Software Quality Methodology to Train Engineers as Evaluators of Information Systems Development Tools

Laura Silvia Vargas Pérez¹, Agustín Francisco Gutiérrez Tornés²*, Edgardo Manuel Felipe Riverón³, Ana María Soto Hernández¹, Norma Ivone Peña Galeana²

¹Investigador del Instituto Tecnológico de Ciudad Madero, Tamaulipas, México
²Investigador de la Universidad Autónoma de Guerrero, México
³Investigador del Centro de Investigación en Computación, Instituto Politécnico Nacional, México

Abstract To have software tools that facilitate the rapid development of applications and generate information systems is a requirement in a globalized world. Worldwide in the engineers training field it is basic to apply measurements and evaluations to the rapid design tools used to obtain applications faster and generate quality information systems in order to determine the best for this purpose. This paper revises MECRAD and proposes its introduction in the formation programs of engineers which is a methodology for the technical evaluation of the visual environments tools to generate information systems where this type of commercial products are evaluated using as reference basic elements of international standards. With this methodology one can evaluate and select in an effective and easy way, those tools and development platforms best suited to create applications in visual environments, in order to generate information systems with quality and sustainability. It is useful for higher education institutions, organizations, companies and system’s end users, among others.

Keywords Software Quality, Technical Evaluation Qualimetric Model, Visual Environment, RAD Tools

1. Introduction

To improve any software, it is necessary to measure attributes through a significant set of metrics that provide indicators that lead to a technical quality assessment of the product. By carrying out this process through a software qualimetric model, it is more probable that one will access the requirements to comply. The way in which the quality characteristics have been defined in most standards models does not allow their direct measurement, so the establishment of metrics to correlate these features in a software product is needed.

The first step in designing a software qualimetric model is the determination of the relevant quality properties. Usually they are described through a hierarchical tree structure where the characteristics appear at the highest level, the sub-characteristics in the intermediate and the attributes in the lowest. Its goal is to facilitate the qualitative and quantitative evaluation of these components [1] [2] [3] [4].

At an international level, tools to facilitate the creation of new information systems for the most diverse applications periodically emerge. It is imperative to identify and evaluate these tools as quick as possible, to determine whether they meet the quality requirements established globally or by the software houses that produce those [5], [6], [7], [8].

1.1. Objectives

When training world-class software and systems engineers, it is necessary to carry out evaluations of various products. Therefore it is essential to perform the evaluations of software applications designing and developing tools to determine, which are the best for the programmers to choose to work on. Actually, even some jobs proposals are focused on the software development processes evaluation.

As an example of such particular instruments we take the RAD (Rapid Application Development) tools, which are commonly part of an IDE (Integrated Development Environment) that is a very popular framework among application programmers for creating information systems. So the training of software or informatic engineers, with a wide knowledge and manipulation of qualimetric models and tools as those that perform the evaluation of RAD tools for the generation of information systems of quality in...
order to select the most suitable has become today and 
updated technical necessity.
That is the reason why, in this case, this article proposes 
the introduction in high level engineering programs of the 
training and manipulation of MECHDAV (Quality 
Evaluation of Application Development Tools in Visual 
Environments) and MECRAD (Rapid Applications 
Development Quality Evaluation Model) which 
particularly allow a comparative analysis of different tools 
of this type. They are based upon the following standards: 
IEEE 1061 (Software Quality Metrics aMethodology) and 
ISO/IEC 9126 (Software Product Quality), 14598 
(Software Evaluation Process) and 25000 
(SQuaRE-Software Quality Requirements and Evaluation). 
MECA (Modelo para la Evaluación de la Calidad), MACS 
(Metodología para el Aseguramiento de la Calidad del 
Software) and SUMI (Software Usability Measurement 
Inventory) models were also taken as reference for its 

1.2. State of the Art and Related Works
During the last three decades a varied type of software 
quality models have been proposed. They are very useful, 
but in turn very generic, so they should be adapted or 
reconfirmed in order to use them to articulate more 
particular and concrete applicable models [21] [22] [23] 
[24] [25] [26] [27] [28] [29].
Regarding RAD we have no knowledge of the existence 
of any other evaluation model or tool but MECHDAV and 
MECRAD. These model and tool have been also used by 
others researchers and academics in their work in different 
areas [16] [17] [18] [19] [20].

2. Methodology
A proper methodology was generated (with models, 
processes, techniques and tools), that allows making 
comparisons and carrying out RAD tools technical 
evaluations, which is briefly describe ahead. The operation 
method is quite simple and the evaluators just must fulfill 
the data asked for by the program. The rest of the process is 
fully atomized. The complete evaluation process is carried 
out and culminates with a technical opinion of the quality 
reached by the product, together with the 
recommendations and criteria to be followed. This result 
can be analyzed by those interested according to the 
purposes of the evaluation, for example acquiring a 
product.

2.1. Evaluation Process
When evaluating software quality, first one establishes 
the quality requirements model under which the evaluation 
is specified, designed and executed. The evaluation 
activities are then indicated in a process. In this proposal it 
comprises five activities as it is shown in Figure 1.
2.2. Compacting MECRAD Pattern

MECRAD’s complete model is shown in Figure 2. Since one of the purposes of this technical model is to provide a range for comparison for any kind of user (expert or beginner) the metrics suggested for beginner users are defined in a subset of this complete model, and are shown in Figure 3.

2.3. Metrics and Evaluation Scale Definition

The quantifiable attributes must be measured, quantitatively, through metrics. The result, the value measured, can then be mapped on a scale. This value does not show in itself the level of satisfaction of the requirements. For this purpose the scale is divided into ranges corresponding to the different degrees of satisfaction. There are several ways to do this. For example one can simply divide the scale in two categories: unsatisfactory and satisfactory, or create a scale with five levels (mandatory categories) for an evaluated product: levels A, B, C, D and E as shown in Figure 4 where level A is the best case, the one that would be the ideal level to achieve. Level B is considered achievable within the reasonable use of the available resources. Level C indicates the control point, the one should be maintained so that the system does not further deteriorate. Level D is the user’s acceptance limit value. Finally, the worst case is level E, where the product does not meet requirements minimum quality.

<table>
<thead>
<tr>
<th>Characteristic/ Sub Characteristic/ Attribute/ metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1. Functionality/ Complteness/ Total contain/ metric</td>
</tr>
<tr>
<td>1.2.1.1. Functionality/ Consistency/ Format of components and functional elements uniformity/ metric</td>
</tr>
<tr>
<td>1.2.2.1. Functionality/ Consistency/ Processing return uniformity/ metric</td>
</tr>
<tr>
<td>1.2.3.1. Functionality/ Consistency/ Vocabulary and symbols used conventions uniformity/ metric</td>
</tr>
<tr>
<td>1.3.1.1. Functionality/ Correction/ Correct operation/ metric</td>
</tr>
<tr>
<td>1.3.2.1. Functionality/ Correction/ Correct utilization of language / metric</td>
</tr>
<tr>
<td>1.3.3.1. Functionality/ Correction/ Correspondence of descriptions with objects / metric</td>
</tr>
<tr>
<td>1.4.1.1. Functionality/ Interoperability/ Components and interfaces exchange/ metric</td>
</tr>
<tr>
<td>1.4.2.1. Functionality/ Interoperability/ Data exchange/ metric</td>
</tr>
<tr>
<td>1.5.1.1. Functionality/ Standardization/ Symbols standardization/ metric</td>
</tr>
<tr>
<td>1.5.2.1. Functionality/ Standardization/ Vocabulary standardization/ metric</td>
</tr>
<tr>
<td>2.1.1.1. Reliability/ Maturity/ Time between failures/ metric</td>
</tr>
<tr>
<td>2.2.1.1. Reliability/ Recoverability/ Options to recover itself / metric</td>
</tr>
<tr>
<td>2.3.1.1. Reliability/ Tolerance of errors or failures/ Degraded processes/ metric</td>
</tr>
<tr>
<td>2.3.2.1. Reliability/ Tolerance of errors or failures/ Errors processing/ metric</td>
</tr>
<tr>
<td>3.1.1.1. Usability/ Attraction/ Attractive interaction/ metric</td>
</tr>
<tr>
<td>3.1.2.1. Usability/ Attraction/ Successful recovery/ metric</td>
</tr>
<tr>
<td>3.1.3.1. Usability/ Attraction/ Time of operation/ metric</td>
</tr>
<tr>
<td>3.2.1.1. Usability/ Diffusion/ Amplitude/ metric</td>
</tr>
<tr>
<td>3.2.2.1. Usability/ Diffusion/ Frequency of operation/ metric</td>
</tr>
<tr>
<td>3.3.1.1. Usability/ Learnability/ Demo/ metric</td>
</tr>
<tr>
<td>3.3.2.1. Usability/ Learnability/ Demo efficiency/ metric</td>
</tr>
<tr>
<td>3.3.3.1. Usability/ Learnability/ Tutorial/ metric</td>
</tr>
<tr>
<td>3.3.4.1. Usability/ Learnability/ Tutorial efficiency/ metric</td>
</tr>
<tr>
<td>3.3.5.1. Usability/ Learnability/ Documentation/ metric</td>
</tr>
<tr>
<td>3.4.1.1. Usability/ Understandability/ Adequate user interface/ metric</td>
</tr>
<tr>
<td>3.4.2.1. Usability/ Understandability/ In line aid/ metric</td>
</tr>
<tr>
<td>3.4.3.1. Usability/ Understandability/ Terminology in agreement to user/ metric</td>
</tr>
<tr>
<td>3.5.1.1. Usability/ Operability/ Help utility/ metric</td>
</tr>
<tr>
<td>3.5.2.1. Usability/ Operability/ Help operability/ metric</td>
</tr>
<tr>
<td>4.1.1.1. Efficiency/ Use of time/ Efficiency in time/ metric</td>
</tr>
<tr>
<td>4.2.1.1. Efficiency/ Use of resources/ Efficiency in resources/ metric</td>
</tr>
<tr>
<td>4.3.1.1. Efficiency/ Scalability / Availability/ metric</td>
</tr>
<tr>
<td>5.1.1.1. Portability/ Instalability/ Installation module/ metric</td>
</tr>
<tr>
<td>5.1.2.1. Portability/ Instalability/ Documentation of installation module/ metric</td>
</tr>
<tr>
<td>5.1.3.1. Portability/ Instalability/ Configuration module/ metric</td>
</tr>
<tr>
<td>5.1.4.1. Portability/ Instalability/ Documentation of configuration module/ metric</td>
</tr>
<tr>
<td>5.2.1.1. Portability/ Adjustability/ Independence of the hardware environment/ metric</td>
</tr>
<tr>
<td>5.2.2.1. Portability/ Adaptability/ Independence of software environment/ metric</td>
</tr>
<tr>
<td>6.1.1.1. Quality in use/ Effectiveness/ Tasks effectiveness/ metric</td>
</tr>
<tr>
<td>6.1.2.1. Quality in use/ Effectiveness/ Tasks performance/ metric</td>
</tr>
<tr>
<td>6.2.1.1. Quality in use/ Productivity/ Productive proportion/ metric</td>
</tr>
<tr>
<td>6.2.2.1. Quality in use/ Productivity/ User relative efficiency/ metric</td>
</tr>
<tr>
<td>6.3.1.1. Quality in use/ Satisfaction/ User favorite psychological effects/ metric</td>
</tr>
</tbody>
</table>

Figure 2. MECRAD’s complete model
Since a metric is defined as "a quantitative measure of the degree to which a system, component or process possesses a given attribute" [5], in order to properly measure the different tool performance one must follow these guidelines:

- Observation of the software performance in order to evaluate the difference between the current execution results and the requirements specification (a view on test and quality validation).
- Unexpected occurrences on performance time or resources utilization during the software operation.
Therefore, evaluating all attributes belonging to a given sub characteristic one obtains an average value that evaluates that sub characteristic in particular. Then, evaluating all the sub characteristics of a given characteristic the user calculates another average value that evaluates that characteristic in particular. Finally, evaluating all the characteristics a new average value that corresponds to the software product as a whole is calculated. The mathematical method is the following:

Quality indicator of the product: 

\[ I_{CC_t} = \frac{\sum_{j=1}^{n} I_{CC_j}}{n} \]

Where:

- \( I_{CC_j} \) is the quality indicator of the characteristic \( j \)
- \( n \) is the number of characteristics in the model

Quality indicator of the characteristic: 

\[ I_{CC_j} = \frac{\sum_{k=1}^{m} I_{CSC_k}}{m} \]

Where:

- \( I_{CSC_k} \) is the quality indicator of the subcharacteristic \( k \)
- \( m \) is the number of subcharacteristics within the characteristic \( k \)

Quality indicator of the subcharacteristic:

\[ I_{CSC_k} = \frac{\sum_{x=1}^{K} VAA_x}{K} \]

Where:

- \( VAA_x \) is the assigned value to the attribute \( x \)
- \( K \) is the number of attributes within the subcharacteristic \( k \).

Thus, when applying the evaluation format you use three types of metrics:

- Direct instructions to the user for carrying out a specific task, taking note of certain indicators (for example: time, number of occurrences of certain event, etc.) The result will be a quantity within the proposed range.
- Direct questions to the user to determine the existence of an essential attribute within the evaluated tool. The result will be an affirmative (1) or a negative (0) one.
- Metrics that depend on the value of certain indicator derived from the realization of a certain task. They serve to calculate a set of parameters with values within the proposed interval.

In order to support the model, forty-four metrics were documented and developed, just as it appears in the formats in Figure 5 and 6. Another eleven metrics were adapted from SUMI [0], making a grand total of fifty-five metrics involved.

**Figure 5.** Documentation of a model metric

<table>
<thead>
<tr>
<th>Characteristic:</th>
<th>1. Functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub characteristic:</td>
<td>1.2 Consistency.</td>
</tr>
<tr>
<td>Attribute:</td>
<td>1.2.3 Uniformity in processing return.</td>
</tr>
<tr>
<td>Metric:</td>
<td>1.2.3.1 Proportion of adequate functions re-establishment from any depth level.</td>
</tr>
<tr>
<td>Formula:</td>
<td>( X = 1 - \frac{A}{B} )</td>
</tr>
<tr>
<td>A</td>
<td>Number of functions changed after introducing operations during a specific period.</td>
</tr>
<tr>
<td>B</td>
<td>Number of specific functions.</td>
</tr>
<tr>
<td>Interpretation:</td>
<td>Stability of functional specifications objective</td>
</tr>
<tr>
<td></td>
<td>( 0 \leq X \leq 1 ); the closer to 1 the better</td>
</tr>
<tr>
<td>Source of reference:</td>
<td>ISO/IEC 9126, MECHDAV</td>
</tr>
</tbody>
</table>
Recording the partial and total results of the software quality evaluation is not an easy task. Simple and understandable formats must be chosen to obtain a quick and reliable assessment of their measurement values. Therefore checklists, simple relationship tables and control matrices are implemented. Checklists are questionnaires where assertions must be confirmed by selecting one of the values given in a scale. These questions are in principle made in such a way that they generate ideas (valuations). They are used to control each separate phase or all the work to be done. A control matrix is a complementary tool related to all aspects of a process that serves to summarize the content and the development of a whole system. It usually includes a control variable (what is measured), the measurement form, place and time, the base standard, who does the analysis, who acts and how to act. The control matrices are important for the design, implementation and maintenance of the control system of the obtained results.

3. Results Discussion

The case studies chosen for the tests of the RAD tools are the commercial visual platforms Visual Studio.Net, NetBeans and Eclipse. The results obtained through the application of the MECRAD tool are the following:

VisualStudio.Net obtained a general average evaluation of 0.89 (89%) for beginners and a punctuation of 0.88 (88%) among experts (See Figure 7). Its weakness lies in portability. This is comprehensible, due to its dependence upon Microsoft’s Windows platform. Its quality classification level is Satisfactory, without recommendations, since it does not require modifications in its design (only updating) and it is accepted thoroughly.

The results obtained from the other two products in their evaluation, have only 2% of variability. The level of quality classification obtained in these development platforms was Excellent for NetBeans (See Figure 8) and Eclipse (See Figure 9).

To provide a more realistic assessment the final result is the combination of different users’ evaluation of the same type (expert or basic). This will allow a more realistic final technical report. It is presented in Figure 10.
Figure 7. Final technical evaluation report of the Visual Studio.NET environment version 20.

Figure 8. Final technical evaluation report of the NetBeans visual environment version 8.0.
Figure 9. Final technical evaluation report of the Eclipse visual environment 7.0

Figure 10. Average report
4. Conclusions and Recommendations

Any of the three visual environment system mentioned above are considered technically advisable for application developments. For that reason, if one requires a decision about the acquisition of some of these environments, one must consider other important parameters, such as cost, platform, systems interacting within the environment, etc.

The model does not contemplate these parameters, since it is limited to the technical quality evaluation of the visual tools themselves. MECHDAV AND MECRAD are already commercially in operation, therefore the information concerning its development and its source code is not available.

As a future work, it would be advisable to make periodic revisions of the model for its improvement, attempting for example to introduce the evaluation of tools in the visual Web sites environment.

As a conclusion we once more state that the inclusion of the training and manipulations of this kind of quality tools will enrich the professional stock of future system and software engineers.

Acknowledgments

Finally, the authors wish to thank the Center of Computer Research (CIC) and the National Polytechnic Institute (IPN), the Autonomous University of Guerrero (UAGrro) and the Technological Institute of Ciudad Madero (ITCM), all of them located in Mexico, for their support in carrying out this research.

REFERENCES


