated with the software system which is used to check the performance and help in development like libraries, external executable code (Grundy, 1999). The fast development in this field is used software reusability. It is a powerful technique to design and develop software components and make them independent. Lots of designed products are stored in the repositories. They are designed for reuse in future deployments. Reusable component is cloned from the existing libraries and developed with some other component to form a new service or software system to their users (Mili et al., 1997).

The components required to be reusable which means that it can instantiate them in any context, and they can control as expected. They should integrally be multi-tenant, which means they don't have data that is global and varying except if it relates to all uses. They should not have much dependence and drag in too many other components. Using the component should not mark remaining software stack, which means they need to run or be able to run in an OSGi or OSGi like vessel and not have naming conflicts with other components (Inoue et al., 2005).

Software component reuse is a significant concept in software development, as it optimizes software development efforts, cost, time and increase flexibility and reliability. Software CBSE defines the reuse of software-components, which could be retrievable and assembled into requests of specific domains.

In existing software-component repositories only retrieve a limited set of software-components and some don't efficient user queries. In-related software-components might exist defined too smaller. The method of the repository itself (Cai et al., 2000) often doesn't reflect semantic relationships among software-components and thus ignores significant component retrieval and regeneration of semantic interrelated software-components.

In this research paper describes the ontology and faced identification based Meta data repository and repository component for retrieval and software components as shown in Fig. 1.

In this research work, the work is on genetic algorithm to optimize the component retrieval data based on fit value. The genetic optimization approach is designed to solve the difficult problems (Complexity, cost, energy and Time consumptions).

* Corresponding Author.
Email Address: dramitverma.cu@gmail.com (A. Verma)
https://doi.org/10.21833/ijaas.2017.010.020
2313-626X/

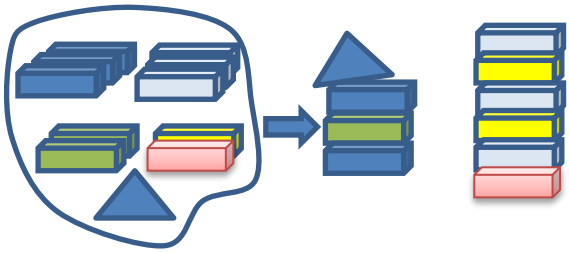

Fig. 1: Component based system

2. Literature review

Dutta and Sengupta (2015) focused on the development of software systems based on component reuse and their storage structures. The author presents the approach for detecting components based on various parameters and provide a method that how to choose components which fulfill the requirements of future developments. Gupta and Kumar (2013) presented a software component reuse model based on a classification technique which helps developers manage the software components in terms of storage, detection of components on semantic domain and taxonomies. The repository system that used in the existing system is not able to provide the information for all the components. It's limited to a set of components. But in the proposed system in this research, analyses the overall input and classified the component based on various parameters and provide efficient output for software component reuse. Basha and Moiz (2012) the author in this research define the difference between the component and domain engineering. The author present change in domain engineering of the technology based on component technologies. Shirali-Shahreza and Shirali-Shahreza (2010) provided the research method in the field of software component which helps developers to use components based on reviewed and surveyed manners. It helps to reuse the components from existing repositories and make the development speed faster and reliable for future developments. Crnkovic et al. (2002) described that the main motive of this conference is aware about the software component reuse techniques and their uses in this domain. The component reuse process provides less time consuming for large scale software development. This domain can save maximum time and cost with much reliability of development of software engineering. Jia and Gu (2002) discussed about the software reuse in a workshop on software engineering. The main concept of this paper is to provide knowledge about software component reuse to the researcher and their benefits for future development that how to store the component, component independence, reusability and their effects on development. It's a powerful concept in software development which provide a higher rate of cost and time saving with reliable development. Wang et al. (2000) presented software management approach for low cost reuse. The development of software systems is more time taking process in this field which can be more costly. But in this research some back-end processing approaches are used to minimize the cost of software reuse. The processed using component reuse in back ends processing which minimize the development cost for the design and development of software systems. Penix et al. (1995) proposed a method which is used to classify software components based on stages. The other parameter which used to classify the software component is their semantic feature which helps to check the working capability and their performance so that it can provide more reliable performance when reuse (Table 1).

Table 1: Comparative analysis techniques for component reuse

Methodology	Author	Purpose/issue
SBA for R of RBC	(Fanchao et al., 2006)	Study the M of BCR
SBRM of RSC	(Al Saiyd et al., 2010)	Define the MP that is related to CBD
SBR of SC through FI	(Şora and Todinca, 2006)	Define the Principles Used in the SoC
ACR and R and A using FS	(Penix, 1998)	IandR of C pertinent to a problem
UGA to IIRS	(Radwan et al., 2008)	Define FF for app(IR)

3. Simulation model

Software component reusability reduces time consumption, efforts, and cost of software development process. Here in this research the proposed approach process component's data to find the software component for reuse. There are various steps followed by the system to find the software components. In this process, the first step is to load raw data from the dataset into a software reuse system and process that data to arrange sub-entries. After processing of all the entries, the system asks the user to enter the component for searching from repository's detail. The entire component processed from fact and rule programming to refine the dataset and eliminate other unwanted entries.

Fact and rule processing used to arrange data in the right manners as the data loaded from the dataset in the form of raw material (Fig. 2).

The processed data pass to the next step and Genetic algorithm process entries on the basis of various sub-properties. The genetic algorithm process and define the priorities and ranking of the component and used to generate optimal output for user search. The software components are retrieved from the dataset and shown according to the quality and data available along with the issues of the software component. After processing the component the proposed approach evaluates result parameters and stops all the objects in memory for another process.

In this methodological focus on genetic algorithm, this algorithm is an initialize the set of size, i.e.,

called a population. Problem Solutions from individual population are used and reserved to new population. This is hope, that the novel population would be better than previous one. Results which are particular to form novel solution, i.e. data stream bits are selected with the help of a fitness function, the suitable phases they have to regenerate.

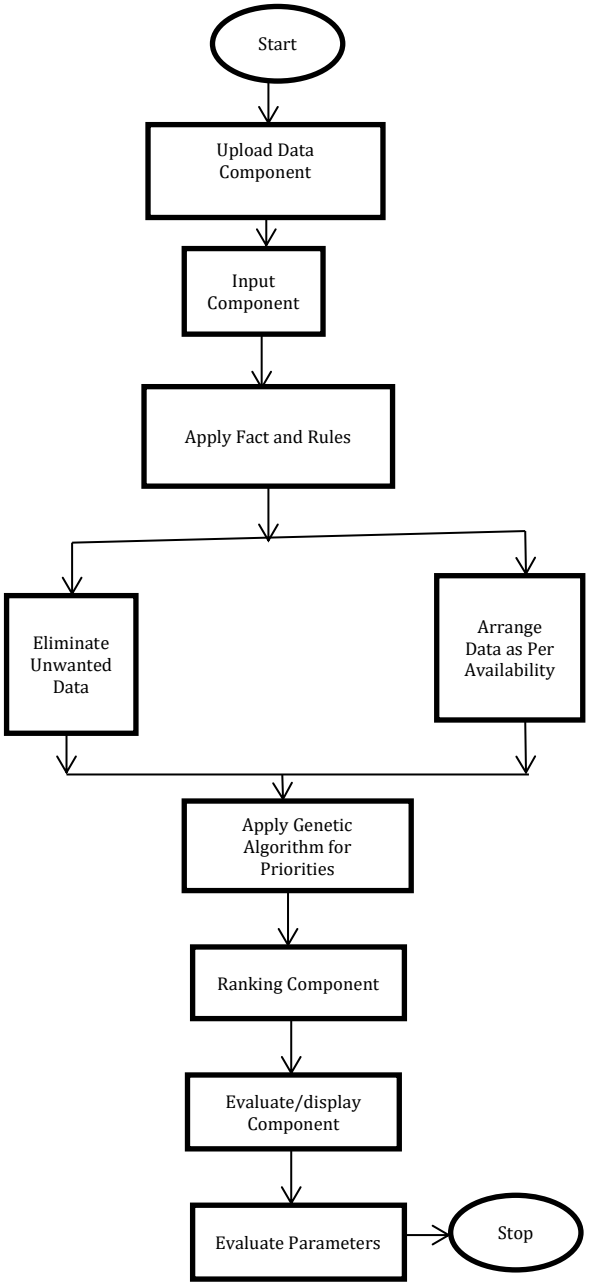

Fig. 2: Proposed flow chart

GO techniques, to solve an optimization issue by repetition the following three operators:

- Selection
- Crossover
- Mutation

4. Simulation results and discussions

In this section, results have been discussed with main form which is used to link all the modules in the software reuse system. The first task in this is to load all the data entries in the memory.The .NET is the technology from Microsoft, on which all other Microsoft technologies will be depending on in the future. It is a major technology change, introduced by Microsoft, to latch the market from the SUN's Java.

System use matrix based approach to manage all the entries on server side (Fig. 3). Once all the entries uploaded to the server for processing it forward the control to the next module for data extraction and verification of data entries.

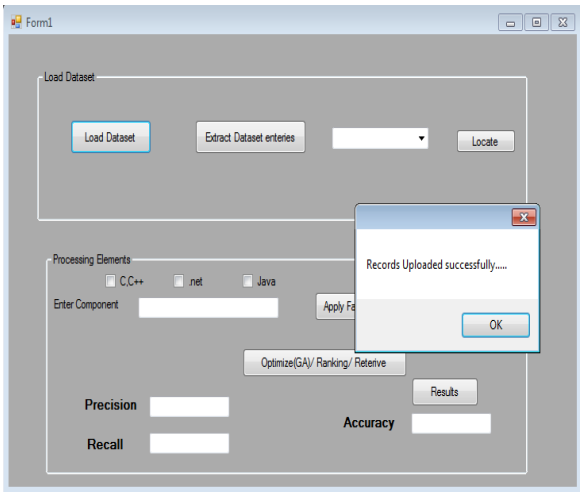

Fig. 3: Component dataset upload

The extract data entries are one of the modules which used to extract data entries from raw dataset. The extraction process used to show data and verify entries through locate button. It also used to arrange the data in matrix form and add joins with other referred entries (Fig. 4).

After processing the data entries system extract all the data from uploading dataset (Fig. 5). The extracted dataset can be verified with the help of locating on the main module. Locate is a verification module that used to check the uploaded data entries. Data entries are in upload content having various sub columns like design and other documents related to the software reuse system.

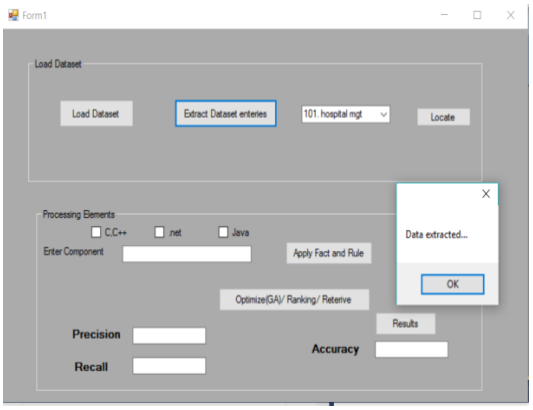

Fig. 4: Extract dataset entries

Dataset passes through the fact and rules module to check the data validations and refine data entries. Fact and rules process data according to the input component, which was entered by the user to

process and find from the software reuse dataset. Fact and rule module eliminate all the data entries which are without software repositories. Other management of fact and rule module is to validate all the data rows based on input software component.

Fig. 5: Process data enterprises

After processing through genetic algorithm the software components retrieved in new with all the details and priorities on the basis of various factors (Fig. 6). Priorities are depended upon the data availabilities of software components and other factors like issues in software component. Selected component will be opened in a new with location on component in software systems.

The graph shows results of proposed approach and comparison with existing approaches (Fig. 7). The results show better performance in terms of accuracy as compare to other existing approaches for component retrieval.

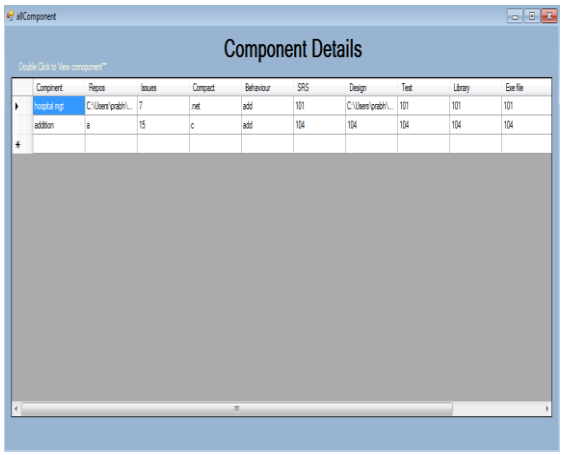

Fig. 6: Process component using GA

In Table 2, other parameters are also calculated in this module to check the performance of the proposed approach. The high rate of precision and recall are shows high performance of a system. Here the working of the proposed approach to extract the component from the uploading dataset.

Table 2 compares the various test cases for existing and proposed approach in terms of precision (Fig. 8). The performance of the proposed approach is higher as compare to other test cases of existing.

Precision calculated in both existing and proposed approaches in this graph with the help of various test cases. The performance of the proposed approach shows better results in all the test cases. The precision rate is stable in all the cases as higher than existing.

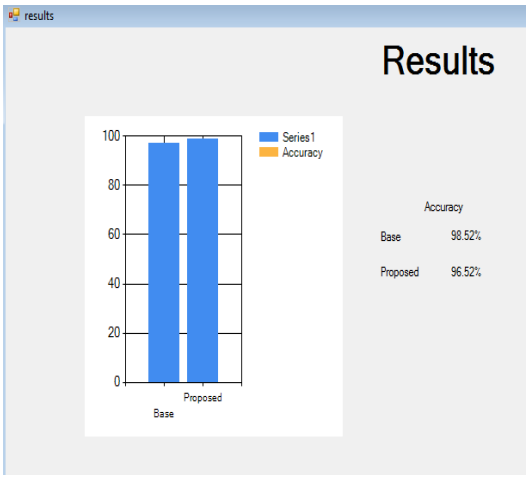

Fig. 7: Result with proposed and base work

Table 2: Test case with precision parameters in component based software engineering

Precision	test1	test2	test3	test4
Base	0.5	0.59	0.568	0.658
Proposed	0.7	0.69	0.66	0.71

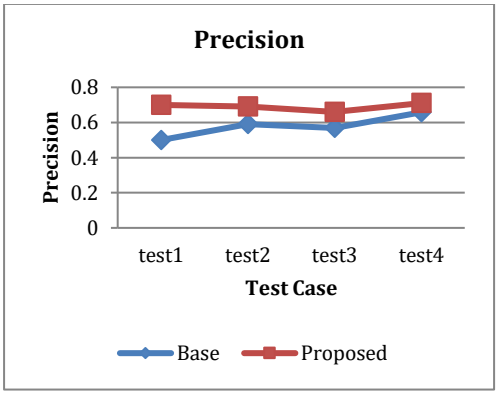

Fig. 8: Precision

Table 3 compares the various test cases for existing and proposed approach in terms of accuracy. The performance of the proposed approach is higher as compare to other test cases of existing.

Accurately calculated in both existing and proposed approaches in this graph with the help of various test cases for retrieving software component on the basis of various extracting rules and optimizations (Fig. 9). The performance of the proposed approach shows better results in all the test cases. The Accuracy rate is stable in all the cases as higher than existing.

Table 3: Test Cases based on base and proposed with accuracy parameter

Accuracy	test1	test2	test3	test4
Base	97.36	96.254	97.65	97.325
Proposed	98.52	98.47	99.36	98.78

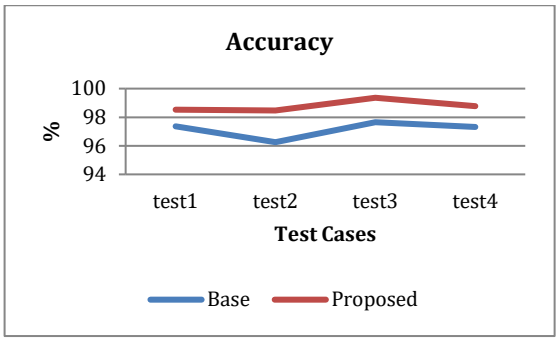

Fig. 9: Accuracy

Table 4 compares the various test cases for existing and proposed approach in terms of recall. The performance of the proposed approach is higher as compare to other test cases of existing. Recall calculated in both existing and proposed approaches in this graph with the help of various test cases. The performance of the proposed approach shows better results in all the test cases. The recall rate is stable in all the cases as higher than existing.

Table 4: Test case in recall based on base and proposed work

Recall	test1	test2	test3	test4
Base	0.785	0.915	0.9012	0.89254
Proposed	0.946	0.948	0.9386	0.9456

5. Conclusion and future scope

The proposed approach performs better as compared to other existing approaches in terms of accuracy and result component on the basis of various components. The component quality and efforts reduced in the proposed approach because various factors are considered to find the component from a raw dataset. The proposed approach minimizes the error in component modules with new columns attach with storage structure to check the reviews of component users and performance in real time (Fig. 10). All these components are used to define raking of a software component and provide the accurate and efficient results as good quality component. The performance accuracy is also suitable for almost all the cases in this field and maximum reach is near about 99.3% of test cases.

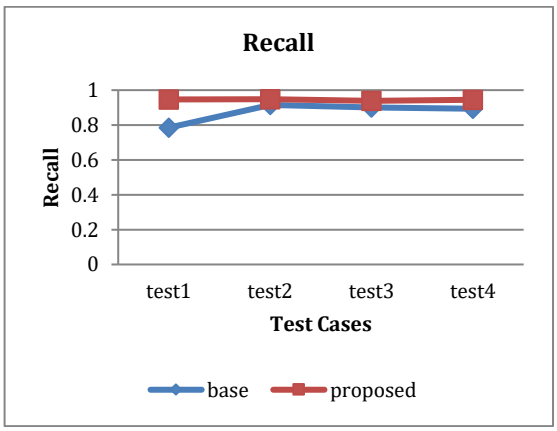

Fig. 10: Recall

In future scope, the processes of the software component can also be increased with the help of software classification techniques. Classification techniques are working on the basis of data features and knowledge bases. So the proposed approach if attached with classification technique than it might enhance the accuracy of detecting software components from the large dataset.

References

Al Saiyd NA, Al Said IA, and Al Takrori AH (2010). Semantic-based retrieving model of reuse software component. IJCSNS International Journal of Computer Science and Network Security, 10(7): 154-161.

Basha NMJ and Moiz SA (2012). Component based software development: A state of art. In the International Conference on Advances in Engineering, Science and Management, IEEE, Nagapattinam, Tamil Nadu, India: 599-604.

Cai X, Lyu MR, Wong KF, and Ko R (2000). Component-based software engineering: technologies, development frameworks, and quality assurance schemes. In 7th Asia-Pacific Software Engineering Conference, IEEE, Singapore, Singapore: 372-379. https://doi.org/10.1109/APSEC.2000.896722

Crnkovic I, Larsson S, and Stafford J (2002) Component-based software engineering: building systems from components at 9th IEEE conference and workshops on engineering of computer-based systems. ACM SIGSOFT Software Engineering Notes, 27(3): 47-50.

Dutta S and Sengupta S (2015). Retrieval of software component version from a software version database: A graph based approach. In the International Conference on Advances in Computer Engineering and Applications, IEEE, Ghaziabad, India: 255-259. https://doi.org/10.1109/ICACEA.2015.7164706

Fanchao M, Dechen Z, and Xiaofei X (2006). A specification-based approach for retrieval of reusable business component for software reuse. International Journal of Computer Science, 1(4): 283-290.

Grundy J (1999). Aspect-oriented requirements engineering for component-based software systems. In IEEE International Conference on Requirements Engineering, IEEE, Limerick, Ireland: 84-91. https://doi.org/10.1109/ISRE.1999.777988

Gupta S and Kumar A (2013). Reusable software component retrieval system. International Journal of Application or Innovation in Engineering and Management, 2(1): 187-94.

Inoue K, Yokomori R, Yamamoto T, Matsushita M, and Kusumoto S (2005). Ranking significance of software components based on use relations. IEEE Transactions on Software Engineering, 31(3): 213-225.

Jia Y and Gu Y (2002). The representation of component semantics: A feature-oriented approach. In: Crnković I, Larsson S, and Stafford J (Eds.) Proc. of the Workshop on Component-based Software Engineering: Composing Systems From Components (a part of 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems), Lund, Sweden.

Mili R, Mili A, and Mittermeir RT (1997). Storing and retrieving software components: A refinement based system. IEEE Transactions on Software Engineering, 23(7): 445-460.

Penix J, Baraona P, and Alexander P (1995). Classification and retrieval of reusable components using semantic features. In the 10th Knowledge-Based Software Engineering Conference, IEEE, Boston, USA: 131-138. https://doi.org/10.1109/KBSE.1995.490128

Penix JJ (1998). Automated component retrieval and adaptation using formal specifications. University of Cincinnati, Cincinnati, USA.

Radwan AA, Latef BAA, Ali AMA, and Sadek OA (2008). Using genetic algorithm to improve information retrieval systems. World Academy of Science, Engineering and Technology, 17: 1021-1027.

Shirali-Shahreza S and Shirali-Shahreza M (2010) Using formal methods in component based software development. In: Sobh T (Ed.), Innovations and Advances in Computer Sciences and Engineering: 429-432. Springer Science and Business Media, Berlin, Germany.

Şora I and Todinca D (2006). Specification-based retrieval of software components through fuzzy inference. Acta Polytechnica Hungarica, 3(3): 123-135.

Wang SF, He ZJ, and Wang KH (2000). Studies on software reuse technology. Computer Engineering and Design, 21: 10-15. Available online at: http://en.cnki.com.cn/Article_en/CJFDTOTAL-SJSJ200005002.htm