The System of Classified and Auto-Mark Source Code for Students' Exams

Nguyen Thanh Binh and Pham The Bao

***Abstract*—There are two main approaches to solve classified problems: using the method of comparing strings and abstract syntax tree (AST). In terms of the method of comparing strings, it has many obstacles because it is not recognized by a wide range of changes such as additions, deletions or editions about information on the number of exams or creating called functions without returning any values. This research is utilized by the AST algorithm in order to match and compare students' source codes to AST tree.**

***Index Terms*—Duplication code (DC), clone code, abstract syntax tree (AST), string matching, semantics, fuzzy means.**

I. Introduction

Scoring students' practical subjects for the faculty of computer science is an indispensable task to assess the studying qualities of students. However, nowadays, for the task, the lectures must do manually by themselves while the copied phenomena of students from others have become popular especially important exams like final exams. Normally, students' practical exams are recognized by changing variables, the name of the function in comparison with original sources to avoid the defection of lecturers.

Scoring manually is very difficult to distinguish between the codes copied each other, time-consuming and labor consuming to assess the sources of tests. This work which frequently repeats makes lecturers not only boring, difficult to have the results of tests, but also be very tough to manage the sources of each class in each semester due to the huge number of tests. Therefore, they are easy to get lost the tests as well as stuck when searching, counting the number of tests which are copied by students.

There are works applied to solve these problems throughout the world to developing software's [1] and some algorithms are recommended simultaneously, such as Text–based, Token–based, Metrics–based [2], [3], and so on. Yet, they only implement tasks which compare two simple sources. To be more precise, such codes are textual degrees, not semantic ones. To tackle these drawbacks, compared algorithms to code for the degree of semantic are given like AST (Abstract Syntax Tree) [2], [4]. The paper, we just focus on AST algorithm because of obtaining the applied results which are better than others, including:

- Processing any copies
- Classified data, which are the code of the present in order to optimize matching time.
- Obtaining accurate with any matching.
- The accelerating of process is acceptable

Precisely, applying AST algorithm starts from the code of program analyzed by textual, for example key words, variables, parameters, called functions… then, categorizing token according to the same codes. After that, the program is transformed such Token to AST tree. If the obtained AST is simple, we can conclude what kind of duplication is (Type 1, 2 or 3) due to the system of classifying duplication by SMC. Whereas, if the obtained AST tree is complicated, we must calculate measures based on degree vector or the Euclide's distance.

Manuscript received September 1, 2014; revised January 19, 2015.

Nguyen Thanh Binh is with Management Information System, College of Finance and Customs, Ho Chi Minh City, Vietnam (e-mail: thanhbinhsp@gmail.com).

Pham The Bao is with Mathematics and Computer Science, University of Science, Ho Chi Minh City, Vietnam (e-mail: ptbao@math.hcmus.edu.vn).

II. Definition of the Copied Types

To mark the effective tests automatically, we divided into two categories: marking accurately and classification relied on the group of tests which are entered to database to extract, filter and sort out the tests have the same criteria or we select a test as pattern/sample data. Then, we are searching other tests on the remaining set. Corresponding to the 2 types making, there are 2 algorithms: scoring accurate and scoring classification.

For scoring accurate, we transfer the problem to the issue of matched strings. Hence, we highly detect accurately for coping type 1 and type 2. In contrast, for the copied type 3, such a method does not give good result. That's why we should use marking classification to optimize the output of software systems.

Based on the statistics of the test results of practical informatics. We can classify all copied from the simple to the complex into 3 types as follows [1]:

Type 1: **"copy exactly"** that means two codes the whole same. This type of cloning was detected easily.

Type 2: "copy syntax" nearly similar to each other, but just changing the statement line of functions such as editing or modifying variable names so on. These changes do not affect the structure of functions. For these basic modifications, this copy is also easily detected.

Type 3: "Modified Copy" means that a new statement line is added or removed a few ones. The structure of the code can be changed and even made the code larger but unchanged content. This copy is difficult to detect because we must understand the full context.

DOI: 10.7763/IJIET.2016.V6.744

III. Scoring Accurate

A. Concept

Scoring accurate source code is changed to become the problem of comparing strings: Strings need to compare (the students' program source) to sample (The lecture's pattern program source or the student's pattern program source that is presented) and the system starts to match strings. Inputting the system is sample code and the set of the student's tests. By matching string algorithm accuracy, if two strings are similar each other, the result of this is that the 2 strings are copied together and verse. Therefore, the lecturer will mark the score by every student's test.

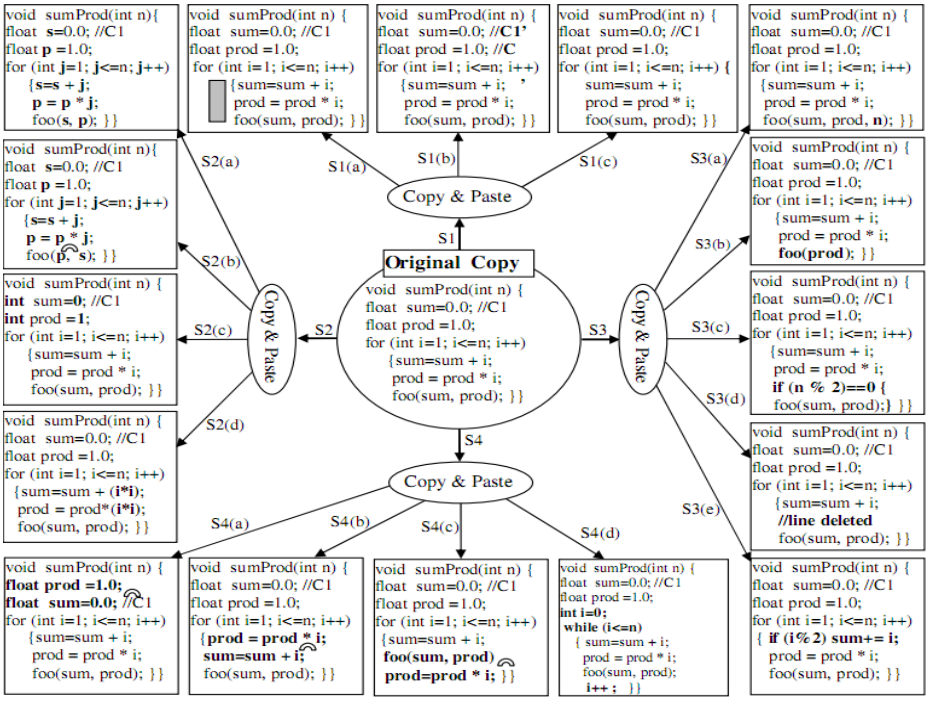

Fig. 1. Describe all the types of copies [5].

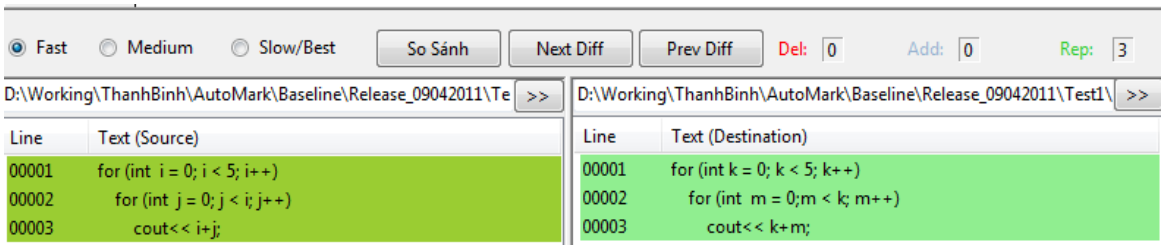

Fig. 2. Result of comparing 2 strings by the software about mark source code.

Example:

Code 1

for (int *i* = 0; *i* < 5; *i*++)
  for (int *j* = 0; *j* < *i*; *j*++)
  cout<< *i*+*j*;

Code 2

for (int *k* = 0; *k* < 5; *k*++)
  for (int *m* = 0;*m* < *k*; *m*++)
  cout<< *k*+*m*;

Two codes matched are the comparison of 2 strings [6]

String 1:

```
for (int i=0; i<5; i++)
  for (j=0; j<=i; j++) cout << i+j;
```

String 2:

```
for (int k=0; k<6; k++)
  for (m=0; m<=k; m++) cout << k+m;
```

Because that manipulate string comparison, each line corresponding results, we count how many changes, such as command code in Fig. 2 for results Rep = 3 i.e. each line are adjusted, changed variable names. Keep scanning over the whole program, this will increase the number of Rep. If Rep belongs to the threshold of what type is (type 1, type 2, type 3), we have concluded exactly how much the percentage of tests are the same of them.

B. Algorithm of Matching Accuracy

Algorithm of matching accuracy [7], [8]:

Step 1: Read each line of the source file.

Step 2: Read the corresponding line of the target file.

Step 3: Compare 2 lines are under any criteria (add, delete, edit) and count the total of criteria by each comparison.

Step 4: Calculate the total of the threshold criteria to determine the type of copy (the measure of each type).

Step 5: Conclusion what copied type is (type 1, type 2 or type 3) and how proportion is the same.

1) Advantages

- Detect quickly and accurate copied codes type 1, type 2.
- Optimized in terms of time and memory

- Efficient and easy implementation, reduced complexity is O(n).

2) Disadvantages

Do not detect accurately with type 3. In other words, if there is a change function name, variable name and moving the position of functions/variables which disorder the structure of the original source code that means we cannot detect the copy of the test.

IV. Marking Classification

A. Cluster Algorithm

The method of marking accuracy cannot be applied to complicated tests or altered many properties such as changes in variable names, adding variables/functions or parameters. In this case, we will gather all the similar properties with the threshold set as a group (cluster) [6], [9]. That means the lectures only mark the representative test of the student of each group instead of the set that of all the ones in such group. There are two solutions to set clusters are that using the method of the exhausted search or combination of the artificial intelligences (AI).

Algorithm 1: The method of exhaustion [10]

From the target set A = {the students' tests) lead to one set = {{the same test}1, {the same test }2, ..., {the same test}N. where: B_i = {the same test}i with i = 1 ... N.

Step 1: select one program, called a_K in set A, grouping all programs are similar to a_K in $A\backslash\{ a_K \}$ creates group B_i

Step 2: Set $A_{mới} = A\backslash B_i$

- If $A_{mới} \neq \varnothing$ Return to step 1.
- If $A_{mới} = \varnothing$ Go to step 3.

Step 3: Stop

Algorithm 1 gives good results if the quantity of A is small. In contrast, that of A is huge will lead to break out combination, so the exhausted algorithm should not affect time costly. To optimize such algorithm, that means the reduced time costly and established the criteria of assessment 2 similar programs by creating AST and responsible measure between them[11].

B. General Structure of a Program

The General structure of a program C/C++ includes 3 main parts as following:

❶	Header: Declare using the library Declare Prototype (if any) [Content of Prototype]
❷	Main: The function calls, commands
❸	[Content of Prototype]

C. Comments

In the student's exam, students can copy all other exams by some ways such as changing variable names, function names, adding function calls in the program, but not called in the main and so on; however, in a program, the commands, the order of function calls in main program are the most important. The function calls have different importance, so we should use an important factor to evaluate. Hence, to compare the two programs of students, we should do pay attention to the main content, and the function is called in main.

D. Constructing Algorithm Compares the Similarity of Two Programs

To solve this problem we have made the improvements AST tree and calculate the value of the analogy of functions between the two programs is a specific value like algorithm 2 so as to give matching results accurate.

For example, we match A and B programs [12]:

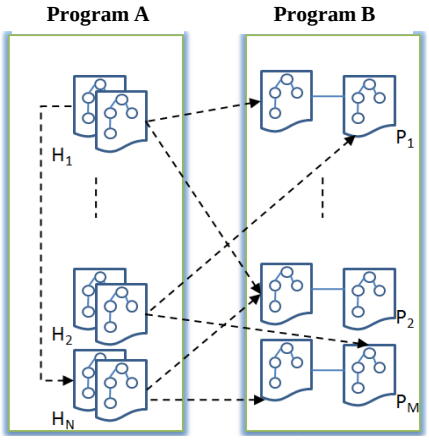

Fig. 3. Comparing the similarity of two programs.

$H_i \in$ {Function of program A}
(i = 1,…N)
$P_j \in$ {Function of program B}
(i = 1,…M)
H_i can responsible to P_1 or P_2 (i = 1,…N; l, k = 1, … M)

Algorithm 1: Extracting data [13]

Step 1: Analysis main function to establish AST_{main}

Step 2: Analysis main function to find H = {the number of functions using in main function}

Step 3:

for (i = 1; i<|H|; i++)
{
Find the code of H_i and input of H_i ;
cout<< AST_{Hi}
}

Each function H_i of program A can resemble to a few functions P_j of program B by identifying combinations means they will produce more comparable. In other words, each function H_i of program A is similar to several functions P_j of program B will get a set of comparable value. As exhausted algorithm, part 3.2.1 we have presented, such algorithm can apply to all cases and obtain accurate results, but leads to combinatorial explosion if the number of function calls is big in the main function. To reduce the problem and costly time we introduce an algorithm to find similar function from program A and program B using AST for the main function and AST for the function to determine the measure.

Algorithm 2: Identify the similarities of the two

programs A and B [14]

Step 1: Assume we have N function calls in order.

Step 2: Create AST tree for the main program (main): AST_{main}

Step 3: Create AST tree for function call H_i (i = 1 .. N): AST_{Hi}

where: H_i includes the return value and parameter list or variable parameter

a_i: important rate of each function.

α_i : Compare the value of the AST tree of H_i in the program A with similar function P_j of program B ($\alpha_i \in [0,1]$)

$$\alpha_i = \left| AST_{H_i} - AST_{P_j} \right| \quad (1)$$

V is a measure of program A to program B

$$V = \frac{\sum_{i=1}^{N+1} a_i \alpha_i}{\sum_{i=1}^{N+1} a_i} \quad (2)$$

From (1) and (2) infer: Corresponding to each function i (i = 1, ... N) we have:

$$V = \{v_1, v_2, \ldots, v_N\}$$

For comparison between the two programs A and B we will rely on the maximum value of the set V (max V) based on three main criteria: The number of command line is added, the number of command line is changed, the number of command line is deleted. Depending on the changes (add, edit, delete) to evaluate our results based on theory and re-evaluate after implement on the scoring tool of source.

Based on the experimental results and then re-evaluate the results of the AST tree to provide accurate results by applying AST algorithm compares two programs

E. The Measure of the Copy Criteria

To determine the threshold V_{max} any copied type, we sort out the criteria like styling pyramid (Fig. 4) to show the importance rate of each criterion by kind of type and assign weights to each criterion. In Table I, added criteria are assigned to the most important rate because it illustrate creativity, no copy, and create complicated rate to match with each other.

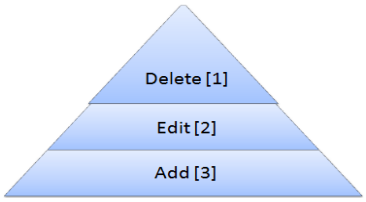

Fig. 4. The weights of pyramid.

After determining the weighting for each criterion, we will decide threshold V_{max} for each criterion as Table I. If the threshold:

$V_{max} \in [0, 2.7)$: copied type 1
$V_{max} \in (2.7, 30]$: copied type 2
$V_{max} \in (30, 60]$: copied type 3

TABLE I: WEIGHTS FOR EACH CRITERION

No	Content	Add	Edit	Delete
1	Type 1	3	2	1
2	Type 2	3	2	1
3	Type 3	3	2	1
4	Sum	9	6	3

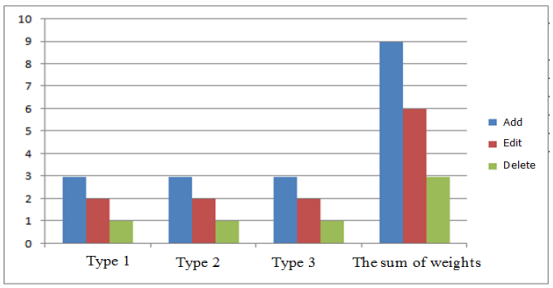

Fig. 5. Weights for each criterion.

TABLE II: WEIGHTS FOR EACH CRITERION

No	Content	Add	Edit	Delete	Threshold
1	Type 1	0	8	0	2.7
2	Type 2	0	90	0	30
3	Type 3	80	40	40	60

To optimize the marking and reduction in processing time. We first perform a test by classifying any copied type before using the matching algorithm. After classifying it into two groups:

X: copied type 1 and 2

Y: copied type 3

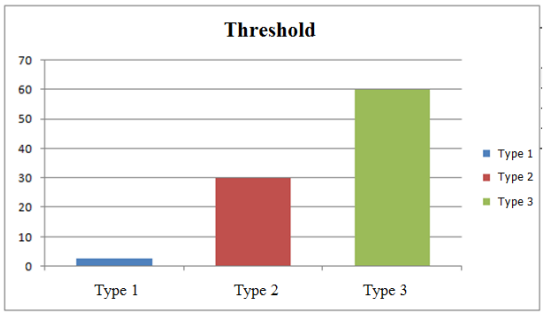

Fig. 6. The threshold of each copied type.

For the program belongs to type X, we can use the string comparison method or the classification methods. For type Y, we use the scoring classified. This is to optimize the processing results and improved effective performance of each scoring method helps to implement the system quickly, accurately and efficiently.

V. RESULTS

To apply this problem to a real one. Firstly, we will set the

rules to name the storage folder of tests. For instance, every exam has more than one class done test, each class will have a topic and an answer. With regard to students, the regulative of tests, which contained in folder, is students' ID. After grouping the tests into groups and name the folder containing all students with the structure as follows: TT_HK1_2014 (TT: Faculty of computer science, HK1: semester, 2014: current year)

Assumingly, there are 100 student examinations of class A and the tests contained in the test folder [15]:

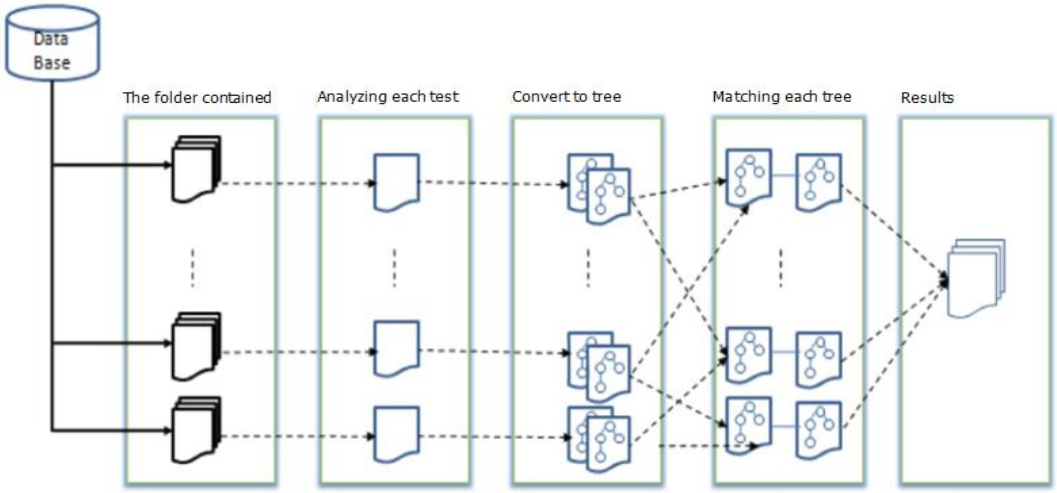

Fig. 7. Illustrating the system of analyzing the table.

From the folder system (Folder), the system will convert to hierarchy graph in software systems stored. The system will find the content duplication or copy of the procedure / function in these trees. For example, we have two trees are two assignments of two students:

For this task, if there are two files, then this tool will analyze each file into three sub_tree as follows:

- AST_{Main} Tree: represents the MAIN function and the method of function calls as well as returns statement presentation.
- $AST_{Function}$ Tree: shown the flow graph of the function in the MAIN function
- $AST_{Statement}$ Tree: corresponding to each statement in the program.

After that, the system will match each tree by the kind of the copied type from small to big. Starting with AST_{Main} tree, next it is $AST_{Function}$, and $AST_{Statement}$. In the process of comparing, the system will automatically cluster the same duplication and keep merging other trees into the mutual duplication. Then, to continue such task for the left. Finally, that's done comparing the AST tree together.

TABLE III: THE THRESHOLD OF THE EXPERIMENT

No.	Test	Total paper	Type1	Type2	Type3
1	Test 1	100	60	25	15
2	Test 2	100	15	60	25
3	Test 3	100	5	35	60
4	Test 4	100	10	55	35
5	Test 5	100	20	45	35

This research has developed a system to assist teacher's statistics that are made the same, clustering programs according to a criterion fastest, most accurate. It also managed to make all of the information as well as scores of students in each class, and any examinations of the department in the semester.

TABLE IV: THE EXPERIMENTAL RESULTS FOR THE SCORING ACCURACY

No.	Content	Count	Type1	Type2	Type3
1	5 tests	500	100	220	180
2	Threshold	100	20	44	36

TABLE V: EXPERIMENTAL RESULTS FROM THE USE OF GROUPING

TEST DATA						THE RESULTS OF SCORING ACCURACY	
No	Test	Quantity	Type 1	Type 2	Type 3	Time	Accuracy
1	Test 1	100	60	25	15	600	87.15
2	Test 2	100	10	60	30	1.200	72.3
3	Test 3	100	5	35	60	2.400	42.6
4	Test 4	100	10	55	35	1.400	67.35
5	Test 5	100	15	45	40	1.600	62.4

TABLE VI: THE TIME CALCULATED USING AST

TEST DATA						RESULT OF AST	
No	Test	Quantity	Type 1	Type 2	Type 3	Time	Accuracy
1	Test 1	100	60	25	15	75	48.6
2	Test 2	100	10	60	30	12.5	98.1
3	Test 3	100	5	35	60	6.25	91.5
4	Test 4	100	10	55	35	12.5	98.1
5	Test 5	100	15	45	40	18.75	93.15

Simultaneously, such system also helps users time-saving to mark students' programs, improve the work efficiency and increase productivity, achieving accurate results and the statistic of students' qualified study. Users can change the criteria to make more flexibility to the tests and enrich the way of that test made.

VI. Conclusion

We will keep developing the well-rounded system, apply this application for the scope of teaching - learning more effectively, simplify the evaluation of students' results for teachers. The focus of the current problem is concentrated duplicate detection in large strings of data as text [3]. We should have further consideration in this field because, these days, the applications of DC have become more popular in a wide range of real life such as images or audios.

Relevant data can be text (press content, documents, administrative documents, textbooks, articles, dissertations, source code), Image (pictures, layout, logo, trade brand, and intellectual property), Sound (audio, video, flash, etc.), etc. Corresponding to each specific type of data we need to have a different process. In the near future, applications such as cloud computing (clouding computer) will be implemented tremendously in businesses, schools, using data to reduce the amount of information and increase the amount of knowledge for humankind.

References

[1] Y. Jia, *Clone Detection Using Dependence Analysis and Lexical Analysis*, Department of Computer Science, King's College London, September 2007, ch. 2, pp. 5-8.

[2] Z. M. Jiang, "Visualizing and understanding code duplication in large software systems," presented to the University of Waterloo, Master of Mathematics in Computer Science, Waterloo, Ontario, Canada, 2006, ch. 2, pp. 19-22.

[3] L. X. Jiang, G. Misherghi, Z. D. Su, and S. Glondu, "Deckard: Scalable and accurate tree-based detection of code clones," University of California, Davis and ENS de Cachan, France, 2009, pp. 1, 3-4.

[4] G. M. K. Selim, L. Barbour, W. Y. Shang, B. Adams, A. E. Hassan, and Y. Zou, "Studying the impact of clones on software defects," Queen's University, Kingston, Canada, 2006, pp. 1-4.

[5] C. K. Roy, J. R. Cordya, and R. Koschkeb, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," School of Computing, Queen's University, Canada and University of Bremen, Germany, 2008, p. 25.

[6] C. K. Roy and J. R. Cordy, "Nicad: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization," School of Computing, Queen's University Kingston, ON, Canada K7L 3N6, 2001, p. 4.

[7] R, Wettel and R, Marinescu, "Archeology of code duplication: Recovering duplication chains from small duplication fragments," LOOSE Research Group, Institute e-Austria Timisoara, Romania, 2005, p. 5

[8] N. Yoshida, T. Ishio, M. Matsushita, and K. Inoue, "Retrieving similar code fragments based on identifier similarity for defect detection," Graduate School of Information Science and Technorogy, Osaka University, Toyonaka, Osaka, Japan, 2008, pp. 1-2.

[9] M. Pfahler, "Improving clone detection for models," Master's thesis in informatics, Verbesserung der Klonerkennung fur Modelle, November 15, 2009.

[10] K. Greenan, "Method-level code clone detection on transformed abstract syntax trees using sequence matching algorithms," Department of Computer Science University of California, Santa Cruz, March 16, 2005, pp. 4-7.

[11] G. G. Koni-N'Sapu, "A scenario based approach for refactoring duplicated code in object oriented systems," Diploma thesis of the Faculty of Sciences, University of Bern, 2001, ch. 4, pp. 36-40.

[12] M. Hutchins and K. Gallagher, "Improving visual impact analysis," CSIRO Mathematical and Information Sciences, GPO Box 664, Canberra, ACT 2601, Australia, 2005, pp. 9-10.

[13] M. Gabel and L. X. Jiang, "Scalable detection of semantic clones," 2008, p. 17.

[14] N. Gode, "Incremental clone detection," Diploma thesis, University of Bremen, Faculty of Mathematics and Computer Science, 1st September 2008, ch. 2, pp. 10-17.

[15] M. Balazinska, E. Merlo, M. Dagenais, B. Lague, and K. Kontogiannis, "Measuring clone based reengineering opportunities," 2003, pp. 6-7.

Nguyen Thanh Binh was born in Thanh Hoa Province, on 2nd March, 1983, Vietnam. From 2003 to 2007, Ms. Binh received her BSc in computer science, Ho Chi Minh City University of Pedagogy, Vietnam and then she achieved her master of applied maths and computer science, Ho Chi Minh City University of Science, Vietnam between 2008 and 2011.

She has been working as a lecturer from 2008, at the Faculty of Management Information System, the College of Finance and Customs (CFC), Vietnam. Her interests are the image processing, algorithms, approach teaching for IT.

Pham The Bao is a lecturer from the Faculty of Mathematics and Computer Science, Ho Chi Minh City University of Science, Vietnam.