CONTROL OF SOFTWARE QUALITY

VIORICA I. MADAN

Software Department
Institute of Mathematics
Academy of Sciences of Moldavia
5 Grosula Street, Kishinev,
REPUBLIC OF MOLDAVIA

MAIA M. UNGUREANU

Software Department
Institute of Mathematics
Academy of Sciences of Moldavia
5 Grosula Street, Kishinev,
REPUBLIC OF MOLDAVIA

ABSTRACT

An abundant growth of software tools is noticed at present. A mechanism ensuring a control of the evaluation and classification processes as well as of the efficient program development is proposed.

1. INTRODUCTION

A very large number of measures analysing diverse program features has been proposed over the last decade. Some measures can be automatically determined from the program texts, while the others are determined by a human expert evaluation. Experimental validations of many measures for different programming languages have been made in both academic and software engineering environments.

The measurement of the program features has a capital importance for the assurance of software quality, for the processes of quality determination and evaluation, by facilitating software control during its entire life-cycle.

No less important is the task of assessment of the already existing programs as a large knowledge base concerning programs (to be referred further on as the program base) in the context of which each program would be characterized by a diversity of quality measures. The existence of a program base of this kind would have made it possible to automate such large processes as:

1. Classification within the space of the already existing programs;
2. Choice of the appropriate program out of this space;
3. Construction of an optimal program.

Whenever a new program appears it is always necessary to make its evaluation and classification, i.e. to locate it within the structure of the already existing programs. In case that the program base has no analogues to the solution of the fixed problem, it is desired to have an automatic means enabling the construction of an optimal program.

Below we proceed to considering into detail the questions of accomplishing the automation of the above-mentioned processes.

2. PARTITION OF PROGRAM SETS INTO CLASSES.

Let's have the following marking:

E stands for the analysed finite set of programs $|E| = n$

where the programs are denoted by $x_1, x_2, \ldots, x_n$.

F is the mark for the set of measures characterizing the quality of program. We assume $|F| = P$ and mark the measures characterizing the x program as $x^1, x^2, \ldots, x^P$.

$M_{x_j}[i] \subset R$ is the ith component of M_{x_j} vector, i.e. the assignment of the i-characteristic to the x_j program.

$M_{x_i} = \{ x_i^1, x_i^2, ..., x_i^p \}$ is the vector representing the co-ordinates of each of $X_i \in E$ programs. The d distance between the correct programs in this space makes:

$$d(x,y) = \left\{ \sum_{i=1}^{p} (M_x[i] - M_y[i])^2 \right\}^{\frac{1}{2}} \quad (1)$$

Within a cluster analysis the d magnitude represents the dissimilarity measure [1].

Below we present the algorithm of the automatic partition and classification of n programs into non-intersecting classes:

1) The d(x,y) distance between programs is computed according to formula (1) (by observing the condition that the M_{x_i}, $i = 1,2,...,n$ vectors for all $x_i \in E$ are obtained);
2) The E set is randomly partitioned into K classes. We receive the $P = (P_1, P_2, ..., P_k)$ partition;
3) A representative space for the given partition is built:

$L = (l_1, l_2, ..., l_k)$. The representative of the i ($l_i = X_i$, $i = 1,2,...,k$) class is chosen to be the weight centre of this class which can be found according to the formula:

$$M_x = \frac{\sum_{i=1}^{n} M_x \delta_{ij}}{\sum_{j=1}^{n} \delta_{ij}} \quad (2)$$

where

$$\delta_{ij} = \begin{cases} 1, & \text{if } x_i \in P_j, P_j \text{ is the i class of the P partition} \\ 0, & \text{otherwise} \end{cases} \quad (2a)$$

4) The adequacy measures between $X \in E$ and $l_i = x_i$ are deduced as $D(x,l_i) = d(x,x_i)$ where $d(x,x_i)$ is deduced according to formula(1).

The value of $W^{(r)}$ criterion is found out to be:

$$W^{(r)}(P,L) = \sum_{i=1}^{k} \sum_{x \in P_i} D(x, l_i) \quad (3)$$

where r is the iteration number, $r = 1,2,...;$

5) $P^{(r+1)} = f(L^{(r)})$ and a new value of the $W^{(r+1)}$ criterion are calculated. For the prescription function f, we choose $L_k \rightarrow P_k$ as follows: the ith class includes that element for which there is $D(x, l_i) < D^k(x,l_j)$ for all $j \neq i$;
6) if $W^{(r+1)} < W^{(r)}$ proceed to step 3);
7) if $W^{(r+1)} = W^{(r)}$ that means that the process has stabilized;

8) All class centres are scanned. Should the class centres be very close to each other two classes were combined and the number of classes were reduced. Should the dispersion of one class be too large it underwent division into two classes and the number of classes increased;

9) The centres of classes are recalculated;

10) Steps 8-9 continue up to the stabilization of the class centres.

Having done the partition of programs space into non-intersectional sets, one can proceed to a multicriteria analysis of programs within each class [2] in order to find "the best" program (a kind of a 'standard') for this class. Different measures of program evaluation can serve as criteria.

Actually, the representative l_i of the ith class is not by all means a real physical point but an averaged index of the class. Such a standard becomes the real program (or some programs) with the best quality coefficients.

3. INSERTION OF A NEW PROGRAM INTO THE PROGRAM BASE.

Let's consider the case of a new y program appearance and its insertion into a certain class of the E space.

First, its characteristics are deduced. Thus, the M_y vector is obtained. In order to determine the class which the program belongs to one should deduce all $d(y,l_i)$, $i=1,2,...,k$. The y program can be included into a certain class of the existing partition only if it satisfies the condition:

$$\sum_{i=1}^{k} d(y, l_i) \leq 0.5 \sum_{\substack{i,j=1 \\ j \neq i}}^{k} d(l_i, l_j) \qquad (4)$$

Otherwise, the y program will be a representative of a new class and an entire reclassification of the E space according to the above described algorithm is needed.

When condition (4) is satisfied then the y program can be included into a class for which $\min_i d(y,l_i)$ is achieved.

The following is possible:

1) If $d(y,l_i)=0$, i.e. the y program characteristics are identical to the ones belonging to a representative of the ith class then it becomes a representative of this class;
2) If there exists such a $x_j \in P_i$ program where $d(y, x_j)=0$ then it signifies that there exists already a realization of this kind and the y program is rejected;
3) If $d(y,l_i) \neq 0$ and $d(y,x_i) \neq 0$ then the necessity arises to compare it with the standard of the class. By applying the multicriteria analysis we can decide whether the program can become a standard for the given class or not. If the statement is positive, the structure of the set of standards is changed.

After the location within the ith class, a reclassification of the entire E space gets necessary.

4. CHOICE OF A PROGRAM OUT OF THE PROGRAM BASE.

Before writing any program it is possible to predict its complexity by using the program specification. One can find different measures of this kind in literature [3]-[5]

A detailed description of these measures falls beyond the goal of this work, that is why we proceed to the next statement of the problem.

Let's say that it is necessary to solve (to program) a certain problem.

This problem could have already been realized and an appropriate program from the E space could correspond to it. For this case we propose the following scheme of choice:

A priori its complexity has been determined many characteristics of trade-offs in realization are subsequently found. We shall denote the obtained prognosis by x_a which stands for a priori evaluation of the problem trade-offs. Before starting to make a choice, it is desirable to check whether x_a point belongs to a class or not and if it does, to what particular class it belongs. Hence a search for L representatives in the space is made and the $l_i \in L$ representative ensuring $\min_i d(x_a, l_i)$ is chosen.

One should further select from the ith class of that x program which implies the smallest distance from x_a. The optimal solution is achieved when x_a is located at the nearest distance from the standard of this class x_e. This proves the existence within the E program space of a program able to solve a similar problem and it is up to the Decision-Maker to make the choice.

In case we fail to find an appropriate class for solving the given problem we proceed to constructing an optimal program for this problem.

5. CONSTRUCTION OF AN OPTIMAL PROGRAM.

Statement: The x program realizing the A algorithm with a priori x_a characteristics can be called optimal if $\min d(x_a,x)$ is achieved for it in a set of $\{x_i\}$ programs which obtain the given algorithm.

Thus, we shall assume x_a to be the standard for the set of $x_i \in E$ points realizing the algorithm and the x_i program being located the closest to x_a will be considered the best one.

The process of the program optimization according to some criteria implies the application of a certain set of rules which makes it possible to discard all kinds of drawbacks from the program, thus avoiding any change in its local and global invariants.

Many drawbacks are discarded by good translators but this happens only in case of object programs. Subsequently, one should accomplish the construction of such software which in an interactive process is able to indicate to the user the "weak" places within the program.

It is apparent that by discarding the drawbacks from the program one accomplishes the optimization of its features and hence the program iteratively "approaches" its optimal presentation.

The optimal program having been constructed it is included in the E space according to the scheme in paragraph 2.

6. CONCLUSIONS

Construction of automatic systems performing accounting and analysis of the existing program base as well as the program quality control at a developing stage are both necessary and possible.

Systems of this kind should include:

- analysts of programs displaying as many as possible characteristics of their qualities;
- classifying factors of programs giving the possibility to make an optimal partition into classes and a fast search for the required program;
- programs controlling the programming process (able to reveal drawbacks, the level of closeness to the a priori estimation, or to the standard of the class) and displaying the program time qualitative profile (i.e. change of the program quality characteristics in case of debugging and testing).

All the above enumerated tools will help to exert a quality control over the software development and support processes.

REFERENCES

[1] DIDAY, E., et al. **Optimisation et Classification Automatique** //INRIA, Domaine de Voluceau, Rocquencourt, 1979.

[2] MADAN, V.I., UNGUREANU, M.M., **Mnogocriterialinii analiz cachestva programm** // sb. Sistemnoe i teoretichescoe programmirovanie, SHTIINTA, 1986.

[3] COULTER, N.S., COOPER, R.B., SOLOMON, M.K., **Information-Theoretic Complexity of Program Specification** // COMPUTER JOURNAL, vol 30, No 3, 1987.

[4] ALBRECHT, A. J., GAFFNA, Y J.G., **Software Function, Source Line of Code and Development Effort Prediction: a Software Science Validation** // IEEE TRANS. ON SOFTWARE ENGINEERING SE-9(6), 1983.

[5] HALSTEAD, M., **Elements of Software Science** // Purdue University. ELSEVIER, New York-Oxford-Amsterdam