International Journal of Software Engineering and Knowledge Engineering
Vol. 0, No. 0 (1994) 000-000
© World Scientific Publishing Company

SOFTWARE PRODUCT AND PROCESS ASSESSMENT THROUGH PROFILE-BASED EVALUATION

MAURIZIO MORISIO
Politecnico di Torino - 24, Corso Duca degli Abruzzi,
10129 Torino Italy
maurizio.morisio@polito.it

IOANNIS STAMELOS
Department of Informatics, Aristotle University,
54124 Thessaloniki Greece
stamelos@csd.auth.gr

ALEXIS TSOUKIAS[1]
Lamsade – CNRS Université Paris Dauphine,
Paris, France
tsoukias@lamsade.dauphine.fr

Received (31 January 2003)
Revised (1 August 2003)
Accepted ()

Software entities (software products or processes) are characterized by many attributes, each one in its turn can be measured by one or more measures. In several cases the software entities have to be evaluated as a whole, thus raising the problem of aggregating measures to give an overall, single view on the software entity. This paper presents a method to aggregate measures, which works comparing the product/process with predefined, ideal entities, 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. It is presented through a simplified case study that deals with evaluating the level of quality of an asset to decide if accepting it in a reuse repository. A plausible way of how the method could be applied to process maturity assessment is also provided. 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.

Keywords: Software evaluation; quality models; multiple criteria decision aiding.

1. Introduction

Software products and processes 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. Alternatively, process attributes may be modeled around key process areas.

[1] This paper was finished while the third author was visiting DIMAC, Rutgers University, under NSF CCR 00-87022 grant, the support of which is gracefully acknowledged.

Sometimes entities need to be evaluated as a whole, not only on each attribute alone. Examples of evaluations are (see also [23]):

- 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 to each other, 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/process. 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 or process is evaluated.

On the other hand, a software entity may represent a broad class of Information Technology concepts (a programming language, a software development approach, a software organization, etc.). Examples of such practical evaluation situations 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 with the solution of 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 entities (e.g. selection of COTS product) or only one. In the latter case, a second entity (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 [11] uses the Analytic Hierarchy Process (AHP) [22], that fits well the hierarchic nature of quality models used in evaluations, but requires also ratio scales on all measures. If we apply such a procedure on ordinal information we obtain meaningless results. A mathematical treatment of the above issues and a detailed analysis on aggregation problems and drawbacks can be found in [3] (see also [15]).

Morisio and Tsoukiàs [13] propose to use an ordinal aggregation method in a COTS product selection evaluation case (see also [24] and [26]), but limit their examples only to ranking problems. 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 software entities 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. A similar approach has already been applied in real world cases (see [15]). In this paper we describe in full detail such an approach, providing also the theoretical justification for its use. In addition, we provide a complete application example and extend the approach to process evaluation as well.

In the following we examine in more detail the concepts of measurement, evaluation, measure, preference, aggregation and their mutual relationships. The paper is organized as follows. Section 2 focus on the differences between measuring objects and evaluating them under a decision perspective, that is establishing a preference relation. Section 3 presents a new method based on comparisons between alternatives and pre-established profiles of categories. A product assessment problem is shown in this section. Section 4 applies the method on a process assessment problem, while section 5 briefly discusses the results obtained.

2. Evaluation Concepts

2.1 Measurement and evaluation

The problem of evaluation of an entity is often addressed in a confusing way. The basic confusion arises between the measurement of attributes of the entity and the evaluation of an entity 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 entities under a decision perspective requires answering questions of the type:

- Who evaluates?
- Why is it necessary to evaluate?
- For what purpose is the evaluation?
- How has the evaluation 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 entities 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 [18]. 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 [25] is usually represented by a binary relation R, $R \subseteq A \times A$, 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 that $r(x,y)$ is true.

When R is a complete binary relation ($\forall$ $x,y \in A$ $r(x,y) \vee r(y,x)$) then it may admit 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 [19]), 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 where the concept of measure intuitively implies that any kind of arithmetic operation is allowed. 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 any type of 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

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

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:

$l(x) \geq l(y) \Leftrightarrow r(x,y)$ (x is better than y if the length of x is bigger than the length of y)
$l(x) \geq l(y)+k \Leftrightarrow r(x,y)$ (x is better than y if the length of x is bigger than the length of y plus a threshold)
$l(x) \geq 2l(y) \Leftrightarrow r(x,y)$ (x is better than y if the length of x is bigger than twice the length of 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 aiding process and represents its subjective dimension (different decision makers might consider different preferences although the lengths are the same). The reader can see more about meaningful and admissible models and operations in [5], [17] and [18].

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 entity. 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, which is

$$v(a)=l(a)h(a)d(a) \text{ and } v(b)=l(b)h(b)d(b).$$

If the three dimensions are prices we may use an average, that is

$$p(a)=l(a)+h(a)+d(a)/3 \text{ and } p(b)=l(b)+h(b)+d(b)/3.$$

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 entities *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

$$g_1 : A \rightarrow [0,1] \text{ and } g_1(a)=0 \text{ and } g_1(b)=1$$
$$g_2 : A \rightarrow [0,2] \text{ and } g_2(a)=2 \text{ and } g_2(b)=1.$$

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.

$$g(a)= (g_1(a)+ g_2(a))/2=1 \text{ and}$$
$$g(b)= (g_1(b)+ g_2(b))/2=1$$

so the two entities 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 whether 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 maintain 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 may consult [5].

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 universal, definitive method.

3. The Evaluation Method: Product Assessment

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". "Unacceptable" occurs when either complexity or documentation becomes "unacceptable".

Of course reusers can choose assets not only according to functional requirements, but also according to the quality level. The reuse manager has information concerning only 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 of 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 entities
- Measurement of entities
- 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.

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
Documentation	Comments on code	(Physical lines of code containing comments) / LOCs	Identity
	Descriptiveness	Unacceptable (U), Acceptable (A), Quite Good (QG), Good (G), Very Good (VG)	VG > G > QG > A > U
	Appropriateness	Unacceptable (U), Acceptable (A)	A > U

*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

In our working example, quality for reusable assets is defined, using a constructive quality model approach [9], in terms of code understandability and code reliability. This model is also influenced by the ISO 9126 standard [6], which lists reliability and maintainability as quality characteristics, and suggests understandability as a decomposition of maintainability.

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 [8]. Refer to Tables 2 and 3 for the complete definition of attributes, subattributes and measures.

3.2 Definition of criteria/attributes/scales

Given that the decision maker is willing to express a quality judgment 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 stating 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). Another common transformation is defining an ordinal scale for the criterion starting from a nominal scale for the attribute. Other transformations are possible (e.g. a preference may be expressed as an interval), but we will not deal with them explicitly 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, possibly using company specific guidelines. 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.

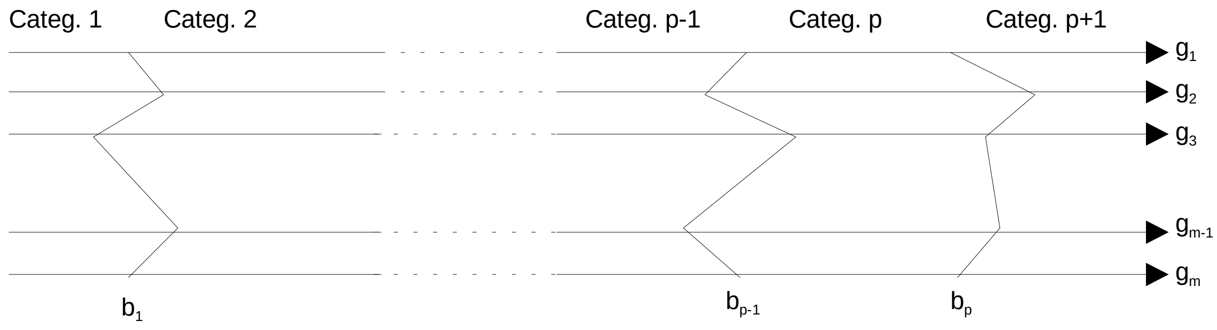

Fig. 1: Definition of categories and profiles

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 example, four profiles and five categories (Very good (VG), Good (G), Quite good (QG), Acceptable (A), Unacceptable (U)) are defined for each composed criterion (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
Documentatio n	Comments on code	10%	20%	30%	40%
	Descripti- veness	A	QG	G	VG
	Appropria- teness	A	A	G	G

Table 5: Profile for criterion Code Understandability

Composed criterion	**Criterion**	**Profile A**	**Profile QG**	**Profile G**	**Profile VG**
Code Understanda- bility	Complexity	A	QG	G	VG
	Documenta- tion	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 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
Docume- ntation	Comments on code	15%	15%	5%	40%
	Descriptiveness	U	U	A	VG
	Appropriateness	U	U	A	A

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 [14], [21], [27]. For similar methods see also [10], [12]. The method uses what is known in literature as an "outranking relation".

Outranking relations represent one of the two basic tools used in order to explore a set of feasible solutions when several criteria have to be considered and a reasonable compromise is requested. Outranking relations simply translate (and expand) in the decision aiding context the concepts of majority procedures used in voting and social choice theory. In other terms, an alternative x is considered to be "at least as good as" y iff the "weighted" majority of criteria agree so. In order to establish a choice or a ranking on a set of alternatives, pairwise comparisons are performed among the alternatives, allowing to establish where the outranking holds and then an ordering relation is constructed. Methods using such an approach expand their flexibility, introducing, besides the positive reasons for which x is supposed to be better than y (the weighted majority), the negative reasons (not supporting the sentence "x is at least as good as") represented by vetos, thresholds and blocking minorities (for more details see [4], [5]).

Typically, in order to establish a ranking, each alternative is compared to all other alternatives in turn. This is accomplished through the basic concept of the algorithm chosen, namely 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 vice versa, and for a set of criteria *G* in which a composed criterion is decomposed:

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

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 concordance relation *C(x,y)* has a different definition if the element (*a*) is compared with the profile (*b*) or vice versa.

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

with:

- *c*: concordance threshold;
- *d*: discordance threshold;
- $c+d \neq 1$;
- $v_j(x,y)$: 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. When the two assignments do not coincide, it means that there are strong incomparabilities between the alternatives and the profiles of the categories, demanding further discussion with the decision maker. The safe ultimate rule is to use the pessimistic assignment.

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 Under-standability	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 code understandability, complexity and documentation nodes. 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 [5] and thus have been introduced in this example, usually it is the decision maker who provides this information).

4. The Evaluation Method: Process Assessment

In this section we show how our profile-based approach may be applied to process assessment methods. Process assessment methods involve the evaluation of software processes as a whole, rather than focusing on individual activities [17]. One of the most known such models is CMM (Capability Maturity model) [16]. According to CMM a software organisation is rated on an ordinal scale: level 1 (Initial), level 2 (Repeatable), level 3 (Defined), level 4 (Managed), level 5 (Optimising). Determination of the appropriate level is accomplished by examining certain Key Process Areas (KPAs), different for each level. For example, among other, you need to master KPA Software Project Planning in order to obtain level 2. The implementation and institutionalisation of a KPA is assessed according to a number of Key Practices, common to all KPAs independently of the level the KPAs belong to. Examples of such practices are Activities performed, Measurement and analysis, etc.

Evaluation in CMM is conducted through CBA-IPI, i.e. the CMM-based appraisal for internal process improvement. The assessment is based on the use of maturity questionnaires, documentation and interviews. Questions ask whether a key practice for a specific KPA is followed and may be answered “yes”, “no”, “don’t know” and “does not apply”. Ratings are subjectively assigned, so team consensus and consolidation of findings is frequently needed [7]. If all key practices are satisfied for a KPA then it is considered that all KPA goals are also satisfied. In case some KPA goal is not satisfied then it is examined whether an alternative organization practice exists that can satisfy the goal. If all KPAs of a given CMM level are satisfied and all KPAs for lower levels are also satisfied then the organization is considered to have reached that level.

CBA-IPI examines both organization project and process profiles in order to build the organization maturity profile. The use of profiles in process assessment is even more explicit in SPICE [20]. However, the assessment method can be considered quite rigid. For example, Pfleeger ([17], p. 549) emphasizes that: “…It is important to remember that capability maturity does not involve a discrete set of possible ratings. Instead, maturity represents relative locations on a continuum from 1 to 5…”, and “…If one part of a process is more mature than the rest, an activity or tool can enhance the visibility of that part and help to meet overall project goals, at the same time bringing the rest of the process up to a higher level of maturity…”. On the other hand, sometimes “big-bang” approaches [7] are suggested in order to reach higher CMM levels, involving significant risks. In the next, we will use CMM terminology, although the ideas presented might be applied to other process assessment methods as well.

The profile-based approach may be used to accommodate extensions of existing process assessment methods, or implement already foreseen profile-based evaluation within such methods. In particular, an extension of a process assessment method might include intermediate values for maturity levels, i.e. an organization might receive ratings between level “repeatable” and level “defined”. While keeping the order of the maturity levels, such an assessment may be useful to show the progress that one organization is making while moving from one level to the higher one and depict the “distance” the organization has to cover in order to reach the next higher level. At a finer grain of detail, assessment might involve the rating of a key practice with intermediate values between “yes” and “no” and the use of other, more refined and, probably, more appropriate measurement scales for factors affecting key practices, in order to produce such intermediate values.

4.1 Definition of criteria, profiles, categories

For process assessment purposes, profiles and categories (see again figure 1) must be defined. A process assessment criteria tree may be defined as follows. Criterion g_0 (process maturity) is decomposed in criteria g_1, g_2, .. g_n, where g_i represents a KPA. A KPA is further decomposed and evaluated according to its preset goals. One may proceed further by evaluating KPA goals according to the key practices.

Process maturity may be rated according to standard levels, i.e. 1 (initial), 2 (defined), …, 5 (optimizing), while additional intermediate levels, i.e. 1-2 (intermediate between initial and repeatable), 2-3 (intermediate between repeatable and defined), etc., may be used to add flexibility to the model. Alternatively, each maturity level might be represented through three profiles, namely Reached (R), On-going (O), Not Reached (NR).

In this example, three profiles and four categories (Satisfied (SA), Improving (IG), Improvable (IE), Starting (ST)) are defined for each KPA. Table 11 shows plausible profiles for KPA Software Project Planning and its goals. For process assessment flexibility purposes, goal satisfaction may be measured in terms of percentage of projects for which the goal is found to be achieved. Of course other means of measurement may also apply. For example, goal satisfaction may receive values on the ordered scale “very good” (VG), “good” (G), “quite good” (QG), “initial” (I), quite similar to the one used in the previous software product assessment example, reflecting the degree of their fulfillment as judged by the process evaluators. Table 12 shows how intermediate maturity levels may be implemented as profiles according to KPA assessment. In Table 12, KPA_k denotes a set representing all KPAs that belong to level *k*.

Table 11: Profiles for one Key Process Area

Composed criterion (Key Process Area)	Criterion (Goal)	Profile ST	Profile IE	Profile IG	Profile SA
Software Project Planning	Estimate documentation	I	QG	G	VG
	Planned and documented project activities and commitments	25%	50%	75%	100%
	Affected groups and individuals agreement	25%	50%	75%	100%

Table 12: Profiles for Process Maturity

Composed criterion	Criterion (KPAs)	Level 2 Repeatable	Level 2-3	Level 3 Defined	Level 3-4	...
Process Maturity	KPA_1	SA	SA	SA	SA	
	KPA_2	ST	IG	SA	SA	
	KPA_3	ST	ST	ST	IG	
	KPA_4	ST	ST	ST	IG	
	KPA_5	ST	ST	ST	ST	

In addition to a single organization certification, other situations may require the comparison of the process maturity of many organizations at once, e.g. when determining the most mature organization during tenders or when an acquisition department creates vendor lists according to predefined profiles. In these cases, as a final step, the aggregation mechanism, described in the previous section should be used.

5. Discussion

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

The application of the method on product evaluation 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 product attributes branch coverage, inspection, and defects correction ratio) this corresponds to defining a predictive model, where the difficulty lies 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, next versions of the evaluation models presented will be built with two profiles and three categories 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.

Process evaluation has also been proven to be an interesting application field for profile-based evaluation. Profiles are inherent in process evaluation, while subjective assessments are scattered throughout the entire evaluation process, adding difficulty to the rating of a process or organization as a whole. Taking into account these difficulties, we believe that process evaluation may benefit from the approach described in this paper.

6. Acknowledgements

The authors would like to thank the anonymous referees for their helpful, constructive comments.

7. References

1. V.B. Basili, H.D. Rombach, The TAME Project: Towards Improvement-Oriented Software Environments, *IEEE Transactions on Software Engineering*, **14,6** (1988) 758-773.
2. M.J. Blin, A. Tsoukiàs, Evaluation of COTS using multi-criteria methodology, in *Proc. of the 6th European Conference on Software Quality* (1999) pp. 429 - 438.
3. M.J. Blin, A. Tsoukiàs, Multicriteria Methodology Contribution to the Software Quality Evaluation, *Software Quality Journal* **9,2** (2001) 113-132.
4. D. Bouyssou, Outranking relations: Do they have special properties?, *Journal of Multi-Criteria Decision Analysis* **5** (1996) 99-111.
5. D. Bouyssou, T. Marchant, P. Perny, M. Pirlot, A. Tsoukiàs, P. Vincke, *Evaluation and Decision Models: a critical perspective*, (Kluwer Academic, Dordrecht, 2000).
6. ISO/IEC JTC1, International Standard 9126 Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for their Use, Geneva (1991).
7. P. Jalote, *CMM in practice*, SEI Series in Software Engineering, (Addison-Wesley, 2000).
8. E.A. Karlsson, *Software Reuse*, (John Wiley & Sons, 1995).
9. B. Kitchenham, Towards a constructive quality model. Part 1: software quality modeling, measurement and prediction. *Software Engineering Journal* (1987) 105-113.
10. M. Köksalan, C. Ulu, An interactive approach for placing alternatives in preference classes, *European Journal of Operational Research* **144** (2003) 429 - 439.
11. J. Kontio, A Case Study in Applying a Systematic Method for COTS Selection, in *Proc. of the 18th Int. Conf. on Software Engineering* (1996) pp. 201-209.
12. O.I. Larichev, , H.M. Moshkovich, An Approach to Ordinal Classification Problems, *International Transactions in Operational Research* **1,3** (1994) 375-385.
13. M. Morisio, A. Tsoukiàs, IusWare: A methodology for the evaluation and selection of software products, *IEE Proceedings Software Engineering* (1997) 162-174.
14. V. Mousseau, R. Slowinski, P. Zielniewicz. A User-oriented Implementation of the ELECTRE TRI Method Integrating Preference Elicitation Support, *Computers & Operations Research* **27,7-8** (2000) 757-777.
15. E. Paschetta, A. Tsoukiàs, A real world MCDA application: evaluating software, *Journal of Multi-Criteria Decision Analysis* **9** (2000) 205 - 226.
16. M. Paulk, B. Curtis, M. Chrissis, C. Weber, Capability maturity model for software, version 1.1., Technical Report SEI-CMU-93-TR-24, Pitsburgh, PA: Software Engineering Institute (1993).
17. S.L. Pfleeger, *Software Engineering: Theory and practice*, 2nd Ed., (Prentice-Hall, 2001).
18. F.S. Roberts, *Measurement theory, with applications to Decision Making, Utility and the Social Sciences* (Addison-Wesley, 1979).
19. M. Roubens, P. Vincke, Preference Modeling, LNEMS 250 (Springer Verlag, 1985).
20. T.P. Rout, SPICE: A framework for software process assessment, *Software Process Improvement and Practice* **1,1** (1995) 57-66.
21. B. Roy, The Outranking Approach and the Foundations of ELECTRE methods, *Theory and Decision* **31** (1991) 49-73.
22. T. Saaty, *The analytic hierarchy process* (Mc Graw Hill, NY 1980).

23. I. Stamelos, A. Tsoukiàs, Software Evaluation Problem Situations, *European Journal of Operational Research* **145, 2** (2003) 273-286.

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

25. P. Vincke, *Multicriteria Decision Aid* (John Wiley 1992).

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

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