Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 2875 – 2879

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineering and Information Science

A Software Decoupling Partition Method Based on Interactive Genetic Algorithm

Zhe Ma[a]*, Kerong Ben[a]

[a]*Department of Computer Engineering, Naval University of Engineering, Wuhan, 430033, China*

Abstract

With the development of software component technology, extraction of components from legacy systems has become a reasearch focus of both software reuse and program comprehension for its lower cost and high efficiency. In this paper, we analyze the independence metrics of component for object-oriented software system from the perspective of coupling, cohesion and scale. Then, we propose a software partition decoupling method based on interactive genetic algorithm. By the user adjusting individuals fitness value, reduce the crossover and mutation probability, our method in obtaining reasonable component partition result while avoiding algorithm premature convergence.

Keywords: software partition; compontent; coupling; cohesion; interactive genntic algorithm

1. Introduction

With the continuous development of computer technology, program code becomes difficult to understand as software scale expanding. Although to some extent, object-oriented software development methodology improves the software encapsulation and increases efficiency of software development. However, from the perspective of legacy system reuse, the object scale is too small, and component technology provides a better way of software reuse with greater granularity. By partition the legacy

* Corresponding author, Zhe Ma Tel.: +86-27-83444626.
E-mail address: mazhe1214@gmail.com.

doi:10.1016/j.proeng.2011.08.541

system into a series of reusable components, component composition can be use to apply funtion for system integration, while reducing maintenance cost and extend the system life cycle.

Component extraction means extract a variety of abstract forms with reverse engineering from the existing software resources, of which semantic in the form of concepts set, pattern description, variable reference relationship, data transformation, etc; and structure in the form of function call diagram, module dependency diagram, class diagram, entity relationship diagrams, etc[1]. Based on these abstract forms, legacy system can be partitioned by component extraction and submodule be selected with high quality as candidate component.

According to the different partition strategies, the existing methods can be devided into knowledge matching-based method and structure analysis-based method[2]. Knowledge matching-based method analyzes the semantic elements in software system, and associates the entites with same or similar semantic as component. For the case of reusable component library and domain semantic model existed, the knowledge matching-based method can support system partition well, but there're difficulties in formal description and knowledge acquisition. Structure analysis-based method implements system partition through classify or cluster some sort of abstract structure of software system, which usually be class diagram, function call diagram and module dependency diagram. Previous research shows that the optimal graph partition or clustering is a NP hard problem, so there isn't a deterministic optimal solution.

Combined with the characteristics of two kind of structure analysis methods, we propose a software partition method based on interactive genetic algorithm (IGA). The method is referenced with clustering idea, according to the independence metric of component, partition object-oriented software into collection of several classes as a component. The use of interactive genetic algorithm, can not only partitioning automatically, but also improving the accuracy of heuristic algorithm through adjust partition plan by users.

2. Independence metrics of object-oriented software component

Software component is a reusable unit with standard interface that can be independently deployed[3]. The independence of component is decided by coupling, cohesion and component scale. The smaller component with high cohesion and lower coupling will be more easy to understand, reuse and maintenance. The following we will discuss component independence metrics from the perspective of coupling and cohesion. In object-oriented software system, component is composed with several classes, component scale can be considered as the number of classes it contained.

2.1. Coupling metric

Call and data coupling between classes contained in component determine coupling relationship of component. Calls between component are implement by access the internal methods or properities of the other components.

Definition 1 (Call relationship between classes) For two classes of system $c_1 \in C, c_2 \in C$, the call relationship between classes $use(c_1, c_2)$ is definited as $\exists m_1 \in M_I(c_1), m_2 \in M_I(c_2)$, satisfy $m_2 \in CM(m_1)$,or $\exists a \in A(m_2)$, satisfy $a \in AR(m_1)$. $AR(m)$ is the properities set of method m accessed.

Definition 2 (Call coupling) For class $c \in C$, the class set of which call coupled with c is definited as $CCD(c) = \{c' \in C - \{c\} \mid use(c, c') \vee use(c', c)\}$.

Definition 3 (Data coupling) For class $c \in C$, the class set of which data coupled with c is definited as $DCD(c) = \{c' \mid a \in A(c) \wedge a \in Par(m) \wedge m \in M(c')\}$, $Par(m)$ is parameter set of method m.

Definition 4 (Class coupling degree) For two classes of system $c_i \in C, c_j \in C$, the class coupling degree is definited as $Cou_{Class}(c_i, c_j) = |\{c' \cup c'' \mid c' \in CCD(c_i) \cap CCD(c_j) \wedge c'' \in DCD(c_i) \cap DCD(c_j)\}|$.

Definition 5 (Component coupling degree) For two components $comp_1 = \{c_1, c_2, \ldots, c_m\}$, $comp_2 = \{c_l, c_{l+1}, \ldots, c_k\}$, the component coupling degree definited as below:

$$Cou_{Comp}(comp_1, comp_2) = \frac{\sum_{i=1}^{m}\sum_{j=l}^{k} Cou_{Class}(c_i, c_j)}{m + k - l}, c_i \in comp_1, c_j \in comp_2$$

2.2. Cohesion metric

Class attribute access function is use to interact with outside, so the interaction will influence component cohesion. In addition, nesting and inheritance between classes are also cohesion factors.

Definition 6 (Class cohesion degree) For class $c \in C$, the class cohesion degree is definited as:

$$Coh_{Class}(c) = (AR(c) \cup DCD(c)) - (AR(c) \cap DCD(c))$$

Definition 7 (Component cohesion degree) For component $comp = \{c_1, c_2, \ldots, c_m\}$, the component cohesion degree is definited as:

$$Coh(comp) = \frac{\sum_{i=1}^{|C|}\sum_{j=1}^{|C|} Cou_{Class}(c_i, c_j) \cdot Coh_{Class}(c_i) \cdot Coh_{Class}(c_j)}{|comp|^2 - |comp|}, c_i, c_j \in comp$$

When there is no interaction between classes in a component, the cohesion degree is 0; when all classes interact with each other, the cohesion degree is 1.

3. Software decoupling strategy

Although reuse software from the greater granularity is the best way to solve the legacy system crisis, the component scale, coupling and cohesion directly affect the quality of software. The smaller scale, lower coupling and higher cohesion make the software easy to understand, reuse and maintenance. Therefore, we calculate the software partition quality from the aspects of component scale, coupling and cohesion.

Definition 8 (Software partition quality) For a partition scheme $P = \{comp_1, comp_2, \ldots, comp_n\}$, the software partition quality is definited as:

$$PQ(S, P) = \sum_{i=1}^{n} Coh(comp_i) \cdot \frac{\sum_{i=1}^{n} SC(comp_i)}{n} - \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Cou(comp_i, comp_j)}{n(n-1)/2}$$

Which of $Coh(comp_i)$ is cohesion degree of component $comp_i$, $Cou_{Comp}(comp_i, comp_j)$ is coupling degree between component $comp_i$ and $comp_j$. It can be seen that the larger n is, the component scale is smaller, the coupling degree is lower, the cohesion degree is higher, and the software partition quality is higher. In this paper, the software partition quality is used to evaluate the individuals fitness as fitness function in IGA.

4. Software partitioning based on IGA

Software partitioning is a particular global optimization problem, due to lack of precise synchronous design document of legacy system to compare, partitioning can only according to coupling and cohesion metrics. But lack of mechanism make it is hard to judge whether some core code can be partitioned, therefore it's diffiuclt to ensure the guarantee of correct and rationality with a clearly definited fitness function. The idea of IGA provides a new way to solve the problem of implicit performance optimization with evaluation of human as the fitness value of individuals in the evolutionary process[4].

4.1. Gene encoding

According to the definition of software partitioning, the system can be sorted in number of all classes. If system has m classes, the population of individuals is a 5m bits binary code strings, each of 5 bits describe that the classification information of a certain class, all of the classification information of m classes constitutes a partition of the software. For example, consider a program contains 5 classes, which can be partitioned for 2 parts, the corresponding gene encoding of binary string is shown in figure 1. By take into account the impact of partitioning granularity to architecture, assume that the number of components in the system is no more than 32, which can be expressed with the 5 bits code. If the number of components is too much, the software can be first partitioned into several subsystems, and then gradually break down.

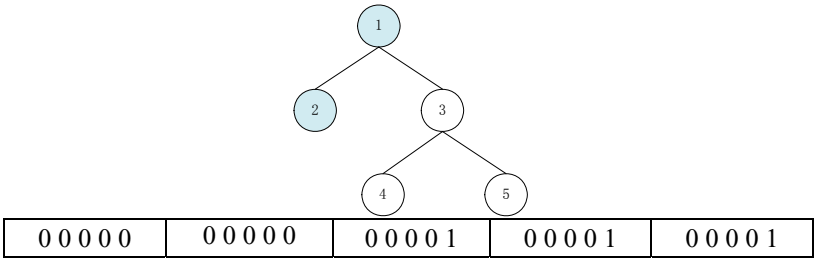

Fig. 1. Individual code of partition scheme

4.2. Software partition algorithm based on IGA

Input: Class set of software

Output: Best partition scheme $P_{best}(comp_1, comp_2, \mathrm{L}\ , comp_m)$

Step 1 Generate initial population;

Step 2 Calculate the individuals fitness value of populations according to adjustment by users and partition quality

Step 3 User adjust the fitness value;

Step 4 Set the probablity of crossover and mutation as $p_m = \sqrt{\frac{k}{E}}$. Produce the next generation of population by random single point crossover and mutation;

Step 5 If the premature convergence $Pre > \varepsilon$, turn to Step 6, else selected individuals randomly to join the next generation From the parent population untill $Pre > \varepsilon$;

Step 6 If the gerenation number $n = N$ or $E < \delta$, algorithm terminated and output the best individual with the highest fitness value, else turn to Step 2.

In the algorithm above, using the population fitness mean square algorithm to determine whether the

premature convergence, $E = \sum_{i=1}^{N} (f(x_i) - \overline{f})^2 / N$ is population fitness mean square, and premature convergence can be judged by $Pre = \Delta E + n / N + \Delta f_c$. In order to avoid premature convergence, we use the strong punishment principle, higher the probability of crossover and mutation to individuals which has low fitness value, and lower the probability of crossover and mutation to individuals which has high fitness value. Thereby, reducing the influence of better individuals in population.

5. Case study

In order to illustrate the process of the software partition, we take the open source list management software Infopad which written by Java for example. Infopad including 1069 lines of code, 29 classes and 73 methods. Each encoding of partition scheme is composed with component classification information in order. The size of solution search space is $29 \times 2^5 = 928$. For validating algorithm performance, we compare our prevent premature convergence-based interactive genetic algorithm (PPCIGA) with classic genetic algorithm (GA) and interactive genetic algorithm (IGA). The experimental results is shown in Figure 2.

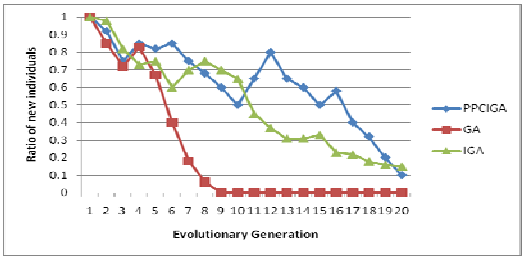

Fig. 2. Comparison of algorithms convergence rate

6. Conclusion

This paper presents a software decoupling partition method based on interactive genetic algorithm. Judge the alforithm whether be premature convergence based on fitness mean square, we also consider the influence to convergence speed by individual fitness and evolutionary generation, adjust probablity of crossover and mutation dynamicly, to obtain a better balance effect between population diversity and rapid convergence. Experiments show that the prevention of premature convergence of genetic algorithms interactive convergence rate is lower, but get more reasonable results than the classical genetic algorithm and interactive genetic algorithm.

References

[1] WANG Jing, Chang I C. Applications of Independent Component Analysis in Endmember Extraction and Abundance Quantification for Hyperspectral Imagery[J]. IEEE Transactions on Geoscience and Remote Sensing, 2006, 44(9): 2601-2616.

[2] Hummel O, Janjic W, Atkinson C. Code Conjurer: Pulling Reusable Software out of Thin Air[J]. IEEE Software, 2008, 25(5): 45-52.

[3] McQuillan J, Power J. Towards the Reusability of Software Metric Definitions at the Meta Level[C]. Proceedings of the 2006 European Conference on Object-Oriented Program. Nantes: IEEE Computer Society Press, 2006: 375-384.

[4] Sheikh R, Raghuwanshi M. Genetic Algorithm Based Clustering: A Survey[C]. Proceedings of the 2008 International Conference on Emerging Trends in Engineering and Technology. Nagpur: IEEE Computer Society Press, 2008: 314-319.