ICT, ELECTRONICS, PROGRAMMING

UDC 004.02

REVIEW OF IRREGULAR CHARACTERISTICS OF PROGRAM CODE

MAKSIM MATSIUSH
Polotsk State University, Belarus

Analysis of the code used to solve the actual problems of determining the quality, searching plagiarism and authorship of source code. For each specific target can be approached only certain methods and algorithms of analyzing the source code of programs.

In the source code of programs, in addition to the standard features, there are also characteristics peculiar to a particular author (naming of variables, the style of writing, the linguitic peculiarities of comments, the number of different data types, stable level of certain metrics, etc.) [1 – 4], so it is appropriate to use the concept "programmer's handwriting".

The concept "programmer's handwriting" is difficult to define as we are to find code optional characteristics that are inherent to a particular programmer. Programmer's handwriting can manifest itself in the architectural solution, for example, in the abuse of any design pattern or coding rules, nomination of variables, methods, and other structural units of the programming language, such as:

– naming of classes, methods and variables, local variables of the method, method parameters;

– the place of initialization the class variables, creating unused class methods (i.e. excess precision that comes from the design, but isn't used in the current implementation);

– the number of comments, the view of comments, the place of comments;

– the length of method names , the use of types;

– the use of transfers;

– the listing of interfaces, but not the use of polymorphism;

– the use of a specific template in all the programs of the author, a certain standard set of code metrics that have the same value in all programs of a specific programmer, etc.;

Programmer's handwriting should first clearly define the website source code, as well as to find the handwriting of the author in search problems of plagiarism. Since under the programmer's handwriting, we also mean a certain level of supported source code metrics, the quality of handwriting has an equivalent relationship with the quality of software code. Thus, this characteristic simultaneously embraces calculation of code quality, searching of plagiarism and authorship.

Let's consider the concept "native programming language". It is the language the programmer is good at when creating a software product. And when a programmer happens to write a program in a so-called non-"native" programming language, the text manifests a programmer's accent. The programmer's accent is an integral part of the programmer's handwriting. It allows the programmer to focus on selection in addition to greater accuracy in search problems of plagiarism and authorship also conduct statistical research on migration programmers with a particular programming language. And the accent allows determining the compliance with the negative coefficient coding standards for the test code, which in turn is the quality of the code.

Heuristic methods for the isolation programmer's accent must allocate a priori set of mandatory and optional features that are specific to a particular programming language, when reduced to a common class of singularities, and the subsequent search for these features in the source code with a great programming language of this class. This feature can also assume the uncertainty in the case where the source code does not match any of the inspected class accents set , even the class with the certain programming language source code , on which will be analyzed. In this case, it is considered that it is not the focus, but a simple ignorance of programmer.

What is clear is that when searching for a accent this process becomes part of the process of defining a unique programmer's handwriting. Therefore necessarily stylistic features of programming is not a panacea to the action, as it was originally a person can follow the rules , and it is the task of searching the emphasis goes into searching programmer's handwriting. When compared to the literary language, such programmers are illiterate. The latter assumption allocates another metric – *programmer's literacy*.

Literacy of programmer is accordance of source code the programmer to standards proposed and widely used for a specific language. This problem is important, for example, in hiring and monitoring the quality of software code. Solution of the problem reduces to determining the criteria of literacy, layout and more – serial programmer check code for compliance. The rules may include naming of variables, functions, or methods, and then adopted language syntax token sequence. It should also be noted immediately that the design rules for one language do not match the rules of registration for another. Because of this, and there is emphasis on the transition from one programming language to another.

Programmer's literacy, on the one hand, is the opposite characteristics to the accent of programming, on the other – an integral part of programmer's handwriting determines its quality. But beyond that literacy determines the correctness of the instructions and rules that are characteristic for most programming languages.

Programmer's handwriting covers all other characteristics, but not uniquely determines the unique feature that is typical for this programmer. This feature is called the unique style of the programmer who will be calculated by finding the unique performance characteristics and previously considered to have both positive and negative tendency to change the quality of the source code. Typical metrics that reflect the unique style of the programmer can be non-standard commenting source code, adding his own prefixes and suffixes to the classes' names, methods and variables.

A distinctive feature of this characteristic is that it can be deliberately varied by programmer himself. This characteristic can be specifically introduced by the programmer for the protection of its source code. You can also divide a programming style for good and bad with respect to the rules of registration code, respectively, and the quality of software code. And in this context can only mean bad that the author of the source text is not specifically follows the general rules of the programming language , maintains its own because of ignorance , or to leave a unique code. Good style is considered a style that is not contrary to the rules of registration, but has redundant comments or instructions that are hallmarks of the author.

Positive dynamic of these characteristics is proportional to the lowest programmer's accent, zero indicators unique style of programmer, high literacy rate programmer and other indicators of good programmer's handwriting. Is proposed that the code "clean" if it is particularly ideal for solving the problem. The main difficulty in finding this criterion – are predefined ideal solution that is utopian. It is therefore proposed to use this feature only to specific instructions and compliance with design patterns.

Features of team programming. Modern high-load projects are the result of working of not one, but a group of programmers. These can be developed as separate unrelated modules and one class (as a syntax language unit) by team. This feature may require using of the above metrics are not one the programmer and to the team. In this case, it will be a general characteristic of the average of all team members. Also in the case of team programming, if they are known to work every programmer and prefilled reference characteristics for each programmer, it will be possible to carry out analysis on the percentage of participation in writing the source code each programmer in the team.

Another feature of the team programming is a different skill levels every programmer that is expressed in a variety of characteristics, namely indicators of handwriting, literacy, accent, etc. This feature allows you to define the maximum and minimum, and to measure the characteristics of the dynamics of change in the code. This figure will more accurately allocate the overall dynamics of changes in indicators of the team changing its participants.

Also it is worth noting that this feature of the team programming is in conjunction with the development of migration between manufacturing companies and software generates precedents plagiarism. It is understood that the proposed non-standard characteristics above software code after their implementation can be identified and such copyright infringement.

REFERENCES

1. Analysis of Algorithms for plagiarism detection in codes of programs written in high level languages // Modern techniques and technologies: XVIII Intern. scientific-practical. conf. [Electronic resource]. – 2012. – Mode of access: http://www.lib.tpu.ru/fulltext/v/Conferences/2012/C2/V2/v2_212.pdf. – Date of access: 10.07.2013.
2. Identification of duplication and plagiarism in the source application // Institute of Control Sciences problems [electronic resource]. – 2005. – Mode of access: http://lab18.ipu.ru/projects/conf2006/ 1/15.htm. – Date of access: 22.08.2013.
3. Search for plagiarism in source programs // Belarusian state university [electronic resource]. – 2011. – Mode of access: http://www.fpmi.bsu.by/ImgFpmi/Cache/36173.pdf. – Date of access: 03.09.2013.
4. Software complexity metrics as criteria for evaluating plagiarism in source programs // Proceedings of the Intern. scientific-practical conference [Electronic resource]. – 2012. – Mode of access of the: http://www.apriori-nauka.ru/uploads/files/RIBANOVK6.pdf. – Date of access: 25.09.2013.