CREAT. MATH. INFORM. **27** (2018), No. 1, 31 - 35
Online version at http://creative-mathematics.ubm.ro/
Print Edition: ISSN 1584 - 286X Online Edition: ISSN 1843 - 441X

An adaptable software quality model

MARA HAJDU-MĂCELARU and IOANA ZELINA

ABSTRACT. Just about everything changes, and so should a software quality model in order to provide a better understanding of quality for any software product to which this model is applied. Due to the diversity of software products, a software quality model pattern that can be adapted to different software products and provide an accurate evaluation of the overall software quality is necessary. This paper presents an adaptable software quality model which is intended to be used for the quality evaluation of different software products.

1. INTRODUCTION

Measuring and evaluating software quality has become a fundamental task. Many models have been proposed to support stakeholders in dealing with software quality [14], [15]. However, in most cases, quality models do not fit perfectly the target application context. Since approaches for efficiently adapting quality models are largely missing, many quality models in practice are built from scratch or reuse the high-level concepts of existing models [15]. The early designs of quality models have followed a hierarchical approach in which a set of factors that affect quality are defined, with little scope for expansion [14]. The first quality models were published by Boehm et al.in 1976 [4] and McCall, Richards and Walters in 1977 [17]. These models used hierarchical decomposition in order to decompose quality into simpler characteristics and sub-characteristics that were easier to manage. These models were the predecessors of ISO 9126 (ISO/IEC, 2001), which in turn was the predecessor of ISO 25010 [18].

2. ADAPTABLE SOFTWARE QUALITY MODEL

The aim of the adaptable software quality model is to allow project managers, developers and the entire team involved in the development of a software product, to adapt the software quality model to their needs, so that, using this model, they would be able to evaluate the quality of the software product and re-use the same model for other software products. The first step to create such a model, is to define the type of users that will be involved in the evaluation of the software quality for a specific projec (project manager, testers, developers). As we know, all models are based on factors that affect the software quality [2]. Several quality models have been defined in the literature and all of them contain a set of factors and sub-factors that the author of the model considers important to be taken into consideration [1], [3], [11], [12]. But these factors stemming from a particular software quality model may not be of the same importance for all platforms. The factors that are important on mobile environment may not be important on desktop or web environment for example. We have here a first type of adaptation of the factors that are involved in the quality model, the adaptation according to the application type: web app, desktop app, mobile app. Based on the application type, a list of factors recommended for that type of application is provided. For each type of application some characteristics

Received: 12.03.2018. In revised form: 13.03.2018. Accepted: 20.03.2018
2010 *Mathematics Subject Classification.* 68N30, 68N99.
Key words and phrases. *software quality model, adaptable quality model, software quality tool.*
Corresponding author: Mara Hajdu-Măcelaru; macelaru.mara@gmail.com

are more important than the others and should be taken into consideration. The software quality attributes list is different based on the application type so the user will be able to agree, or modify the list if he considers he needs to remove or add factors. The recommended list will be based on studies made in the literature related to which factors are more important for the chosen type of application.

Because quality is a very generic term and depends on the specific of the software product that is developed, an adaptive quality model will be very useful. For example, we may have software products in which the end-user opinion is more important (such as a scenario whereby a product is made on request by a user). We may have situations in which the user opinion is not important compared to that of the developer, for example an application created for aerospace, in which the quality level measured by the developer should be very high, as the errors or delays may be fatal in real time. Again we see the need to have an adaptive model depending on which is the most important actor in evaluating the quality. For this purpose we use a group evaluation criteria based on importance. By default we will have two groups: team involved and final users and the importance will be $1:1$ meaning both groups' opinion have the same importance. Those examples show models that use software quality factors based on application type, and adapt the evaluation of the quality based on group type and its importance. Most of the existing models use the distribution of the quality factors /sub-factors on levels, layers or phases of development [5]. Those models are not suitable for an agile development process where specifications are poor and changes are made at smaller intervals of time. In this case, it will be a big effort to evaluate quality factors after each change, so a solution for this issue no matter the process model implemented is needed. For this case, an adaptive model can be created following five steps:

1. Choose the application type and based on this type a list of factors $\{f_1, f_2, \dots, f_n\}$ will be defined.

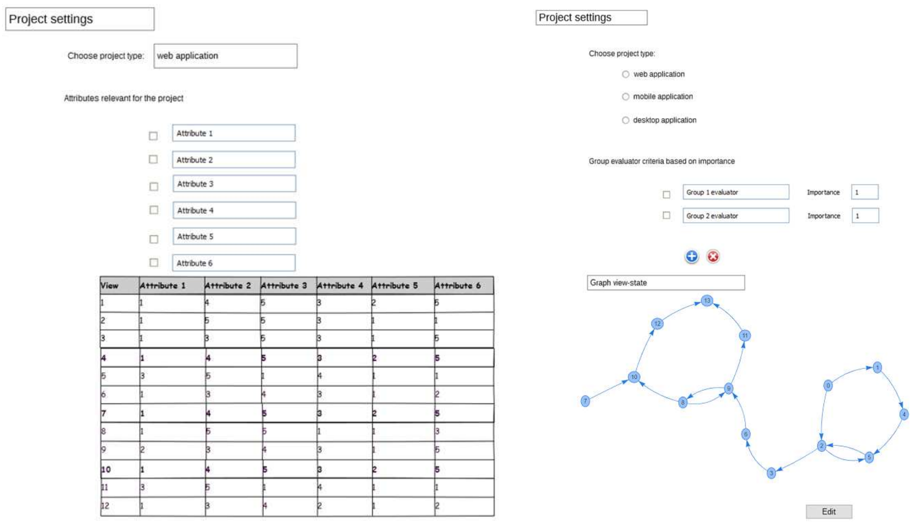
2. Choose the evaluation groups and their importance $\{(g_1, p_1), (g_2, p_2), \dots, (g_m, p_m)\}$, where g_i are the evaluation groups and p_i their priority, $i = \overline{1, m}$.

3. The project manager will have to add all the state-views and the correlations between them. For each state-view, the project manager will have to define a priority related to the specifications. Every application has functionalities that are more important/critical than others, so, depending on the specifications, each group will have to define a priority for each functionality. Also, for each state-view the manager will have to define which attributes from the list defined in Step 2 need to be applied, and their importance. As a result, the view-state functionality graph $G = (V, E)$ is created, where $V = \{v_1, v_2, ..., v_k\}$ is the set of the view-states and the edges between the vertices are given by the relations between those view-states, those relations being defined by the links in the application.

4. Add for each view-state a priority related to the specifications from 1 to 5, 1-very low importance, 2-low as importance, 3-medium importance, 4-high importance, 5-very high importance. These priorities define each view's importance for the project, as a functionality. Every project has some main functionalities that are more used, and some functionalities that are not used very often by the end user.

5. Add for each view-state list of attributes the priority that should be taken into consideration for that view-state: 1-very low importance, 2-low as importance, 3-medium importance, 4-high importance, 5-very high importance (see Figure 1).

For each attribute a method of evaluation is set. Some metrics are provided by default with a description scheme and reference for a better measurement, but the project manager can add more metrics or delete the default ones based on the needs and project particularities. Also, the project manager can associate test plans to be filled to measure the specific attributes. For each attribute he will define a set of metrics/test plans, their



Project settings

Choose project type: web application

Attributes relevant for the project

- ☐ Attribute 1
- ☐ Attribute 2
- ☐ Attribute 3
- ☐ Attribute 4
- ☐ Attribute 5
- ☐ Attribute 6

View	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6
1	1	4	5	3	2	5
2	1	5	5	3	1	1
3	1	3	5	3	1	5
4	**1**	**4**	**5**	**3**	**2**	**5**
5	3	5	1	4	1	1
6	1	3	4	3	1	2
7	**1**	**4**	**5**	**3**	**2**	**5**
8	1	5	5	1	1	3
9	2	3	4	3	1	5
10	**1**	**4**	**5**	**3**	**2**	**5**
11	3	5	1	4	1	1
12	1	3	4	2	1	2

Project settings

Choose project type:

- ○ web application
- ○ mobile application
- ○ desktop application

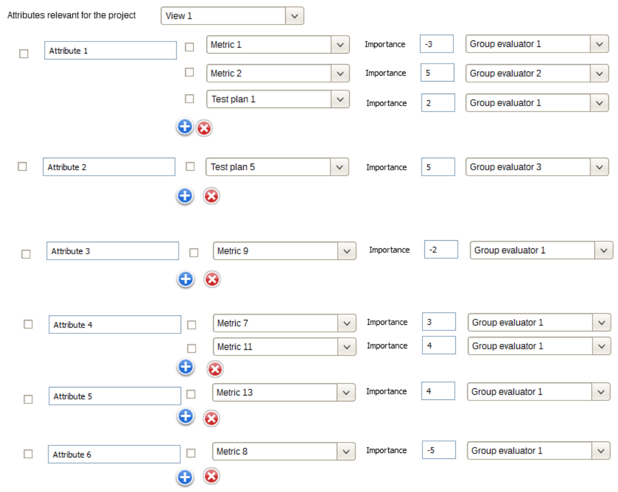
Group evaluator criteria based on importance

- ☐ Group 1 evaluator Importance 1
- ☐ Group 2 evaluator Importance 1

Graph view-state

FIGURE 1. Interface 1 - adaptable software quality

importance to the attribute evaluated, and the evaluation group in charge of filling the test plan, or compute the metric. The importance value is: $+1$ or -1 very low importance, $+2$ or -2 low as importance, $+3$ or -3 medium importance, $+4$ or -4 high importance, $+5$ or -5 very high importance. The mapping between the attributes of the software quality model and the metrics is described by associating a list of related and highly related metrics with each quality attribute. For each metric, the way it affects the attribute, directly or indirectly, is known. In case of positive importance value, the metric affects directly the attribute, otherwise, in case of negative importance value, the metric affects indirectly the attribute.



Attributes relevant for the project: View 1

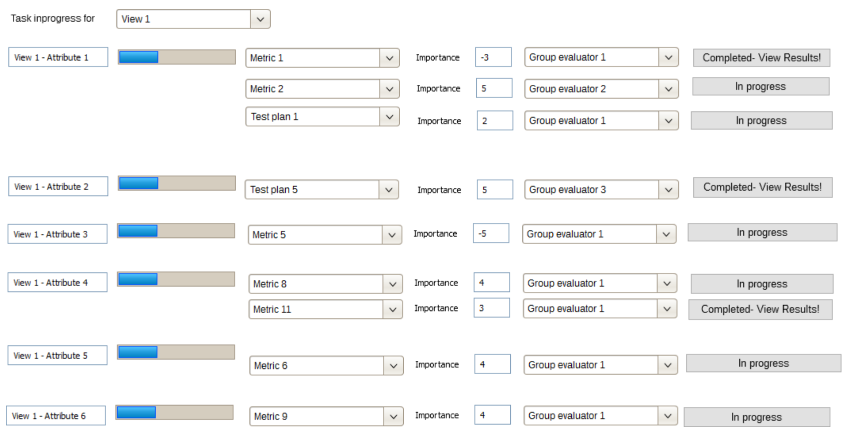
☐ Attribute 1	☐ Metric 1	Importance -3	Group evaluator 1
	☐ Metric 2	Importance 5	Group evaluator 2
	☐ Test plan 1	Importance 2	Group evaluator 1
☐ Attribute 2	☐ Test plan 5	Importance 5	Group evaluator 3
☐ Attribute 3	☐ Metric 9	Importance -2	Group evaluator 1
☐ Attribute 4	☐ Metric 7	Importance 3	Group evaluator 1
	☐ Metric 11	Importance 4	Group evaluator 1
☐ Attribute 5	☐ Metric 13	Importance 4	Group evaluator 1
☐ Attribute 6	☐ Metric 8	Importance -5	Group evaluator 1

FIGURE 2. Interface 2 - adaptable software quality

For each view, the metrics/test plans that should be used for evaluation are defined. It is also useful to create the relationship chart, to see how attributes relate with each other

[21]. All attributes relevant for the project are added in the relationship chart as it follows: if attribute A is directly correlated with attribute B then we set 1 in the relationship chart, if it is neutral the value is 0 and if it is indirectly correlated the value is -1. This way we know how the improvement of an attribute affects other attributes.

Each evaluation group will receive the tasks they need to do from the manager, regarding the metrics or test plans (automatic or manual). When the evaluation of a metric is done, the group will state that the task is finalized and give the information about the results using their account. The information provided is about how the results should look percentage or specific value.



Task inprogress for View 1

View 1 - Attribute 1	Metric 1	Importance	-3	Group evaluator 1	Completed- View Results!
	Metric 2	Importance	5	Group evaluator 2	In progress
	Test plan 1	Importance	2	Group evaluator 1	In progress
View 1 - Attribute 2	Test plan 5	Importance	5	Group evaluator 3	Completed- View Results!
View 1 - Attribute 3	Metric 5	Importance	-5	Group evaluator 1	In progress
View 1 - Attribute 4	Metric 8	Importance	4	Group evaluator 1	In progress
	Metric 11	Importance	3	Group evaluator 1	Completed- View Results!
View 1 - Attribute 5	Metric 6	Importance	4	Group evaluator 1	In progress
View 1 - Attribute 6	Metric 9	Importance	4	Group evaluator 1	In progress

FIGURE 3. Interface 3 - adaptable software quality

Based on the information given by the user and the results provided, we will be able to provide several information related to the quality and guidance on how it can be improved. The proposed model also solves the situations in which specifications change, in this case we just need to modify the view-state and update the info related to them and to the connected view-states.

3. CONCLUSIONS

We defined the specifications for building an adaptable software quality model that will allow the possibility of enhancing a quality model that adapts to the environment, to the user importance perspective, to the process development implemented, to the importance of the functionalities and will also be able to provide real time estimations related to the software product.

REFERENCES

[1] AL-Badareen, A. B., Selamat, M. H., Jabar, M. A, Din, J. and Turaev, S., *Software Quality Models: A Comparative Study*, In: Communications in Computer and Information Science , January 2011

[2] Al-Qutaish, R. E. *Quality Models in Software Engineering Literature: An Analytical and Comparative Study*, In: Journal of American Science, 2010

[3] Berander, P., Damm, L.-O., Eriksson, J., Gorschek, T., Henningsson, K., Jönsson, P., Kågström, S., Milicic, D., Mårtensson, F., Rönkkö, K. and Tomaszewski, P., *Software quality attributes and trade-offs*, In: Compendium Blekinge Institute of Technology, June 2005

[4] Boehm, B. W., Brown, J. R., Lipow, M. L., Quantitative Evaluation of Software Quality. Proceedings of the 2nd international conference on Software engineering, San Fransisco, California, United States, 592-605, 1976, IEEE Computer Society Press.

[5] Chappell, D., *The three aspects of software quality: functional, structural and process*, paper Sponsored by Microsoft Corporation

[6] Côté, M.-A., Suryn, W. and Georgiadou, E., *Software Quality Model Requirements for Software Quality Engineering*, In: 14th international conference on the software quality requirement, 2003

[7] Darmawan, N., Chong, A. Y.-L. and Ooi, K.- B. and Boon In Tan, *Factor Strategy Model: Proofs of prototype concept for software quality evaluation*, In: The Journal of Computer Information Systems, Spring 2010

[8] Gordieiev, O., Kharchenko, V. and Fusani, M., *Evolution of Software Quality Models*, Green and Reliability Issues

[9] Habib, B. and Ashfaq, R. A. R., *Relationship between Factors of Quality Models and the System Development Life Cycle*, In: International Journal of Computer Applications, **81**, (2013), No. 10

[10] Hajdu-Măcelaru, M. D., *Quality evaluation of the software product approach*, Creat. Math. Inform, **17** (2008), No. 3, 420–426

[11] Hajdu-Măcelaru, M. D., *Software quality evaluation using the fuzzy theory*, Scientific Studies and Research, Series Mathematics and Informatics, 2013

[12] Hajdu-Măcelaru, M. D., *A comparative study on existing software quality models*, Creat. Math. Inform, **25** (2016), No. 1, 57–62

[13] Jung, Ho-Won, *Validating the external quality subcharacteristics of software products according to ISO/IEC 9126*, In: Computer Standards & Interfaces, **29**, 2007

[14] Khaddaj, S. and Horgan, G., *A Proposed Adaptable Quality Model for Software Quality Assurance*, J. Comput. Sci., **1** (2005), No. 4, 482–487

[15] Klas, M., Lampasona, C. and Munch, J., *Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results*, Proceedings of the 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications, p. 341–348, August 30-September 02, 2011

[16] Lee, M. C., *Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance*, In: British Journal of Applied Science & Technology, 2014

[17] McCall, J. A., Richards, P. K., Walters, G. F., *Factors in software quality*, Griffiths Air Force Base, N.Y. : Rome Air Development Center Air Force Systems Command, 1977

[18] Miltiadis, G. S., Kyriakos, C. Ch., Andreas, L. S., *QATCH - An adaptive framework for software product quality assessment*, Expert Systems with Applications , **86** (2017), 350–366

[19] Ortega, M., Perez, M. A. and Rojas, T., *Construction of a systemic quality model for evaluating a software product*, In: Software Quality Journal, July 2003

[20] Sharma, K. and Sharma, K., *Comparison Of Various Software Quality Models*, In: Proc. of the Intl. Conf. on Recent Trends in Computing and Communication Engineering, 2013

[21] Suma, V., Shubhamangala, B. R., Manjunatha R. L., *Customization of quality models in software projects to enhance the business value*, Advance Computing Conference (IACC), 2013 IEEE 3rd International

[22] Tripathi, S., *A Survey on Quality Perspective and Software Quality Models*, In: Journal of Computer Engineering, **16**, No. 2, 2014

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
NORTH UNIVERSITY CENTER AT BAIA MARE
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
VICTORIEI 76, 430122 BAIA MARE ROMANIA
E-mail address: macelaru.mara@gmail.com
E-mail address: ioanazelina@yahoo.com