A new method to evaluate software artifacts against predefined profiles

Maurizio Morisio
Politecnico di Torino
24, Corso Duca degli Abruzzi
10129 Torino Italy
+ 39-011-5647033
morisio@polito.it

Ioannis Stamelos
Dept. of Informatics
Aristotle University
54006 Thessaloniki Greece
+30-310-998227
stamelos@csd.auth.gr

Alexis Tsoukiàs
Lamsade - CNRS
Université Paris Dauphine
Paris, France
+33-1-44054401
tsoukias@lamsade.dauphine.fr

ABSTRACT

Software artifacts are characterised by many attributes, each one in its turn can be measured by one or more measures. In several cases the software artifact has to be evaluated as a whole, thus raising the problem of aggregating measures to give an overall, single view on the artifact.

This paper presents a method to aggregate measures, that works comparing the artifact with predefined, ideal artifacts, or profiles. Profiles are defined starting from ranges of values on measures of attributes. The method is based on two main phases, namely definition of the evaluation model and application of the evaluation model, and is presented in a simplified case study that deals with evaluating the level of quality of an asset to decide if accepting it in a reuse repository. The advantages of the method are that it allows using ordinal scales, while it deals explicitly with preferences expressed, implicitly or explicitly, by the evaluator.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *Complexity measures, Performance measures, Process metrics, Product metrics.*

General Terms

Management, Measurement

Keywords

Software evaluation, quality models, multicriteria decision aid.

1. INTRODUCTION

Artifacts in the software process are complex items with many attributes, each one can be characterized by a measure. For instance a code module could be characterized by size, functionality, complexity, modularity, and the related measures. A software product could be characterized by functionality, reliability and cost.

Sometimes artifacts need to be evaluated as a whole, not only on each attribute alone. Examples of evaluations are (see also [13]):

- Decide if a management information system (MIS) should be kept, or changed. The existing MIS is compared with the new, expected one.
- Decide which commercial off the shelf (COTS) product to buy, to fulfill a need. COTS are compared among them, and possibly they are compared with the ideal one, fulfilling the need.
- Decide if a code module can be accepted, as far as its quality level is concerned. This evaluation could be performed by the quality assurance function of a company, the module is compared with a virtual, ideal module described in a company document, or in a standard.
- Certify a software product. This case is in fact a variation of the case above. The evaluation is performed by an independent entity, an international/national standard is used, a whole product is evaluated.

On the other hand, a software artifact may represent a broad class of Information Technology concepts (a programming language, a software development approach, a software organisation, etc.). Examples of such practical evaluation situtations are:

- choice of a programming language to be used in a project
- choice of open-source or close-source approach in developing a new system
- determination of the capability maturity level a software company belongs to

We can recognize some common patterns in the evaluation cases listed above.

- An evaluator (project manager, quality assurance manager, certification body, etc.) is charged of solving a decision problem.
- The decision problem can be Boolean (keep- buy, accept – reject, certify – not certify) or implies a choice (select a COTS product).
- The evaluation involves many artifacts (selection of COTS product) or only one. In the latter case, a second artifact (the ideal one) is often used for comparison. In other words the evaluation is not absolute, but uses a reference for comparison.
- The starting point of the evaluation is a set of simple attributes where measures are available. For instance, to decide if a code module can be accepted, internal attributes (such as size, complexity, number of defects, etc.) are measured. But the final decision is Boolean, accept-reject. We call this problem aggregation. Simple measures have to be aggregated in a single view to help the decision.

In the literature, evaluations, and specifically aggregations, are mostly dealt with using the Weighted Average Sum (WAS) approach. The problem with WAS is that it requires that the measures have interval scales. In real world cases measures with ordinal scales, or judgements on ordinal scales (such as good, average, bad) are much more common. If one or more ordinal scales are involved, the aggregation should be made as if all scales were ordinal. Otherwise, ordinal scales will be treated as if they were ratio, therefore introducing arbitrary information that makes the evaluation unfair.

Kontio [8] uses the Analytic Hierarchy Process (AHP) [12], that fits well the hierarchic nature of quality models used in evaluations, but requires also ratio scales on all measures.

Morisio and Tsoukiàs [9] propose to use an ordinal aggregation method in a COTS product selection evaluation case (see also [14] and [16]). The advantage is that ordinal scales can be used, and that preferences are clearly distinguished from measures. Starting from this work we propose in this paper a method that compares artifacts with predefined profiles. The method applies to any situation where preferences are expressed on an ordinal scale and where alternatives are not compared between them, but to "profiles" in order to fit them in pre-defined categories. Such an approach has already been applied in real world cases (see [10]).

In the following we examine in more detail the concepts of measurement, evaluation, measure, preference, aggregation and their mutual relationships.

2. EVALUATION CONCEPTS

2.1 Measurement and Evaluation

The problem of evaluation of an artifact is often addressed in a confusing way. The basic confusion arises between the measurement of attributes of the artifact and the evaluation of an artifact based on these attributes for any decision purpose. In the first case a measurement is expected to be performed, while in the second the decision maker's preferences have to be modeled. These are two completely different activities (Tab. 1) and have to be treated as such (for a detailed discussion see [2] and [3]).

The construction of a measure requires:

- The definition of the semantics of the measure (what do we measure?);
- The definition of the structure of the metric (what scale is used?);
- The definition of one or more standards (how the measurement is performed?).

On the other hand, evaluating a set of artifacts under a decision perspective requires to answer questions of the type:

- Who evaluates?
- Why it is necessary to evaluate?
- For what purpose is the evaluation?
- How the evaluation has to be done?
- Who is responsible for the consequences?
- What resources are available for the evaluation?
- Is there any uncertainty?

A measure is a unary function $m: A \rightarrow M$ mapping the set of artifacts A to the set of measures M. The set M is equipped with a structure which is the scale on which the measure is established. Such scales can be nominal, ordinal, ratio, interval or absolute. Each type is univocally defined by its admissible transformations. Measuring the elements of A can be done only if M is defined. So, an external reference system and standards are necessary (represented by M).

A preference is usually represented by a binary relation R, $R \subset A \times A$, so that the set A is mapped to itself. We obviously need to know under which conditions $r(x,y)$ $x,y \in A$ is true, but there is no need of external reference system. Typically, an evaluator can decide that he prefers x to y, basing the decision on simple judgement, or using a measure, in both cases this establishes $r(x,y)$ is true.

When R is a complete binary relation ($\forall x,y \in A\ r(x,y) \vee r(y,x)$) then it admits a numerical representation which depends on what other properties R fulfills. For instance if R satisfies the Ferrers property and semi-transitivity (for such concepts see [11]), then it is known that $\exists v: A \rightarrow \Re : r(x,y) \Leftrightarrow v(x) \geq v(y)+k$ (k being a real constant). A typical confusion is to consider the function v as a ``measurement" applied on the set A. Actually there exist an infinity of functions v representing the relation R and any one could be chosen. Since there is no standard (or metric) any of such functions v is just a numerical representation of R, but not a measure. For instance if on the preference relation x is indifferent to y, y is indifferent to z, but z is preferred to x, then two numerical representations of such preferences are

$u(x)=10,u(y)=12,u(z)=14,k=3$ and
$v(x)=50,v(y)=55,u(z)=60,k=6$.

We call criterion a preference relation with a numerical representation.

Finally, if for a given set A a measurement function exists, it is always possible to infer a preference relation from the measurement. However, such a preference relation is not unique

(the fact that two objects have a different length, which is a measure, does not imply a precise preference among them).

Suppose that $\exists\, l{:}A \rightarrow \Re$ (a measure mapping the set A to the reals, let's say a length). Then the following are all admissible:

$r(x,y) \Leftrightarrow l(x) \geq l(y)$
$r(x,y) \Leftrightarrow l(y) \geq l(x)$
$r(x,y) \Leftrightarrow l(x) \geq l(y)+k$
$r(x,y) \Leftrightarrow l(x) \geq 2l(y)$

These are all admissible preference relations, but with an obvious different semantic. The choice of the ``correct'' one depends on the answers on the evaluation questions. An evaluation therefore is always part of a decision aid process and represents its subjective dimension.

Table 1. Properties of measures and preferences

	Measure	**Preference**
Definition	Function	Binary relation
Used for	Measurement	Evaluation
Constraints	Representation condition, meaningfulness	Properties of the binary relation
Obtained by	Measurement (reference system)	Established by the evaluator (possibly using a measure)
Scale	Nominal to absolute	Ordinal to absolute (defined for the corresponding criterion, not for the preference)
Value obtained by	Measurement (reference system)	From measure or from judgement
Choice of aggregation operator	Function of scales of measures and semantics	Function of scales of criteria and semantics

2.2 Aggregation

The differences between measurement and evaluation (seen as preference modeling) reflect also the possibilities we have in order to obtain an aggregated measure or an aggregated preference from sets of measures or sets of preferences. Typically sets of preferences or measures regard a set of attributes that characterize an artifact. But a comprehensive measure or preference relation is needed, which may represent all the different dimensions we want to consider. It is surprising how often the choice of the aggregation operator is done without any critical consideration about its properties. Let's take two examples.

Suppose you have two three dimension objects a,b, for which their dimensions (length, height and depth) are known ($l(a),l(b),h(a),h(b),d(a),d(b)$). In order to have an aggregate measure of each object dimension we may 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 dimensions are prices we may 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 have an idea of the price of a,b as it will make no sense to compute an arithmetic mean in order to have an idea of the volume of a,b. The choice between the geometric and the arithmetic mean depends on the semantics of the single measures and of the aggregated one.

For the next example, suppose you have two artifacts a,b evaluated on two attributes. For each one a complete preference relation (r_1, r_2) is defined.

Let's pass to the numerical representation, defining the criteria g_1 and g_2

$g_1 : A \rightarrow [0,1]$ and $g_1(a)=0$ and $g_1(b)=1$
$g_2 : A \rightarrow [0,2]$ 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 (weighted average sum) to infer the global preference relation.

$g(a)= (g_1(a)+ g_2(a))/2=1$ and
$g(b)= (g_1(b)+ g_2(b))/2=1$

so the two artifacts result to be indifferent. 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). Under the usual hypothesis of equal importance of the two criteria we obtain now $g(a)=1/2$ and $g(b)=3/4$ meaning that b is preferred to a.

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 judgements (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). Second, even if the ratio transformation were admissible, the concept of criteria importance is misleading. In a ``weighted arithmetic mean'' (as the average is) the weights are constants representing the ratios between the scales of the criteria.

In the example, if we reduce g_2 to g'_2 we have to give to g'_2 twice the weight of g'_1 in order to keep true the concept of ``equal importance''.

In other words it is not possible to speak about importance of the criteria (in the weighted arithmetic mean case) without considering the cardinality of their co-domains.

From the above examples we can induce a simple rule. In order to choose appropriately an aggregation operator it is necessary to take in consideration the semantics of the operator and of each single preference or measure and the properties (axiomatic) 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. For a detailed discussion on the above problems the reader can see [4].

Uncertainty can be considered using intervals, fuzzy measures, possibility and/or probability distributions etc., instead of exact evaluations. For each such case, precise procedures apply. In this paper we present a principle of ordinal preference aggregation, not a complete method. To this end, we chose an easy example in order to show how such a family of methods works, not for presenting a definitive method.

3. THE EVALUATION METHOD

In this section we present the evaluation method using a simplified real life case as working example. This case is a variation of the third evaluation type presented in the introduction, 'Decide if a code module can be accepted ...'.

A reuse repository contains reusable assets. These are made of source code and documents describing design and functionality of the asset.

The reuse manager receives the potential assets, and has to verify their quality level to accept them in the repository, or not. For this purpose, the reuse manager, helped by the quality assurance function, builds a quality model. His intuition is to establish a judgement of the type "very good" (VG), "good" (G), "quite good" (QG), "acceptable" (A), "unacceptable" (U) and introduce to the repository assets judged at least "A". Of course reusers can choose assets not only according to functional requirements, but also to the quality level. The reuse manager only has information concerning specific attributes of the assets and finds difficult to define the comprehensive judgements. Actually the reuse manager is facing a problem of measurement aggregation from the single quality attributes to the comprehensive ordinal scale "VG > G >QG > A > U".

In this section we briefly present the method adopted consisting in the following steps (we identify the reuse manager as a decision maker):

Phase 1 - definition of evaluation model
- Definition of quality model
- Definition of criteria
- Definition of profiles and categories

Phase 2 - application of evaluation model
- Selection of artifacts
- Measurement of artifacts
- Aggregation of measures

3.1 Definition of the Evaluation Model

The evaluation model is established defining a hierarchy of attributes and the associated measures. Measures can have any scale, from nominal to absolute.

In our working example, quality for reusable assets is defined, using a constructive quality model approach [7], in terms of code understandability and code reliability. This model is also influenced by the ISO 9126 standard [5], that lists reliability and maintainability as quality characteristics, and suggests understandability as a decomposition of maintainability.

Table 2. Attributes and measures for Code Understandability

Attribute	Subattribute	Measure	Criterion scale
Code understandability			
Complexity	Algorithmic complexity	Mc Cabe's cyclomatic number	Inverse
	Size	LOCs*	Inverse
	Fan out	Number of functions called, not contained in the asset	Inverse
Docume-ntation	Comments on code	(physical lines of code containing comments) / LOCs	Identity
	Descriptive-ness	Unacceptable (U), Acceptable (A), Quite Good (QG), Good (G), Very Good (VG)	VG > G > QG > A > U
	Quantity	Number of pages of documents associated to source code	Identity

*LOCs = Physical lines of code, less comments and blank lines

Table 3. Attributes and measures for Reliability

Attribute	Subattribute	Measure	Criterion scale
Reliability	Branch coverage	Branch coverage (percentage of statements and decisions exercised by test cases)	Identity
	Inspection	Yes (the source code was formally inspected) No	Yes > No
	Defects correction ratio	(Number of defects fixed after release) / (Number of defects reported after release)	Identity
	MTTF	Mean Time To Failure	Identity

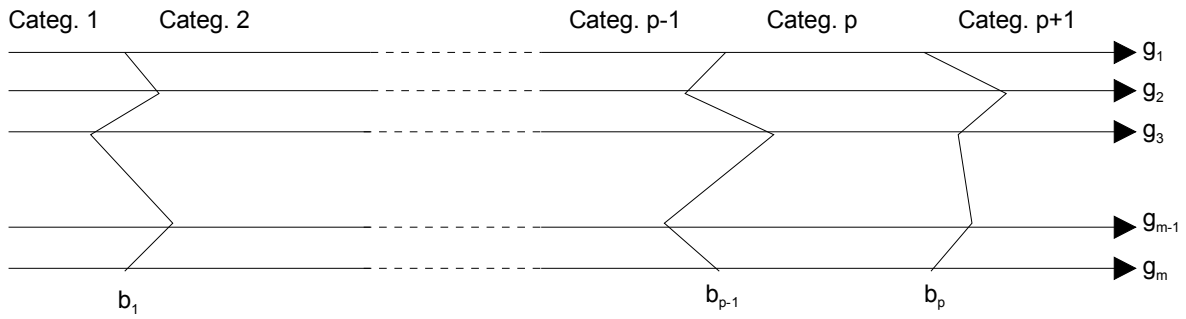

Figure 1: Definition of categories and profiles

Code understandability is further decomposed in complexity and documentation. Next, each leaf quality attribute (complexity, documentation, reliability) is characterized through a number of measures. This step uses a GQM approach [1] and is also influenced by the Reboot reusability model [6]. Refer to Tables 2 and 3 for the complete definition of attributes, subattributes and measures.

3.2 Definition of Criteria/Attributes/Scales

The decision maker willing to express a quality judgement on an ordinal scale, all attributes have to be equipped with at least ordinal scales of measurement. Further on, since the final scale is both a measurement and a criterion (in the sense that obviously VG objects are preferred to G objects, etc.) we have to associate to each attribute a preference model.

For each attribute a correspondent criterion has to be defined, with its scale. While an attribute is neutral, a criterion expresses a preference by an evaluator. For example code size is an attribute that allows to state that a 200 Loc source code module is of larger size than a 100 Loc module. A criterion based on size expresses the preference of an evaluator for larger or smaller modules. In one context an evaluator could prefer larger modules, in another smaller ones.

A criterion can have the same scale as the attribute (identity transformation, larger modules are preferred to smaller modules), or the inverse scale (small modules are preferred to large ones). The same holds for (Documentation) Quantity: a user might prefer to define a more suitable documentation attribute (e.g. Documentation Appropriateness, measured on an ordinal scale), not strictly depending on the number of pages. Another common transformation is defining an ordinal scale for the criterion starting from a nominal scale for the attribute. Other transformations are possible, but we will not deal with them in this paper.

The rightmost column of tables 2 and 3 shows how the scale of the criterion was defined starting from the scale of the attribute. The attribute Descriptiveness uses an ordinal scale, and depends on the judgement of the reuse manager. The attribute Inspection uses a measure with nominal scale (values yes no), the corresponding criterion uses an ordinal scale. For all other criteria the scale is the same as for the attribute, or the inverse one.

3.3 Definition of Profiles and Categories

Next, profiles and categories (see figure 1) have to be defined. The criteria of the evaluation model compose a tree, for instance criterion g_0 decomposes in criteria $g_1, g_2, .. g_n$. A profile for g_0 is a set of values, one for each criterion g_i. In figure 1 $g_1..g_m$, indicate generic criteria, $b_1..b_p$ generic profiles, that define p+1 categories. In our method b_h represents the upper limit of category C_h and the lower limit of category C_{h+1}.

In our working example, four profiles and five categories (Very good (VG), Good (G), Quite good (QG), Acceptable (A), Unacceptable (U)) are defined for each composed criterion, see tables 4, 5 and 6.

Table 4: Profiles for criteria Complexity and Documentation

Composed criterion	Criterion	Profile A	Profile QG	Profile G	Profile VG
Complexity	Algorithmic complexity	8	6	4	2
	Size	10000	5000	2000	1000
	Fan out	20	10	7	5
Documentation	Comments on code	10%	20%	30%	40%
	Descriptiveness	A	QG	G	VG
	Quantity	0	10	100	1000

Table 5: Profile for criterion Code Understandability

Composed criterion	Criterion	Profile A	Profile QG	Profile G	Profile VG
Code Understandability	Complexity	A	QG	G	VG
	Documentation	A	QG	G	VG

Table 6: Profile for criterion Reliability

Composed criterion	Criterion	Profile A	Profile QG	Profile G	Profile VG
Reliability	Branch coverage	20%	40%	60%	100%
	Inspection	No	Yes	Yes	Yes
	Defects correction ratio	50%	70%	80%	100%
	MTTF [hours]	1000	5000	8000	10000

3.4 Selection, Measurement

At this point Phase II starts. Elements to be evaluated are selected and identified. In our working example assets are produced and submitted to the reuse manager. Next, elements are measured on each attribute of the evaluation module. In the example, these measures are taken partially by the project that produces the asset, partially by the reuse manager. As already noted, some attributes are judged and not measured, such as Descriptiveness. Table 7 reports values for four assets to be evaluated on Code Understandability.

Table 7: Values for attributes related to Code understandability

Composed criterion	Criterion	Asset p0	Asset p1	Asset p2	Asset p3
Complexity	Algorithmic complexity	2	2	5	2
	Size	2378	4277	9501	1010
	Fan out	6	15	20	5
Documentation	Comments on code	15%	15%	5%	40%
	Descriptiveness	U	U	A	VG
	Quantity	0	0	50	1000

3.5 Aggregation

The aggregation phase assigns an element to be evaluated to a category of the root criterion in the tree. The aggregation is performed using an algorithm inspired by the ELECTRE-TRI procedure [17], defined in the Multi Criteria Decision Aid approach [15].

The basic concept of the algorithm chosen is the Outranking relation S, which has to be read as *is at least as good as*, and has to be computed between each element and each profile. The outranking relation holds if the concordance and non-discordance tests are satisfied.

The *concordance test* is the majority strength to be reached in order to be able to establish with a certain degree of confidence the outranking relation. Such a majority is generally computed using the relative importance (weight) of each criterion.

The *non-discordance* test is the minority strength not to be reached in order to be able to establish the outranking relation. Such a minority is generally computed using the relative importance of each criterion.

Formally, for each ordered pair *(x, y)*, where *x* and *y* stand for *a* and b_h or viceversa, and for a set of criteria *G* in which a composed criterion is decomposed:

$$\begin{cases} G^{+} = \{ g_j \in G : p_j(x, y) \} \\ G^{=} = \{ g_j \in G : i_j(x, y) \} \\ G^{-} = \{ g_j \in G : p_j(y, x) \} \\ G^{\pm} = G^{+} \cup G^{-} \end{cases}$$

where $p_j(x, y)$ means that *x* is preferred to *y* on criterion g_j while $i_j(x, y)$ means that *x* and *y* are indifferent on criterion g_j.

Let w_j be the relative importance of a criterion, with $\sum w_j = 1$,

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

the non-discordance relation is:

$$\neg D(x,y) \Leftrightarrow \sum_{j \in G^{-}} w_j \leq d \wedge \forall g_j \in G : \neg v_j(x,y)$$

The discordance relation *C(x,y)* has a different definition if the element (*a*) is compared with the profile (*b*) or viceversa.

$$\begin{cases} C(a,b) \Leftrightarrow \left(\sum_{j \in G^{\pm}} w_j \geq c \wedge \sum_{j \in G^{+}} w_j \geq \sum_{j \in G^{-}} w_j \right) \\ C(b,a) \Leftrightarrow \left(\sum_{j \in G^{\pm}} w_j \geq c \wedge \sum_{j \in G^{+}} w_j \geq \sum_{j \in G^{-}} w_j \right) \vee \left(\sum_{j \in G^{+}} w_j > \sum_{j \in G^{-}} w_j \right) \end{cases}$$

with:

- *c*: concordance threshold;
- *d*: discordance threshold;
- c+d ≠ 1;
- $v_j(x,y)$: veto, expressed on criterion g_j, of *y* on *x*.

When the relation *S* is obtained, the assignment of an element to a category can be done in two ways:

1) Pessimistic assignment:

- a is iteratively compared with b_i, for i = p, p-1, ..., 0,
- as soon as a profile b_h exists for which $S(a, b_h)$ then a is assigned to the category C_h.

2) Optimistic assignment:

- a is iteratively compared with b_i, for i = 1, 2, ..., p,
- as soon as a profile b_h exists for which $S(b_h, a) \wedge \neg S(a, b_h)$ then a is assigned to category C_{h-1}.

The pessimistic procedure finds the profile for which the element is not worst. The optimistic procedure finds the profile against which the element is surely worst. If the optimistic and pessimistic assignments coincide, then no uncertainty exists for the assignment. Otherwise, an uncertainty exists and should be considered by the evaluator.

Let's show how this works on our example. Aggregation will be limited to the Code Understandability criterion. Consider asset **p0** and the sub-node complexity. The performance vector of **p0** is [2, 2378, 6] (from Table 7.). The best profile to which **p0** is "at least as good as" is G ([4, 2000, 7]), therefore the pessimistic assignment is in class G. The worst profile which is strictly better than **p0** is VG ([2, 1000, 5]), therefore the optimistic assignment is in class G.

Table 9: Categories of assets for Complexity and Documentation

Composed criterion	Criterion	Asset p0	Asset p1	Asset p2	Asset p3
Code Understandability	Complexity	QG	QG	A	VG
	Docume-ntation	A	A	A	VG

Table 10: Categories of assets for Code Understandability

Criterion	Asset p0	Asset p1	Asset p2	Asset p3
Code Understandability	A	A	A	VG

Tables 9 and 10 show the allocation of assets to categories on the nodes code understandability, complexity and documentation. In all cases the pessimistic and the optimistic assignment coincide. For all composed criteria, composing criteria have the same weight. In all cases the thresholds used are 70% for the concordance threshold, 28% for the discordance threshold (these figures are commonly used in literature and thus introduced in this example, usually it is the decision maker who provides this information).

4. DISCUSSION

A new method to evaluate software artifacts has been presented. The method distinguishes between measures and preferences and uses an ordinal aggregation operator. Both points are essential, as evaluations are decision problems that, even if they use measures as a starting point, involve judgement; and because real life evaluation models often use ordinal measures that require ordinal aggregation operators.

The application of the method has shown that the definition of the evaluation model is a difficult task, probably the most difficult in an evaluation problem. One problem is the decomposition in attributes and subattributes. In some parts (for instance attributes branch coverage, inspection, and defects correction ratio) this corresponds to defining a predictive model, where the difficulty is in validating it.

Another problem lies in the definition of profiles, and therefore categories. We have discovered that four profiles and five categories are probably too many. Both empirical and intuitive evidence of how the value of a measure discriminates assets and therefore defines profiles is missing. Accordingly, the next version of the evaluation model will be built with two profiles and three categories (reject, acceptable, good) only.

Initially, reliability and understandability were supposed to be aggregated in a final evaluation considering both of them. Actually, this further aggregation was not performed, because it did not correspond with the need of the final user of an asset who decides to use an asset in function of understandability only. The evaluation on reliability is used by the reuse manager to reject some assets, then the user selects on understandability only. In other words, two evaluation models are actually used, one on understandability, by the user and the reuse manager, one on reliability, by the reuse manager only.

This situation could change in a safety critical systems context, where a user could be constrained to select an asset in function of the class of risk of the project, or part of project. Reliability categories of assets would be mapped to classes of risk, and the user should select accordingly. This situation will be the object of further research.

5. ACKNOWLEDGMENTS

Our thanks to the anonymous reviewers for their helpful comments

6. REFERENCES

[1] Basili V.B., Rombach H.D. (1988). The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on software engineering, 14,6 (June 88 758-773.

[2] Blin M.J., Tsoukiàs A., ``Evaluation of COTS using multi-criteria methodology'', in Proceedings of the 6th European Conference on Software Quality (1999) 429 - 438.

[3] Blin M.J., Tsoukiàs A. Multicriteria Methodology Contribution to the Software Quality Evaluation, Software Quality Journal 9,2 (June 2001) 113-132.

[4] Bouyssou D., Marchant Th., Perny P., Pirlot M., Tsoukiàs A., Vincke Ph., Evaluation and Decision Models: a critical perspective, Kluwer Academic, Dordrecht (2000).

[5] ISO/IEC JTC1, International Standard 9126 Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for their Use (1991) Geneva.

[6] Karlsson, E.A. Software Reuse. John Wiley & Sons (1995).

[7] Kitchenham, B. Towards a constructive quality model. Part 1: software quality modeling, measurement and prediction. Software Engineering Journal (July 1987) 105-113.

[8] Kontio, J. A Case Study in Applying a Systematic Method for COTS Selection, in Proceedings of the 18th Int. Conf. on Software Engineering (1996) 201-209.

[9] Morisio, M., Tsoukiàs, A. IusWare: A methodology for the evaluation and selection of software products. IEE Proceedings Software Engineering (June 1997) 162-174.

[10] Paschetta E., Tsoukiàs A., ``A real world MCDA application: evaluating software'', Journal of Multi-Criteria Decision Analysis, 9 (2000) 205 - 226.

[11] Roubens M., Vincke, Ph. Preference Modeling, LNEMS 250, Springer Verlag (1985).

[12] Saaty, T. The analytic hierarchy process. Mc Graw Hill, NY (1980).

[13] Stamelos, I., Tsoukiàs, A. Software Evaluation Problem Situations. Cahier du LAMSADE, No 156, Université Paris Dauphine, to appear in European Journal of Operational Research.

[14] Stamelos I., Vlahavas I., Refanidis I., Tsoukiàs A., ``Knowledge Based Evaluation of Software Systems: a case study'', Information and Software Technology, 42 (2000) 333 - 345.

[15] Vincke, Ph. Multicriteria Decision Aid. John Wiley (1992).

[16] Vlahavas I., Refanidis I., Stamelos I., Tsoukiàs A., ``ESSE: an expert system for software evaluation", Journal of Knowledge Based Systems, 12 (1999) 183 - 197.

[17] Yu, W. Aide multicritere a la decision dans le cadre de la problematique du tri: methodes et applications LAMSADE, Université Paris Dauphine, Paris (1992).