

Retrieval of Most Relevant Reusable Component Using Genetic Algorithms

Rajesh Bhatia
Thapar Institute of Engineering &
Technology
Patiala, INDIA

Mayank Dave
National Institute of Technology
Kurukshetra, INDIA

R. C. Joshi
I. I. T. Roorkee
Roorkee, INDIA

Abstract

Software reuse has become very popular in the software development because of its immense advantages, which results in improving software product quality, decreased product cost and schedule. Reusable components stored in a repository are useful in developing early prototypes with better quality. One of the most fundamental problems in software reusability is locating and retrieving software components from a large repository. To reuse a software component, you first have to find it. Retrieval of component should be less time consuming and efficient. This paper addresses the issue of most appropriate component retrieval from a component repository. Appropriateness here is precision and quality. Two-step process is used for retrieval of appropriate component from repository. The initial step is Keyword based retrieval for finding all candidate components according to the user requirement. Keyword based search is used to narrow down the search space, which further improves initial population for Genetic Algorithm based second step. As Genetic Algorithms are applied to only extracted candidate components, initial population will be precise as compared to random, which will help in early convergence. Thirty-two attributes of reusable component are considered for selecting the appropriate component. A Fitness function is calculated on the basis of weight vectors and attribute vectors associated with each component to select the appropriate component.

Keywords: Genetic algorithms, Software Reuse, Software Component Software Repository

1. Introduction

Developing software from scratch is a tedious, time consuming and a costly process. Vast research is going on in the field of software reuse. Successful reuse requires a wide variety of high-quality generalized components, proper classification and retrieval

mechanisms. It also requires sufficient and proper documentation of components, a flexible means for combining components, and a means of adapting components to specific needs. Challenges in the context of repositories are; techniques to locate and retrieve components efficiently, e.g., classification schemes, retrieval techniques, and to integrate them in software systems. Software reuse is not so successful, because the effort needed to create reusable software components, locate them, adapt them and integrate them in a specific application has been usually greater than the effort needed to create the application from scratch. However, there are many factors that limit the practice of software reuse, among them is the lack of tools to construct reusable software components and the lack of tools for fast and effective retrieval of components. But still, software reuse is the best answer to software crisis [1, 11, 12].

The term component is used in this work in a generic sense. Reusable software components not only include generic code, basic procedures, module, but also systems, subsystems, software specification, requirements and design specifications.

Various techniques have been used by researchers to improve the retrieval performance from a repository [2, 3]. One such technique is described in the proposed system. The proposed system uses a new hybrid approach. Where components are retrieved in two steps. The first step is retrieval based on user-defined keywords; the next step is optimizing the retrieved components using genetic algorithms. This approach enhances the chance of retrieving appropriate component from the repository that can be reused.

In this paper the next section is system overview, which gives the complete working of the system. The section 3 component based repository gives knowledge about component description. Section 4, is about the first step of the proposed system, called as keyword based retrieval. Section 5, includes optimization of components using genetic algorithm which results in finding the optimal solutions, Section 6, describes the experimentation and results of the system.

2. System Overview

The proposed system is a component retrieval system. Information is stored in a reusable library called as repository. In this repository thousands of components are stored. These can be used for reuse during development of new software. But the proper retrieval of software component is a cumbersome task. Selection of appropriate component according to their attributes is even hard. Attributes are defined as the properties belong to each component. Each attribute has certain weights. These weight vectors show the importance of that attribute in the component. In the proposed system a component repository is developed, which has thousands of components. From this repository we have to select the appropriate component. Retrieval of a appropriate component from the repository can not be done efficiently, because the repository size is often very large and complex [17, 18, 19]. This affects the performance of the system. The best solution to this is to use a hybrid approach. In the initial step all the possible solutions are retrieved according to the user query. But we can not get the best solution from these. So, we require

another approach. In proposed system we apply genetic algorithms for getting the appropriate component among the possible solutions. Genetic algorithm operations like reproduction, crossover, and mutation are used. Ranking of the appropriate components is also done for user knowledge.

3. Component Based Repository

In the proposed system, various components are stored in a component repository. Our approach for component description is based on how software component is reused through the reuse model in order to establish a classification framework over the applicable component domain [4]. Component retrieval becomes important as Component Based software development requires a large repository to supply components [15]. Repositories should provide highly sophisticated methods for storing and retrieving components. This is important because the size of a repository is very large. Properly managed component specification results in easy retrieval. Component description used in the proposed system is shown in figure 1.

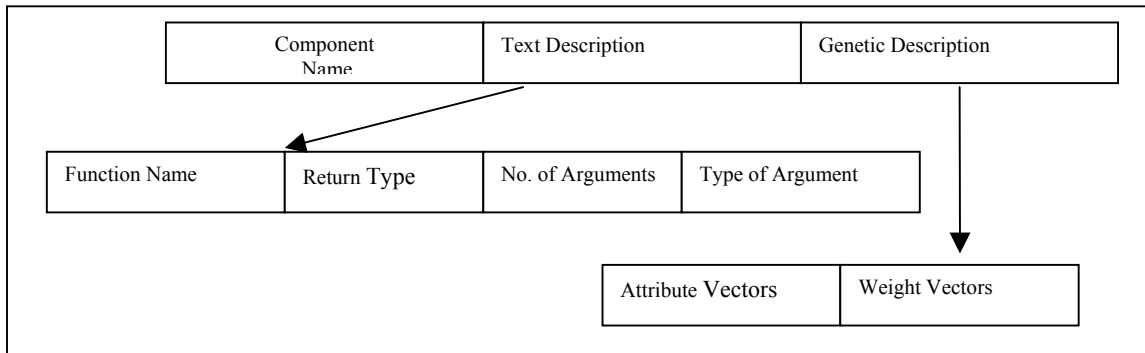


Figure 1: Component Description

System has been built with the following functionality:

1. Descriptions of components are constructed using the classification framework.
2. Users are able to search for components, based on the descriptions stored in the system.

The component description is divided into three parts:

Component name: Unique name given to a component.

Text description: It focuses on the

-Function performed by the component and synonym corresponding to them.

-Return type used by the function,

-Number of argument used by the function,

-Type of arguments.

Genetic description: Attributes of the component are part of genetic description; the weights corresponding to the attributes are also stored in the description.

Figure 2, shows the genetic description. Weight of

each attribute is used to indicate the importance of that attribute in the component [6, 10].

A1	A2	A3	A4	A5	A6	A7	A8
1	1	0	1	0	1	1	0

Attribute Vector

A1	A2	A3	A4	A5	A6	A7	A8
0.5	0.2	0.4	0.25	0.3	0.6	0.1	0.7

Weight Vector

Figure 2: Genetic description

4. Keyword-Based Retrieval

Localizing components in small software libraries can be simple. User can learn quickly where the available components are and can select them by name or browsing the library. Finding reusable components in large libraries is not as simple. Browsing the library can be very laborious and so, mechanism which allows

faster searching is needed to retrieve a component through the specification of its main features.

The system uses keyword based retrieval technique for initial component retrieval. In Keyword based retrieval, user selects the keywords according to their requirement. These selected keywords are matched in this phase with the repository. All the possible components are retrieved and are shown. This step helps in reducing the domain search. In certain cases user does not know the exact keyword. This problem is solved by using synonyms. Synonyms help the user in proper selection of keywords.

5. GA Based Optimization

Genetic algorithms are non-deterministic search algorithms based on the mechanics of natural selection and natural genetics in a biological system. Genetic algorithms have been used in modeling evolving systems or combinatorial optimization problems. Genetic algorithms are robust in many application areas and search a huge problem space while exploiting historical information to speculate on new search points with expected improvement of performance [7, 8].

The genetic algorithms attempts to find a very good or appropriate solution to the problem by genetically breeding the population of individuals. The genetic algorithm transforms a population of individual objects, each with an associated fitness value, into a new generation of the population using the Darwinian principle of reproduction and survival of the fittest and naturally occurring genetic operations such as crossover and mutation. Each individual in the population represents a possible solution to a given problem [8, 20].

Before run genetic algorithms, we define a relevant encoding of chromosome to solve a problem, design an objective function for fitness, and construct genetic operators. In order to run Genetic Algorithms, we generate an initial population consisting of chromosomes and evaluate these chromosomes using the objective function designed. And we select two chromosome randomly and crossover and mutate them and replace a low quality chromosome with a new one of high quality. As these processes have been repeated, the population consists of high quality chromosomes.

5.1 Fitness Function

Fitness Function of a chromosome determines how fit a chromosome is for each problem and which chromosomes survive in the next generation. Genetic algorithms favor chromosomes, which are of high quality.

Fitness Function in genetic algorithms is the function that is optimized using the genetic process. Choosing an appropriate fitness function is very important.

For each individual in the population, Fitness Value is evaluated as:

$$\sum_{j=1}^8 A_j W_j$$

‘A’ denotes 8 bit attribute vector, Attribute vectors are in binary form, i.e. 0 or 1 only.

‘W’ denote weight vector corresponding to the attribute vector. Its value ranges from 0 to 1, it has real values.

5.2 Genetic Modification

In this step, genetic operators are applied to the individuals in the previous generation to generate the next generation of individuals. It involves three stages.

1. *Selection and reproduction*: All individuals in the previous generation were made available for reproduction in the next generation. The roulette wheel reproduction process was used to select individuals for reproduction.

2. *Crossover*: A Single-site crossover was followed. In this cross-site is selected randomly along the length of the mated strings and bits next to the cross-sites are exchanged. Since the knowledge of the appropriate site is not known and it is selected randomly [9].

3. *Mutation*: After cross over, the strings are subjected to mutation. Mutation of a bit involves flipping it changing 0 to 1 and vice versa with small mutation probability P_m . Mutation Probability is the probability of mutation, which is used to calculate number of bits to be muted. The mutation operator preserves the diversity among the population, which is also very important for the search.

$$\text{Number of bits to be changed} = \text{Mutate Probability} * \text{pop size} * \text{string length}$$

5.3 Termination Criteria

In genetic algorithms, the termination criteria for stopping the process is required. Genetic Process has large number of iteration. Termination Criteria is important and vital step. This gives information about when to stop the process.

Two Termination Criteria’s are used according to the requirement.

1. *Fitness Convergence*: A Termination Method that stops the evolution when the fitness is deemed as converged. Fitness is deemed as converged when the

difference between average fitness across the current population and previous population is less than the value specified. We have used fitness convergence value of 0.01 for termination criteria. Average fitness is calculated as:

$$\frac{\sum_{i=1}^{popsize} \sum_{j=1}^8 A_j W_j}{popsize}$$

Where A_j denote 8 attribute vectors, W_j is weight vectors corresponding to attribute vectors, $popsize$ is the population size.

$$| \text{AvgFitness}_{\text{current population}} - \text{AvgFitness}_{\text{previous population}} | \leq 0.01$$

2. Generation Number: In certain cases, above termination criteria is not achieved after large number of iteration, in that case another termination criterion is used i.e. Generation Number. Generation Number is a termination method that stops the evolution when the specified maximum numbers of evolution have been run.

6. Experimental Results

In this section experimental results of the proposed system are shown. The repository used for the system contains five thousand components. Each component has description used for keyword retrieval, and genetic description for genetic algorithm operations.

According to user query and selected keywords, the system selects 35 components as possible solution. But we have to randomly select initial population from these possible solutions for the genetic algorithms operation. Here we are selecting only 6 components for showing the results. In the first step of genetic algorithms, the fitness values of the components present in the initial population are calculated, average fitness value of the initial population is also calculated. Average fitness value is required for termination of genetic process. Fitness value is calculated using Attribute vectors (AV) and Weight vectors (WV) according to the given formula. Fitness value and average fitness value for initial population is:

Table 1: Fitness value of initial population

Component Name	C1	C2	C3	C4	C5	C6
Fitness Value	1.8	0.9	2.7	1.5	2.4	1.1

Average Fitness value: 1.73

Once the fitness value is calculated, the genetic operations start on the initial population until the said criteria is not met. The genetic algorithm operations

used are reproduction, crossover and mutation. The genetic operations are repeated for number of generations. Crossover Point is selected randomly for partially exchanging the string. Mutation rate used is 0.02. In the said case after 28th generations the process is terminated, because the difference between the average fitness value after 27th and 28th generation ≤ 0.01 . The results are shown in table 2.

Table 2: Fitness Value after 1st, 27th and 28th generation

Component Name	Fitness Value after 1 st Generation	Fitness Value after 27 th Generation	Fitness Value after 28 th Generation
C1	1.8	2.5	2.6
C2	1.0	1.5	1.5
C3	2.7	2.6	2.8
C4	2.7	2.7	2.7
C5	2.6	3.4	3.1
C6	2.7	2.8	2.8

Average Fitness after 1st Generation: 2.25

Average Fitness after 29th Generation: 2.58

Average Fitness after 30th Generation: 2.58

The genetic process is terminated, but still the task of ranking the components according to their fitness value is left. After ranking, the appropriate components we got are shown in table 3.

Table 3: Appropriate Components

Appropriate Component	C5	C3	C6
Max Fitness	3.1	2.8	2.8

7. Results and Tentative Future Investigations

In this paper we described a method for utilizing genetic algorithms in retrieval of reusable components, specifically how the genetic algorithms can be used to optimize the search in order to find appropriate component. The major contribution of this work is integration of keyword and genetic algorithm based retrieval. Software components are denoted by features, attributes and weights corresponding to them. Current work uses a small repository containing only code fragments, but same technique with more number of component attributes specific to work product like design documents, UML diagrams, test cases etc can be applied.

8. References

- [1] Jun-Jang Jeng Betty H. C. Cheng, Specification Matching for Software Reuse: A Foundation, *ACM*, 1995, pp 97-105.
- [2] D.Vijay Rao, V.V.S.Sarma, A rough-fuzzy approach for retrieval of candidate components for software reuse, *Pattern recognition letters* 24, 2003, pp 875-886.
- [3] Praveen Pathak, Michael Gordon, Weiguo Fan, Effective Information Retrieval Using genetic Algorithms based matching functions adaptation, *Proceedings of the 33rd Hawaii international conference on system sciences*, 2000, pp 1-8.
- [4] Sathit nakkrasae, Peraphon Sophatsathi, William R.Edwards, Fuzzy subtractive clustering based indexing approach for software components classification.
- [5] Rune Meling, E James Montgomery, Pon Sudha ponnusamy, Eva Beverly Wong, Daniela Mehandjiska, Storing and retrieving Software Components: A Component Description Manager.
- [6] Donald H.Kraft, Frederick E. Petry, Bill P. Buckles, Thyagarajan Sadasivan, The use of genetic programming to build queries for information retrieval, *IEEE*, 1994, pp 468-473.
- [7] Goldberg, D. E, Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA: Addison-Wesley Publishing Co, 1989.
- [8] Byung-Jeong Lee, Byung-Ro Moon, Chi-Su Wu, Optimization of multi-way clustering and retrieval using genetic algorithms in reusable class library.
- [9] Dana Vrajitoru, Crossover improvement for the genetic algorithm in information retrieval, *Information Processing & management*, 1998, vol. 34, No. 4, pp 405-415.
- [10] Jorng-Tzong Horng, Ching-Chang Yeh, Applying genetic algorithms to query optimization in document retrieval, *Information Processing and Management* 36, 2000, pp 737-759.
- [11] Gordon, M, Probabilistic and Genetic Algorithms for Document Retrieval, *Communications of ACM* (31:2), 1988, pp 152-169.
- [12] Giancarlo Succi, Carl Uhrlik, Tullio Vernazza, A formal view to classification and retrieval mechanism for reusable objects”, pp 27-32.
- [13] Weiguo Fan, Michael D. Gordon, Praveen Pathak, “A Generic Ranking Function Discovery Framework by Genetic Programming for Information Retrieval”, *Information Processing And Management*, 2004, pp 587–602.
- [14] Weiguo Fan, Michael D. Gordon, Praveen Pathak, Automatic Generation of a Matching Function by Genetic Programming for Effective Information Retrieval, 1998.
- [15] Ju An Wang, Towards Component-Based Software Engineering, November 2000.
- [16] Rubén Prieto-Díaz, Software Reuse: Issues and Experiences, *American Programmer vol.6*, No.8, pp 10-18, April 1993.
- [17] C.J.Van Rijsbergen, Information Retrieval, 2nd Edition, *London: Butterworths*, 1979.
- [18] Amit Singhal, Modern Information Retrieval: A Brief Overview, *IEEE, Computer Society Technical Committee On Data Engineering*, 2001.
- [19] Weiguo fan, Michael d. Gordon, Praveen Pathak, Personalization of search engine Services for effective retrieval and knowledge management.
- [20] Holger Billhardt, Daniel Borrajo, Victor Maojo, Using Genetic Algorithms to Find Suboptimal Retrieval Expert Combinations, *SAC 2002, Madrid, Spain*, pp 657-662, 2002.
- [21] A. Mili, R. Mili and R.T. Mittermier, A survey of software reuse libraries, *Annals of Software Engineering* 5(1998) 349-414
- [22] Jiang Guo, Luqi. "A Survey of Software Reuse Repositories", 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2000, p. 92.