Table of Contents

19TH INTERNATIONAL CONFERENCE WITH SUMMER SCHOOL
INFORMATION TECHNOLOGIES AND PROGRAMMING
Sofia, Bulgaria, June 26 — July 2, 1994

The Problem of Objectivity in Software Quality Evaluation

Avram Eskenazi, Institute of Mathematics, Sofia
E-mail: ESKENAZI@BGEARN.BITNET

1. Introduction

Boehm [1,2] was the first at the beginning of 70-ies to pose and systematically consider the problem of software quality. During the next several years a few models have been created based on the same principles. The software (SW) quality is represented as an hierarchical structure of characteristics. As far as the quality is most often defined as the degree of satisfaction of users' needs and requirements, in each model it directly depends on users' views about its particular characteristics (usually called **factors**). The latter depend on a set of program characteristics, called **criteria**. Further in this hierarchy the various models propose either one, or two levels. But in all cases there are at the lowest level some kind of **elementary characteristics**.Sometimes there is an intermediate level called the **metrics** level. The word "metric" in this case only means a protocol of the results of the measurement or the evaluation of several close (by content) to each other elements. The SW quality evaluation procedure of such a model [3,4] is approximately the following. Experts evaluate (or in rare cases measure) all **rating elements** (elementary characteristics). By using weighed sums the values of the criteria are calculated. (Certainly, if the level of metrics exists, this level is calculated similarly). Then by using again weights predefined by the experts the values of the factors are calculated. Finally, a single value - the weighed sum of all factors - is got and this is the measure of the SW quality itself. Usually, values and weights are normalized, so the intermediate and final results are in a predefined interval, e.g. [0-1].

Obviously, such a procedure has its drawbacks - e.g. it is cumbersome: the number of elements in [3] is more than 300, and in [4] - 257. But probably the most dangerous disadvantage of this type of models is their subjectivity. The aim of our further analysis is to check whether this subjectivity can be reduced on behalf of the inclusion of some objective procedures in given steps of the respective methods.

2. The M-model

As already mentioned, the hierarchical methods principally resemble each other. As [4] is relatively new, balanced, well developed and rather well documented, we will further on rely on it for our considerations. For brevity, we will call this model the M-model. The end conclusions we will finally reach will be valid for the other hierarchical models, too.

The M-model has been developed by experts from the former European COMMECON countries. Its basic idea provides the evaluation of not only the final SW product, but also of its intermediate states. The factors are: flexibility (**F**), correctness (**C**), reliability (**R**), maintainability (**M**), ease of use (**U**), efficiency (**E**). (The original acronyms respectively are Г, K, H, C, У, E). There is an exact definition of the hierarchy, including criteria, metrics and rating elements (RE). A notation is

used of the form ***Fmmii***, where ***F*** is the code of the factor, ***mm*** - the number of the metric inside this factor and ***ii*** - the number of the RE. For each factor and each phase of the SW life cycle an hierarchy diagram is given down to the level of metrics, incl. the numbers of the metrics. A classification of the methods to determine the RE-s' values is developed. It is important to note that only 18 RE-s have to be determined by means of calculations, other 20 - either by calculations or by means of experts' estimations and 3 - by means of experts' estimations or through measurements. The remaining 216 RE-s are determinable only by means of experts' estimation. This is about 84% and unambiguously confirms the high subjectivity of the model and its respective method.

3. Metrics

In the Software Engineering literature the term "metric", has been used for the last twenty years in one more sense. According to [5], a "measure" is said to be a quantitative measure of one ore more quality characteristics. A metric in the Software Engineering is usually understood as a method for finding out the corresponding measure. It is clear that we should make a difference between this metric and the metric (protocol) defined in the M-model or other hierarchical SW quality models (see 1. above). That's why from now on till the end of this paper we will call a metric the notion just defined (sometimes for clarity we will precede it by "classical"). The metric in the sense of the M-model will be denoted as a "M-metric".

If we ignore a very small number of classical metrics concerning the characterization of natural language texts (e.g. [6] - 1948 and [7] - 1962) and later on used to measure the clarity and understandability of SW documentation, the basic metrics appeared in the mid 70-ies, proposed by Halstead [8], McCabe[9], McCall[10]. Further on several tenths of metrics appeared. For example, in [11] on the base of 124 references 75 different metrics were classified. In [12] 50 metrics were very systematically described. We should note that the intersection of the first 75 and the other 50 is not very large. Besides these, other metrics are well developed and known - e.g. [13]. We are not in a position to include in our study new metrics, particularly concerning the object oriented programming as for example "Depth of inheritance tree", "Response of a class", etc.[32] because of the lack of sufficient information.

4. Setting the problem for increasing the evaluation objectivity.

The question about how to reach a greater objectivity can be formulated as follows: is it possible to find out a link between the great number of existing metrics and the elements of the M-model ? More precisely: is it possible for a part of the RE (and particularly these 216 of them whose values are determined through expert estimation) to get their values from one or more metrics ? In other words, if $M_1, M_2, \ldots, M_k$ are metrics (of Halstead, McCabe, McCall, etc.) and V_p is the result (obviously a number) of M_p (p=1,2, ...,k) for a given program, we would like to get the value of a given RE ***Fmmii*** for the same program as

$Fmmii = F(V_1, V_2, ..., V_k)$.

In particular cases it is also possible to have

$Fmmii = F(V_a)$,

i.e. the value of RE could be obtained as a function of a single metric M_a.

Let's consider an example. In [14] the metric of DeYoung-Kampen (DYK) is described. According to it the readability R of a program is determined as follows:

$R = 0.295\ VAR - 0.499\ NSL + 0.13\ CYCLO$,

where ***VAR*** is the average normalized length of variables

NSL is the number of lines containing statements

CYCLO is the total number of branches + 1 (McCabe's cyclomatic number)

If we analyze the M-model we will come to the maintainability factor (**M**), then, following the tree - to the criterion "simplicity", then, on the next level to the metric **M10** - coding simplicity, and finally, on the last level we will discover three RE-s:

M1001 - whether a high level programming language is used

M1002 - total number of branches (***A***)

M1003 - total number of executable statements (***B***).

It is clear that **M1002** practically matches ***CYCLO*** and **M1003** - ***NSL***. **M1001** has to do with DYK as far as DYK has been created for high level programming languages. But, at this place in the M-model the variables' length is not considered. Due to typographical impreciseness in [4] one cannot say to which RE exactly refers the formula ***(1-A/B)***, but anyway it is clear that when increasing the number of branches the simplicity of the program is decreased. The same applies to DYK, where the readability of the program decreases when the number of branches increases. Consequently, it is possible in the M-model to obtain **M10** by using DYK, whose reliability and objectivity may be considered as higher than that of the respective procedure in the M-model concerning **M10.** However, we should not ignore that two of the RE-s (**M1002** and **M1003**) are calculable. Nevertheless, obtaining ***R*** of DYK is more objective, because the coefficients in the formula are the result of a regression analysis, whilst nothing is said about the weights of **M10ii** in the M-model.

What else implies this example ? First of all, one can notice, that an M-metric - "coding simplicity" - exists also in the subtree of the flexibility factor (**F**), "modifiability" criterion **F10**. **F10** depends on a large number (7) RE-s, all they are experts estimated, even those which evaluate the conditional and unconditional branches. Probably, one could try to evaluate **M10** by using DYK, but here this would be far more difficult because of the big number of RE-s, not participating in the DYK formula. On the other hand the effect would be more substantial, because an entirely subjective estimate would be replaced by an objective one. In this sequence of thoughts it is quite natural to ask whether it is really necessary to look for several metrics (as proposed above) in order to replace a single RE. We should not forget that the main aim of the M-model is to find a final value for the quality at level 0 and that the intermediate values on the other levels have only an

intermediate character. Hence, it would be quite acceptable if ***F(V1,V2,...,Vk)*** gave a value for the M-model not on level 4 (RE) but on level 3 and why not even on a higher level. Thus, by the way, we justify the approximate correspondence between **M10** and DYK that we have already established.

From a general point of view such a strategy seems to be more promising, as far as level 4 (RE) is the level of the elementary (atomic) characteristics, whilst the classical metrics usually combine several parameters corresponding to such atomic characteristics.

5. Establishing the correspondence.

After the last conclusion we think that the most reasonable strategy is to consider in parallel the entire hierarchy of the M-model together with a representative set of metrics, e.g. as systematized in [12]. First of all we will take those calculable RE or M-metrics which are calculable and we will try to find corresponding classical metrics.

One of them is related to the efficiency factor (**E**). The criterion is "exactness of the calculations" and it is determined by a single M-metric - **E02** with the same name. There is also a single RE determining this M-metric - **E0201**, "number of digits after the decimal point in the result". This RE is either calculable, or experts estimated. One can find a distant relation with the metric described in [10]. In the latter more characteristics are taken into account, but the majority of them are estimated quite subjectively.

The largest group of calculable RE-s belong to the reliability factor (**R**) subtree. Let's take **R05** - "testing completeness" as the sole M-metric determining the criterion "working capacity", as well as **R07** "tools for error detection" as one of the M-metrics determining the criterion noise-resistance".

Several metrics measuring the testability are known, e.g. those described in [20,21,22,23, 24]. In all cases the complexity of the program is described (from the point of view of the branches and the possible paths in the program, as well as in [24] - on a macro-level, i.e. considering the modules and the links between them). On this basis the efforts (actually the number of operations) to perform the testing are estimated, as well as the expected efficiency, i.e. how exhaustively the program will be tested.

Unfortunately, it is not possible here to make a comparison with the criterion "testability" of the factor "correctness" (**C**). The reason is that the M-metrics **C10, C11, C12, C13**, which determine this criterion are not described at the lowest level (that means that no RE-s are given for them). Hence, they remain if not undefined, then at least - unclear.

Consequently we can only deal with **R05**. The two calculable RE-s - **R0503** and **R0504** are quite specific and have nothing to do with the already mentioned classical testability metrics. They try to measure the degree of testing of logical blocks and modules performed (and not the expected effort) as the ratio of the tested number of blocks (or modules) against the respective total number

The two other RE-S - **RE0501** and **RE0502** are still farther and a little bit fuzzy - hence they should be also left.

Let's consider **R07**. Generally speaking in the M-model the reliability is defined first of all as an estimate of the features created for filtration of non-valid data, for proper reactions to hardware and software failures, for recovery and program restart and for testing already performed.

The classical metrics described in [25,26,27,28,29,30] aim at predicting the operation reliability (e.g. as the mean time between two failures or as the average number of expected errors for a given time period). This is done on the basis of some kind of extrapolation (We should note that [25] refers only to the development process). Having this in mind we establish the following common characteristics of both approaches.

R0701 evaluates the probability for an error-free operation according to the formula

$P = 1 - Q/N$,

where Q is the number of registered errors and N - the number of experiments. There is a similarity with [29], where the measure is to some extent the opposite - it is the ratio between the number of successful trials and their total number. Therefore the replacement of **R0701** with the metric of [29] is useless.

R0702 predicts the total number of errors through the formula

$B = K_o * R/1000$,

where R is the total number of statements in the program, K_o is a factor of the number of errors for 1000 statements, which has to be determined by experts (recommended values in the interval (0.25 - 10). This approach resembles the metric described in [25], where the analog of R is obtained in a more complicated way (practically it is proportional to the volume V of the program as determined by Halstead [8]) and K_o is 1/3 for the validation process and 4/3 for the whole life cycle. Actually for **R0702** the value of K_o is up to the experts. They could determine it by using a classical metric (particularly that of Halstead). We cannot state which approach is the better, but anyway, in the metric of [25] things seem to be more fixed and, hence, less subjective. As far as the time factor is concerned, there is hardly any difference. The M-model does not say anything about the period of prediction but we can accept that a given period of the life cycle is considered.

Here we are obliged to make the following remark. During the elaboration of the M-model we happened to have numerous contacts with some of the acting experts (particularly the Bulgarian ones). We have at our disposal various working materials, some of them very detailed. Nevertheless we rely exclusively on the official version as described in [4].

R0703 is an indicator of the program stability against damaging or distorting impacts. It is determined by the formula

$P(Y) = 1 - D/K$,

where D is the number of trials in which the impacts have led to failures and K - the total number of trials with damaging impacts. This resembles very much the metric described in [28]. Still in the

latter all possible input data sets N are expected to be encompassed and m to be the number of input data sets reducing incorrect outputs. Consequently, the formula

$1 - m/N$

is the definition of this metric. Therefore D and m are identical, but K and N are not - in **R0703** it is known in advance that each input data set is a damaging one. From a practical point of view **R0703** seems to be more applicable than the metric of [28].

R0704 is the mean recovery time. The respective formula contains the ratio of the recovery time and the number of failures, but without saying what is the interval containing this number. A similarity can be found with the metric of [31]. The metric there is calculated as $T/(T+F)$, where T is the mean time until the failure and F - the mean recovery time. It seems that the metric is more clear and unambiguous, but nevertheless, after some clarification and transformation the **R0704** formula can be brought to the metric.

We cannot find any correspondence between a classical metric and the RE-s **R0705** and **R0706**. By the way no formula is given for **R0706,** but probably it is obtained through transformation of **R0705** by taking into account the number of trials. (**R0705** concerns the duration of the data input - data output transformation and **R0706** - the mean value for the same transformation.)

We still have a few RE-s obtained by experts estimation, for which, more or less, a correspondence with a classical metric can be established.

U0201 is the RE "completeness and understandability of the documentation" in the frame of the M-metric **U02** "documentation to be assimilated" and the criterion "ease of assimilation". Each of the metrics in [6,7,15,16] could be used for the objective obtaining of a value of **U0201**, unfortunately only for the part "understandability", yet not for the "completeness". More general is the metric described in [17]. It would have given not only the value of **U0201**, but also that of **U0501** - "evaluation of the style" and of **U0504** - "clarity of the logical structure". But we have to be careful with this classical metric, because in its procedure there are elements of an obvious subjectivity (there are four parameters which are supposed to be estimated by experts on a five-degree scale - i.e. we come back to the usual subjectivity of the M-model).

We have already discussed **F10**. **F08** is "dependence on the basic software" in the frame of the criterion "mobility". **F08** is determined by three RE-s: **F0801** - "use of dedicated programming languages", **F0802** - "dependence of the program on the operating system" and **F0803** - "dependence on other dedicated software". Here two metrics of McCall might be applied, particularly that one concerning the software independence [10]. Two of the four parameters of this metric fully correspond to **F0801** and to **F0802**, and a third one - to **F0803** to some extent. It is true that no automatic tools are available for the determination of these four parameters' values. But as far as they are the result of simple counting, they might be considered as fully objective. The conclusion is that **F08** might obtain an objective value by using the McCall software independence metric.

We have found one more correspondence for the "flexibility" factor. The M-metric **F14** "modules' independence", which is part of the "modifiability" criterion, is determined by five RE-s. We consider [18], where a metric is described, measuring how strongly the change of a variable in a module affects the other modules. The conclusion is that **F1401** "transfer of control information by means of parameters", **F1402** "transfer of input data by means of parameters" and **F1403** "transfer of results between modules" can be far more objectively and formally obtained by using the metric just mentioned. Although **F14** takes into account two other RE-s and [18] - a few other characteristics, it is possible to get an objective estimate for at least an essential part of **F14** trough [18]. Another solution is to modify **F14.**

Concerning the flexibility a correspondence with the M-model might be sought even at level two - with the "mobility" ("portability") criterion. To this end we consider the Gilb metric described in [19]. The respective formula is very simple - the ease of portability P_s is determined as

$$P_s = 1 - (E_t / E_r),$$

where E_t are the resources needed to move the program system to the target environment and E_r - the resources needed to create the program system for the resident environment. Unfortunately there is here some fuzziness - resources are defined too generally - manpower, time, machine resources and no formal procedure is defined for their evaluation.

6. Conclusion

Let us summarize the results obtained. As is shown above, for no more than 10 calculable RE-s a correspondence with one or more classical metrics might be sought. For the majority of these ten RE-s the link is not close even in the case when an attempt is made the classical metric to be applied at the higher level of the M-model (i.e. the level of the M-metrics). In two to three cases the RE estimation could become more objective by using a classical metric. In other two to three cases this could hardly be achieved. We also established that in other about ten cases experts estimated RE-s might be evaluated trough classical metrics. In some cases (e.g. for **F08**, **F14** and even for the "mobility" criterion) this could be achieved at level three, or even two of the M-model.

On the other hand there are lots of classical metrics which cannot be linked to any of the elements of the M-model because of the very complicated relationship (such as the multiple program complexity metrics - they have something to do with the "maintainability" factor and the "structurness" and "simplicity" criteria).

All this proves that the M-model, as well as the other hierarchical software quality models have already reached the limit of their objectivity, probably due to their nature. Hence, if we want to increase the evaluation objectivity (which is by no means necessary), we have to look for approaches based on different principles. We made an attempt in this direction [33].

This work has partly been supported by the Ministry of Science and Education under contract I24.

References

1. Boehm B.W., Software and its Impact: A Quantitative Assessment. Datamation, Vol.19, No.5, May 1973,p.49-59.

2. Boehm B.W., Software Engineering. IEEE Trans.on Computers, Vol.C-25, No.12, December 1976, p.1226-1241.

3. Bowen T.P., G.B.Wigle, J.T.Tsai, Specification of Software Quality Attributes - Software Quality Evaluation Guidebook, RADC-TR-85-37, Vol.III, 1985.

4. Общая методика оценки качества программных средств. МПК по сотрудничеству социалистических стран в области ВТ, Координационный центр, Бюлл. 1(37), М.1988. (A general methodics for software quality evaluation. COMMECON Intergovernmental Commission on Cooperation in Computing Technologies, Bull.1(37), Moscow, 1988, in Russian).

5. European Organization for Quality Control - Glossary of Terms Used in the Management of Quality. Bern, 1981.

6. Dale E., Chall J.S., A Formula for Predicting Readability. Educational Research Bulletin, Vol.27, Feb 1948, p.221-233.

7. Gunning R.,Technique of Clear Writing. New York, McGraw-Hill, 1968.

8. Halstead M.L., Elements of Software Science. New York, Elsevier, 1977.

9. McCabe T.J., A Complexity Measure. IEEE Transactions on SE, Vol.SE-2, No.4, Dec.1976, p.308-320.

10. McCall J.A., P.K.Richards, G.F.Walters, Factors in Software Quality. Vol.1: Concepts and Definitions of Software Quality. Vol.2: Metric Data Collection and Validation. Vol.3: Preliminary Handbook on Software Quality for an Acquisition Manager. Springfield, Va, NTIS, AD-A0 49014, Nov.1977.

11. Cote V.,P.Bourque, S.Oligny, N.Rivard, Software Metrics: An Overview of Recent Results. J of Systems and Software, Vol.8 No.2, March 1988, p.121-131.

12. Hoecker H.,W.Itzfeldt, M.Schmidt, M.Timm, Comparative Description of Software Quality Measures. GMD Studien No.81, March 1981.

13. Rechenberg P., Ein neues Mass fuer die softwaretechnische Komplexitaet von Programmen. Informatik Forschung und Entwicklung, Band 1, Heft 1, 1986, p.26-37.

14. DeYoung G.E., G.R.Kampen, Program Factors as Predictors of Program Readability. Proc. Computer Software and Applications Conference, IEEE, 1979, p.668-673.

15. Kincaid J.P., J.A. Aagard, J.W. O'Hara, L.K.Cottrell, Computer Readability Editing System. IEEE Trans on Professional Communications, Vol. PC-24, No.1, March 1981, p.38-41.

16. Steiwer L., Ausarbeitung von Lesbarkeitsformeln fuer die deutsche Sprache. College d'Enseignement Moyen et Professionel du Nord, 1976.

17. Steinbach I., I.Langer, R.Tausch, Merkmale von Wissens- und Informationstexten im Zusammenhang mit der Lerneffektivitaet. Zeitshcrift fuer Entwicklungspsychologie und Paedagogische Psychologie, Vol.IV, Heft 2, 1972, p.130-139.

18. Yau S.S., J.S.Collofello, Some Stability Measures for Software Maintenance. Proc. Computer Software and Applications Conference, IEEE, 1979, p.674-679.

19. Gilb T., Software Metrics. Cambridge, Mass, Winthrop Publishers, Inc, 1977.

20. Mohanty S.N., M.Adamowicz, Proposed Measures for the Evaluation of Software. Microwave Research Institute of the Polytechnic Institute of New York, Proc. Symposium of Computer Software Engineering, 1976, p.485-497.

21. Paige M., A Metric for Software Test Planning. Proc. Computer Software and Applications, IEEE, 1980, p.499-504.

22. Pimot S., J.C.Rault, A Software Reliability Assessment Based on Structural and Behavioral Analysis of Programs. Proc 2nd Intl Conference on Software Engineering, 1976, p.486-491.

23. Woodward M.R., D.Hedley, M.A.Hennel, Experience with Path Analysis and Testing Programs. IEEE Trans. on SE, Vol SE-6, No.3, May 1980, p.278-286.

24. Yin B.H., Software Design Testability Analysis. Proc. Computer Software and Application Conference, IEEE, 1980, p.729-734.

25. Ottenstein L.M., Quantitative Estimates of Debugging Requirements. IEEE Trans. on SE, Vol.SE-5, No.5, 1979, p.504-514.

26. Musa J.D., The Measurement and Management of Software Reliability. Proc. IEEE, Vol.68, No.9, Sept.1980, p.1131-1143.

27. Remus H., S.Zilles, Prediction And Management of Program Quality. Proc. 4th Intl. Conference on Software Engineering, 1979, p.341-350.

28. Nelson E.C., A Statistical Basis for Software Reliability Assessment. TRW Software Series, TRW-SS-73-03, IEEE, Long Beach, Ca., June 1977.

29. Hecht H., Measurement, Estimation, and Prediction of Software Reliability. Software Engineering Techniques, Infotech Intl., Maidenhead, Berkshire, England, 1977.

30. Schick G.J., R.W.Wolverton, An Analysis of Competing Software Reliability Software Models. IEEE Trans. on SE, Vol.SE-4, No.2, March 1978, p.104-120.

31. Littlewood B., How to measure Software Reliability and how to... Proc. 3rd. Intl. Conference on Software Engineering, Atlanta, Georgia, May 1978, p.37-45.

32. Chidamber S.R., C.F.Kemerer. Towards a Metrics Suite for Object Oriented Design. SIGPLAN Notices, 26(11), Oct.1991, p.197-211.

33. A.Eskenasi. Evaluation of Software Quality by Means of Classification Methods. J. of Systems and Software, vol.10, No.3, October 1989, p.213-216.