C++ Class Restructuring Using the Neural Networks

Kwang-Baek Kim, Bong-Gi Jun and **Young-Ju Kim,** *Member, KIMICS*

Abstract—Classes are apt to include useless codes and inadequate inheritance relationship between them when they are being updated, inserted and deleted during the evolution process of object-oriented software, leading to lots of errors. Conventional class restructuring methods degrade the effectiveness of reusability since they go with preprocesses such as dependency analysis and estimation of class cohesion and run statically. In this paper, we propose a new C++ class-restructuring algorithm that does not require those preprocesses and runs dynamically by improving ART learning algorithm in the artificial neural networks.

Index Terms—C++ Class, ART, Software Restructuring, Object Oriented Program.

I. INTRODUCTION

One of the important concepts of object-oriented method is class, which is a model of substance in subject possession. And this has advantage to correct and not to take effect on outside if interface is maintained through hidden information. Also when inheritance correct and expand similar, existing class, it is easy to invent new class. However, improper relationship of class modeling and inheritance can be obstruction of system development[1]. If you are going to maintain or develop new system by using it or change and expand through inheritance after initial design, this also can be disadvantage of reusability and maintenance because it is based on improper original design[2]. Mean-while, in the aim of reducing cost of software maintenance by reusing software parts in other system or expanding life of software, it is called "software restructuring" to make it understandable and changeable, and update software to avoid error of inheritance change[3]. There have been a lot of studies for class restructuring which increase reusability of class and reduce maintenance cost by applying such software restructuring to object-oriented language. The methods of class restructuring are as follows: First method of class restructuring is to distinguish classes and members of classes through input of source code from object-oriented program then analytical dependency between members of classes. Second method is to estimate class cohesion by structure tree, understand potential classes against low-class cohesion base the test result, and define a new class. Third method is to divide non-simple method into simple[3,4,5].

There have been a lot of studies for class restructuring which increase reusability of class and reduce maintenance cost by applying such software restructuring to object-oriented language. The methods of class restructuring are as follows: First method of class restructuring is to distinguish classes and members of classes through input of source code from object-oriented program then analytical dependency between members of classes. Second method is to estimate class cohesion by structure tree, understand potential classes against low-class cohesion base the test result, and define a new class. Third method is to divide non-simple method into simple [5,6,7]. But conventional methods have disadvantage, that is it needs all process of describing class-member dependency graph or class structure tree through source code input and testing class cohesion by using this structure tree, and runs restructuring statically. In this paper, we propose a new C++ class-restructuring algorithm that does not require preprocesses such as dependency analysis and estimation of class cohesion has additional learning ability for dynamic running by using ART learning algorithm in the artificial neural networks.

II. RELATED STUDIES

A notion of software re-engineering that is changing as object-oriented program has appeared for increasing extensibility, reusability, maintainability of systems developed conventional paradigm being general and having interest of lots of people. Re-engineering is to combine analysis and plan sampling quality of an inverse engineering and restructuring ability about data, structure and logic. Restructuring is a process changing other expressional form of equal conjecture step as remain the outside act in functional or meaning system. Restructuring is a part done lots of studies among re-engineering operation. Restructuring runs for purpose of progress of interpretation, simplifying of logic, decreasing of test / debugging time, following of programming standard, simplifying of programming change, decreasing of danger and software maintainability cost, improving adaptability on request of maintenance progress of program quality and user-satisfaction, improving of spirit stimulation of maintainer and a feeling of satisfaction, decreasing interdependence between them, preparing for transplantation of software and maintaining property value of software[4,9].

Manuscript received August 14, 2003.

K. B. Kim is with the Computer Engineering Department, University of Silla, Busan, Korea, (corresponding author to provide phone: +82 -051-309-5052; E-mail: gbkim@silla.ac.kr)

B. G. Jun is with the Computer and Information Engineering Division, University of Silla, Busan, Korea, (E-mail: bgjun@silla.ac.kr)

Y. J. Kim is with the Computer Engineering Department, University of Silla, Busan, Korea, (E-mail: yjkim@silla.ac.kr)

In general, programs needed restructuring are inadequate code, uneasy code for interpretation, change and test, over reaching code for an error rate, correction time and cost, requesting code particular number, and system that is important strategically, expensive cost and frequent change.

Restructuring methods are as follows:

- Removal GoTo department, dead code, recursion, infinite loop and so on.
- Formation program modularize, hierarchically ordered, well structured.
- Base graph theoretical approaching method.
- Formation flow graph of non-structure from reading original code.
- Application of structuring change step that "if-sentence" repeated most deeply is taken start-point.

These methods have been restructuring the purpose of changing structure code from conventional non-structure code. However, since recently object-oriented paradigm is generalized, inadequate class modeling and inheritance relation in object-oriented between them has taken a growing interest of object-oriented program and has been effected a lot of studies because of being an obstacle to develop system[1]. When class including inadequate class modeling and inheritance is made use of that way at beginning structure or system is maintained by changing and extending through inheritance or new system is developed [2]. Therefore restructuring is requested to find inadequate parts existing on conventional structure and to have it change desirable form [1,4,5]. Especially they are leading to lots of errors in time of reusability because of including useless codes and maintaining inadequate inheritance relationship when they are being inserted, deleted and updated during the maintainability process of object-oriented program. Therefore restructuring classes are requested and also have been lots of studies.

However, conventional C++ class restructuring method is to store in class table by discrimination classes from input source and also to be formed graph expressing inheritance relationship between classes and member, usability relationship between members happening in functional call and data reference time and to make use of this graph then if it exists inheritance, member and usability relationship is restructuring class by fractionating it a high class and a low class that class gets through dependency analysis discrimination dependency relationship of that class. Another method is to define new classes on potential class as dividing a class for improving class cohesion on a low class cohesion after displaying as structure tree taken class input from source code and testing class cohesion by basing this structure tree. However conventional methods for efficient restructure have a part being improved as follows:

1. Running of statically.
2. Restructuring of a class.
3. Preprocesses of dependency analysis and estimation of class cohesion.
4. Selecting question of restructuring class.

Therefore, it is asked to develop of a new class restructuring algorithm that doesn't require preprocess for restructuring, runs dynamically in real time when add and/or delete operation is happen and is able to reinforce by means of being restructuring not only a class but also whole classes.

III. CLASS RESTRUCTURING ALGORITHM USING NERUAL NETWORKS

A. The Similarity for C++ Class Restructuring

The similarity of conventional ART is got with percentage of norm, times of storing vector and input vector on input vector and the formula is as follow:

$$\frac{\|T \bullet X\|}{\|X\|} \tag{1}$$

where, T is the internal vector and X is the input vector.

The similarity is admitted on condition that this similarity is higher value than vigilance parameter[10]. However an existed method has an effect on similarity comparison when the value is only "1" pattern but otherwise the value is "0", the method has no an effect on it since being equal to "1" approximate ratio in case of the binary scale. For that reason, it is happened a problem not to be divided definitely and to be recognized as a same vector against vectors that are divided and recognized clearly in real image. Therefore it is difficult for C++ class restructuring to apply it [11].

In this paper, similarity is divided of using how many nodes is turned as a same value not to be divided on "1" approximate ratio of (1). In other words, it is practiced that percentage of norm between storing vector and input vector to input vector on the occasion of being considered of "logical operation" form. The proposed equation is as follow:

$$\frac{\|T \otimes X\|}{\|X \otimes X\|} \tag{2}$$

B. The C++ Class Restructuring using the Proposed ART Algorithm

The subject of both stability and plasticity practices principle of producing a class by improving ART learning algorithm in artificial neural networks and grouping regular vector of input, that is, all nodes can be practiced efficiently of restructuring a class because of an object with an attribute of itself and a method. The learning processes propose being able to restructure a class after improving the ART learning algorithm, namely a model in unsupervised learning algorithm of the artificial neural networks. When the learning is during a class is added and deleted and then the next class gets input through learning and inference, it can estimate the relation between already learned classes.

It's to make 3 vectors (a vector of member function and variable, a vector of named class and a vector of succeeded beginning class) and to learn on each vector. It's learning whether this class is in occurred with already learned class by using improved ART learning method proposed in this paper. In time of being deleted, it is to carry out deletion to search for a relation by basing on a learned subject. After deletion, it is to possess information of existing class by over-learning. Therefore, C++ class restructuring doesn't require preprocessing and runs dynamically by learning in the whole classes.

The proposed C++ class-restructuring algorithm is as follows:

(1) Main Menu Class

```
Class Main Menu: Public Menu {
Public :
      RuningMenu*   _running;
      LearningMenu*  _learning;
      DeleteMenu*   _delete;
      InsertMenu*   _insert;
      ExitMenu*   _exit;
      void check(int x, int y);
      virtual void run();
};
```

(2) Menu Class

```
Class Menu {
Public:
      . . . .
Protected:
     Tree*  _tree;
Public:
      virtual void run();
      void PrintNode(Node*);
      void PrintInsert();
      . . . . . .
};
```

(3) Display Class

```
Class Display {
Public :
      Tree*   _tree;
      MainMenu*   _mainmenu;
      Message*   _message;
Public:
      . . . . . .
      void run();
};
```

(4) NodeList Class

```
Class NodeList {
Private:
      node*   _grad;
Public:
      char   _matrix[MAX][MAX];
      . . . . . .
Public:
      void MakeList();
      void DeleteList();
      void Addition(node* ptr);
      void Del(char num);
      Node* check(int, int);
Private:
      . . . . . .
};
```

(5) Node Class

```
Class Node {
Public:
      . . . . . .
      classmember _member;
      classposition _position;
Public:
      . . . . . .
      Node* check(int, int);
      . . . . . .
};
```

(6) Class restructuring

```
Class Network {
Public:
      float  tweight[MAXNODE][CLASS];
     int _classtable[MAXNODE][CLASSBIT];
      float  _acts[CLASSBIT];
      float  _total[MAXNODE];
      int  _new_pat[CLASSBIT];
Private:
      int   _classnode;
      int   _classbit;
      int   _test;
      float  threshold;
      int   _active_nodes;

Public:
      void start_class();
              // Generate basic class
      void make_class(int);
              // Generate new class
      void digit_compute_edu();
          // Calculate number of digits
      int digit_edu();
              // Vector matching
      void update_weight(int, int);
               // Update weight value
      void learning();  // Learning
      void Running(char, int*);
    // Class matching using weight
```

IV. EXPERIMENTS AND RESULTS

In this paper, C++ class restructuring algorithm using the artificial neural networks is tested by Visual C++ language under Windows 2000 environment, is able to show a structure between member variable and member function of class by viewing function, and is learning that restructuring can be run in whole domain of class when a user chooses it on an occasion that an addition, a deletion and a correction of class are occurred.

In this paper, it deals with restructure of class member but doesn't deal with detailed restructure of its C++ source code.

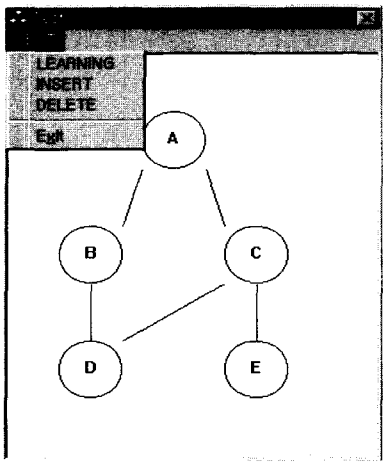


Fig. 1 Initial Structure of Class

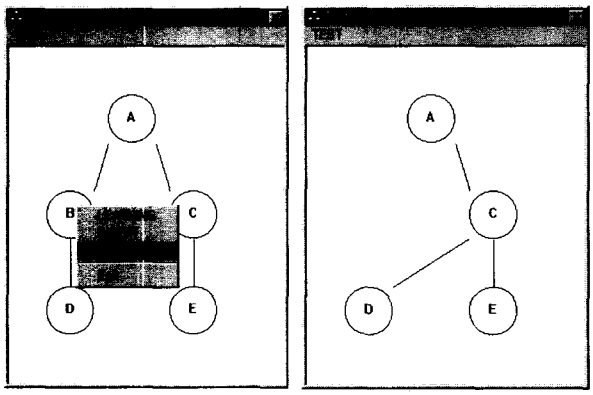


Fig. 2 Restructured State after Class Deletion

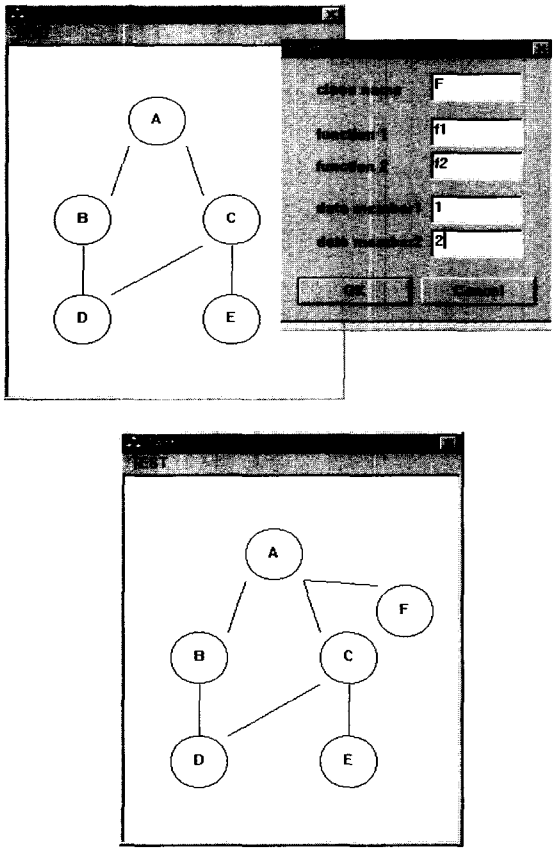


Fig. 3 Restructured State after Class Addition

V. CONCLUSION

C++ classes are not defined independently but defined by using inheritance relationship basing of similarity with other classes. Therefore class restructuring is to understand whether inadequate inheritance, is included it about inheritance relationship between members in time of happening insertion, deletion and update of a new class in maintain step and then to exclude inadequate inheritance. Also class restructuring is one of methods for being formed a new class object by needed property when programmer reuses classes. Classes restructuring are methods for improving reusability and reducing maintain cost, while on the other conventional methods have defects since they go with preprocesses such as dependency analysis and estimation of class cohesion and run statically.

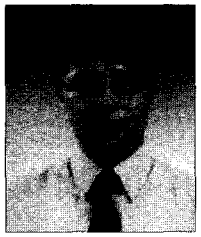
In this paper, we propose, realize and inspect a new C++ class restructuring algorithm that does not require those preprocesses and have additional learning function for running dynamically in the whole possession of class structure by using ART (Adaptive Resonance Theory) learning algorithm in the Artificial Neural Networks. Restructuring algorithm will be the effectiveness of reusability system more than use sub-system of C++ class reusability. Subjects of study from this time on will be expanded C++ class restructuring construction of member into a detailed restructuring of source code. Class and source code are formed automatically by expanding class-restructuring editor of window in environment conformity with an intelligence system. Also they are to be run class restructuring in practice time and to study of a correctly distinguishable learning algorithm.

REFERENCES

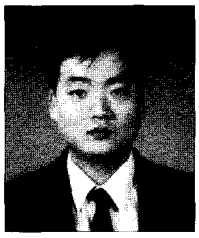
[1] E. Casais, "Managing Class Evolution in Object-Oriented Systems," Technical report, University of Geneva, 1990.
[2] Synder, "Encapsulation and Inheritance in Oject-Oriented Programming Language," Proc. of OOPSLA, 1986.
[3] Robert S. Arnold. "Software Restructuring," IEEE Software. Vol.77, No.4, pp.607-617, 1989.
[4] Ralph E. Johnson and Brian Foote, "Designing Reusable Classes," Journal of Object-Oriented Programming, Jun-Jul, 1996.
[5] Karl Michael Bieman, "Measuring Software Data Dependency Complexity," PH. D. Thesis, Univ. of Louisiana, 1984.
[6] Gianluigi Caldiera and Victor R. Basili, "Identifying and Qualifying Reusable Software Component," IEEE Software, Vol.8, No.2, pp.61-72, 1991.
[7] Ellot J. Chikofsky and James H.Cross II, "Reverse Engineering and Design Recovery: A Taxonomy," IEEE Software, Vol.7, No.1, pp.13-17, 1990.
[8] Lionel Briand et al., "Defining and Validating High-Level Design Metrics," Technical report, Univ. of Myryland, 1994.
[9] David W. Embley and Scott N. Woodfield, "Assessing the Quality of Abstract Data Types Written in Ada," Proc. of International Conference on Software Engineering, 1988.

[10] G. A. Carpenter, S. Grossberg and J. H. Reynolds, "ARTMAP : supervised real-time learning and classification of nonstationary data by a self-organizing neural network," Neural Networks, Vol. 4, pp.565-588, 1991.

[11] T, K. Kim and K. B. Kim, "The Recognition of Student Identification Card based on Improved ART Algorithm and Educational Matters Administration System on The Web," Journal of Korea Association of Small Business Innovation Research, Vol. 3, No. 2, pp.129-128, 2003.



Kwang-Baek Kim
Received his M. S. and the Ph.D. degrees in Department of Computer Science from Busan National University, Busan, Korea, in 1993 and 1999, respectively. From 1997 to present, he is an Associate Professor, Department of Computer Engineering, and Silla University in Korea. His research interests include Fuzzy Neural Networks and Application, Image Processing, Biological Signal Processing and Biomedical System.



Bong-Gi Jun
Received his M.S. and the Ph.D. degrees in Department of Computer Engineering from Busan University, Busan, Korea in 1993 and 2003, respectively. From 1993 to 1998, he joined at Korea Telecom Research Center, where he worked as a manager of technical staff. In 2003, he joined the Division of Computer and Information Engineering, Silla University, Korea, where he is presently a full time instructor. His research interests include Geographic Information Systems, Moving Object Databases and Mobile Information Systems, etc.

Young-Ju Kim
Received his B.S. and M.S. degrees in Computer and Statistics from Busan University in 1988 and 1990, respectively, and Ph.D. degree in Computer Science from Busan University in 1999. From 1990 to 1995, he joined at Qnix Computer Application System Research Center, where he worked as Senior researcher. In 2000, he joined the Dept. of Computer Engineering, Silla University, Korea, where he is presently an associative professor. His research interests include Multimedia Communication and Embedded System, etc.