Evaluation of Software Product Quality by Means of Classification Methods

Avram Eskenasi

Institute of Mathematics, Acad. Bonchev str., Sofia, Bulgaria

A new method for software products' quality (SP) evaluation is proposed. It is based on known, refined or newly developed classification methods. Several well-known software products of a given type are described by using a binary scale for a set of characteristics. These SP are then distributed into a few classes of quality. Classification methods are used to determine which is the appropriate class for each new SP, described in the same way.

1. INTRODUCTION

As is well-known, the production of software has been developing for several decades. Some aspects of this production have already been theoretically explored. But the specific task of software-quality evaluation is a relatively new problem of great importance.

Up until now, the methods for quality evaluation of software products (SP) have been based on the weighted sum of various measures that assess the degree to which certain characteristics exist in the program [1]. By software quality we mean a set of characteristics of the software product or service that shows their capacity to satisfy certain needs [2, 3]. When the number of the characteristics is low, the final sum is obtained easily, but is not precise. When the characteristics are numerous, they are often broken down into more elementary ones, and therefore many calculations are necessary in order to obtain the final sum step by step. In this case, the evaluation is more exact but the procedure is more expensive and time-consuming. The result depends on the particular scores as well as on the accepted weights, and since both are determined by experts, a strong element of subjectivity is introduced. It should also be pointed out that it is rather difficult to join all elementary scores into a final one, and this is why, sometimes, only separate scores on several characteristics are given and compared in the form of charts, diagrams, etc.

Address correspondence to Avram Eskenasi, Institute of Mathematics, Acad. Bonchev str., Bl. 8, 1113 Sofia, Bulgaria.

Our purpose is to propose a new method that is simple, inexpensive, and sufficiently precise. We feel that the existing concept of good product quality is based not so much on a high rating of the evaluation function, but on positive comparison results with the best products of the same type. Very often the important thing is not to get a precise rating, corresponding to the product quality, but to get an approximate evaluation whether to accept or to reject the product. Hence, it is likely that the decision to be made will be based on past experience, and therefore classification methods for solving the problem of SP quality could be applied.

2. THE METHOD PROPOSED

Obviously, only software products of the same type should by compared. The simplest way to start the evaluation is to assign each characteristic either 0, if the product does not possess the considered property, or 1 if it does. In cases where there is a lack of information, it is also possible to assign a third value—designated by "x."

Let us suppose we have several well-known products. We assign the values (0, 1, or in some cases x) to their characteristics, which have been determined and fixed in advance. We shall call these products "samples" or "standards." We divide these standards into several classes depending on their quality (e.g., "excellent," "good," and "poor," or only "good" and "poor"). Whenever a new product is to be evaluated, a 0, 1, or x is assigned to each of its characteristics on the basis of an unambiguous procedure. We then try to link the new product to one of the already defined classes, thus obtaining a quality evaluation for it, by applying classification methods on the "standards" information (represented by the 0–1–x "baseline" vectors). The possibility of distributing the standards into classes using a formal procedure shall not be considered. Only the experience and judgment of experts and users as well as

The Journal of Systems and Software 10, 213–216 (1989)

0164-1212/89/$3.50

Table 1.

"Standards"	Characteristics												
	1	2	3	4	5	6	7	8	9	10	11	12	13
I = II	1	1	1	1	1	1	0	1	1	1	1	0	1
VII	1	1	0	1	x	1	0	1	1	0	0	0	0
III	1	1	1	1	0	1	1	0	0	1	0	1	0
IV	1	1	1	1	0	1	1	1	1	1	1	1	0
VIII	0	1	0	0	1	1	1	x	0	0	1	x	0
V	0	0	1	0	0	0	0	0	0	x	0	x	0
IX	0	1	0	1	0	1	0	1	0	x	x	x	0
X	0	0	0	1	0	0	0	1	0	x	x	x	0

other pertinent information, if available, are taken into account. This obviously introduces a certain degree of subjectivity which however is on a different level compared to the subjectivity of the "classical" methods.

3. THE MATHEMATICAL MODEL

Let M be a set of objects. M is divided into subsets, K_1, K_2, $\cdots$, K_s, called classes so that $M = \cup K_i$, $K_i \cap K_j = \emptyset$ for $i \neq j$. The class K_g is represented by the objects $s_{m_{g-1}+1}, \cdots, s_{m_g}$, which we shall call standards or samples ($m_0 = 0$, $m_s = m$). The description of every object is given by the values of n characteristics: $s_i = (a_{i1}, a_{i2}, \cdots, a_{in})$ for $i = 1, 2, \cdots, m$, and $a_{ij} \in \{0, 1, x\}$, where 0 denotes that the characteristic is missing, 1 denotes that it is present, and x denotes that no information about this characteristic is available. The description of the standards $s_1, s_2, \cdots, s_m$, grouped in classes, defines the teaching table T_{mns}. This table is acceptable if there are no identical rows in the different classes. Using the table T_{mns} and the description of a new object $s = (a_1, a_2, \cdots, a_n)$, the teaching algorithm A determines which class the new object s belongs to. In some cases, the algorithm could reject any classification.

One of the existing types of algorithms uses the so-called combinational logical approach in which the central notion is the "terminal test." Many such algorithms that use different heuristics and are based on some measure of "adjacency" and some voting procedures [4] have been developed.

We shall call "test" of T_{mns} a subset of columns of T_{mns}, such that every two rows of T_{mns}, including only these columns, are different, if they belong to different classes. A "terminal test" is a test, no subset of which is a test. Terminal tests of T_{mns} in Table 1 are {7, 9}, {1, 3, 5}, etc. The number of tests grows exponentially with the number of characteristics. Therefore, it is acceptable to find out all terminal tests for only a limited number of characteristics, e.g., 30. Some mathematical results about $|T|$ are available [5], but they are in fact not applicable because of their asymptotical character. Let us denote the set of all terminal tests of T_{mns} by T, where $|T| = \tau$, and the set of all terminal tests, containing the column j, by T_j, where $|T_j| = \tau_j$. The determination of all terminal tests of a given T_{mns} is a purely combinatorial task [4].

We have applied two algorithms for the SP quality evaluation. Generally speaking, with the first one we are looking for a maximal matching between the new object and the standards of a class, whereas with the second we look for a minimal difference.

The A1 algorithm realizes a voting procedure. Let j_1, j_2, $\cdots$, j_k be the characteristics, forming a terminal test. Then the mean number of votes, given by the representatives of the class g for the new object s, is calculated by the function:

$$\gamma_g(s) = \frac{1}{m_g - m_{g-1}} \sum_{(j_1, \cdots, j_k) \in T} \sum_{i=m_{g-1}+1}^{m_g} a_{j_1}^{a_{ij_1}} \cdots a_{j_k}^{a_{ij_k}}$$

where

$$a_{j_t}^{a_{ij_t}} = \begin{cases} 1 & \text{if } a_{j_t} = a_{ij_t} = 0 \text{ or } a_{j_t} = a_{ij_t} = 1 \\ 0 & \text{else} \end{cases}$$

The new object s belongs to the class which has given the maximal number of votes—if this maximum exists and is unique. Otherwise, no decision can be taken.

The A2 algorithm finds out the differences between the new object s and the representatives of every particular class with respect to every characteristic. The number of matches defines a "distance" as a weighted sum. The weight of each characteristic is calculated as the part of those terminal tests that contain this characteristic: $p_j = \tau_j / \tau$. The distance is expressed by

$$d_g(s) = \frac{1}{m_g - m_{g-1}} \sum_{i=m_{g-1}+1}^{m_g} \sum_{j=1}^{n} (a_{ij} * a_j) p_j$$

where "*" is the operation:

*	0	1	x
0	0	1	0
1	1	0	0
x	0	0	0

The new object s belongs to the class for which $d(s)$ has a minimal value—if this minimum exists and is unique. Otherwise, no decision can be taken.

It should be pointed out that the task would have been far easier, if the "baseline" vectors were in some sense monotonous. If such was the case, it would have been possible to only use some kind of a very simple measure for distance instead of the terminal tests and the proposed algorithms. Unfortunately, the interrelation between the characteristics of real SP is far more complicated.

4. EXPERIMENTS

As an experiment, we selected a set of payroll program products, using the documentation available at the National Library of programs in Sofia. We designated the "standards" by I, II, ···, XII and determined the baseline vectors by using information obtained from the documentation and from a few interviews with users and experts. We started by defining 30 characteristics. In the final analysis, only 13 appeared to be of any importance for our experiment. Some characteristics were cancelled as inappropriate (e.g., "optimal method"). Others got the same value for all objects (i.e., SP). Out of the 22 characteristics that remained (see corresponding table T_{mns} in Table 1), nine (marked by an asterisk in Table 2) did not take part in any terminal test and are therefore not included in the classification procedure. There is thus no need to assign any value to these nine characteristics for any new SP, this simplifies the task of the expert assigning the values, and significantly reduces the computing time for the classification. The fact that the nine characteristics are cancelled does not mean that they do not influence the SP quality—but only that they do not participate in the algorithm in this particular case.

It is advisable to keep table T_{mns} up-to-date. If some new SP of the same type is implemented on a large scale, it should be included in the table. A recalculation should then follow, and it is quite probable that the set of the terminal tests will change. We carried out numerous experiments to this effect—replacing a baseline vector or adding a new one—but no substantial change has been noticed. This could be explained by several factors, the most important of which is the relative stability of the characteristics of the type of SP that was used. We are at present trying to formulate a number of recommendations that the user should follow, but it is already clear that the maximal number of characteristics when using an IBM/PC/XT should not exceed 28, and the optimal number of "standards" is 8–10.

Table 2.

Number	Characteristic
1	Minimal input
2	Simplicity of input coding
* 3	Input verification
* 4	Machine independence
5	Degree of automation
6.	Applicability when conditions change
* 7	Readability of source text
8	Optimal data organization
9	Rationality
10	Execution speed
11	Product independence
12	Ease of running
* 13	Ease of learning
14	Modularity
* 15	Preciseness
* 16	Reliability
* 17	Availability of common control block
* 18	Common memory used
19	Interface possibilities
* 20	Availability of comments
21	Availability of mnemonics
22	Data protection

In carrying out an experiment, the standard SP were divided into 3 classes: perfect, good, and poor. The 22 characteristics are listed in Table 2. A new SP with the description

(1, 0, 0, 0, 1, 1, 0, 1, 1, 0, x, 0, 0)

is classified as the first class by both A1 and A2, obtaining $y_1 = 3$, $y_2 = 0$, $y_3 = 0$, and $d_1 = 1.027$, $d_2 = 2.306$, $d_3 = 1.928$. In some rare cases, the results obtained by A1 and A2 may be different, due to "x" – s, which are treated differently by A1 and A2.

In further developing the method proposed, it is possible, at least in the case of some of the characteristics, to assign not only 0, 1, and x to each characteristic, but also a number from the set 0, 1, 2, ···, k. Obviously, k should be reasonably small—otherwise our method will begin to resemble the classical methods—with all of their inconveniences. A few other algorithms have been developed in addition to A1 and A2. All methods proposed have been implemented on a mainframe and on IBM/PC.

ACKNOWLEDGMENT

The author is very indebted to V. Angelova for her active participation in this work and in its further development.

REFERENCES

1. Th.P. Bowen, G. B. Wigle, J. T. Tsai, Specification of Software Quality Attributes, Software Quality Evaluation Book, Report RADC-TR-85-37, Vol. 3, Boeing Aerospace Company.
2. F. J. Buckley and R. Poston, Software Quality Assurance, *IEEE Trans. Software Eng.* SE-10, 33–41, 1984.
3. V. V. Lipaev, *Software Quality*, Moscow, Finansi i statistika, 1983. (In Russian.)
4. J. I. Juravlev et al., Recognition and Classification Problems with Standard Teaching Information, *J. Comput. Mathem. Phys.* (Jurnal vichislitelnoi matematiki i matematicheskoi fisiki), 20(5), 1294–1309 (1980). (In Russian.)
5. E. V. Dukova, Asymptotically Optimal Test Algorithms in Recognition Problems, *Cybernetics Problems* ("Problemi kibernetiki"), 39, 165–199 (1982). (In Russian.)