J. New Gener. Comput. Syst. 3 (1990) 1, 47–53 Akademie-Verlag Berlin

A New Method for Software Quality Evaluation

A. Eskenasi, V. Angelova

Institute of Mathematics,
Bulgarian Academy of Sciences, Sofia

Abstract: Existing methods for software quality evaluation are based on hierarchical models, and as a consequence are rather expensive and cumbersome. A new method, based on classification procedures is proposed. Experts determine a set of binary characteristics for a given type of software, describe a few well known program products of this type as baseline binary vectors and break them down into a few classes of quality. When a new product comes it is similarly described, and then eight different algorithms are applied in order to classify the new product into one of the quality classes. An example, illustrating the method, is given, some arising problems are discussed and appropriate heuristics to solve them are proposed.

Key Words: Software Engineering, Quality, Classification, Program Product

Новый метод для оценки качества программного обеспечения

Резюме: Существующие методы для оценки качества пакетов программ (ПП) основываются на иерархических моделях и являются трудоемкими и дорогими. Метод, предлагаемый нами базируется на классификационных процедурах. Эксперты определяют множество двоичных признаков для данного класса ПП, описывают несколько хорошо знакомых ПП в виде двоичных векторов и разбивают их на несколько классов качества. Каждый новый ПП описывается таким же способом, и применяются восемь алгоритмов для его классификации. Дан пример, обсуждаются некоторые возникающие проблемы и предлагаются эвристики для их решения.

Ключевые слова: Технология программного обеспечения, качество, классификация, пакет программ

1. Introduction

By software quality we will understand a set of characteristics of the software product or service which shows their capacity to satisfy certain needs [1], [2]. Software quality evaluation is based on hierarchical models. The quality itself is defined by several factors which represent the user's views on the essential attributes of the product. Each factor is determined by several criteria which characterize programming aspects corresponding to this factor. Each criterion is determined by a few metric elements. In some models there is an additional intermediate level. The evaluation process for a particular program product is the following: experts give an assessment for every element. By using

predefined weights for each particular type of software, weighted sums are obtained which represent criteria values. After that factor values are similarly obtained as weighted sums of criterion values, the end result being the quality value as a weighted sum of the factor values. This method obviously is rather subjective as all weights and the majority of the particular element values are determined by experts. It is also expensive and cumbersome – in [3] the number of metric elements is 257, and in [4] – even higher. It should be kept in mind that such value accuracy (usually one hundredth in the interval [0, 1]) is not necessary in all cases. Very often the user requires only approximate information as to whether the product is perfect, good or poor, while the professional only needs to know how the product ranks in comparison with several already well known products of the same type.

These general considerations rise to an attempt to develop another method for software quality evaluation. Well known tools of recognition theory [5] were used – some of which were further developed. In some cases new ones were proposed. Although the principles of the method have already been explained (e.g. in [6]), it is worthwhile reiterating some of them in order to make the paper more understandable.

2. Quality Model

Let us consider a given type of software product (text editors, payroll programs, games etc.). We determine for this type a set of characteristics – $j_1, j_2, \ldots, j_n$. Each characteristic can be assigned the value 1 if the product does possess the corresponding property, 0, if it does not, and x, if no information is available.

Let us also suppose that several products of the same type are very well known and that the value of each characteristic can be determined for every product (to be called furtheron *standard*). Certainly, it is preferable to minimize the number of x-s. Then, on the basis of the existing opinion of users and professionals, these standards are broken down into several classes, depending on their quality. In practice two (good and poor), or three (perfect, good, and poor) classes are most often determined, but a higher number is also possible. In this way each standard is represented by the vector $E_i = (a_{i_1}, a_{i_2}, \ldots, a_{i_n})$, where n is the number of characteristics and a_{ij} belongs to $\{0, 1, x\}$. Moreover, if s is the number of classes, then E_i belongs to one and only one class K_g, where $g = 1, 2, \ldots, s$. The intersection of these classes is empty. The input data described above can be represented as a table T_{mns}, called "teaching" table (Table 1).

Once created, this table is relatively stable for the particular type of software. Let us

Table 1. Teaching table

		j_1	j_2	$\cdots$	j_n
	E_1	a_{11}	a_{12}	$\cdots$	a_{1n}
K_1	$\vdots$				
	E_{m_1}	$a_{m_1 1}$	$a_{m_1 2}$	$\cdots$	$a_{m_1 n}$
	$\vdots$				
	$E_{m_{s-1}+1}$	$a_{m_{s-1}+11}$	$a_{m_{s-1}+12}$	$\cdots$	$a_{m_{s-1}+1 n}$
K_s	$\vdots$				
	E_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}

now suppose that a *new* product E is given. We determine its baseline vector $E = (a_1, a_2, \ldots, a_n)$, i.e. we determine the value of each characteristic for E. The main idea of the method proposed is that by using table T_{mns} and the description (baseline vector) of E, the new product E is classified into one of the s predefined classes. This supplies us with information about its quality, which in many cases is quite sufficient.

There are two notions in the classification procedures which are essential. 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 (TT) is a test, no subset of which is a test.

We shall call representative sample for a given class on a subset of columns that part of a baseline vector (row of T_{mns}), which in the subtable of T_{mns} constituted by these columns, is met only in the given class. A terminal representative sample (TRS) is a representative sample, no part of which is a representative sample. In Table 3 e.g. {2}, {10, 12} are TT, and (0) in column 10, as well (0) in column (12) are both TRS-s for the class K_2, (whereas (1) neither in column 10, nor in column 12 is a TRS). $\{j_1, \ldots, j_t\}$ is a representation of TT or TRS, where j_1 $(i = 1, \ldots, t)$ are numbers of columns of T_{mns}. To obtain of all TT-s and TRS-s is a problem that has already been solved [7]. After we have found them we apply eight different classification algorithms (A1–A8).

3. Algorithms

One group is formed by A7–A8 which use the so-called voting procedures. A7 uses the set T of all TT (we designate by τ the number of all TT, i.e. $|T|$). In a similar way A8 makes use of the set of all TRS for each class. For example with A7 we calculate:

$$y_g(E) = \frac{1}{|K_g|} \sum_{\{j_1,\ldots,j_t\} \in T} \sum_{E_i \in K_g} F(a_{ij_1}, \ldots, a_{ij_t}, a_{j_1}, \ldots, a_{j_t})$$

for every $g = 1, 2, \ldots, s$. Here $|K_g|$ is the number of standards in the K_g class and

$$F(a_{ij_1}, \ldots, a_{ij_t}, a_{j_1}, \ldots, a_{j_t}) = \begin{cases} 1, \text{ if } a_{ij_r} \neq a_{j_r} \neq x \text{ for every } r \\ 0, \text{ else} \end{cases}$$

Thus, the new product E obtains votes from the K_g class only if for the given TT the values of E and a standard from K_g in the columns, constituting this TT, match fully. If a value of x is present, there is no matching. The role of $|K_g|$ is to obtain a normalized result at the end. E will be assigned to the class having been given the maximum number of votes.

The two other groups of algorithms (A1–A3 and A4–A6) make use of the so-called information weights of the characteristics. The simplest weight is $p_j = \tau_j/\tau$, where $\tau_j = |T_j|$, i.e. the number of TTs, in which the j-th characteristic (column) takes part. But since this weight does not take into account the lengths of the TTs, we introduced a new weight $q_j = 1/\tau \sum_{\nu=1}^{n} \tau_j^\nu/\nu$, where τ_j^ν is the number of TTs with a length of ν which contain the j-th characteristic. Hence, we count separately the TTs with a length of ν, then divide by ν. The final sum is divided once more by τ in order to obtain a normalized result. In that way the characteristics, represented in longer TTs will get smaller q_j in comparison with characteristics represented in shorter TTs. With p_j this difference, which is intuitively clear, is not reflected.

The third weight – r_j, proposed by us [8], is based on TRS and resembles q_j.

The algorithms of the first group (A1–A3) use in a similar way the weights p_j, q_j, and r_j and calculate the so-called information weight of a standard. For example A1:

$$J(E_i) = \sum_{j=1}^{n} a_{ij} p_j \qquad (a_{ij} \neq x)$$

By using the weights thus obtained, we determine the threshold numbers $c_1, c_2, \ldots, c_s$, which delimit the various classes. If a new product E has an information weight $J(E)$ in the interval $(c_{g-1}, c_g]$, then we conclude that E belongs to the class K_g. Unfortunately in some cases it is possible that not all weights of the standards belong to the interval $(c_{g-1}, c_g]$. This particularity will be discussed later on.

The last group of algorithms (A4–A6) calculates a *distance* to a class, by using respectively p_j, q_j, and r_j. For example A4 uses the weight p_j:

$$d_g(E) = \frac{1}{|K_g|} \sum_{E_i \in K_g} \sum_{j=1}^{n} (a_{ij} \ast a_j)\, p_j$$

where again $|K_g|$ is the number of standards in the K_g class and $(a_{ij} \ast a_j) = |a_{ij} - a_j|$ (the value of x is considered as a 0). The new product E belongs to the class for which the distance obtained has a minimal value – if this minimum exists and is unique. Otherwise no decision will be taken.

Certainly, the question arises which algorithm among A1–A8 is the best for each particular teaching table. As far as the whole method has an heuristic character, the choice of the most appropriate algorithm is based on heuristic considerations. Such considerations are particularly necessary if the solutions obtained by the various algorithms are different. It should not be forgotten that the final user must obtain an unique solution regarding the quality class to which the new product belongs. It is not possible to describe here all the heuristics proposed by us and that is why we will give only an example, which will also illustrate our method as a whole.

4. Example

Let us consider software products of the *text editors* type. In Table 2 we describe the characteristics selected by us (name, meaning, how the values of 0 and 1 are assigned). Note that we have reduced the initial number of characteristics to 15 for the sake of clarity. In the original experiments 28 characteristics were determined. In this paper we have omitted *profile*, *screen options*, *keyboard options*, *memory used*, *portability to other operating systems*, etc. Most of the ignored characteristics are featured by all the text editors considered, therefore they get the value 1 and do not further influence the procedures of the method. Obviously they have to be checked every time a new product is classified, because if it does not contain one or more of these properties, the teaching table T_{mns} must be modified accordingly. In our example we consider them in the way already mentioned, divide them into two classes (good and poor) and put the result obtained into Table 3. This table has 38 TTs. Furthermore, there are 86 TRSs for the K_1 class and 28 TRSs for K_2. Characteristics 1 and 14 have the highest weights, and 2 and 3 – the smallest. In Table 4 are the information weights $J(E_i)$, calculated with p_j, q_j, and r_j respectively. It is obvious that the information weights of the standards calculated by using p_j are not well distributed among the classes. ($J(E_4)$ has a higher weight – 0.65 – than $J(E_2)$ with 0.64 and $J(E_3)$ with 0.54). Therefore $A1$ is not expected to give good results. Indeed, the new object $E = (1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1)$ is grouped into class K_1 by $A1$ and into class K_2 by the remaining seven algorithms $A2$–$A8$.

Table 2. Characteristics set

#	name	meaning	values
1	macros	capability for creation and updating of macrodefinitions	1 – yes 0 – no
2	tabulation	capability for creation and updating of tabulation positions	1 – yes 0 – no
3	masks	capability for input mask definition	1 – yes 0 – no
4	windows	capability for window opening when editing	1 – yes 0 – no
5	basic commands	availability of copy, move, search, and replace commands	1 – all 0 – some
6	advanced commands	capability for creation of user's commands	1 – yes 0 – no
7	cursor control	capability for cursor control	1 – big 0 – poor
8	additional commands	availability of fill, justification, centering, and split commands	1 – all 0 – some
9	carry over	availability of automatic carry over of a word to the next line	1 – yes 0 – no
10	register exchange	capability for register exchange	1 – yes 0 – no
11	additional files	availability of commands for inclusion and output of additional files	1 – all 0 – some
12	stack	capability for stack processing	1 – yes 0 – no
13	buffer	availability of a maximal buffer	1 – >64 K 0 – <64 K
14	openness	capability for interface with other programs	1 – yes 0 – no
15	laconicism	capability to use menus, acronyms, keys, etc.	1 – good 0 – poor

Table 3. Text editors – teaching table

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K_1	E_1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
	E_2	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
	E_3	0	1	1	1	1	0	0	1	0	1	1	1	1	1	0
K_2	E_4	1	0	0	0	1	1	1	1	1	1	1	0	1	0	0
	E_5	0	0	0	1	1	0	1	0	0	0	1	0	0	0	1
	E_6	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

Table 4. Information weights of standards

	p_j	q_j	r_j
$J(E_1)$	0.83	0.85	0.86
$J(E_2)$	0.64	0.70	0.83
$J(E_3)$	0.54	0.60	0.65
$J(E_4)$	0.65	0.59	0.59
$J(E_5)$	0.37	0.36	0.35
$J(E_6)$	0.17	0.19	0.21

Hence, for this particular table, it is necessary to exclude the $A1$ algorithm just after the initial procedures and prior to the first classification of the new product.

The other heuristic considerations take into account the number of standards in the different classes, the number of TRSs in the different classes, the number of votes for the standards themselves, the distances among the standards and the other classes, the number of matchings between the descriptions (baseline vectors) of the standards, etc. These considerations remain transparent for the final user. When he/she starts classification (evaluation) of *his/her* new product, all considerations have already been made either by the experts, who have determined the characteristics, standards, and the teaching table, or by the program, implementing the method described. This program has been realized on IBM/PC in MS DOS, as well as on a VAX compatible minicomputer. Using this program we have carried out multiple experiments with different types of program products. The results are encouraging. This was the reason to design a program product which is now on the market.

Acknowledgement

The authors are much indebted to Mr. V. Jelesov. The example in Table 2 is based on his investigations.

This work has been partly supported by the Ministry of Culture, Science and Education under contract Num. 36.

References

[1] Липаев В. В.: Качество программного обеспечения. Финансы и статистика, Москва 1983г.

[2] Buckley, F. J.; Poston, R.: Software quality assurance. IEEE Trans. Software Engrg. (1984) 1, 33–41.

[3] Общая методика оценки качества программных средств. Межправительственная комиссия по вычислительной технике, СЭВ бюллетень 1 (37), Москва 1988г.

[4] Bowen, T. P.; Wigle, G. B.; Tsai, J. T.: Specification of software quality attributes. Software Quality Evaluation Book Report # RADC-TR-85-37, Vol. 3; Boeing Aerospace Company, 1985.

[5] Дмитриев А. Н.; Журавлев Ж. И.; Кренделев П.: О математических принципах классификации предметов и явлений. Дискрет. анализ (1986) 7, 3–11.

[6] Eskenasi, A.: Evaluation of software products quality by means of classification methods. J. Systems Software (1989) 10, 216–219.

[7] Константинов Р. М.; Королева З. Е.; Кудрявцев В. Б.: О Комбинаторно-логический подходе к задачам прогноза рудоносности. Проблемы кибернет. 31 (1976), 5–33.

[8] Angelova, V.: Weight estimations of the features. In: Mathematics and Mathematical Education, Proceedings of the 16th Spring Conf. of the Union of the Bulgarian Mathematicians, Sofia, 1987; 301–305. (in Bulgarian)

Received: May 2, 1989

Authors' address:
Dr. A. Eskenasi,
Dr. V. Angelova
Institute of Mathematics,
Bulgarian Academy of Sciences
Acad. G. Bonchev str., bl. 8
1113 Sofia
Bulgaria

Avram Eskenasi

was born in Varna (Bulgaria) in 1946. He received the NSc in mathematics in 1969 and the PhD in mathematics in 1978, both from the University of Sofia. He is currently head of the Software Engineering Department at the Institute of Mathematics of the Bulgarian Academy of Sciences. He is the Bulgarian representative to IFIP TC 2 on Programming as well as the Bulgarian co-ordinator to the software technology project (KNP-8) within the frame of the new generation computer systems developments in the CMEA.