Scientific Programming 13 (2005) 299–316
IOS Press

XMatch: A language for satisfaction-based selection of Grid services

Sergio Andreozzi[a,b,*], Danilo Montesi[b] and Rocco Moretti[b,c]
[a]*Istituto Nazionale di Fisica Nucleare (INFN) – CNAF, Viale Berti Pichat 6/2, 40127 Bologna, Italy*
[b]*Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy*
[c]*Department of Pure and Applied Mathematics, University of Padova, Via G. Belzoni 7, 35131 Padova, Italy*

Abstract Grid systems enable the sharing of a large number of geographically-dispersed resources among different communities of users. They require a mapping functionality for the association of users requests expressed in terms of requirements and preferences to actual resources. This functionality should deal with a potentially high number of similar resources and with the diversity of the perceived satisfactions of users. We propose XMatch, a query language enabling the expression of the user request in terms of the expected satisfaction over XML-based representation of available resources. This language offers a compact way for users to express their preferences for Grid resources and enable the maximization of the global preference.

Keywords Service evaluation, service selection, Grid, LSP, XQuery

1. Introduction

Grid systems follow a new paradigm of distributed computing that enables the sharing of resources and services that are not subject to centralized control, are geographically dispersed and can dynamically join and leave virtual pools [1]. Users typically express their requests as constraints and preferences using a well-known set of terms defined in an information model and describing the resource characteristics. A resource request may involve different types of resources or more items of the same resource and it requires the capability of expressing inter-dependencies among them. Such inter-dependencies may be verified before starting the service provision or may require the knowledge about the status of resources during the service provision. Given the large number of possible attributes and the need for making an automatic selection, it is essential to define mechanisms enabling the efficient evaluation of the degree to which resources satisfy the request.

In our previous work [2], we have proposed a model for the rigorous representation of service characteristics, for the association of each of their possible values with the user satisfaction and for the aggregation of the single satisfactions in an overall score using a particular logic. In this paper, we propose XMatch, a query language enabling the expression of the user requirements and preferences based on the defined model. This language is inspired by XQuery [3] reusing a set of constructs useful for our goal and providing clauses based on our service evaluation model. Each XMatch query is executed on a snapshot representation of the available Grid resources. In a scenario involving a workflow requiring the selection of resources not only at its start, but also during its execution, different XMatch queries can be defined at different instants. The evaluation of how XMatch can be used together with specific workflow description languages [4] is not in the scope of this paper.

*Corresponding author: Sergio Andreozzi, Istituto Nazionale di Fisica Nucleare (INFN) – CNAF, Viale Berti Pichat 6/2, 40127 Bologna, Italy. Tel.: +39 328 8462871; Fax: +39 051 6092746; E-mail: sergio.andreozzi@cnaf.infn.it.

ISSN 1058-9244/05/$17.00

The paper is structured as follows: in Section 2 we set out our service evaluation model used to measure the satisfaction; in Section 3, we introduce the XMatch language with a number of examples; in Section 4, we use the language in two important use cases concerning the Grid computing area; in Section 5, we present a number of works that are relevant to our area of research; in Section 6, we draw up our conclusions and plan for future work; in Appendix 7, the EBNF (Extended Backus-Naur Form) grammar definition of our language is given.

2. Background on the service evaluation model

In order to understand the goal of the language that we are proposing, it is important to set out the core part of the service evaluation model proposed in [2]. It is an approach tailored to the problem of identifying a service satisfying a request from the requester viewpoint. In particular, we define a model for the rigorous representation of service characteristics, for associating each of their possible values to the satisfaction perceived by a specific viewpoint and for the aggregation of the single satisfactions in an overall score using a particular logic.

Our model does not make assumptions as regards the definition of the attributes and the way values are associated to them. Specific representations are considered by different domains like the area of information modeling, knowledge representation, ontologies and expert systems. As regards the Grid community, GLUE (Grid Laboratory Uniform Environment) Schema [5] and JSDL (Job Submission Description Language) [6] are meaningful proposals in the information modeling area; an ontology-based matchmaker is presented in [7]; Component-Expert technology [8] is proposed in the area of expert systems.

The measurement theory [9] and the Logic Scoring of Preferences (LSP) method [10,11] are the building blocks of our model. The former supports the rigorous representation of properties of interest by defining a mapping from the empirical world into its formal representation, thus enabling to perform meaningful analysis and forecasts. The latter supports the synthesis of the global satisfaction by associating elementary satisfactions to each property and by providing a flexible aggregation mechanism.

2.1. Measurement theory

In this section, we introduce the basic concepts of the measurement theory that are useful to model in a rigorous way the attributes characterizing the entities to which the user is interested in expressing satisfaction predicates. These concepts enable to define an unambiguous machine-processable representation of the domain of interest. We start by considering the following definition [9]:

Definition 1. Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.

A proper insight of this definition requires the investigation of the concepts of *entity* and *attribute*. The former is an object or event of the real world, whereas the latter is a feature of this object or event. For each entity under investigation, each of its attributes needs the expression of criteria in order to determine its measurableness and define its measurement. Empirical relationships are inferred from attributes of entities. Therefore, respective formal relationships must be based on the formalization of these attributes in the models of the entities. We define an empirical relationship as the relationship among entities of the real world based on rules derived from experience. The first step towards the representation of the aspects of interest consists in defining an abstraction of the reality under investigation. An empirical relational system can be defined in the following way:

Definition 2. An empirical relational system is an ordered tuple $\mathcal{ERS} = (\mathcal{EO}, er_1, \ldots, er_n, \star_1, \ldots, \star_m)$ where *$\mathcal{EO}$ is a set of empirical objects*, er_i ($i = 1, \ldots, n$) are k-ary empirical relationships ($k \geqslant 2$) defined in $\mathcal{EO}$, $\star_j$ ($j = 1, \ldots, m$) are binary closed operations between empirical objects in $\mathcal{EO}$.

An empirical relational system describes the objects of the real world ($\mathcal{EO}$) and our empirical knowledge of their attributes (the empirical relationships er_i). We introduce the formal relational system enabling to model each empirical object, relationship and operation:

Definition 3. A formal relational system is an ordered tuple $\mathcal{FRS} = (\mathcal{FO}, fr_1, \ldots, fr_n, \diamond_1, \ldots, \diamond_m)$ where $\mathcal{FO}$ *is a set of formal objects,* fr_i ($i = 1, \ldots, n$) are k-ary formal relationships ($k \geqslant 2$) defined in $\mathcal{FO}$, $\diamond_j$ ($j = 1, \ldots, m$) are binary closed operations between formal objects in $\mathcal{FO}$.

In the formal relational system, empirical objects are mapped into the formal objects, empirical relationships are modeled by the respective formal relationships and empirical operations are mapped into formal operations. The link between the empirical relational system and the formal relational system is represented by the following concepts of *measure* and *measurement scale*.

Definition 4. Let $\mathcal{EO}$ and $\mathcal{FO}$ be a set of empirical objects and a set of formal objects respectively. A measure $\mu : \mathcal{EO} \rightarrow \mathcal{FO}$ is a function mapping each empirical object $eo \in \mathcal{EO}$ into a formal object $\mu(eo) \in \mathcal{FO}$ (the measurement value).

The measure μ enables to define the measurement scale expressing a mapping between the empirical and the formal relational system.

Definition 5. Let $\mathcal{ERS} = (\mathcal{EO}, er_1, \ldots, er_n, \star_1, \ldots, \star_m)$ and $\mathcal{FRS} = (\mathcal{FO}, fr_1, \ldots, fr_n, \diamond_1, \ldots, \diamond_m)$ be an empirical relational system and a formal relational system respectively. Let $\mu : \mathcal{EO} \rightarrow \mathcal{FO}$ be a measure. $\mathcal{S} = (\mathcal{ERS}, \mathcal{FRS}, \mu)$ is a measurement scale if and only if $\forall i \in [1, n], \forall j \in [1, m]$ and $\forall eo_1, \ldots, eo_k, b, c \in \mathcal{EO}$ ($k \geqslant 2$) the following properties hold: $er_i(eo_1, \ldots, eo_k) \iff fr_i\big(\mu(eo_1), \ldots, \mu(eo_k)\big)$ and $\mu(b \star_j c) = \mu(b) \diamond_j \mu(c)$. If $\mathcal{FO} = \mathbb{R}$ the measurement scale $\mathcal{S}$ is a real measurement scale.

Five measurement scales are reputed to be particularly meaningful [9]: (1) the *nominal scale* is used when the measured objects need to be partitioned in classes; (2) the *ordinal scale* is a nominal scale with an order relation among the classes; (3) the *interval scale* is an ordinal scale that preserves the difference among the values associated to the measured objects; (4) the *ratio scale* is an interval scale that maintains constant the ratio of the values associated to the measured objects; finally, (5) the *absolute scale* is a ratio scale representing the only possible scale suitable to measure the attribute under investigation.

We have considered software measurement theory [9] in order to quantitatively express attributes and properties of interest. Without this step, no evaluation method can be used for obtaining meaningful information, therefore we consider the adoption of this theory as a requirement that has to be satisfied before starting to apply our evaluation method.

2.2. LSP

We present two steps included in the Logic Scoring of Preferences (LSP) method [10,11]. The first step that we consider is the definition of the elementary criteria of satisfaction that are functions associated to the values of measurements concerning the attributes of the entities to examine. These functions map each possible value into a number $e \in [0, 1]$ (i.e., $e = 0$ means 'no satisfaction', whereas $e = 1$ means 'full satisfaction') expressing the satisfaction for each possible value. Therefore, the elementary criteria of satisfaction are defined depending on the set of possible measurement values. There are three main ways to define an elementary criterion of satisfaction [10]: an enumeration of all possible values returned by a measurement of an attribute, an absolute classification of these values and a relative classification.

The second step of the LSP method is the synthesis of the satisfaction concerning a feature that can be modeled only by a set of measurements. The method prescribes the definition of functions returning a global satisfaction $E \in [0, 1]$. Such functions are based on the satisfactions $e_1, \ldots, e_n$ defined by n elementary criteria to be aggregated and by their respective weights $w_1, \ldots, w_n$. These weights can be selected in order to reflect the relevance of the attribute that the satisfaction refers to. Besides, they must be positive (i.e., $w_i > 0 \ \forall i$) and normalized (i.e., $\sum_{i=1}^{n} w_i = 1$). A wide spectrum of nonlinear multi-criteria scoring functions has been defined in order to model simultaneity (i.e., e_i and e_j have to be simultaneously greater than a threshold), neutrality, replaceability (i.e., a high value of e_i can compensate a low value of e_j) and other input relationships. These scoring functions, defined in [10], have the following generalized form:

(1) $E = (w_1 {e_1}^r + \ldots + w_n {e_n}^r)^{\frac{1}{r}}$

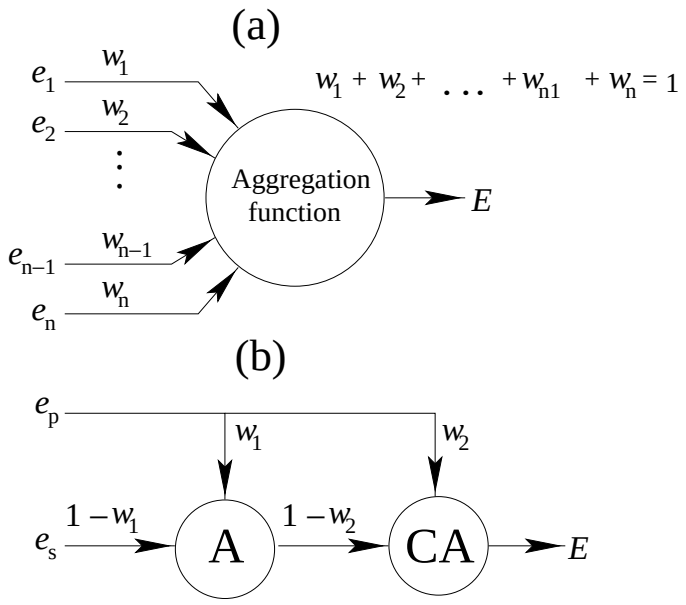

Fig. 1. Aggregation functions: (a) Notation to represent the aggregation functions. (b) The conjunctive partial absorption function.

where $-\infty \leqslant r \leqslant +\infty$. The power r is a parameter selected in order to capture the desired logical relationship and intensity of polarization of the aggregation function. The calculation of the value assigned to the r parameter is based on the number n of satisfactions to be aggregated and on the expected degree of conjunction [10,11]. A useful notation based on [11] is presented in Fig. 1(a). An aggregation function is represented by a circle; for each satisfaction e_i a weighted entry arc is defined ($i = 1, \ldots, n$; $\sum_{i=1}^{n} w_i = 1$); finally, for each circle we have a single exit arc (the synthesized global satisfaction). Besides, the aggregation functions can be composed in order to produce other functions enabling to define aggregation criteria based on a particular logic.

By using this notation, in Fig. 1(b) we introduce a particular scoring function defined in [10]: the *conjunctive partial absorption* function suitable for aggregating a primary satisfaction e_p and a secondary satisfaction e_s. The arithmetic mean function A ($r = 1$) is the input of the medium quasi-conjunction function CA ($r = -0.72$) and the weights are placed as presented in the figure. Apart from the value of the secondary satisfaction e_s, we have $E = 0$ if the principal satisfaction e_p is equal to 0; on the contrary, if $e_p > 0$ the global satisfaction E is positive. Finally, we have the largest global satisfaction ($E = 1$) if $e_p = e_s = 1$.

2.3. The service evaluation model

We start by providing a number of definitions that are the building blocks of our service evaluation model. The first definition associates each attribute to its measurement scale. By this association, the meaningful predicates can be identified.

Definition 6. Let A be the set of attributes of the entities involved in the evaluation. AM is the set of pairs: (a, S_a) where $a \in A$ and $S_a = (\mathcal{ERS}_a, \mathcal{FRS}_a, \mu_a)$ is a measurement scale.

The next definition concerns the set of elementary criteria of satisfaction.

Definition 7. Let A be the set of attributes of the entities involved in the evaluation. EC_A is the set of elementary criteria of satisfaction for each $a \in A$.

The following equivalence relation is defined in order to identify equivalence relevance classes for the attributes.

Definition 8. Let A be the set of attributes of the entities involved in the evaluation, a_i and a_j be elements of A, then $\sim_{REL}$ is an equivalence relation on A defining a collection of pairs (a_i, a_j) where $a_i \sim_{REL} a_j$ means that the satisfaction concerning the values of a_i is as relevant as the satisfaction concerning the values of a_j.

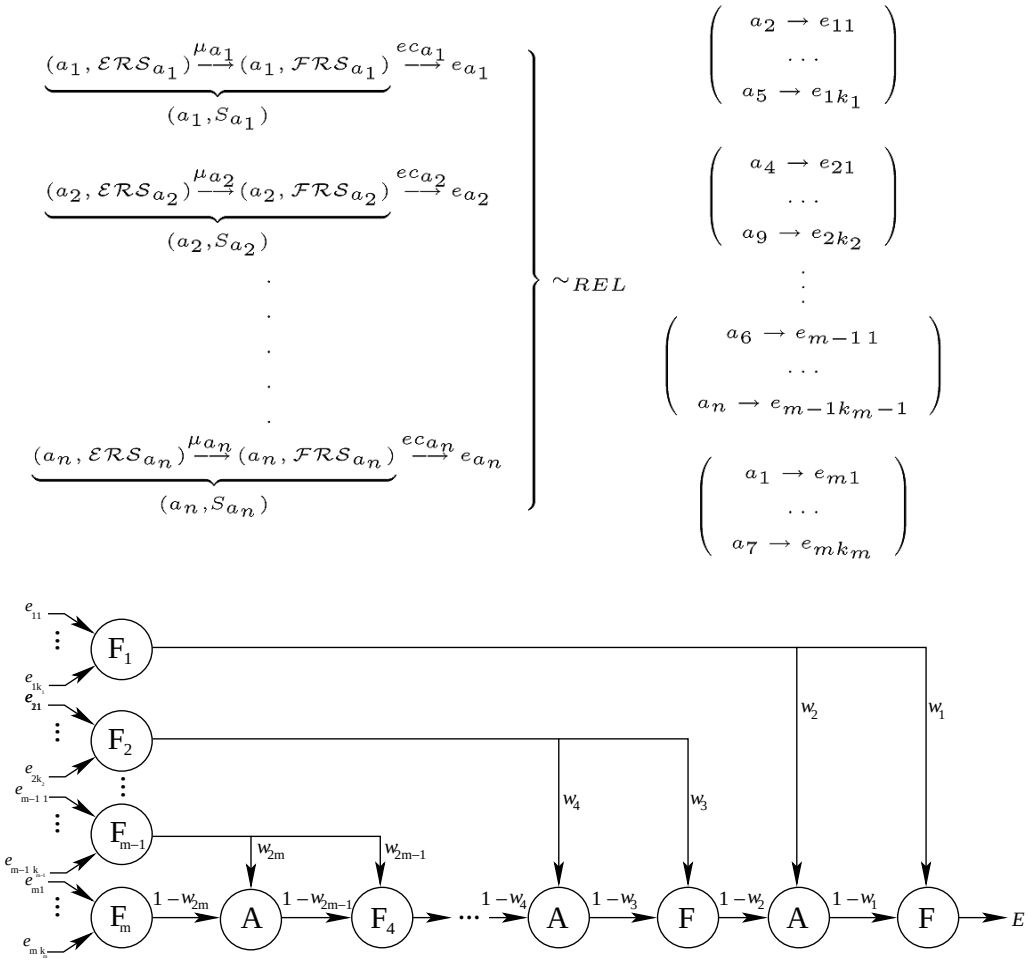

Fig. 2. From attribute values to the overall satisfaction degree.

The next definition introduces the functions needed to aggregate the values returned by the elementary criteria of satisfaction.

Definition 9. AF_{EC_A} is the set of functions for the aggregation of the satisfaction synthesized by the elementary criteria in EC_A.

The definitions given above compose our service evaluation model (SEM) synthesized by the following function:

(2) $SEM : (AM, \sim_{REL}, EC_A, AF_{EC_A}) \rightarrow [0, 1]$

Figure 2 expresses how the arguments of the function SEM have to be used in order to obtain the overall satisfaction degree. Considering the figure: (1) the measurement scales for the attributes under investigation are defined; (2) the satisfaction concerning the value of each attribute is produced by its respective elementary criterion of satisfaction; (3) the satisfactions reputed to be of the same relevance are grouped together by means of $\sim_{REL}$; (4) the overall score is obtained by using the general pattern by firstly (4a) aggregating the satisfactions belonging to the same category and secondly (4b) by performing the aggregation of the satisfactions synthesized for each category.

3. The XMatch language

In this section, we introduce the XMatch language enabling to express queries over XML-based representations of Grid resources by considering also the satisfaction degree that a user perceives as regards the possible values of the attributes of interest. In Appendix 7, the complete grammar is given. The grammar rules are given only for symbols starting with the prefix `XM`, while the other symbols are taken from the XQuery W3C specification [3]. The core part of the XMatch language is an expression defined in the grammar by the symbol `XMExpr` as follows:

(3)
```
XMExpr ::= XMForClause XMLetClause+ XMWhereClause XMReturnClause
```

The rule defining this symbol is inspired by the FLWOR (For-Let-Where-Order by-Return) expression of the XQuery language [3]. The first clause (`XMForClause`) of this expression generates an ordered sequence of tuples of bound variables called the tuple stream. For each set of bound variables generated in this step, the second clause (`XMLetClause`) enables to bound one or more variables to the value returned by an elementary criterion of satisfaction as regards elements belonging to the tuple stream. The third clause (`XMWhereClause`) enables to associate each variable defined in the second clause (i.e., an elementary satisfaction) with a relevance category, thus defining the aggregation pattern. The fourth and last clause (`XMReturnClause`) is used to return the result as an XML document and allows to select a subset of solutions based on their overall satisfaction.

3.1. Generation of the tuple stream

As stated in the previous section, the clause `XMForClause` generates an ordered sequence of tuples of bound variables, called the tuple stream. This is a simplified version of the XQuery `ForClause`, where only the basic constructs are maintained to select elements from an XML document and to generate set of elements by using joins. For each XMatch expression, only one `XMForClause` is allowed with one or more variables to be bound to different types of nodes. The `URILiteral` is defined in the XQuery specification and should refer to a URI that can be resolved to a file containing the data in XML format from which the set of important fragments are extracted. The `OrExpr` is also part of the XQuery specification.

(4)
```
XMForClause     ::= <"for" "$"> VarName "in" XMDocCall
                    ("," "$" VarName "in" XMDocCall)*
XMDocCall       ::= "doc(" URILiteral ")" XMPathExpr? XMPredicate?
XMPathExpr      ::= ("/" QName)+
XMPredicate ::= "[" OrExpr "]"
```

3.2. Expressing the Satisfaction

In this section, we describe the symbol `XMLetClause` as the essential construct for defining a single elementary criterion of satisfaction. The three main categories are supported (see Section 2): an enumeration of all possible values returned by a measurement of an attribute, an absolute classification of these values and a relative classification [10].

(5)
```
XMLetClause    ::= "let" ( XMSimpleEnum | XMCompEnum | XMRange )
XMSimpleEnum   ::= XMPathExpr ValueComp XMElement "satisfies" XMSatLiteral
                   ( "," XMElement "satisfies" XMSatLiteral )*
XMCompEnum     ::= XMPathExprList ValueComp XMElementList "satisfies"
                   XMSatLiteral ( "," XMElementList "satisfies" XMSatLiteral )*
XMRange        ::= XMPathExpr "in" XMElement "to" XMElement "satisfies" "with"
                   ( "linear" "increment" | "linear" "decrement" | "around")
XMPathExprList ::= "(" XMPathExpr ( "," XMPathExpr )+ ")"
XMElementList  ::= "(" XMElement ( "," XMElement )+ ")"
XMElement      ::= ( Literal | XMPathExpr | XMFunctionCall )
XMSatLiteral   ::= ( "0"? "." Digits | "1" )
XMFunctionCall ::= ( "max"|"min"|"avg"|"sum"|"count" ) "(" XMPathExpr ")"
XMPathExpr     ::= ( "/" QName )+
```

The values that are input for an elementary criterion of satisfaction are determined by a simplified XPath expression (`XMPathExpr`). A step in this expression consists only of a child forward step in its abbreviated form with a name node test according to the definitions given in the XPath 2.0 specification [12]. As stated in this specification, the result of an XPath expression is any sequence allowed by the data model. An important characteristic introduced by the new data model [13] is that there is no distinction between an item (i.e., a node or an atomic value) and a singleton sequence containing this item (an item is equivalent to a singleton sequence containing the item and vice versa). As regards XMatch, we only admit as result of an `XMPathExpr` expression a sequence of atomic values. Besides, we have to specify how an elementary criterion of satisfaction is applied to such a type of result. Given the simplified path expression, the resulting sequence is composed by elements with the same qualified name at the same distance from the root node.

We start by describing the elementary criteria of satisfaction when the result of a path expression is a singleton. The first category of elementary criterion of satisfaction can be used when the number of possible values for which the user wants to express an explicit satisfaction is finite. For instance, a user may want to express a number of acceptable possibilities for the operating system type where its job should be executed; the most preferred one is `Scientific Linux`, but `RedHat Enterprise Linux` is acceptable with lower satisfaction. This could be expressed as in Example 6 by using the `XMSimpleEnum` clause.

(6)
```
let $e1 := $CS/OSName eq "Scientific Linux" satisfies 1, "RedHat Enterprise
Edition" satisfies 0.8
```

A more complex use case supported in XMatch is the association of an elementary satisfaction by enumeration to a compound comparison predicate. For instance, if a user wants to express a satisfaction associated to both the name and version of an operating system, this can be done as in Example 7 by using the `XMCompEnum` clause.

(7)
```
let $e1 := ($CS/OSName, $CS/OSVersion) eq ("Scientific Linux", "3") satisfies 1,
("RedHat Enterprise Edition", "3") satisfies 0.8
```

The second category of elementary criterion of satisfaction can be used when the expression of satisfaction is based on parameters independent from attribute values under investigation. The possible functions that map the range of values into a satisfaction are infinite. Our choice is to express in the language three meaningful functions. They are linear and capture an increasing satisfaction, a decreasing satisfaction or a satisfaction centered around a value. An example of usage for the first function (linear increment) is a user requiring a storage service with at least 50 GB of available disk space. Values up to 70 GB produce an increasing satisfaction, while values greater than 70 GB are considered to be equal and fully satisfying. This can be captured in XMatch as in Example 8 by using the `XMRange` clause. The satisfaction is 0 if the value of the attribute is lower than or equal to the lower bound of the range, while it is 1 if the value of the attribute is equal to or higher than the upper bound.

(8)
```
let $e2 := $SS/FreeSpace in 50 to 70 satisfies with linear increment
```

The second linear function (linear decrement) is similar to the previous function, except that for values lower than or equal to the lower bound, the satisfaction is 1 and for values equal to or higher than the upper bound, the satisfaction is 0. For attribute values in the range, the satisfaction linearly decreases. For instance, let us consider a user that is fully satisfied if the response time of a service is lower than or equal to 5 ms, but he will accept values up to 30 ms (see Example 9).

(9)
```
let $e3 := $CS/EstimantedResponseTime in 5 to 30 satisfies with linear decrement
```

The last function should be used when a high satisfaction is associated to values close to the one reputed to be the optimum. For instance, let us consider a user requiring a bandwidth for a network service around 200 Mb/s with a 10% tolerance (see Example 10).

(10)
```
let $e4 := $NS/Bandwidth in 180 to 220 satisfies with around
```

The third category of elementary criterion refers to a relative comparison among the attribute values of all entities in the evaluation context. This criterion requires the introduction of aggregation functions (e.g., min or max). They can be used in both the first and second elementary criteria given above. For instance, let us consider a user requiring the storage service offering the highest free storage capacity (see Example 11).

(11)
```
let $e5 := $SS/FreeSpace eq max($SS/FreeSpace) satisfies 1
```

The use case of a relative elementary criterion concerning a range of values can be expressed by enabling the possibility of using aggregation functions to define the range bounds. For instance, a user may be fully satisfied when he is assigned for the service with the lowest response time among the available ones, but his satisfaction linearly decreases up to 0 when the highest is given.

(12)
```
let $e6 := $CS/EstimantedResponseTime in min($CS/EstimantedResponseTime) to
    max($CS/EstimantedResponseTime) satisfies with linear decrement
```

In the Examples from (6) to (12), the hypothesis is that the path expression returns a singleton, while in the remaining part of this section, we generalize the behavior of the `XMLetClause` to path expressions returning a sequence with multiple atomic values. In order to perform a meaningful evaluation of the satisfaction, we need to introduce the following definitions.

Definition 10. Two distinct elements el_1 and el_2 of an XML document are in intra-evaluation relationship if and only if the qualified name of el_1 is equal to the qualified name of el_2 and they are siblings.

Definition 11. Two distinct elements el_1 and el_2 of an XML document are in inter-evaluation relationship if and only if they are not siblings and the path from the root to el_1 and the path from the root to el_2 are equal.

According to the last definition, two elements in inter-evaluation relationship are identified by the same `XMPathExpr` expression, but they are not siblings. In order to exemplify the two definitions, let us consider the following example.

(13)
```
 1:  <DataSet>
 2:           <A>
 3:              <B>
 4:                 <C>10</C>
 5:                 <C>2</C>
 6:              </B>
 7:              <B>
 8:                 <C>1</C>
 9:              </B>
10:              <B>
11:                 <C>2</C>
12:                 <C>3</C>
13:              </B>
14:          </A>
15:          <A>
16:              <B>
17:                 <C>7</C>
18:                 <C>4</C>
19:              </B>
20:          </A>
21:  </DataSet>
```

The path expression `/DataSet/A/B/C` returns a sequence with seven `C` elements. As an example in this sequence, the first and second element (lines 4 and 5) are in intra-evaluation relationship because they share the same parent node (line 3); the second and the third one (lines 5 and 8) are in inter-evaluation relationship because they have different parent nodes (lines 3 and 7 respectively), but are identified by the same path expression. We

consider values in intra-evaluation relationships as alternative values when determining the satisfaction concerning the attribute defined according to our service evaluation model (see Section 2.3). Therefore, the evaluation of an elementary criterion is made for each value and the higher satisfaction will be selected for the attribute under consideration. As regards values in inter-evaluation relationships, they are reputed to refer to different alternative entities according to our service evaluation model. Therefore, the evaluation of the global satisfaction must be based on only one entity among the alternative ones. Such an entity is selected maximizing the global satisfaction. In order to exemplify the concepts presented by the Example 13, let us consider a user requiring an A element. The user has a linearly increasing satisfaction from 0 to 1 for the values of C from 0 to 10. This elementary criterion can be written in XMatch as expressed in Example 14.

(14)
```
let $e1 := $A/B/C in 0 to 10 satisfies with linear increment
```

By means of the `XMForClause`, the variable `$A` is bound to each `A` element in the document presented in the Example 13. Let us consider the first `A` element: according to Definition 10, the `C` elements in lines 4 and 5 are in an intra-evaluation relationship. The same relationship holds for the `C` elements in lines 11 and 12. For each subset of the elements that are in intra-evaluation relationship (i.e., 10 and 2 in lines 4 and 5; 1 in line 8; 2 and 3 in lines 11 and 12), a representative providing the maximum satisfaction is selected and returned as part of the sequence assigned to the `$e1` variable, that is (1, 0.1, 0.3). This demonstrates how the intra-evaluation relationships can be resolved within the single elementary criterion. According to Definition 11, the three satisfactions in the sequence refer to attributes that are in an inter-evaluation relationship. This implies that in order to select the satisfaction participating to the aggregation producing the global satisfaction, we need to relate the values returned by different elementary criteria considering also their relevance. A complete example is given in Section 4. Following the same rules, the elementary criterion applied to the second `A` element will bound the variable `$e1` with the singleton sequence (0.7).

3.3. Classifying the satisfactions by relevance

In this section, we explain how the association of an elementary satisfaction to a relevance category is modeled in the XMatch language. Potentially, the relevance categories can be infinite, but only three of them are introduced as they are sufficient for meaningful use cases. They are defined in the XMatch language grammar by using the following string literals: `essential`, `desirable` and `optional`. The advantage of such a definition is the improvement of the legibility of XMatch queries. A possible approach to generalize the language to an high number of relevance classes is to use natural numbers to label the relevance categories. The lower is the natural number associated to a relevance category, the more important is the satisfaction.

(15)
```
XMWhereClause    ::= "where" XMRelevanceExpr ( "and" XMRelevanceExpr )*
XMRelevanceExpr ::= ( "$" VarName "is" | XMVarNameList "are" )
                        ( "essential" | "desirable" | "optional" )
XMVarNameList    ::= "(" "$" VarName ( "," "$" VarName )+ ")"
```

Given a set of `XMLetClause` expressions defining elementary satisfactions, each of them can be associated to a relevance category by using the `XMWhereClause` (see Example 16). This provides the meaningful information for building the aggregation pattern.

(16)
```
where ( $e1, $e2 ) are essential and $e3 is desirable
```

Weight and power parameters used in the aggregation pattern (see Fig. 2) are considered to be part of the query processor. Let us consider Example 17 showing an XML document having elements in both intra- and inter-evaluation relationships.

(17)
```
 1:  <DataSet>
 2:           <A>
 3:              <B>
 4:                 <C>10</C>
 5:                 <C>2</C>
 6:                 <D>6</D>
 7:              </B>
 8:              <B>
 9:                 <C>1</C>
10:              </B>
11:              <B>
12:                 <C>2</C>
13:                 <C>3</C>
14:                 <D>9</D>
15:              </B>
16:          </A>
17:          <A>
18:             ...
19:          </A>
20:  </DataSet>
```

We consider a user that wants to select an `A` element for which it is essential that a `C` element is higher than 6 and it is desirable that a `D` element is higher than 8. This expectations can be written in XMatch as showed in the following example:

(18)
```
for $A in doc("data.xml")/DataSet/A,
let $e1 := $A/B/C gt 6 satisfies 1
let $e2 := $A/B/D gt 8 satisfies 1
where $e1 is essential and $e2 is desirable
```

The values returned by the expression `$A/B/C` are in both intra- and inter-evaluation relationships. The same condition holds for the values returned by the expression `$A/B/D`. Considering the rules defined in Section 3.2, the variable `$e1` is bound to the sequence (1, 0, 0), while the variable `$e2` is bound to the sequence (0, 1). In order to choose the satisfaction participating in the aggregation for each elementary criterion, we introduce the ancestor relationship for performing such a choice. This relationship applies to the values of an XML document selected among two or more `XMLetClause`'s and is presented in Definition 3.3.

Definition 12. A value v_{ih} ($h \in [1, m]$) in the i-*th* sequence $(v_{i1}, \ldots, v_{im})$ returned by a path expression $/X/Y$ (where X and Y are path expressions) is in an ancestor relationship with a value v_{jk} ($k \in [1, n]$) in the j-*th* sequence $(v_{j1}, \ldots, v_{jn})$ returned by a path expression $/W/Z$ (where W and Z are path expressions) if and only if the path X is equal to the path W and the number of steps from the element v_{ih} to the element v_{jk} is equal to the number of steps in Y plus the number of steps in Z. If v_{ih} is in an ancestor relationship with v_{jk}, then the same relationship holds for their respective elementary satisfactions.

In order to exemplify this definition, we consider the two path expressions in Example 18 ($/A/B/C$ and $/A/B/D$) applied to the XML document in Example 17. They identify the sequences (10,2,1,2,3) and (6,9) respectively. Considering the Definition 12, X is equal to $/A/B$, Y is equal to C, W is equal to $/A/B$ and Z is equal to D. The first and the second value of the first sequence (i.e., 10 and 2) are in an ancestor relationship with the first value of the second sequence (i.e., 6), while they are not in an ancestor relationship with the second value (i.e., 9). For the final aggregation, we have to select only one value for each elementary criterion. The choice is made by selecting the n-tuple of elementary satisfactions (n is the number of elementary criteria of satisfaction) that are in ancestor relationships and that maximize the global satisfaction. It may happen that it is not possible to produce a complete tuple because the structure of the XML document may lead to the identification of a set of elementary satisfactions that are in ancestor relationship whose cardinality is less than n. In this situation, the missing elementary satisfactions are considered to be equal to 0. In Exampl 18, we have two elementary criteria, therefore we must produce pairs

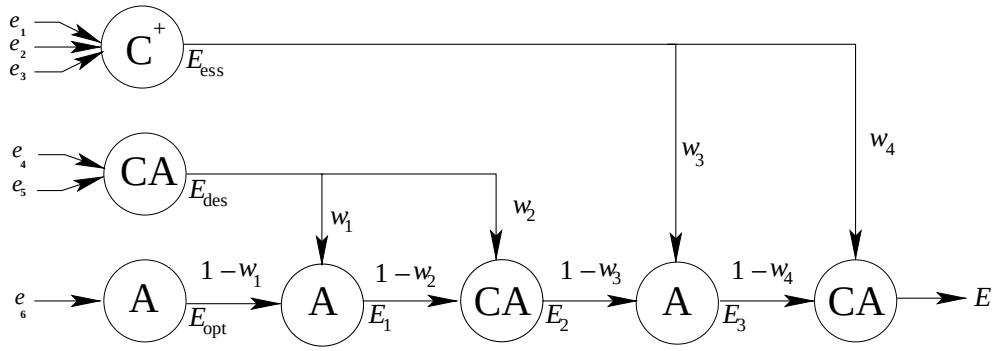

Fig. 3. Example aggregation pattern given by the XMWhereClause.

of elementary satisfactions (i.e., $n = 2$). The sequence $e_1 = (1, 0, 0)$ associated to the sequence of values v_1 is composed of three values, while the sequence $e_2 = (0, 1)$ associated to the sequence of values v_2 is composed of two values. In particular, we can point out that the second value of e_1 (i.e., $e_{12} = 0$) has no respective satisfaction in e_2 for which an ancestor relationship holds. Therefore, the related pair is $(e_{12}, \ )$, where the symbol ' ' expresses the lack of an elementary satisfaction (to be replaced with 0 during the aggregation phase). Summarizing, the tuple of elementary satisfactions that are candidate for the aggregation are: $(1, 0)$, $(0, \ )$ and $(0, 1)$. The tuple that maximizes the global satisfaction is the first one because its first value is essential and equal to 1.

3.4. Constructing the result

In this section, we describe how the result of the query is constructed and returned. Our decision is to define a clause that does not provide any transformation capability. The transformation of the result can be achieved by adding a postprocessing phase using languages like XQuery or XSLT. The `XMReturnClause` clause returns an XML document with a predefined structure as presented in Example 19.

(19)
```
<Results>
        <Result E="0.98">
        ...
        </Result>
        <Result E="0.94">
        ...
        </Result>
    </Results>
```

Each `Result` element contains a set of elements as generated by the `XMForClause` and an `E` attribute with the overall satisfaction associated to the solution. These elements are given following a decreasing order with respect to the value of `E`. Moreover, the number of results can be limited in two ways: by asking the 'Top K' results and by dropping all solutions that do not reach a minimum overall satisfaction.

(20)
```
XMReturnClause ::= "return" "top" digits ( "with threshold" XMSatLiteral )?
```

4. Use cases

In this section, we present two meaningful use cases in the area of Grid computing. The first use case refers to the selection of a set of three different Grid services that better satisfy the user expectations. The XML representation of these services is simplified in the sense that neither intra-evaluation nor inter-evaluation relationships are present. The second use case refers to the selection of a single storage service, but its structure is more complex and presents intra/inter-evaluation and ancestor relationships.

4.1. Computing, network and storage resources

We consider a simplified scenario where core services referring to computing, storage and network resources are defined as follows: the *computing service* is a uniquely identified Grid service that can provide a user software application for computing power in a certain execution environment; the *storage service* is a uniquely identified Grid service that manages storage extents to be used for storing data; finally, the *network service* is a uniquely identified service that offers unidirectional communication capability between network domains that are sets of services sharing the same connectivity (we refer to the model defined in [14]). The following XML fragments represent a computing, a storage and a network service. They are based on a schema and represent the mapping to a formal relational system. In a real scenario, a broker service maintains and continuously updates a cache with the representation of the available services [15]. The broker also receives requests from users and, based on their requirements and preferences, performs the matchmaking phase during which suitable solutions are prepared and one is selected.

(21)
```
<data>
    <CS>
        <URI>httpg://hostname.cnaf.infn.it/CS/</URI>
        <DomainURI>/INFN/CNAF</DomainURI>
        <Type>CS</Type>
        <OSName>Scientific Linux</OSName>
        <OSVersion>3.0.3</OSVersion>
        <ProcessorFamily>Intel Pentium</ProcessorFamily>
        <AssignedJobSlots>30</AssignedJobSlots>
        <WaitingJobs>10</WaitingJobs>
        <RunningJobs>30</RunningJobs>
        <FreeJobSlots>0</FreeJobSlots>
    </CS>
    <NS>
        <URI>/NS/1</URI>
        <Type>NS</Type>
        <SourceDomainURI>/INFN/CNAF</SourceDomainURI>
        <DestDomainURI>/INFN/MI</DestDomainURI>
        <Bandwidth Unit="Mb/s">300</Bandwidth>
        <RoundTripTime Unit="ms">3</RoundTripTime>
    </NS>
    <SS>
        <URI>httpg://hostname.mi.infn.it/SS</URI>
        <Type>SS</Type>
        <DomainURI>/INFN/MI</DomainURI>
        <Durability>0.6</Durability>
        <AvailableSpace Unit="GB">300</AvailableSpace>
        <DataAvailability>0.8</DataAvailability>
    </SS>
</data>
```

Let us consider a user that requires a computing service offering the `Scientific Linux` operating system in its version 3.0.3, but also `CentOS` version 4.0 is acceptable with lower satisfaction. Then, the user requires an `Intel Pentium` processor family. The application is expected to store permanent data for around 220 GB in a storage service. A network service with around 70 Mbit/s of bandwidth and a small RTT (Round Trip Time) is desirable. Finally, the minimization of the waiting time of the computing service is optional. In Example 22, we show a possible XMatch query expressing these requirements.

(22)
```
for $NS in doc("data.xml")/data/NS,
    $CS in doc("data.xml")/data/CS[DomainURI=$NS/SourceDomainURI]
    $SS in doc("data.xml")/data/SS[DomainURI=$NS/DestDomainURI]
```

```
let $e1 := ($CS/OSName, $CS/OSVersion) eq ("Scientific Linux", "3.03") satisfies
           1, ("CentOS", "4.0") satisfies 0.8
let $e2 := $CS/ProcessorFamily eq "Intel Pentium" satisfies 1
let $e3 := $SS/AvailableSpace in 200 to 250 satisfies with linear increment
let $e4 := $NS/Bandwidth in 50 to 100 satisfies with linear increment
let $e5 := $NS/RoundTripTime in 0 to 10 satisfies with linear decrement
let $e6 := $CS/FreeJobSlots in 0 to max($CS/FreeJobSlots) satisfies with linear
           increment
where ($e1, $e2, $e3) are essential and ($e4, $e5) are desirable and $e6 is
optional return top 10 with threshold 0.6
```

Each `let` clause is used to express an elementary criterion of satisfaction, while the `where` clause describes the aggregation pattern in Fig. 3 and corresponds to the following equation of our service evaluation model:

$$E_{ess} = \left(w_{11}e_1^{r_{ess}} + w_{12}e_2^{r_{ess}} + w_{13}e_3^{r_{ess}}\right)^{\frac{1}{r_{ess}}} \tag{23}$$

$$E_{des} = \left(w_{21}e_4^{r_{des}} + w_{22}e_6^{r_{des}}\right)^{\frac{1}{r_{des}}} \tag{24}$$

$$E_{opt} = e_5 \tag{25}$$

$$E_1 = \left((1 - w_1)E_{opt} + w_1E_{des}\right) \tag{26}$$

$$E_2 = \left((1 - w_2)E_1^{r_{aggr}} + w_2E_{des}^{r_{aggr}}\right)^{\frac{1}{r_{aggr}}} \tag{27}$$

$$E_3 = \left((1 - w_3)E_2 + w_3E_{ess}\right) \tag{28}$$

$$E = \left((1 - w_4)E_3^{r_{aggr}} + w_4E_{ess}^{r_{aggr}}\right)^{\frac{1}{r_{aggr}}} \tag{29}$$

4.2. Storage resource

In this section, we consider a scenario concerning a user interested in selecting a storage service. The expected characteristics of this service are: the access protocol must be `gridftp` (`/SS/AccessProtocol/Type` element), the data availability should be as high as possible (`/SS/SA/DataAvailability` element), the storage area must be of type `permanent` and with an available space greater than 60 GB (`/SS/SA/AvailableSpace` and `/SS/SA/Type` elements), the user must be able to access the storage area, that is the virtual organization (VO) he belongs to (`urn:vo:1`) must be authorized (`/SS/SA/AuthorizedVO` element). The following XML document represents two examples of storage services:

(30)
```
 1:  <data>
 2:          <SS>
 3:             <UniqueID>urn:cnaf.infn.it:storage.service</URI>
 4:             <SA>
 5:                <LocalID>storage.area.001</LocalID>
 6:                <AvailableSpace>70</AvailableSpace>
 7:                <AuthorizedVO>urn:vo:1</AuthorizedVO>
 8:                <AuthorizedVO>urn:vo:2</AuthorizedVO>
 9:                <Type>permanent</Type>
10:                <DataAvailability>0.7</DataAvailability>
11:             </SA>
12:             <SA>
```

```
13:                <LocalID>storage.area.002</LocalID>
14:                <AvailableSpace>90</AvailableSpace>
15:                <Type>permanent</Type>
16:                <AuthorizedVO>urn:vo:3</AuthorizedVO>
17:             </SA>
18:             <SA>
19:                <LocalID>storage.area.003</LocalID>
20:                <AvailableSpace>50</AvailableSpace>
21:                <Type>volatile</Type>
22:                <DataAvailability>0.7</DataAvailability>
23:                <AuthorizedVO>urn:vo:4</AuthorizedVO>
24:                <AuthorizedVO>urn:vo:5</AuthorizedVO>
25:             </SA>
26:             <AccessProtocol>
27:                <URI>gridftp://grid001.cnaf.infn.it:2811</URI>
28:                <Type>gridftp</Type>
29:             </AccessProtocol>
30:          </SS>
31:          <SS>
32:             <UniqueID>urn:cern.ch:storage.service</URI>
33:             <SA>
34:                <LocalID>storage.area.001</LocalID>
35:                <AvailableSpace>30</AvailableSpace>
36:                <Type>permanent</Type>
37:                <DataAvailability>0.9</DataAvailability>
38:                <AuthorizedVO>urn:vo:4</AuthorizedVO>
39:                <AuthorizedVO>urn:vo:5</AuthorizedVO>
40:             </SA>
41:             <SA>
42:                <LocalID>storage.area.001</LocalID>
43:                <AvailableSpace>80</AvailableSpace>
44:                <Type>permanent</Type>
45:                <DataAvailability>0.9</DataAvailability>
46:                <AuthorizedVO>urn:vo:1</AuthorizedVO>
47:             </SA>
48:             <AccessProtocol>
49:                <URI>gridftp://grid005.cern.ch:2811</URI>
50:                <Type>gridftp</Type>
51:             </AccessProtocol>
52:          </SS>
53: </data>
```

An XMatch query expressing the user expectations in the use case presented in Example 30 is the following one:

(31)
```
for $SS in doc("data.xml")/data/SS
let $e1 := $SS/SA/AuthorizedVO eq "urn:vo:1" satisfies 1
let $e2 := $SS/SA/AvailableSpace gt 60 satisfies 1
let $e3 := $SS/SA/Type eq "permanent" satisfies 1
let $e4 := $SS/SA/DataAvailability in 0 to 1 satisfies with linear increment
let $e5 := $SS/AccessProtocol/Type eq "gridftp" satisfies 1
where ( $e1, $e2, $e3, $e5 ) are essential and $e4 is desired
return top 10 with threshold 0.6
```

For each storage service in the XML document, the five elementary criteria of satisfaction are applied to the five attributes of interest. According to the Definitions 10, 11 and 12, we notice that intra-evaluation relationships

hold among the elements returned by the path expression `$SS/SA/AuthorizedVO`, while inter-evaluation and ancestor relationships hold among the elements returned by the path expressions `$SS/SA/AuthorizedVO`, `$SS/SA/AvailableSpace`, `$SS/SA/Type` and `$SS/SA/DataAvailability`.

For the first storage service (from line 2 to line 30), the `XMLetClause`'s generate the following sequences of satisfaction: $e_1 = (1, 0, 0)$, $e_2 = (1, 1, 0)$, $e_3 = (1, 1, 0)$, $e_4 = (1, 0)$ and $e_5 = (1)$. The tuples of elementary satisfactions that are candidate for the aggregation are: $(1, 1, 1, 1, 1)$, $(0, 1, 1, \ , 1)$ and $(0, 0, 0, 0, 1)$. Among these three candidate tuples, the first one is selected in order to determine the global satisfaction. For the second storage service (from line 31 to line 52), the `XMLetClause`'s generate the following sequences of satisfaction: $e_1 = (0, 1)$, $e_2 = (0, 1)$, $e_3 = (1, 1)$, $e_4 = (0.9, 0.9)$ and $e_5 = (1)$. The tuples of elementary satisfactions that are candidate for the aggregation are $(0, 0, 1, 0.9, 1)$ and $(1, 1, 1, 0.9, 1)$. The second candidate tuple is selected in order to determine the global satisfaction.

Summarizing, the representative tuple of satisfactions for the first storage service is $(1, 1, 1, 1, 1)$, while the representative tuple of satisfactions for the second storage service is $(1, 1, 1, 0.9, 1)$. Because of the monotonicity of the weighted power mean [16], we can assert that the first storage service will be selected as it provides the highest satisfaction among the considered services. The XMatch query given in Example 31 has also a representation in terms of an aggregation function as defined in the previous use case (see Eqs (23–29) and Fig. 3).

5. Related work

Relevant works for our activity are present in two main areas: the database area and the Grid computing area. In the first area, the literature about preference queries grew up in the last twenty years. The Agrawal and Wimmers proposal [17] is targeted at giving users the ability of quickly and flexibly express their preferences over search queries in the Web. This work was proposed in the context of searching on-line information about products and services where the number of items is a critical factor for the query response time. The concept of score was defined in terms of values belonging to the set $[0, 1] \cup \{\bot, \natural\}$ representing the possible user preference. The symbol '$\bot$' represents the indifference while the symbol '$\natural$' represents a veto. Given the structure of the description of a set of entities, the user can express a set of preference functions by assigning scores to possible combinations of values in the description and a way to combine such preferences is provided. The concept of score given in this work relates to the user preference as in our proposal. Conversely, their global preference is the result of a combination of the user preferences considered by relevance (i.e., first vetoes, then values from 1 to 0 and finally the indifference), while in our proposal the global satisfaction is the result of an aggregation taking into account the relevance of each attribute and specific conjunction degrees. As regards the mapping to data models, the relational one was selected, while we have opted for XML. Kießling proposed a formal language for formulating preference queries based on the Best-Matches-Only (BMO) query model [18,19]. It developed a set of constructors and combinators that can be used to write preference expressions. An algebra modeling such operators was defined and extensions for both SQL (Preference SQL [20]) and XPATH (Preference XPATH [21]) were proposed. In our work, we focused on the XML data model and we have proposed a new language based on XQuery. Chomicki proposed a logical framework for formulating preferences as strict partial orders by using arbitrary logical formulas [22]. In order to embed such formulas into relation algebra, a single winnow operator that can be parameterized by a preference formula was defined. This enables the rewriting of preference formulas as SQL queries. Our language is targeted at semi-structured data and our approach has been to define a new language.

In the Grid computing area, there are several works trying to introduce quality aspects in service selection. In [23, 24], a framework to support QoS management in service-oriented Grids is proposed. It includes support for resource and service discovery based on QoS properties. The approach is to model these properties in the WSDL (Web Service Description Language) document describing a service, then to use an extended UDDI (Universal Description and Discovery Integration) to enable QoS-based discovery and selection. The expression of the desired service is based on the boolean logic, therefore this framework enables the partition of services in two categories: the one that satisfy the user expectations and the one that do not satisfy the user expectations. Our approach enables to use a continuous logic for expressing the satisfaction associated to each service. In this way, the services can be partitioned in an arbitrary number of categories. Another relevant work is a framework for the reputation management in Grid

systems that provides a distributed mechanism for resource selection [25]. This framework relies on the concepts of trust and reputation based on community experiences to classify, select and tune the allocation of entities. Trust is defined as the underlying principle for a security mechanism applicable in a global context and helps in reducing risks in unknown situations. Reputation refers to the history of the trust exhibited by an entity. For each entity that can be requested, an overall grade is computed as the aggregation of the elementary grades assigned by service requesters. The algorithm is an adaptation of a mechanism for evaluating trust in peer-to-peer networks. As regards the elementary grade, there is no specification about the way this can be evaluated, however an algorithm to compute the decay of trust values depending on time and on the global reputation returned by service requesters is provided. This work provides a meaningful way to synthesize a community-based reputation concept, while our work focuses on a user-based satisfaction concept. Reputation and satisfaction differ because the former is a possible metric that can be applied to an attribute, while the latter is the acceptance degree concerning the value returned by a measurement for such a metric. Finally, we mention an ontology-based matchmaker [7] enabling a flexible and extensible approach for performing Grid resource selection. It allows the definition of ontologies describing resources and user requests. The ontology-based matchmaker performs a semantic-based selection with a ranking phase using the attribute specified in the `RankBy` element, while we propose a syntax-based selection with a ranking phase supporting the expression and aggregation of satisfactions over a number of different attributes.

6. Conclusions

Grid systems require a mapping functionality for the association of users requests expressed in terms of requirements and preferences to actual resources. Our work started by proposing a model for the evaluation of the satisfaction perceived by a potential user for a set of services. In this paper, we have presented a mapping of such a model to a language for querying XML-based representations of available resources. This language provides a bi-directional mapping with our model, thus offering a tailored solution for its application. It also offers a compact way for users to express their preferences for Grid resources and enable the maximization of the global preference. Future work will be targeted at defining a method for rewriting XMatch in terms of XQuery in order to exploit the available XQuery processor implementations. In this way, we will be able to evaluate, test and tune the suitability of our language. After that, it will be possible to design and implement a dedicated processor providing an optimized execution of XMatch queries. Another important aspect is the extension of the language to enable the definition of the weights and the power parameters. Finally, a meaningful activity is to include the XMatch language in the metamodeling framework for engineering Grid services proposed in [26].

7. XMatch grammar

In this appendix, we define the complete grammar of the XMatch language specified in the simple Extended Backus-Naur Form (EBNF). The grammar rules are given only for symbols starting with the prefix `XM`, while the other symbols are taken from the XQuery W3C specification [3].

(32)
```
XMExpr          ::= XMForClause XMLetClause+ XMWhereClause XMReturnClause
XMForClause     ::= "for" "$" VarName "in" XMDocCall ( "," "$" VarName "in"
                    XMDocCall )*
XMDocCall       ::= "doc(" URILiteral ")" XMPathExpr? XMPredicate?
XMPredicate ::= "[" OrExpr "]"
XMLetClause     ::= "let" ( XMSimpleEnum | XMCompEnum | XMRange )
XMSimpleEnum    ::= XMPathExpr ValueComp XMElement "satisfies" XMSatLiteral
                    ( "," XMElement "satisfies" XMSatLiteral )*
XMCompEnum      ::= XMPathExprList ValueComp XMElementList "satisfies"
                    XMSatLiteral
                    ( "," XMElementList "satisfies" XMSatLiteral )*
```

```
XMRange          ::= XMPathExpr  "in" XMElement "to" XMElement "satisfies" "with"
                     ( "linear" "increment" | "linear" "decrement" | "around" )
XMPathExprList   ::= "(" XMPathExpr ( "," XMPathExpr )+ ")"
XMElementList    ::= "(" XMElement ( "," XMElement )+ ")"
XMElement        ::= ( Literal | XMPathExpr | XMFunctionCall )
XMFunctionCall   ::= ( "max"|"min"|"avg"|"sum"|"count" ) "(" XMPathExpr ")"
XMPathExpr       ::= ( "/" QName )+
XMWhereClause    ::= "where" XMRelevanceExpr ( "and" XMRelevanceExpr )*
XMRelevanceExpr ::= ( "$" VarName "is" | XMVarNameList "are" )
                     ( "essential" | "desirable" | "optional" )
XMVarNameList    ::= "(" "$" VarName ( "," "$" VarName )+ ")"
XMReturnClause   ::= "return" "top" digits ( "with threshold" XMSatLiteral )?
XMSatLiteral     ::= ( "0"? "." Digits | "1" )
```

References

[1] Z. Németh and V. Sunderam, Characterizing Grids: Attributes, Definitions, and Formalisms, *Journal of Grid Computing* **1**(1) (2003), 9–23.

[2] S. Andreozzi, P. Ciancarini, D. Montesi and R. Moretti, *An Approach to the Quantitative Evaluation of Grid Services*, To appear in Journal of Concurrency and Computation: Practice and Experience, 2005.

[3] S. Boag, D. Chamberlin, M.F. Fernndez, D. Florescu, J. Robie and J. Simeon, *XQuery 1.0: An XML Query Language*, W3C Candidate Recommendation, 3 Nov 2005.

[4] J. Yu and R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Mar 2005.

[5] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Kónya, M. Mambelli, J.M. Schopf, M. Viljoen and A. Wilson, GLUE Schema Specification – Version 1.2, 2005.

[6] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher and A. Savva, *Job Submission Description Language* (*JSDL*) *Specification*, Version 1.0, 2005. draft 25.

[7] H. Tangmunarunkit, S. Decker and C. Kesselman, *Ontology-Based Resource Matching in the Grid – The Grid Meets the Semantic Web*, In Proceedings of the 2nd International Semantic Web Conference (ISWC 2003), 20–23 Oct, Sanibel Island, FL, USA, 2003.

[8] L. Dutka and J. Kitowski, *Application of Component-Expert Technology for Selection of Data-Handlers in CrossGrid*, In Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, pages 25–32, London, UK, 2002. Springer-Verlag.

[9] N.E. Fenton and S.L. Pfleeger, *Software Metrics: a Rigorous and Practical Approach*, (2nd edition), Course Technology, 1998.

[10] J.J. Dujmovic, *A Method for Evaluation and Selection of Complex Hardware and Software Systems*, In Proceedings of the International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems (CMG96), San Diego, CA, USA, Vol. 1, pages 368–378, Dec 1996.

[11] S.W. Su, J.J. Dujmovic, D.S. Batory, S.B. Navathe and R. Elnicki, A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems, *ACM Transaction on Database Systems* **12**(3) (1987), 472–520.

[12] A. Berglund, S. Boag, D. Chamberlin, M.F. Fernández, M. Kay, J. Robie and J. Siméon, XML Path Language (XPath) 2.0. W3C Candidate Recommendation, 3 Nov 2005.

[13] M. Fernández, A. Malhotra, J. Marsh, M. Nagy and N. Walsh, XQuery 1.0 and XPath 2.0 Data Model. W3C Candidate Recommendation, 3 Nov 2005.

[14] S. Andreozzi, A. Ciuffoletti, A. Ghiselli and C. Vistoli, *Monitoring the Connectivity of a Grid*, In Proceedings of the 2nd International Workshop on Middleware for Grid Computing (MGC 2004), Toronto, Canada, Oct 2004.

[15] P. Andreetto, S. Andreozzi, G. Avellino, S. Beco, S. Borgia, V. Ciaschini, F. Giacomini, A. Giannelle, A. Guarise, A. Krenek, D. Kuril, A. Maraschini, M. Marchi, L. Matyska, M. Mezzadri, S. Monforte, M. Mordacchini, M. Mulac, F. Pacini, M. Pappalardo, G. Patania, J. Pospisil, F. Prelz, D. Rebatto, E. Ronchieri, M. Ruda, Z. Salvet, J. Sitera, J. Skrabal, M. Sgaravatto, A. Terracina, M. Vocu and L. Zangrando, *Pratical Approaches to Grid Workload and Resource Management in the EGEE Project*, In Proceedings of the Conference on Computing in High Energy and Nuclear Physics (CHEP 2004), Interlaken, Switzerland, Sep 2004.

[16] P.S. Bullen, *Handbook of Means and Their Inequalities*, (Second edition), Kluwer Academic Publishers, 2003.

[17] R. Agrawal and E.L. Wimmers, *A Framework for Expressing and Combining Preferences*, In International Conference on Management of Data, Dallas, pages 297–306, May 2000.

[18] W. Kießling, *Foundations of Preferences in Database Systems*, In Proceedings of the 28th Very Large Database System (VLDB) Conference, Hong Kong, China, 2002.

[19] B. Hafenrichter and W. Kießling, *Optimization of Relational Preference Queries*, In Proceedings of the 16th Australasian Database (ADB05) Conference, Newcastle, Australia, 2005.

[20] W. Kießling and G. Köstler, *Preference SQL – Design, Implementation, Experiences*, In Pro-ceedings of 28th International Conference on Very Large Databases (VLDB), Hong Kong, China, Aug 2002.

[21] W. Kießling, B. Hafenrichter, S. Fischer and S. Holland, *Preference XPATH: a Query Language for E-Commerce*, In Proceedings of 5th Internationale Tagung Wirtschaftsinformatik, Augsburg, Germany, Sep 2001.
[22] J. Chomicki, Preference Formulas in Relational Queries, *ACM Transaction on Database Systems* **28**(4) (2003), 427–466.
[23] A. ShaikhAli, O.F. Rana, R. Al-Ali and D.W. Walker, *UDDIe: an Extended Registry for Web Services*, In Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT Conference, Florida, USA, Jan 2003.
[24] R.J. Al-Ali, O.F. Rana, D.W. Walker, A. Jha and S. Sohail, G-QoSM: Grid Service Discovery Using QoS Properties, *International Journal of Computing and Informatics* **21**(4) (2002), 363–382.
[25] B. Alunkal, I. Valjkovic, G. Von Laszewski and K. Amin, *Reputation-based Grid Resource Selection*, In Proceedings of the Workshop on Adaptive Grid Middleware (AGridM 2003), New Orleans, LA, USA, Sep 2003.
[26] S. Andreozzi, P. Ciancarini, D. Montesi and R. Moretti, *Towards a Metamodeling Based Method for Representing and Selecting Grid Services*, In Proceedings of the 1st International Conference on Grid Services Engineering and Management (GSEM'04), Erfurt (Germany), pages 78–93, Oct 2004.