Contribution of Multi-criteria Methodology to Software Quality Evaluations

Marie-José Blin, Alexis Tsoukiàs
Lamsade – CNRS, Université Paris Dauphine
Place du Maréchal de Lattre de Tassigny
75775-Paris Cedex 16
E-mail : blin|tsoukias@lamsade.dauphine.fr

Abstract

Industrial evaluations of COTS software mostly use the quality models provided by international standards. But the context and objectives of COTS evaluations are fundamentally different from those primarily defined by the standards. Some key issues are often forgotten: 1°) the existence of several evaluators and several quality models sharing common factors, criteria and measures, 2°) the purpose of the evaluation model, 3°) measures of different types, and 4°) the recursive nature of the model since each node is an evaluation model itself. We had the occasion to study the results of real standard-based COTS evaluations. Faced with the difficulties of exploiting them, we experimented the use of multi-criteria methodology. This work allows us to understand some of the problems generated by applying the standards to COTS evaluations, and to propose new principles for evaluating software quality that should be considered in an evolution of the standards. This paper reports our experiment.

1. Introduction

The software quality evaluation is defined in the ISO 9126 [ISO 9126, 1991] and IEEE 1061 [IEEE, 1992] standards, which propose a quality model and an evaluation methodology. Quality is modeled by a tree where a node (which is "weighted") represents one standardized aspect of the quality evaluation, and where the edges represent the relationships between the different aspects (Figure 1). The quality is evaluated by calculating a score for each node of the tree (but not for the leaves). A node score is the result of the weighted sum of the scores of its child nodes . Leaves are expected to be associated with measurable software aspects (metrics). The different nodes and leaves of the quality model are standardized and the evaluation is supposed to be managed by one person, generally the quality control manager.

Standards are primarily studied to control the quality of specific software along the different phases of its development. In fact, they are also widely used by industrials to evaluate the quality of COTS software. But, the context and objectives of such evaluations are fundamentally different from those primarily defined by the standards: 1°) most of the time, several evaluators are involved, 2°) the different evaluators may have different objectives: for example, one evaluator orders the different products, another looks for the best product or calculates a quality score for each product, 3°) each evaluator may propose a specific quality model, 4°) the different models generally have some common factors, criteria and metrics, 5°) most of the time, factors, criteria and metrics are not those proposed by the standards. For example, Figure 2 shows a quality model for the evaluation of word processing software. Quality is evaluated for three main aspects: functionality, learning and conviviality. Functionality concerns editing, formatting, recording and printing. Editing is associated with five metrics: search processing, integration of objects, "undo" possibilities, copy, pasting. The metrics may be of different types, for example : quantitative, qualitative, Boolean.

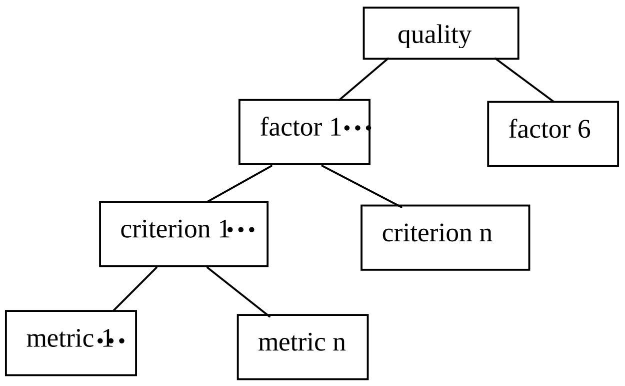

Figure 1: outline of the quality model proposed by standards

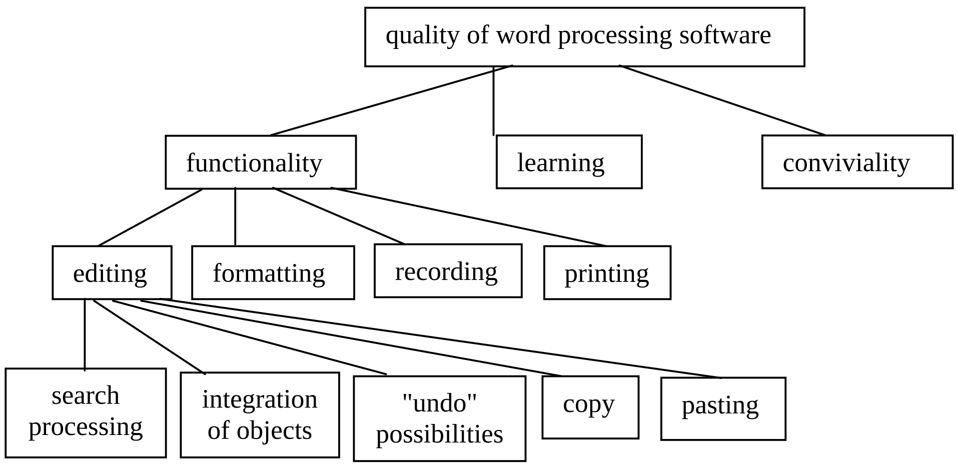

Figure 2 : an example of a quality model for COTS evaluation

Moreover, one key issue is often forgotten in applying standards for software artifacts evaluations: the model is naturally recursive since each node is an evaluation model itself. We claim that the definition of an evaluation model implies a number of choices including the aggregation procedures to be used. Such choices are not neutral and have to be rigorously justified. Otherwise, when results are not totally surprising, they are very difficult to understand and to exploit.

We had the opportunity to study the results of real standard-based COTS evaluations. Faced with the difficulties of using standards, we experimented the use of multi-criteria methodology. This work allows us to understand some of the problems generated by the application of the standards to COTS evaluations and to propose new principles for evaluating software quality that should be considered in an evolution of the standards.

This paper reports the experiment. It is organized as follows: section 2 summarizes the ISO 9126 and IEEE 1061 standards; section 3 presents the case study, analyses the drawbacks of the aggregation procedures and describes how we apply the multicriteria methodology; section 4 describes the lessons learned from the experiment; section 5 discusses important general issues relevant to the design of the quality evaluation models; section 6 presents some related work and finally, section 7 concludes.

2. The standards

The ISO 9126 and IEEE 1061 standards, based on research work in software quality [Boehm, 1978] [Mc Call, 1977] propose: 1°) a common definition of the concept of quality, which is "the set of properties and characteristics of a software which defines its capability to satisfy some expressed or implicit needs", 2°) a common quality model (Figure 1), which defines the quality as an expression of several elements -the quality factors.

Quality factors are defined from the user's point of view. Six factors are proposed: efficiency, functionality, maintainability, portability, reliability and usability. As the quality measurements are carried out on the components of the software, each quality factor is associated with several quality criteria that express the quality from the software point of view. One or several metrics are finally associated with each quality criterion.

A list of quality criteria for each factor and a list of metrics associated with each criterion are provided by the standards. Tables 1 and 2 show examples of such lists.

Factor	Criteria
functionality	completeness correctness security compatibility interoperability
maintainability	consistency correctness modularity traceability expandability
efficiency	time economy hardware resource economy software resource economy communication economy

Table 1: example of quality factors and associated criteria proposed by the standards

Criterion	Metrics
completeness	• ratio of number of completed documents to total number of documents • ratio of number of completed software components to total number of software components • ratio of number of implemented functions to total number of required functions • ratio of number of implemented user interfaces to total number of required user interfaces
traceability	• ratio of number of software components of this phase that can be traced to the previous phase, to total number of software components of this phase • ratio of number of documents of this phase that can be traced to the previous phase, to total number of documents of this phase • ratio of number of functions that can be traced to the requirements, to total number of functions • ratio of number of user interfaces that can be traced to the requirements, to total number of user interfaces

Table 2: example of metrics proposed by the standards

Each metric is calculated from answers to a set of questions. Their values (comprised between 0 and 1) are the result of the following operation :

number of positive answers to the set of questions / total number of questions

A weight is associated with each metric, each criterion and each factor. It represents the relative importance of the associated element in relation to its siblings. For example, in Figure 3, factor A is split into three criteria a_1, a_2 and a_3. Criteria a_1 with the weight 4 is considered four times more important than criteria a_2 and twice as important as criteria a_3.

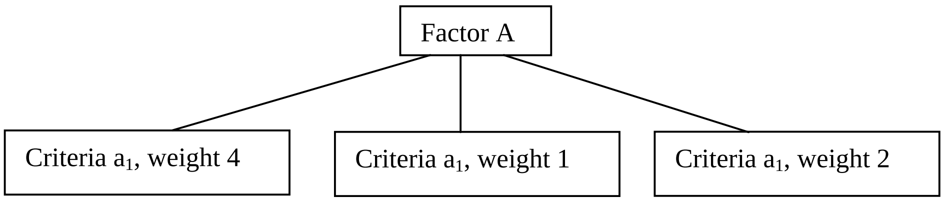

Figure 3: relative importance of the different criteria associated with a factor

An aggregation function is used to calculate the measure of each quality criterion using the weight of the metrics associated to the criterion. In the same way, the measure of each quality factor is calculated using the evaluation and the weight of its criteria. Finally, the global quality measure is provided using the evaluation and the weight of each of the factors. The aggregation function proposed by the standards is the weighted sum.

3. COTS evaluation cases.

The case study is presented first. Then, the drawbacks of the aggregation procedures are discussed. Finally, the use of multi-criteria methodology to evaluate COTS is described see also [Blin and Tsoukiàs, 1999].

3.1. The case study

The case study belongs to a French Laboratory which provides comparative studies of COTS and of computer material for publishing. Each evaluation involves six or seven softwares and two or three evaluators. They are processed in accordance with the ISO 9126 standard using a five level hierarchical quality model (Figure 4). The different evaluators use the same hierarchy but give different weights to the elements. We worked only on a sub-tree of the model, which contains from 200 to 300 nodes and leaves (the grey sub-tree of Figure 4).

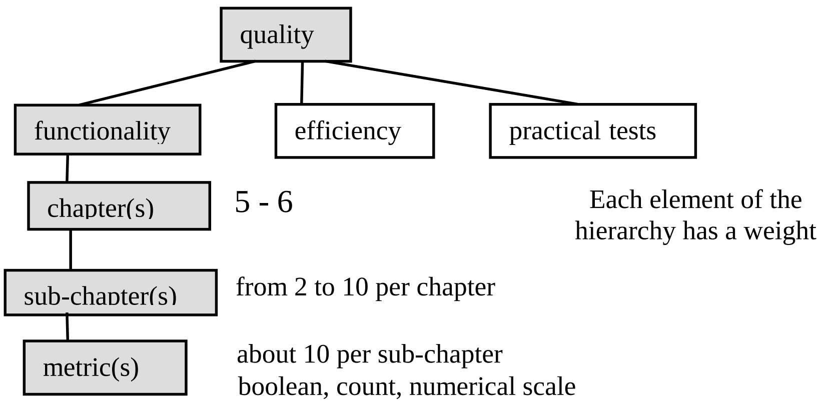

Figure 4: the case study quality model

The measures of the metrics can be: counts, Boolean or ranges of numerical scales. Ranges are sometimes obtained by applying a numerical scale to counts as in Figure 5.

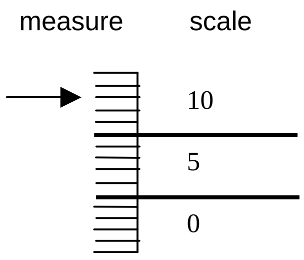

Figure 5 : example of a numerical scale

As the metrics may be of different types, the use of the weighted sum to aggregate the metrics of a sub-chapter requires the normalization of their measures. So, each measure of the metrics is transformed into a mark using the formula below:

mark = (measure / the highest measure of the metric) / the sum of the weight of the metrics of the sub-chapter) * the weight of the metric

For example, Table 3 shows the evaluation of project management tools. It shows the measures and the normalization of three metrics of the "Task Editing"chapter. The first metric -number of user action cancellation levels- is a count and provides the maximum number of user actions that a tool is able to undo. The second metric -simultaneous updating of several tasks- is the numerical scale {0, 5, 10}. It provides a measure of the task updating function usability. Bad usability is represented by 0, good usability by 10. The last metric is Boolean and indicates whether a repetitive task concept exists (value 1) or not (value 0). In the example in Table 3, two products A and B are evaluated.

metrics	type	weight	product A		product B	
			measure	mark	measure	mark
number of user action cancellation levels	count	1	100	$\frac{100}{100} \times \frac{1}{8}$	10	$\frac{10}{100} \times \frac{1}{8}$
simultaneous updating of several tasks	range	2	5	$\frac{5}{10} \times \frac{2}{8}$	10	$\frac{10}{10} \times \frac{2}{8}$
existence of repetitive tasks	Boolean	5	0	$\frac{0}{1} \times \frac{5}{8}$	1	$\frac{1}{1} \times \frac{5}{8}$

Table 3: measure normalization

3.2. Drawbacks of aggregation

At one point our client became aware of several drawbacks of the weighted sum being used as basis for their evaluations. Just to mention an example, consider again Table 3. It is clear that by using the weighted sum the only discriminate evaluation is the “existence of repetitive tasks”, since an object taking the value 1 on this attribute can never be worse than any other whatever values they have for the other attributes. Is this correct? Does it correspond to the client’s feeling? The use of an aggregation procedure frequently implies undesired effects that are often ignored. We will briefly discuss some of these effects.

Aggregating measures or preferences is a very common activity. Observations and/or evaluations provide measures or preferences for several distinct attributes or criteria. But we need a comprehensive measure or preference relation that may represent all the different dimensions we want to consider. It is surprising how often the choice of the aggregation operator is made without any critical consideration of its properties. Let us take two examples.

Example 1. Suppose one has two three dimensional objects *a,b* of which the dimensions are known (*l(a),l(b),h(a),h(b),d(a),d(b)*). In order to obtain an aggregate measure of each object size, one may naturally compute their volume, that is *v(a)=l(a)h(a)d(a)* and *v(b)=l(b)h(b)d(b)*. If the three dimension are prices, one may , however, use an average, that is *p(a)=l(a)+h(a)+d(a)/3* and *p(b)=l(b)+h(b)+d(b)/3*.

From a mathematical point of view, both operators are admissible (when *l(x),h(x),d(x)* are ratio scales as in our example). However, the semantics of the two measures are quite different. It will make no sense to compute a geometric mean in order to get an idea of the price of *a or b* as it will make no sense to compute an arithmetic mean to get an idea of the size of *a or b*. The choice between the geometric and the arithmetic means depends on the semantics of the single measures and of the aggregated ones.

Example 2. Suppose there are two objects *a,b* and two criteria (in the Multi-criteria Decision Aid Methodology, a criterion is a preference relation with a numerical representation) g_1 and g_2 such that, $\forall x,y,\ p_j(x,y) \Leftrightarrow g_j(x) > g_j(y)$. Moreover $g_1: A \rightarrow [0,1]$ and $g_1(a)=0$ and $g_1(b)=1$ and $g_2:A \rightarrow [0,2]$ and $g_2(a)=2$ and $g_2(b)=1$. Under the hypothesis that both criteria are of equal importance, many people will compute the average and infer the global preference relation. In our case, *a* is indifferent to *b,* since $g(a)=g_1(a)+g_2(a)/2=1$ and $g(b)=g_1(b)+g_2(b)/2=1$. However, if an average is used, it is implicitly assumed that g_1 and g_2 admit ratio transformations. Therefore, it is possible to replace g_2 by $g'_2:A \rightarrow [0,1]$ so that $g'_2(a)=1$ and $g'_2(b)=1/2$ (known as scale normalization, see also the case study). Under the usual hypothesis of equal importance of the two criteria, we now obtain that *b* is preferred to *a,* since $g(a)=1/2$ and $g(b)=3/4$. Where is the problem?

The problem is that the average aggregation was chosen without verifying if the conditions under which it is admissible hold. First of all, if the values of *a* and *b* are obtained from ordinal evaluations (of the type good, medium, bad, etc.), then the numerical representation does not admit a ratio transformation (in other words we cannot use its cardinal information). Secondly, even if the ratio transformation was admissible, the concept of criteria importance is misleading. In a "weighted arithmetic mean" (as the average is) the "weights" are constants representing the ratio between the evaluation scales. In the example, if we reduce g_2 to g'_2, we have to give g'_2 twice the importance given to g_1 in order to keep the concept of "equal importance" true. In other words, it is not possible to speak of importance of the criteria (in the weighted arithmetic mean case) without considering the cardinality of their co-domains.

From the examples given above, we can induce a simple rule. *In order to choose an aggregation operator appropriately, it is necessary to take into consideration the semantics of the operator and of each single preference or measure and the properties (axiomatics) of the aggregation operator*. In other words, if the aggregation operator is chosen randomly, neither the correctness of the result, nor its meaningfulness can be guaranteed.

3.3. The use of multi-criteria methodology

We reused the quality models and the existing measures of the metrics of the studied cases, and we alternatively substituted three aggregation procedures to the weighted sum of the scores (which is an arithmetic mean): ordinal aggregation, geometric mean and dual geometric mean. This section briefly describes such procedures and explains how we applied them.

Ordinal aggregation. The ordinal aggregation procedure used belongs to the family of the ELECTRE methods [Roy, 1991]. A detailed description of the method can be found in Appendix A.

Such a procedure was applied from the leaves to the root of the quality model. At each level of the tree, an ordering of the alternatives (the software to be evaluated) is calculated for each node of the level. These orderings are used at the next level to calculate new orderings and so on (aggregation of preferences) (Figure 6).

Only the concordance formula was used. The person responsible for the evaluation in the Laboratory was not able to indicate any veto condition on the criteria. Moreover, as most of the criteria were ordinal, it was very difficult to state any veto threshold. Finally the person responsible considered that the existence of a veto could act as an a priori' elimination, in which case the set of products to evaluate should be considered as badly chosen.

In order to be able to repeat the calculation at each level of the quality model, the outranking relation obtained at each node of the hierarchy was transformed into a weak order using the "Score method". This method consists, for each alternative, in subtracting the number of times this alternative outranks the others and the number of times it is outranked ("final score" of each alternative) (Table 4). The weak order is established on the basis of the final score of each alternative.

The ordinal aggregation may conceal situations of incomparability, which have to be analyzed before calculating the final order of the alternatives at each level of the hierarchy. When incomparable alternatives were detected, a sensitivity analysis was applied. Every alternative better or worse than the incomparable alternatives was set aside. The incomparable alternatives and every alternative ordered between them were retained and the calculations were made again with a new concordance threshold until all incomparabilities disappeared. The idea of the sensitivity analysis is to verify at which level of confidence all the alternatives can be compared. In fact, the decision maker wanted to verify if the incomparability was due to the imposition of a high confidence level or to intrinsic characteristics of the alternatives.

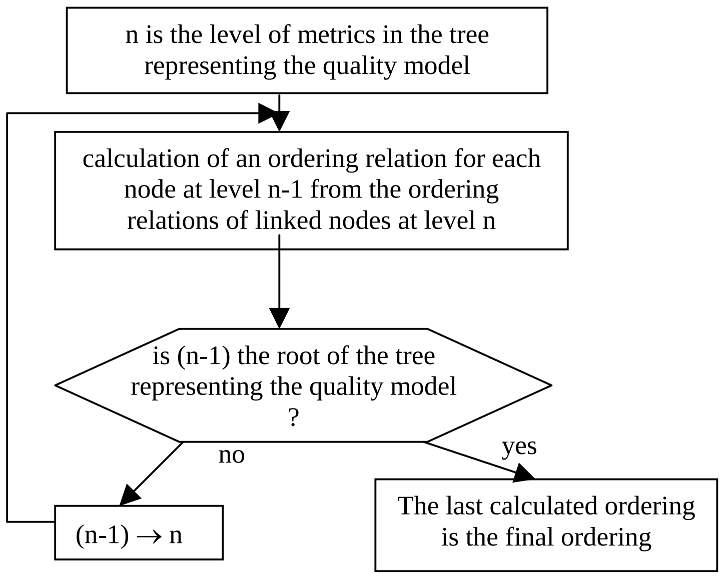
n is the level of metrics in the tree representing the quality model
calculation of an ordering relation for each node at level n-1 from the ordering relations of linked nodes at level n
is (n-1) the root of the tree representing the quality model ?
no
yes
(n-1) → n
The last calculated ordering is the final ordering

Figure 6 : applying Electre II for comparing COTS

Geometric means. The geometric mean procedure calculates the score of each node as the result of the weighted product of its child node scores, while the dual geometric mean computes the weighted product of the complement of each child node score. More specifically, the two formulas below were used:

$u(x) = \prod_j (u_j(x))^{w_j}$ geometric mean

$u(x) = 1 - \prod_j (1 - u_j(x))^{w_j}$ dual geometric mean

where:

u(x): score of alternative x on the parent node;

$u_j(x)$: score of alternative x on the child node j;
w_j: relative importance of the child node j.

Obviously, the formulas make sense when all of the evaluations are expressed in the interval [0,1]. In our experiment, however, we avoided the extreme values 0 and 1 since the presence of just one of them in the child nodes will keep the global score at 0 or 1 independently from the rest of the evaluations (in other words we attenuated the non compensation effect of the formula).

	A1	A2	A3	A4	number of alternatives outranked by x
A1	1	0	0	0	1
A2	1	1	1	1	4
A3	1	0	1	1	3
A4	1	0	0	1	2
number of alternatives outranking x	4	1	2	3	
final score	-3	3	1	-1	
final order used in the next level of the hierarchy	4	1	2	3	1 (being the best)

Table 4: an example of the use of the Score method

4. Lessons learned

We first analyzed the difficulties encountered with the use of quality standards for COTS evaluations and then, the advantages and weaknesses of the multi-criteria methodology for this kind of evaluations. These results can be completed by the reading of [Fenton, Schneidewind, 1996] who discuss the weak and strong points of standards. For similar experiences see also [Paschetta and Tsoukiàs, 2000].

4.1. Difficulties in using the standards for COTS evaluations.

COTS are generally used in organizations that purchase them and it is impossible to define and to simulate all the applications a software will go through. Therefore, the evaluation has to mainly consider the software features rather than their behavior in a real context. The evaluation of COTS is often a long process involving several actors, for example, the final users, the purchase manager, the software maintenance people, the manager responsible for the integration of the software in the organization or in the technical environment. Each of the actors has his own point of view and his own quality model. Of course, several models often have common parts.

Generally, each actor builds an a priori quality model with a great number of factors, sub-factors and criteria from his knowledge of the domain and from his experience. But, it is very difficult for him to determine the decisive elements of the model and to award "weights" to factors, sub-factors and criteria. So, the model includes a great number of elements and is difficult to exploit. Moreover, the different elements forming a quality model are not always independent. For example, a same criterion may be associated with several factors. As a result, this criterion has in fact a greater weight in the quality model than the weight given by the evaluator (if the weighted sum is used). Actually, the use of additive aggregation operators is an error if preferential independence is not verified.

Usually, the evaluators proceed by trial and error in order to determine the correct choices. So, the quality model is iteratively modified all along the decision process (Figure 7). The evaluator will be efficiently helped if the evaluations provide information to refine and simplify the model (for example, the significant items of the model or the real weight of each item). But, the standards do not provide such means.

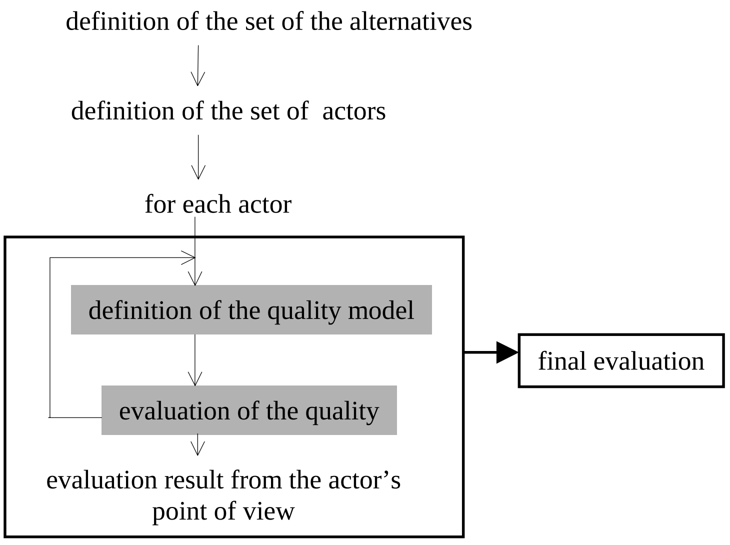
definition of the set of the alternatives
definition of the set of actors
for each actor
definition of the quality model
evaluation of the quality
evaluation result from the actor's point of view
final evaluation

Figure 7 : the iterative quality evaluation process

As we have seen, the measures are transformed into homogeneous numerical evaluations (in order to use the weighted sum as the aggregation procedure). However, such transformations are often arbitrary and the result is therefore meaningless. Some research has pointed out such a drawback (for a discussion see [Kitchenham, Pfleeger 1996]). Moreover, as already shown in the previous section, the weighted sum suffers severe drawbacks and may produce totally misleading results if used incorrectly.

4.2. Advantages and weaknesses of the multi-criteria methodology

The use of the multi-criteria methodology presents three positive features and a negative one:

1°) Ordinal aggregation enables meaningful homogenous handling of non homogeneous information, since it does not impose any restriction on the information expressed on the criteria (sub-criteria, etc.).

2°) In addition, ordinal aggregation highlights situations of incomparability, which otherwise could be concealed during the aggregation. Therefore, the ordinal aggregation can be a way of validating the quality model.

3°) Geometric and dual geometric means provide another way of refining and validating the model. Geometric mean brings out specific "bad" performances of the alternatives since the global score deteriorates exponentially with respect to the importance of the criterion on which the "bad" score is expressed (conversely the dual geometric mean will bring out alternatives with "good" evaluations). Both of these means introduce a non linear compensation effect among the criteria and therefore can also be used as measures of "attractiveness" in the interval [0,1] in the presence of ordinal information. They may also be replaced with other kinds of ordered statistics and triangular means, which are meaningful as "attractiveness" measures.

4°) The information contained in each criterion (sub-criterion, etc.) is often richer than the simple order of the alternatives. It is sometimes a ratio or interval information on the comparison of the alternatives, other times an external measurement or a qualitative judgment, but in all such cases it contains knowledge about a metric and its properties. A purely ordinal aggregation at every level eliminates this information since it focuses on the order of the alternatives. This may lead to a poor conclusion from the point of view of the decision-maker. Particularly, in our case, although he was aware that a large part of his criteria were purely ordinal, the decision-maker would have preferred to have measures of the distances between the alternatives.

We conclude that it is interesting to exploit the evaluation information provided by different aggregation procedures. The evaluators should be able to use several of them. Successive evaluations will enable the evaluator to bring out significant elements for simplifying the model and for having a clearer vision of the evaluation situation. For example, the knowledge of the decisive elements that cause a fall in quality or that place one software ahead of others in a league, may efficiently help the decision makers. Thus, each node of a quality model should be associated with the aggregation procedures, which can be used in relation to the type of measures and to the evaluation objectives. But, such procedures must be chosen with great care. The following section gives the evaluators some useful hints to this end.

5. Software Evaluation Models

As already discussed in the previous sections, software evaluation uses a complex hierarchical quality model. Moreover, the evaluation may concern parts of the software itself (or the whole), may be carried out with different dimensions and may be made for different purposes [Morisio, Tsoukiàs, 1997], [Stamelos, Tsoukiàs, 1998].

Provided a first idea of the evaluation to be performed is outlined, it is necessary to establish an evaluation model (actually one for each node of the hierarchy). Following the IUSWARE specifications [Morisio, Tsoukiàs, 1997], we may consider that a set of alternatives A (a set of software, or parts to be evaluated), a set of dimensions or ”points of view" V under which the evaluation has to be made and a problem statement Π describing the purpose of the evaluation are available. We call such a triple a ”problem formulation" at time t $\Gamma_t = \langle A,V,\Pi \rangle$.

An evaluation model consists of a n-tuple $M = \langle A^*,D,M,E,G,R \rangle$, where A^* is the set of alternatives to evaluate after an eventual preliminary screening, D is the set of attributes considered instantiating V, M and E are any measures and relative scales to be used, G is a coherent set of criteria to represent preferences and R is a set of aggregation operators to apply on M and/or G. As we have already said, the construction of such a model may be a long process including different activities, feedback and revisions. We will concentrate our attention on two specific activities.

Building the sets M and G. This is usually inferred from *D,* which provides an explicit representation of the different points of view (set *V*) introduced in the problem formulation. The set *D* usually is obtained by using international standards (as the ones included in ISO 9126 and IEEE 1061) and/or the client's specific knowledge about the kind of the software to evaluate.

Two processes are performed simultaneously. The first, top-down, in which general dimensions (factors in the IEEE terminology) are dis-aggregated to specific sub-dimensions and so on until sub-n dimensions are reached on which the client is able to express or gather or build up some information. The second, bottom-up, from the client, who identifies subsets of evaluation dimensions as sub-dimensions of a dimension on a higher level from his specific knowledge. The result of the two processes is the definition of a hierarchy of the type presented in the previous sections.

However, in such an activity it is necessary to pay attention not only to the semantic relevance of the child nodes of a parent node, but also to the verification of their independence. The basic and necessary independence condition to meet is the "separability" of the "son-nodes" (the nodes to be aggregated into a parent node). Intuitively, the notion of separability means that if two objects are perfectly equivalent for all son-nodes except one, then the difference for such single son-node should be reflected on the parent node. In other words, every single son-node should be able to discriminate two objects alone. If such a condition is not verified, then the set of son-nodes has to be reconsidered. Further independence conditions can be imposed, but they deal with specific aggregation operators and will not be discussed here (see [Roberts, 1979], [Von Winterfeldt, Edwards, 1986], [Vincke, 1992] and [Roy, 1996]).

To summarize, given a more or less intuitively created evaluation hierarchy, both a semantical and independence verification has to be performed before it can be included in the evaluation model M.

Defining the set R. Associating a subset of nodes to a parent node implicitly assumes that an aggregation operator is also associated with the parent node such that the information contained in the son-nodes is propagated to the parent node. As already observed at the beginning of this section, the choice of an aggregation operator is not neutral and has to be made appropriately. A first distinction to be made is between preference and measurement aggregation.

Preference aggregation. If the parent node is expected to represent preferences on the set A^*, then the son-nodes have to carry such information. If this is not the case, then a preference relation has to be associated to each son-node. If any measures are available and if they are at least expressed on an ordinal scale, then a preference relation can be inferred applying such a measurement on the set A^* (for instance, if the measure m_j is available, we can define $p_j(x,y) \Leftrightarrow m_j(x) > m_j(y)+k$ or $p_j(x,y) \Leftrightarrow m_j(x) > 2\, m_j(y)$, etc.).

If the measures available on a son-node are just nominal, then an ad hoc preference model has to be established on the set A^*. Once each son-node is equipped with a preference model, a preference aggregation method can be used, acting either on the numerical representations of the preference relations (if they exist), or directly on the binary relations. Some basic rules should be remembered:

- if at least one of the numerical representations is obtained from an ordinal scale, then only ordinal aggregation operators can be used.
- if a linear multi-attribute value function is going to be used, then linear preferential independence for the set G has to hold, a compensation principle is accepted and weights are trade-offs.
- if an ordinal aggregation has to be used and a complete global preference relation is required, then it will be either dictatorial, or it will not respect the independence from irrelevant alternatives [Arrow, 1951].

For a comprehensive discussion, see [Keeney, Raiffa, 1976] and [Vincke, 1992].

Measurement aggregation. If a new measurement scale is defined on the parent node, then the basic operation is to establish the semantics of the new scale and its structure. Once a metric has been chosen, the most appropriate aggregation operator can be identified. Again some basic rules should be remembered:

- the result of an aggregation cannot carry more information than that contained in the aggregated nodes (for instance, it is not possible to construct a ratio scale aggregating ordinal scales).
- if the aggregation operator requires scale transformations, these have to be compatible with the admissible transformations of any single aggregated measurement (for instance, an interval transformation is not admissible for a ratio scale).
- if weighted statistics are used, weights should respect scale ratios (provided that such a ratio makes sense).

In order to give an example, let us consider a scale whose values are 10, 20 and 30. If it is a ratio scale, then a linear transformation of the type $\alpha x+\beta$, with $\alpha =1$ and $\beta=5$, will yield the new scale with values 15, 25 and 35 where the ratio information is lost. Such a problem is particularly relevant when normalization of the scales is required due to various reasons. Unfortunately this is possible only if the scales are of the same type.

Once an evaluation model is constructed, a validation process should be performed in order to verify if the model effectively yields the expected results. As far as the aggregation operators are concerned, from our experience, two types of results are questionable and have to be discussed with the client:

1. unexpected measures are obtained for some objects on some nodes. Clearly, the aggregated measure, as defined in the model, does not apply suitably to the set A^* and the measures for these nodes have to be redefined.

2. an incomparability emerges for a couple of objects on a node. If it is unexpected, we have to look for the reasons: a) perhaps, very different objects are considered and the set A^* has to be discussed and eventually redefined, b) the quality model may represent strongly conflicting evaluations and a new compromise has to be established, c) the information available is not sufficient and further investigation is necessary.

6. Related work

This section presents some papers concerned with the quality evaluation issues we developed in the previous sections : the importance and difficulty of determining the significant items of a quality model and simplifying it, the importance of integrating several points of view in the model and the right use of aggregation methods.

Definition of the quality model. In the system described in [Meskens, 1994], an ISO 9126 like quality model (including factors, sub-factors, criteria, weights and aggregation methods) is dynamically built and proposed to the evaluator who may adapt it. The process uses a knowledge base testing the domain of the application and the target software environment. Each criterion is associated with metrics concerning different elements of the code and elements of checklists, such as the uniqueness of the meaning of each variable, the indentation of the code, the use of a programming method. The measure of each element of the checklists is calculated from elementary metrics using the rules of the knowledge base. Aggregation methods are defined by other rules of the knowledge base which specify how to calculate the quality of each criterion from its associated metrics, of each sub-factor from its associated criteria, of each factor from its associated sub-factors and of total quality from the factors. However, no discussion concerning the relevance of the aggregation methods for the different possible evaluations is presented.

In [Erikkson, McFadden, 1993], the Quality Function Deployment method (a Japanese method introduced in 1972) is described. The main purpose of this method is to allow all employees of an organization to participate in the design of new products. The higher level of the model defines customer quality requirements. Each of these requirements is linked to software characteristics, which are themselves linked to software sub-characteristics. At both lower levels, design and implementation of product features, relevant metrics are proposed. The model is represented at each level by matrices.

Simplification of the model. A method to calculate a minimal model is presented in [Anderson, Chen, 1997]. The authors are interested in calculating a statistical evaluation of software user satisfaction. They present a model to compare COTS and a method to simplify the model. Software are categorised and six attributes measure their quality. Several hundred users of software products were asked to evaluate each attribute of the product(s) they own, according to their satisfaction using scaled replies ranging between 1 to 10. Each software is evaluated using the weighted sum of the average measure of each attribute. The weight of the attributes is statistically calculated from the users' answers. A method based on principal component analysis allows to discover components of attributes. It also allows to observe that three components are enough to evaluate software (useful components are not necessarily the same for different categories of software). This work allows the software editors to improve the products or to adapt them to the expectations of the users.

Introduction of the concept of point of view. In [Verner and al., 1996], the current state-of-practice for software quality in Information Systems Departments in Hong Kong is analyzed from replies to 175 questionnaires sent to a wide variety of occupation groups and enterprises. The authors underline the importance of considering several points of view in defining software quality, for example, the developer, the buyer, the user, the maintainer, the project manager, the accountant and different specialists like lawyers, and the need to adapt the quality model to the evaluator. In fact, the authors report that: 1°) most of the quality factors proposed in the ISO standards were important for the people questioned, but a few of them are ignored, while new factors are added, 2°) if some of the factors used by the different groups to evaluate the software quality are the same like reliability, maintainability and functional correctness, other ones are specific and the relative importance of the factors is always different.

In [Feuk and al., 1995], the metrics are disconnected from the model to allow the building of different quality models using the same metrics and the same results of metrics. The structure of the model is equivalent to the structure proposed in the ISO and IEEE 1061 standards, except that the model does not have only one root, but as many roots as different points of view.

In [Paulussen, 1995], the author underlines that a software interests several groups of people, for example the users and the maintainers and that each of these groups does not consider the same characteristics of the software as essential. Only one quality model is defined but it is the result of the merged opinions of the people concerned.

Aggregation methods. Most of the work that focuses on the quality model does not reference the aggregation methods used to calculate the values of the different elements of the model. In fact, aggregation methods are not really discussed in the papers on software quality. In [Meskens, 1994], the author states that the model includes the aggregation method and the relative importance of factors and of criteria, but unfortunately, every example described in the paper uses the weighted sum.

Researchers belonging to the multicriteria community have published interesting papers about aggregation methods but these studies do not consider any quality model. For example, in [Le Blanc, Jelassi, 1994], the results of using the LWA (Linear Weighted Attribute) and the MAUT (Multi-Attribute Utility Theory) aggregation methods for the evaluation of a small set of software are compared. Both methods give the same results but the statistical analysis of the MAUT method results gives more precise and more detailed information about the differences of the software to the evaluator.

7. Conclusion

The paper presents and discusses some experiments conducted in the evaluation of software products, mainly COTS. The problem arises since the application of ISO (and other) standards for software quality evaluation appears to be unsatisfactory: a single evaluation model is applied all along the hierarchy of evaluation nodes neglecting important variations due to the presence of different evaluators, of software components and of evaluation purposes. Further on, a major problem highlighted by our experience concerns the unjustified use of a single aggregation procedure all along the hierarchy (the weighted sum), although severe drawbacks can be observed.

A specific case study, concerning the evaluation of COTS by a large French company specialised in publishing of software benchmarking studies is presented. Different aggregation procedures originating from the multi-criteria methodology were used. Ordinal aggregation techniques and geometric means were tested. Lessons learned through such an experiment are reported in the paper. Further on, some general guidelines for the design of the evaluation model are presented mainly as far as the definition of measures, preferences and their aggregation is concerned.

We claim that such results should be considered in the evolution of the standards for software quality evaluation. It will be the subject of our forthcoming research.

References

E. E. Anderson, Yu-Min Chen, (1997), *Microcomputer software evaluation: An economic model,* Decision Support Systems 19, pages 75-92, 1997

K. Arrow , (1951), *Social Choice and Individual Values,* J. Wiley, New York.

B. W. Boehm, (1978), *Characteristics of Software Quality,* North Holland Publishing Company

M.-J. Blin, A. Tsoukiàs*, Evaluation of COTS using multi-criteria methodology*, in Proceedings of the 6th European Conference on Software Quality, pages 429 - 438, 1999.

I. Erikkson, F. McFadden, (1993), *Quality function deployment: a tool to improve software quality,* Information and Software Technology, Volume 35, number 9, September 1993, pages 491 - 498

N. Fenton and N.F. Schneidewind, (1996), *Do Standards Improve Quality?,* IEEE Software, January 1996, pages 22 - 24

N. Feuk, R. Whitty and Y. Ilruka, (1995), *Applying the Goal/Question Metric paradigm in the experience factory,* International Thomson Computer Press, London, pages 23-44

The Institute of Electrical and Electronics Engineers, (1992), *Standard for a Software Quality Metrics Methodology,* December 1992

International Organization for Standardization, (1991) *ISO 9126: Information Technology - Software product evaluation - Quality characteristics and guidelines for their use*

R. Keeney, H. Raiffa, (1976), *Decision with multiple objectives: preferences and value trade-offs,* J. Wiley, New York.

B. Kitchenham, S.L. Pfleeger, (1996), *Software Quality: The Elusive Target,* IEEE Software, January 1996, pages 12 - 21

L. le Blanc, T. Jelassi, (1994), *An empirical assessment of choice models for software selection: a comparison of the LWA and MAUT techniques,* Revue des systèmes de décision vol. 3 - no.2, pages 115 - 126

J. A. Mac Call, (1977), *Factors in Software Quality, Journal of General Electric*, no. 77, C15-OL, June 1977}

N. Meskens, (1994), *A knowledge-based system for measuring the quality of existing software,* Revue des systèmes de décision vol. 3, no. 3, pages 201 - 220

M. Morisio, A.Tsoukiàs, (1997), *IusWare: a methodology for the evaluation and selection of software products,* IEE.-Softw. Eng. Vol. 144, pages 162 - 174

E. Paschetta, A. Tsoukiàs, (2000), *A real world MCDA application: evaluating software*, Journal of Multiple Criteria Decision Analysis, vol. 9, pages 205 - 226.

R. M.C. Paulussen, (1995), *The Quint Approach to the Specification of Software Quality,* 1st World Congress for Software Quality, San Francisco, June 20-22, 1995

F.S. Roberts, (1979), *Measurement Theory with Applications to Decision Making, Utility and the Social Sciences,* Addison Wesley, New York.

B. Roy, (1991), *The outranking approach and the foundations of ELECTRE methods,* Theory and Decision, vol. 31, pages 49-73

B. Roy, (1996), *Multicriteria Methodology for Decision Aiding,* Kluwer Academic Publishers, Dordrecht

J. Stamelos, A. Tsoukiàs, (1998), *Software evaluation problem situations,* cahier du LAMSADE No 156, Université Paris Dauphine, submitted.

J. Verner, T. Moores, A.R. Barrett, (1996), *Software quality: perceptions and practices in Hong Kong,* Working Paper Series, City University of Hong Kong, Faculty of Business Department of Informations systems, WP96/02, April 1996.

Ph. Vincke, (1992), *Exploitation of a crisp relation in a ranking problem,* Theory and Decision, vol. 32, pages 221-240.

D. Von Winterfeldt, W. Edwards, (1986), *Decision Analysis and Behavioral Research,* Cambridge University Press, Cambridge MA.

Appendix A: The ELECTRE method applied in the experiment

The ELECTRE family methods provide a complete or a partial ordering of equivalence classes from the best to the worst ones. It considers ties and incomparable classes. Equivalence classes are composed of alternatives characterised by criteria. The ELECTRE methods compute a binary relation for all possible pairs built on the alternatives set and constructs a preference relation on such a set.

More precisely, for any pair of alternatives *(x,y)*, we have

$$S(x,y) \Leftrightarrow C(x,y) \wedge \neg D(x,y)$$

where:

S(x,y): the alternative *x* is at least as good as *y* (*x outranks y*);

C(x,y): concordance condition (in our specific case):

$$C(x,y) \Leftrightarrow \frac{\sum_{j \in J^{\geq}_{xy}} w_j}{\sum_j w_j} \geq c \wedge \frac{\sum_{j \in J^{>}_{xy}} w_j}{\sum_{j \in J^{<}_{xy}} w_j} \geq 1$$

with:

$J_{xy}^{\geq}$: criteria for which $S_j(x,y)$ holds;

$J_{xy}^{>}$: criteria for which $\neg$ $S_j(y,x)$ holds;

$J_{xy}^{<}$: criteria for which $\neg$ $S_j(x,y)$ holds.

and *D(x,y)*: discordance condition

$$D(x,y) \Leftrightarrow \exists g_j : v_j(x,y)$$

where:

$v_j(x,y) \Leftrightarrow g_j(x) > g_j(y) + v_j$ or

$v_j(x,y) \Leftrightarrow \dfrac{g_j(x) - g_j(y)}{t_j - b_j} \geq v_j$

with:

$v_j(x,y)$: veto condition for criterion g_j;

v_j: veto threshold for criterion g_j;

t_j: max value of criterion g_j;

b_j: min value of criterion g_j.

The conditions under which $S_j(x,y)$ holds depend on the preference structure of each criterion. If g_j; is:

- a weak order: $S_j(x,y) \Leftrightarrow g_j(x) \geq g_j(y)$
- a semi order: $S_j(x,y) \Leftrightarrow g_j(x) \geq g_j(y) - k_j(y)$
- and so on with interval orders, pseudo orders, etc.

Once the global outranking relation is obtained, we can deduce:

- a strict preference relation *p(x,y)* between *x* and *y*: $p(x,y) \Leftrightarrow s(x,y) \wedge \neg s(y,x)$
- an indifference relation *i(x,y)* between *x* and *y*: $i(x,y) \Leftrightarrow s(x,y) \wedge s(y,x)$
- an incomparability relation *r(x,y)* between *x* and *y*: $r(x,y) \Leftrightarrow \neg s(x,y) \wedge \neg s(y,x)$

The definition of the relation *s(x,y)* is such that only the property of reflexivity is guaranteed. Therefore, neither completeness nor transitivity holds, and thus *s(x,y)* is not an order on the set *A*. In order to obtain an operational prescription, the relation *s(x,y)* is transformed into a partial or a complete order through an ”exploiting procedure" which can be of a different nature (see Vincke, 1992). In this case we adopted a score based procedure.

$\sigma(x) = |\{y: s(x,y)\}| - |\{y: s(y,x)\}|$, where $\sigma(x)$ is the “score of x”.