A Data Model for Algorithmic Multiple Criteria Decision Analysis

Olivier Cailloux, Tommi Tervonen, Boris Verhaegen, François Picalausa

Noname manuscript No.
(will be inserted by the editor)

A Data Model for Algorithmic Multiple Criteria Decision Analysis

Olivier Cailloux · Tommi Tervonen · Boris Verhaegen · François Picalausa

Received: date / Accepted: date

Abstract Various software tools implementing Multiple Criteria Decision Analysis (MCDA) methods have appeared over the last decades. Although MCDA methods share common features, most of the implementing software have been developed independently from scratch. Majority of the tools have a proprietary storage format and exchanging data among software is cumbersome. Common data exchange standard would be useful for an analyst wanting to apply different methods on the same problem. The Decision Deck project has proposed to build components implementing MCDA methods in a reusable and interchangeable manner. A key element in this scheme is the XMCDA standard, a proposal that aims to standardize an XML encoding of common structures appearing in MCDA models, such as criteria and performance evaluations. Although XMCDA allows to present most data structures for MCDA models, it almost completely lacks data integrity checks. In this paper we present a new comprehensive data model for MCDA problems, implemented as an XML schema. The data model includes types that are sufficient to represent multi-attribute value/utility models, ELECTRE III/TRI models, and their stochastic (SMAA) extensions, and AHP. We also discuss use of the data model in algorithmic MCDA.

Keywords Multiple Criteria Decision Analysis (MCDA), Data model, Multi-Attribute Value Theory, Utility Theory, ELECTRE methods, Stochastic Multicriteria Acceptability Analysis (SMAA), XML

O. Cailloux
Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands
Industrial Engineering Laboratory, École Centrale Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry Cedex, France E-mail: olivier.cailloux@uva.nl

T. Tervonen
Econometric Institute, Erasmus School of Economics, Erasmus University Rotterdam, The Netherlands

B. Verhaegen · F. Picalausa
Department of Computer and Decision Engineering, Université Libre de Bruxelles, Belgium

1 Introduction

The research in Multiple Criteria Decision Analysis (MCDA) has produced a large amount of methods to support decision making processes (Figueira et al 2005; Hwang and Yoon 1981; Keeney and Raiffa 1976). Applicability of the methods has been shown in reported real-life applications (Wallenius et al 2008), but most of them have been one-off decisions in disciplines where models can be constructed and calculated manually or with a general purpose software. The limited application of MCDA in new disciplines can be due to difficulty of integrating existing MCDA software with rest of the decision support technology (Tervonen 2014).

Various practical studies have proposed software frameworks for MCDA or Decision Support Systems (DSSs) implementing MCDA methods (Fedorowicz and Williams 1986; Gauthier and Néel 1996; Georgopoulou et al 1998; Jiménez et al 2006; Jármai 1989; Martin and Fuerst 1984; Minch and Sanders 1986; Natividade-Jesus et al 2007; Spengler et al 1998; van Valkenhoef et al 2013; Zopounidis and Doumpos 2000). However, these have been developed for specific problem types and/or methods, and aimed towards the end user. Furthermore, they lack a data interchange standard that could be used for sharing models over users and software. Theoretical interest has been shown for gathering several MCDA methods in a common software framework; already Teghem et al (1989) analyzed several MCDA methods and studied the properties they satisfy. These were used for deriving a decision tree to choose the most appropriate method. Similarly, Hong and Vogel (1991) proposed a taxonomy of several MCDA methods and considered chaining compatible decision rules. However, neither of these works considered the sharing of input models over different MCDA methods and/or implementations. Considering the diversity of the needs an MCDA end user can face, it is hardly conceivable a single team of researchers could produce a software general enough to satisfy all of them. Therefore it should be made possible that different software tools would allow data exchange in a standard manner.

An appropriate data model for MCDA would allow evaluating results of different MCDA methods with similar inputs. This would be useful in an educational setting and it could also enable easier integration of MCDA methods in industrial applications. The approach of defining a common data interchange standard has proven succesful in the data mining community, who have defined the Predictive Model Markup Language (PMML) (Guazzelli et al 2009) as a means to encode predictive and data mining models in a vendor-independent way, thereby facilitating interoperability of software developed for the domain. Similarly, the OS standard has been proposed for representing mathematical programs together with a framework for their distributed computing (Fourer et al 2009, 2010a,b). Initial steps for uniform data structures in MCDA have already been taken in the XMCDA standard (cf. `www.decision-deck.org/xmcda`).

PMML, OS and XMCDA allow encoding models in their corresponding domains, and therefore they all enable data sharing among various software tools. Unlike PMML and OS, XMCDA defines a single XML schema that can contain an arbitrary number of possibly unrelated elements. For example, a valid XMCDA instance could have a set of alternatives and a set of weights, or a performance table. Every implementation that allows XMCDA input has to use this universal schema. However, most MCDA methods have conceptually different input models. These differences in the inputs cannot be specified using XMCDA.

Moreover, instead of considering only complete models to be used as input to software tools, as in PMML and OS, the XMCDA standard aims to enable *algorithmic MCDA*: the design of computational components performing independent computational steps used in one or multiple MCDA methods. Algorithmic MCDA considers the decision support methods as workflows composed of sequential and parallel executable components. Such workflows can be managed e.g. with the diviz software (see Bigaret and Meyer 2012, or `http://www.diviz.org`). However, the components need to communicate through an interface, for which XMCDA is currently used. As different input types are not differenciated in the XMCDA schema, the middleware is unable to match outputs from one component to input to the next component in the workflow. Flow composition is thus non-trivial as the user has to know which component inputs and outputs can be connected. Such type-safe flow composition cannot in general be enables solely by designing a new data standard: the components themselves need input and output schemas defined with types of the data model. A mechanism need to be provided to enable this; only then can middleware detect incompatibilities in order to help the user in building correct workflow instances.

To overcome the shortcomings of XMCDA, this paper proposes a new core data model for MCDA data and results specification. The main idea of our proposal is that the data model contains only a small set of entities for modeling the main concepts (alternatives, attributes, criteria, relations) recurring in most MCDA methods, and that these concepts are grouped in sets to allow easy data interchange among components in algorithmic MCDA. We develop a model implemented as an XML schema and discuss its use in algorithmic computation of MCDA methods. Our data model does not provide representations for every data type used in all currently existing MCDA methods. Instead, it allows component developers to define the required inputs and outputs using a subset of the standard types, which enables both precise input and output specifications and interoperability of the components. The developed data model aims at enabling (i) easier exchange of data across different software packages, and (ii) decomposition of MCDA methods in smaller, individually executed components that communicate with interfaces defined through the data model. Our emphasis is on multi-attribute value/utility theory, ELECTRE-III/TRI methods, their stochastic extensions SMAA-2/TRI (Lahdelma and Salminen 2001; Lahdelma et al 1998; Tervonen et al 2009), and AHP. With this scope we cover two preference structures applied in majority of MCDA applications (Wallenius et al 2008): utility/value theory and outranking, all three problem statements: choice, ranking, and sorting (Roy 1996), and most methods currently used in practical applications (Wallenius et al 2008).

2 Context

Most MCDA methods have the same base context. A set of decision alternatives A are evaluated on a set of attributes Z. The performance of alternative $a \in A$ on attribute $z \in Z$ is given by a performance function $g_z : A \Rightarrow X_z$, where X_z is the evaluation scale bound to attribute z. The performances can be categorical, e.g. $X_z = \{\text{“Bad”}, \text{“Medium”}, \text{“Good”}\}$, or numerical, with X_z being a subset of $\mathbb{R}$. Performances can be imprecise and modeled using, for example, an interval in $\mathbb{R}$ or a Gaussian distribution. MCDA methods also need a way to represent the

preferences of the Decision Maker (DM). To achieve this, a set of criteria I, each corresponding to one attribute, is defined and preference information is attached to each $i \in I$. Most methods also include preference information that expresses relative importance of the criteria through weights. Usually MCDA methods are presented using only the concept of a criterion, but distinguishing criteria from attributes allows for clearer separation of performances and preference information. The former are mostly objective measurements whereas the latter model subjective views regarding outcome preferability, trade-offs or risk. Furthermore, clear separation of attributes and criteria allows to distinguish several preference model instances for a single attribute. This can be useful in a multiple DM context or if multiple ways to represent the DM preferences are needed.

In Utility Theory (UT) each criterion i, corresponding to an attribute z, is associated with a utility function $u_i : X_z \to [0, 1]$. For an evaluation $x_i \in X_z$, $u_i(x_z)$ is the DM's utility for performance x_z according to criterion i. In the multi-attribute additive utility model each criterion is also associated with a weight w_i. This permits to compute the overall utility $u(a)$ of an alternative a using the values given by the partial utility functions $u_i, i \in I$, and the associated weights $w_i, i \in I$. Once each alternative is associated with an overall utility, they can be rank-ordered. Note that not all UT models associate each criterion with a single weight, e.g. multilinear models include additional weights for sets of criteria. Value functions in value theory are different from utility functions in utility theory in that they do not quantify DM attitudes towards risk, but with respect to data modeling the two are equal. For this reason, in what follows, we use the term "utility" to refer uniformly to both utility and value (-functions, -theory).

In Outranking Theory (OT), including Electre (Roy 1991) and Promethee (Brans et al 1984) family of methods, criteria are associated with threshold parameters. Depending on the method, two or three types of threshold parameters are usually considered: preference, indifference, and possibly veto thresholds. Each criterion $i \in I$ is associated with preference functions of pre-defined shapes, depending on the threshold values. Similarly to UT, these methods also include criteria weights to quantify their relative importances. Electre preference functions, and most Promethee preference functions, may be defined using solely the three threshold parameters included in our data model (Cailloux 2010).

3 Data model

In this section, we present a data model appropriate for a set of common MCDA concepts. For each MCDA concept (such as an alternative or a performance table), we first present a UML diagram. This shows the attribute values that can be associated with the entity, and its relations to other entities. The XML schema specifications corresponding to all diagrams and the complete schema of the data model can be downloaded at `http://github.com/tommite/pubs-code/tree/master/mcdadm-aor`. Diagrams throughout this paper use UML 2.0 notation: the entities (Section 3) as class diagrams, the sample instantiations (Section 4) as object diagrams, and the method flow (Section 4) as a component diagram. All inheritance is implemented as "extension" in the XML schema, and therefore we omit the stereotype from the diagrams.

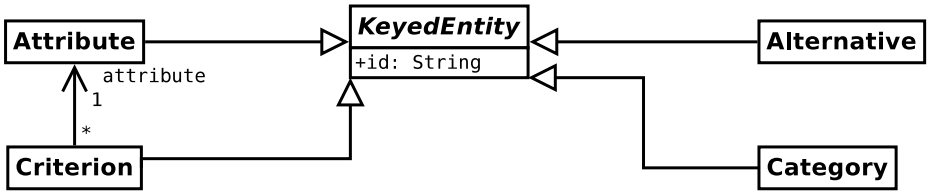


Fig. 1 Data model keyed entities: alternatives, attributes, categories and criteria.

Listing 1 A schema "example" with one attribute and one criterion. The header and namespace declarations have been omitted.

```
<xs:schema>
  <xs:element name="example">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="attribute" type="x3:attributeType" />
        <xs:element name="criterion" type="x3:criterionType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 2 An instance of "example" where criterion c1 references attribute a1. The header and namespace declarations have been omitted, as well as the type of the criterion (criteria hierarchy are defined later in the text).

```
<example>
  <attribute>
    <id>a1</id>
  </attribute>
  <criterion>
    <id>c1</id>
    <attribute ref="a1" />
  </criterion>
</example>
```

The most important entities of our data model are the four keyed ones: alternative, attribute, criterion and category. A keyed entity has an identifier that serves both as its name and as a unique identifier to refer to. We use inheritance with keyed entities to avoid duplication of other types described later on. Keyed entities are displayed in Figure 1.

Listing 1 provides an example XML schema that uses these entities: the example contains one attribute and a single criterion. This schema can be used for unambiguously defining such hypothetical models, and the developer of a MCDA software tool could define the accepted input models trough this schema. One such model is provided in Listing 2: there is a criterion c1, attribute a1, and c1 is explicitly referring to a1.

Our data model defines various measurement types, presented in Figure 2, to encode both performance and relation values. We use inheritance with measurements to enable uniform encoding of imprecise inputs in Stochastic Multicrite-

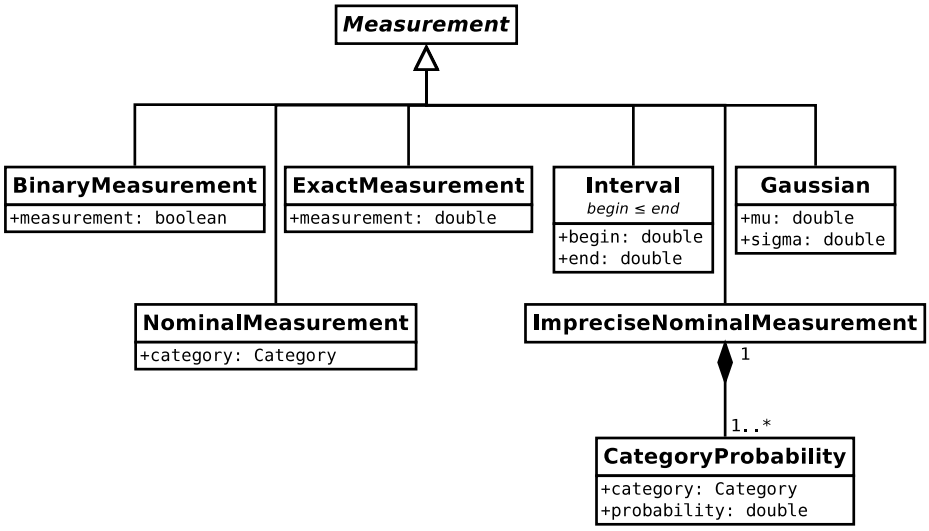


Fig. 2 Data model measurements.

ria Acceptability Analysis (SMAA) models (Tervonen and Figueira 2008). Many MCDA methods allow some model parameters to be imprecise or uncertain, and often these are represented as discrete or continuous probability distributions. Discrete uncertain outcomes can be encoded as imprecise nominal measurements, and continuous ones with intervals or Gaussian measurements. Note that the data model does not include more distributions or other ways to represent imprecise measurement, as even the most common distributions are quite numerous and to avoid bloating the model a non-trivial inclusion choice would have to be made. XML extension mechanisms allow users of our schema to define additional measurement types in external namespaces.

Any two keyed entities can be related with a measurement as a valued pair, e.g. alternative "car1" can have value 120.0 regarding attribute "speed", representing its max speed, or alternative "car1" can have a binary value true for alternative "car2", representing a holistic preference statement car1 $\succ$ car2. As XML inheritance does not allow generic types, the valued pair is repeated for each type of measurement, allowing to specify exactly what type of value the relation has to refer to. We refer to a valued pair with a measurement of type M using the notation <M>ValuedPair. ValuedPair refers to a <M>ValuedPair with measurement type ExactMeasurement. NominalValuedPair refers to a <M>ValuedPair with measurement type NominalMeasurement. Sets of valued pairs form valued relations, that represent a relation between two sets of objects (e.g. performances of a set of alternatives on a set of attributes). Keyed entities can also have a value connected to themselves only, which is represented as a valued entity (e.g. a criterion and a weight). Valued pairs, relations and entities are presented in Figure 3.

Associating two keyed entity types within a valued pair instead of more specific keyed entity types (such as alternative) loses type safety. A practical solution to this issue is to constrain references to strongly typed sets of keyed entities with XPath expressions and XML Schema keys. Listing 3 shows an example of such

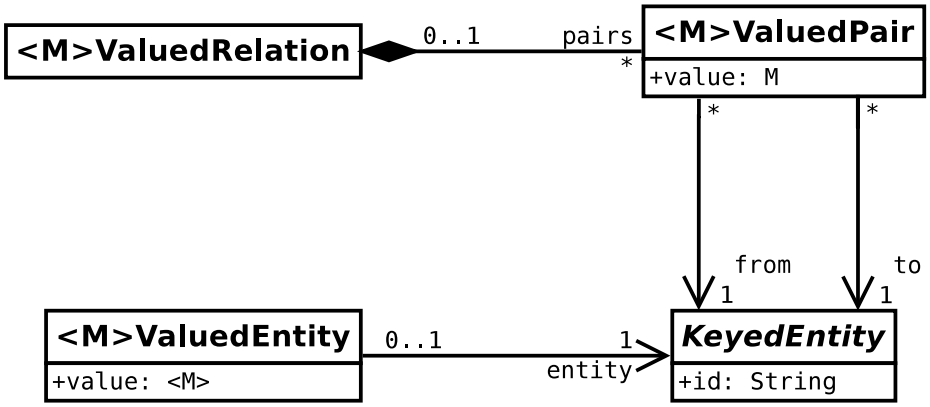


Fig. 3 Data model valued entity, pair and relation. Note that <M> represents a measurement type, and in the XML schema implementation the enclosing type is repeated for each sub-type of measurement (Exact, Interval, ...), as well as one for the abstract measurement type.

Listing 3 An XML schema with path constraints.

```
<element name="exampleOne">
  <complexType>
    <sequence>
      <element name="alternativeSet" type="x3:alternativeSetType" />
      <element name="attributeSet" type="x3:attributeSetType" />
      <element name="performanceTable" type="x3:valuedRelationType" />
    </sequence>
  </complexType>

  <key name="alternativeKey">
    <selector xpath="alternativeSet/alternative" />
    <field xpath="id" />
  </key>
  <key name="attributeKey">
    <selector xpath="attributeSet/attribute" />
    <field xpath="id" />
  </key>
  <keyref name="performanceTableAlternativeKeyRef" refer="x3e:alternativeKey">
    <selector xpath="performanceTable/valuedPair/from" />
    <field xpath="@ref" />
  </keyref>
  <keyref name="performanceTableAttributeKeyRef" refer="x3e:attributeKey">
    <selector xpath="performanceTable/valuedPair/to" />
    <field xpath="@ref" />
  </keyref>
</element>
```

a schema defining a set of alternatives, a set of attributes, a set of valued pairs, and path constraints making sure the valued pairs refer to an alternative and an attribute. With no path constraints the schema would also validate if the valued pairs referred to e.g. two alternatives.

In order to increase readability and reduce verbosity of the encoding, the measurement is defined as optional in a binary relation: a missing measurement value

Listing 4 An XML data file with a set of alternatives and a binary relation. The header and namespace declarations have been omitted. Assuming the binary relation is non reflexive and supposed to be defined on the whole set of alternatives, it may be deduced that the pairs $(a1, a3)$ and $(a2, a1)$ are not part of the relation.

```
<exampleTwo>
  <alternativeSet>
    <alternative>
      <id>a1</id>
    </alternative>
    <alternative>
      <id>a2</id>
    </alternative>
    <alternative>
      <id>a3</id>
    </alternative>
  </alternativeSet>

  <relation>
    <valuedPair>
      <from ref="a1" />
      <to ref="a2" />
    </valuedPair>
    <valuedPair>
      <from ref="a2" />
      <to ref="a3" />
    </valuedPair>
    <valuedPair>
      <from ref="a3" />
      <to ref="a1" />
    </valuedPair>
    <valuedPair>
      <from ref="a3" />
      <to ref="a2" />
    </valuedPair>
  </relation>
</exampleTwo>
```

is considered to be true. Also, when the binary relation is supposed to be defined on some known set, for example, in the case of a binary relation to be defined on a set of alternatives given as input, missing valued pairs are considered to have a false value. With these rules the binary relations can be represented more naturally as only the pairs that are part of the relation have to be included. Listing 4 shows an example encoding of a binary relation. The binary relations may also be used to represent assignments in a sorting problem by using a relation on a set of alternatives and a set of categories. Path constraints may be used to ensure that an alternative is assigned to exactly one category if crisp assignments are required.

The data model criteria contain preference information related to measurements for a certain attribute. For example, if an attribute is top speed, we still need a criterion to represent the preference direction either implicitly with a utility criterion containing an increasing function, or explicitly with a directed criterion. Including meaningful names for the criteria allows to distinguish different parameterizations of the same preference model and to treat the criteria uniformly with other keyed entities. The data model criteria are presented in Figure 4. We do not

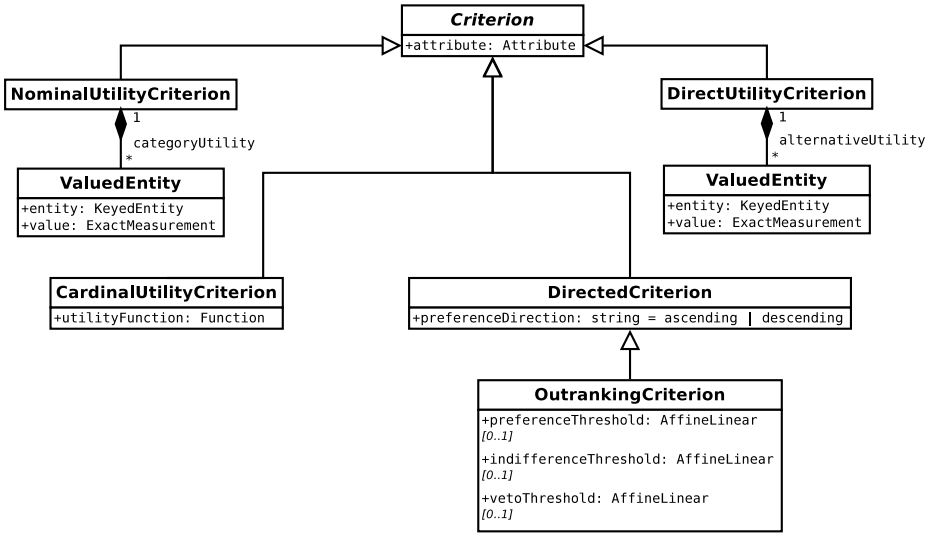


Fig. 4 Data model criteria.

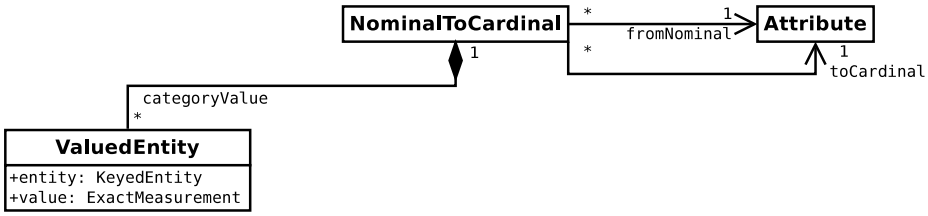


Fig. 5 Data model nominal to cardinal attribute conversion.

distinguish between cardinal and nominal attributes. However, if alternatives evaluated with nominal measurements (represented by categories) are to be used with a criterion expecting cardinal measurements, a mapping of the nominal measurements to cardinal ones is necessary, as well as relating the new measurements with another attribute. The type used for this is presented in Figure 5. The mapping is defined thanks to valued entities associating values to categories.

We support three classical shapes for encoding partial utility functions u_i. Affine linear functions of the form $u_i(x) = ax + b$ need only the slope and offset parameters. A piecewise linear function may be defined using at least a pair of two-dimensional points. Exponential functions $u_i(x) = 1 - e^{-ax}$ require only a single parameter. Figure 6 shows these three types.

The data model contains set versions of most concepts: AlternativeSet, CategorySet, AttributeSet, CriterionSet which may contain any type of criterion, OutrankingCriterionSet which contains only outranking criteria, and similarly for the other criteria types, and ValuedEntitySet. Note that sets of valued pairs are valued relations. A summary of the data model is included in Appendix A.

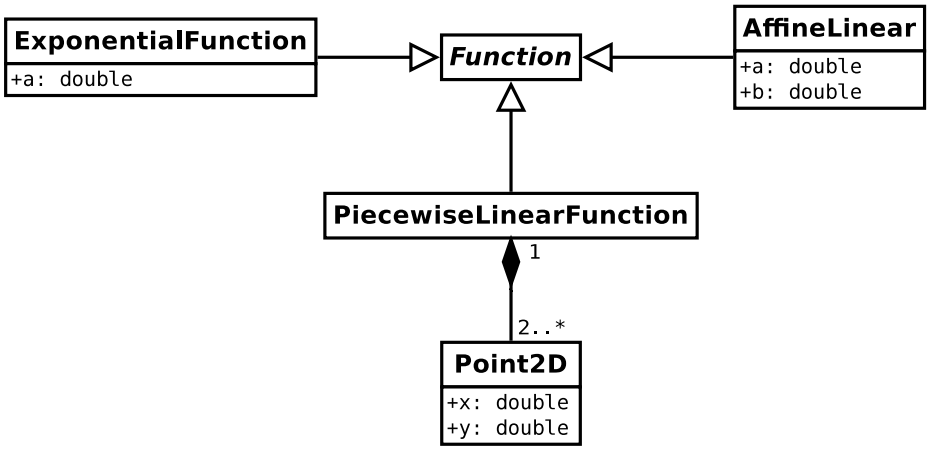


Fig. 6 Data model functions.

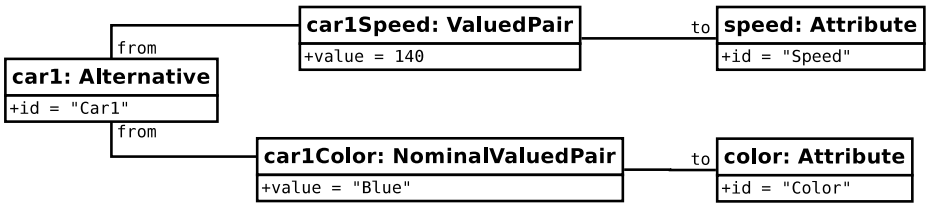


Fig. 7 Evaluation of car1 performances. The attributes of car1Speed and car1Color are actually references to the corresponding entities, that are omitted for brevity.

4 Model application

In this section we present application of the model in two parts. First we illustrate with various use cases the encoding of attribute performance measurements and related preference structures, and afterwards how the data model can be used in algorithmic MCDA to connect multiple components within a workflow representing a complete method.

4.1 Preference and measurement encoding

Let us consider a multi-criteria problem of choosing between two cars, car1 and car2. There are two relevant attributes for this decision, speed and color. Car1 has speed 140 and color "Blue", car2 has speed 120 and color "Red". The performance evaluations of car1 are presented in Figure 7 in terms of the data model. The choice between the two cars depends on the DM preferences that are elicited considering a certain preference model. Let us illustrate instantiation of the data model for utility theory (UT) and outranking theory (OT) models by considering the following use cases (we refer to "user" as the one encoding the model).

UC1: UT with nominal categories. The user wants to encode a utility function from the set of colors to the range $[0, 1]$. Assume that the DM considers blue color to

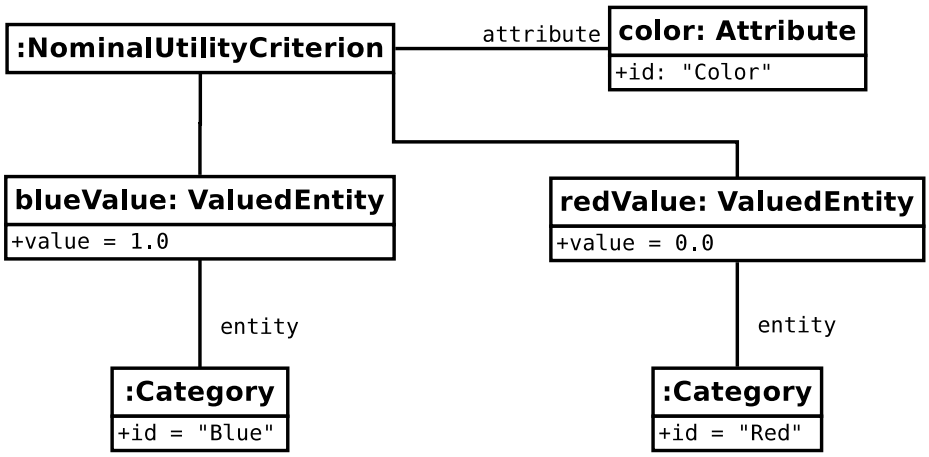


Fig. 8 Sample preference evaluation of UC1 (utility theory with nominal categories).

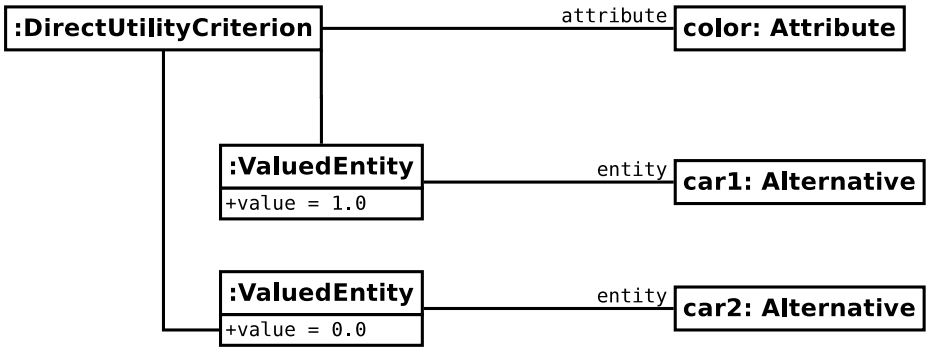


Fig. 9 Sample preference evaluation of UC2 (utility theory with direct encoding).

be associated with the maximal utility (value 1) and red color with the minimal one (value 0). This is presented in Figure 8. The NominalUtilityCriterion is connected to the Attribute named Color to indicate that this criterion contains preference informations relating to that attribute. The NominalUtilityCriterion entity contains two valued entities. The first one links category Blue to the ExactMeasurement value one. The second one links category Red to value zero.

UC2: UT, direct encoding. The user does not care of (or does not know) the exact color of each car, but she knows the utility of each car's color. She wants to encode directly the utility function from the set of alternatives to the range $[0, 1]$ with no encoding of the color attribute. Note that this use case appears naturally when a proxy attribute is difficult to find (e.g. the beauty of a landscape, or the comfort of a car). The use case is presented in Figure 9.

UC3: OT, with recoding. Assume the colors of each car are known, as displayed in Figure 10. The user wants to recode the colors (e.g. blue=3, red=10). To do this, she wants to encode the recoding function (i.e. the association between some set of possible colors and some set of numbers). Figure 11 presents the

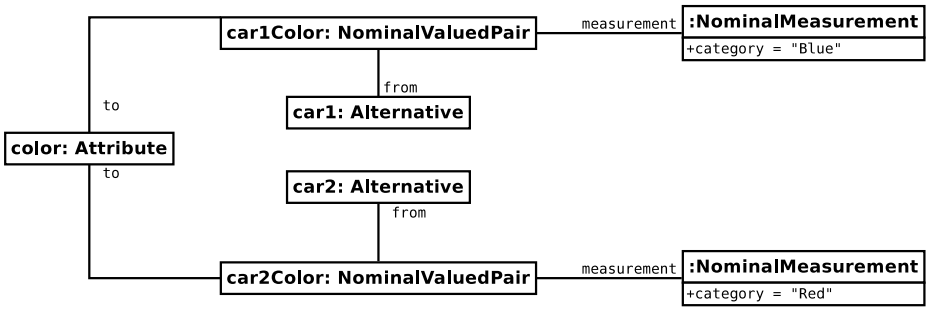


Fig. 10 Sample performance evaluation of UC3 (outranking theory with recoding). The cars are evaluated with a nominal attribute.

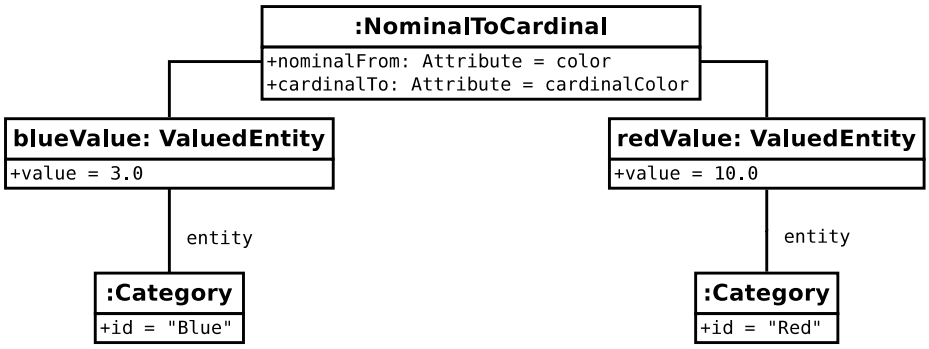


Fig. 11 Sample recoding function for UC3 (outranking theory with recoding).

NominalToCardinal entity required for the recoding: two valued entities are used. The first one indicates that the category Blue maps to value 3 and the second one indicates that the category Red maps to value 10. Observe that this information together with the nominal performances defines a new attribute with cardinal performances. In algorithmic MCDA such conversions can be implemented as components taking as input the nominal performances and the recoding function represented by the NominalToCardinal entity, and outputting a new Attribute (here, named CardinalColor) with corresponding cardinal performances. This connects to the next use case.

UC4: OT, with recoding and thresholds. The user wants to recode the colors (e.g. green=1, blue=3, red=10), and on top of that to define an outranking criterion quantifying the preference structure with thresholds. Figure 12 presents this case. Observe that an OutrankingCriterion is supposed to be linked with an Attribute that is associated with cardinal evaluations, otherwise its threshold values are not interpretable. Thus, for this use case, the user should encode the cars' measurements as cardinal values instead of nominal ones. This can be done for example by using the output of the component defined in the previous use case.

UC5: UT, with cardinal performances. The user wants to code a utility function directly on the range of values of the car speeds using a linear utility function

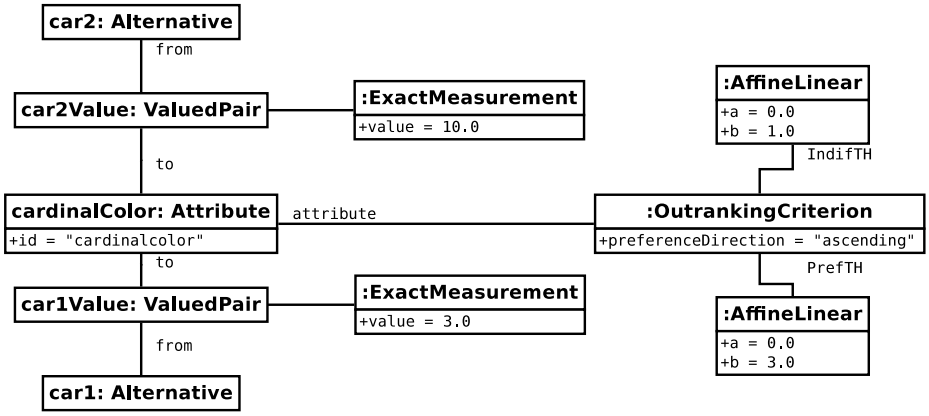


Fig. 12 Sample preference evaluation of UC4 (outranking theory with recoding). The car evaluations have been recoded to a cardinal attribute (see Figures 10 and 11), which permits to define thresholds.

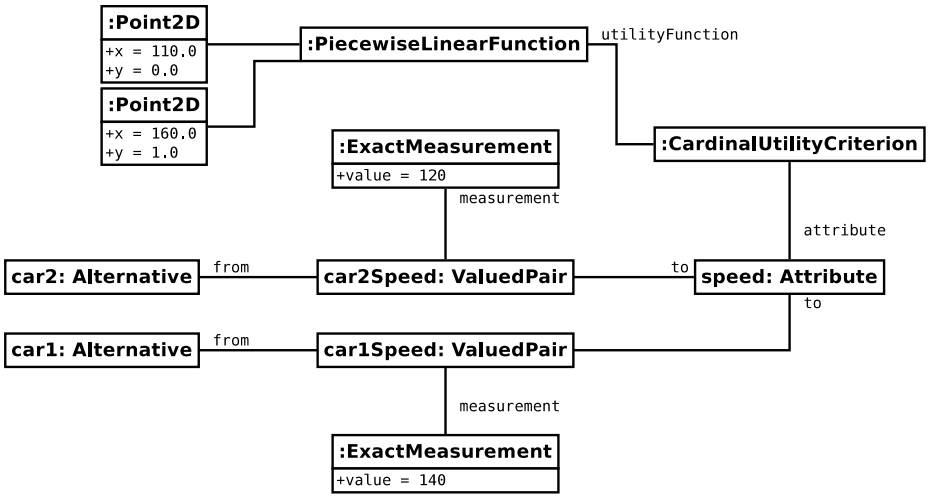


Fig. 13 Sample preference evaluation of UC5 (utility theory with direct ratings).

Fig. 14 Sample preference evaluation of UC6 (holistic preference statement).

with utility 0 at speed $\leq$ 110 and 1 at speed $\geq$ 160. This is illustrated in Figure 13.

UC6: UT or OT, with holistic evaluations. The user wants to provide holistic information car1 $\succ$ car2, that can subsequently be used to infer a set of preference models compatible with the information, for example as in the UTA family of methods (Greco et al 2008). For this case only the binary valued pair needs to be encoded as shown in Figure 14.

Encoding performance/preference information given as matrix of relative importances in methods such as AHP is trivial, as the valued relation can represent such information completely. We omit the presentation of these use cases for brevity.

4.2 Algorithmic MCDA

Algorithmic MCDA refers to composing the methods (workflows) from independent components communicating through strictly defined interfaces. Such components can be implemented as e.g. webservices. Workflow composition becomes considerably easier if there is only a small amount of different types available. For this purpose our XML implementation of the data model contains set versions of concepts presented in Section 3, such as AlternativeSet or AttributeSet. Although it is simple in XML to define such sets by using sequences, defining these types in the XML implementation of the data model permits to easily check for type compatibility when using the output of one component as an input to another component.

Let us now describe an example of composing an Electre III outranking relation computation using multiple components. The complete workflow is presented in Figure 15. Electre III, like most other outranking methods, consist of discrete steps of concordance computation, discordance computation, aggregation of the results into an outranking relation, and an exploitation of the outranking relation (Roy 1991). The top component in Figure 15 computes the concordance valued relation. It requires a set of alternatives, a set of attributes, alternative performances on the given attributes, a set of criteria (one per attribute), criteria weights, and preference and indifference thresholds for each criterion. Therefore, the criteria used as input are outranking criteria. The component output is a valued relation, where the from and to attributes refer to alternatives given as input. This concordance valued relation may then be sent to a component called cut_relation, which also takes as input a majority threshold parameter represented as an exact measurement, and cuts the valued relation into a binary one. Observe that this component is not specific to Electre and may hence be useful in other contexts as well.

The discordance component requires as input a set of alternatives, a set of attributes, performances for every alternative and attribute, and a set of criteria corresponding to the attributes. Each of the criteria may have an associated veto threshold. In Electre III, it is allowed to have criteria with no veto values defined on them, in which case the veto condition will never apply for that particular criterion. The discordance component computes a set of relations represented as valued relations, one for each criterion. To represent this, no ready made types exist in our data model, and a new type needs to be defined. However, it is easy to re-use existing types in the data model to define DiscordanceRelation elements to contain a sequence of pairs of criteria and valued relations. Finally, the binary concordance relation and valued discordance relations are sent to an outranking component that aggregates them and outputs an outranking relation in form of a binary relation.

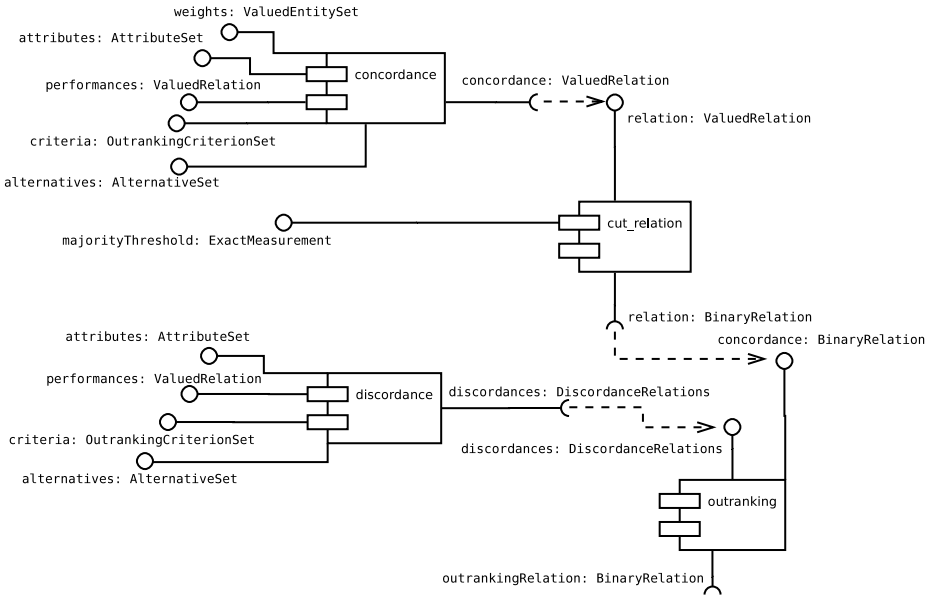


Fig. 15 Electre flow with four components: concordance computation, fuzzy-to-crisp relation cut, discordance computation, and outranking relation computation.

5 Discussion

Although most MCDA methods share common elements such as alternatives, attributes and criteria, many also include method-specific concepts, and representing each of these in a standard data model would be pointless. The data model we propose in this paper contains the entities needed to represent data most commonly encountered in MCDA. This permits to have a reasonable size data model while enabling strong data typing. Because the inputs and outputs of different MCDA methods vary, each MCDA component should define its own input and output XML schemas. For example, the required input for electre concordance component is similar to the input required for electre discordance component, but not identical. The XML schemas may include elements having types defined in our data model, other standard types, or ad hoc types. For example, a component which outputs mathematical programs should reuse the adequate type to represent mathematical programs defined by the OS standard (Fourer et al 2009). When no standard exists, for example due to the concept being specific to a new family of MCDA methods, a developer may define her own type. A separate XML namespace should be used to ensure that this new type has its own identity. Note that it is possible for the developer to share the newly defined type with other component developers working on similar methods.

As the data model enables strongly typed component interfaces, some types are not compatible even though subsets of their possible values have equal mathematical values. We suggest to use data converters in such cases. For example, an interval or an affine linear function can be represented in a particular case as an exact value. Should an ExactValue be used instead of an AffineLinear, as it does

not inherit from AffineLinear, a data converter is required for transforming the ExactValue to an equivalent AffineLinear entity (i.e. one having the x coefficient zero). Similar technique is used in the rapidminer software.

Note that some of the types could be decomposed in looser coupled entities, e.g. outranking criteria could have the thresholds as separate types attached to the criterion entities. Such approach would allow for better decoupling and to define more data integrity constraints in component input schemas. However, the workflow design effort grows with the amount of decoupled types, and for this reason we have chosen to include the thresholds with the outranking criteria as optional attributes, and likewise functions within utility criteria.

The components of algorithmic MCDA could be implemented as web services and subsequently made available through a central registry (e.g. of the Decision Deck consortium). This would permit to create software that uses the components published as web services. The definition of standard types would make connecting components in a type safe manner trivial, for example to compose MCDA workflows in the diviz software, because it is easy to detect potential compatibility between the output of a component and the input of another one simply by looking at the types of the corresponding elements in the component XML schemas. Furthermore, our data model enables data interchange between different software and, for example, easy use of computational libraries of JSMAA (Tervonen 2014) and J-MCDA (Cailloux 2010) in more domain-specific analysis software such as ADDIS (van Valkenhoef et al 2013), that applies MCDA as the last component in a workflow of analyzing benefits and risks of medical treatments.

References

Bigaret S, Meyer P (2012) Diviz: an MCDA workflow design, execution and sharing tool. Intelligent Decision Technologies Journal 6(4):283–296

Brans J, Mareschal B, Vincke P (1984) PROMETHEE: a new family of outranking methods in multicriteria analysis. In: Brans J (ed) Operational Research, IFORS 84, North Holland, Amsterdam, p 477–490

Cailloux O (2010) ELECTRE and PROMETHEE MCDA methods as reusable software components. In: Antunes CH, Insua DR, Dias L (eds) Proceedings of the 25th Mini-EURO Conference on Uncertainty and Robustness in Planning and Decision Making (URPDM 2010)

Fedorowicz J, Williams GB (1986) Representing modeling knowledge in an intelligent decision support system. Decision Support Systems 2(1):3–14, DOI 10.1016/0167-9236(86)90116-8

Figueira J, Greco S, Ehrgott M (eds) (2005) Multiple Criteria Decision Analysis: State of the Art Surveys. Springer Science+Business Media, Inc., New York

Fourer R, Gassmann HI, Ma J, Martin RK (2009) An XML-based schema for stochastic programs. Annals of Operations Research 166(1):313–337, DOI 10.1007/s10479-008-0419-x

Fourer R, Ma J, Martin K (2010a) Optimization services: A framework for distributed optimization. Operations Research 58(6):1624–1636, DOI 10.1287/opre.1100.0880

Fourer R, Ma J, Martin K (2010b) OSiL: an instance language for optimization. Computational Optimization and Applications 45(1):181–203, DOI 10.1007/s10589-008-9169-6

Gauthier L, Néel T (1996) SAGE: an object-oriented framework for the construction of farm decision support systems. Computers and Electronics in Agriculture 16(1):1–20, DOI 10.1016/S0168-1699(96)00018-X

Georgopoulou E, Sarafidis Y, Diakoulaki D (1998) Design and implementation of a group DSS for sustaining renewable energies exploitation. European Journal of Operational Research 109(2):483–500, DOI 10.1016/S0377-2217(98)00072-1

Greco S, Mousseau V, Słowiński R (2008) Ordinal regression revisited: multiple criteria ranking using a set of additive value functions. European Journal of Operational Research 191(2):415–435, DOI 10.1016/j.ejor.2007.08.013

Guazzelli A, Zeller M, Chen W, Williams G (2009) PMML: An open standard for sharing models. The R Journal 1

Hong IB, Vogel DR (1991) Data and model management in a generalized MCDM-DSS. Decision Sciences 22(1):1–25, DOI 10.1111/j.1540-5915.1991.tb01258.x

Hwang C, Yoon K (1981) Multiple Attribute Decision Making: Methods and Applications; A State-of-the-Art Survey. Springer

Jiménez A, Ríos-Insua S, Mateos A (2006) A generic multi-attribute analysis system. Computers & Operations Research 33(4):1081–1101, DOI 10.1016/j.cor.2004.09.003

Jármai K (1989) Single- and multicriteria optimization as a tool of decision support system. Computers in Industry 11(3):249–266, DOI 10.1016/0166-3615(89)90006-7

Keeney R, Raiffa H (1976) Decisions with multiple objectives: Preferences and value tradeoffs. J. Wiley, New York

Lahdelma R, Salminen P (2001) SMAA-2: Stochastic multicriteria acceptability analysis for group decision making. Operations Research 49(3):444–454, DOI 10.1287/opre.49.3.444.11220

Lahdelma R, Hokkanen J, Salminen P (1998) SMAA - stochastic multiobjective acceptability analysis. European Journal of Operational Research 106(1):137–143, DOI 10.1016/S0377-2217(97)00163-X

Martin M, Fuerst W (1984) Effective design and use of computer decision models. Management Information Systems Quarterly 8(1)

Minch RP, Sanders GL (1986) Computerized information systems supporting multicriteria decision making. Decision Sciences 17(3):395–413, DOI 10.1111/j.1540-5915.1986.tb00233.x

Natividade-Jesus E, Coutinho-Rodrigues J, Antunes CH (2007) A multicriteria decision support system for housing evaluation. Decision Support Systems 43(3):779–790, DOI 10.1016/j.dss.2006.03.014

Roy B (1991) The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31(1):49–73

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

Spengler T, Geldermann J, Hähre S, Sieverdingbeck A, Rentz O (1998) Development of a multiple criteria based decision support system for environmental assessment of recycling measures in the iron and steel making industry. Journal of Cleaner Production 6(1):37–52, DOI 10.1016/S0959-6526(97)00048-6

Teghem J, Delhaye C, Kunsch PL (1989) An interactive decision support system (IDSS) for multicriteria decision aid. Mathematical and Computer Modelling 12(10-11):1311–1320, DOI 10.1016/0895-7177(89)90370-1

Tervonen T (2014) JSMAA: open source software for SMAA computations. International Journal of Systems Science 45(1):69–81, DOI 10.1080/00207721.2012.659706

Tervonen T, Figueira JR (2008) A survey on stochastic multicriteria acceptability analysis methods. Journal of Multi-Criteria Decision Analysis 15(1–2):1–14, DOI 10.1002/mcda.407

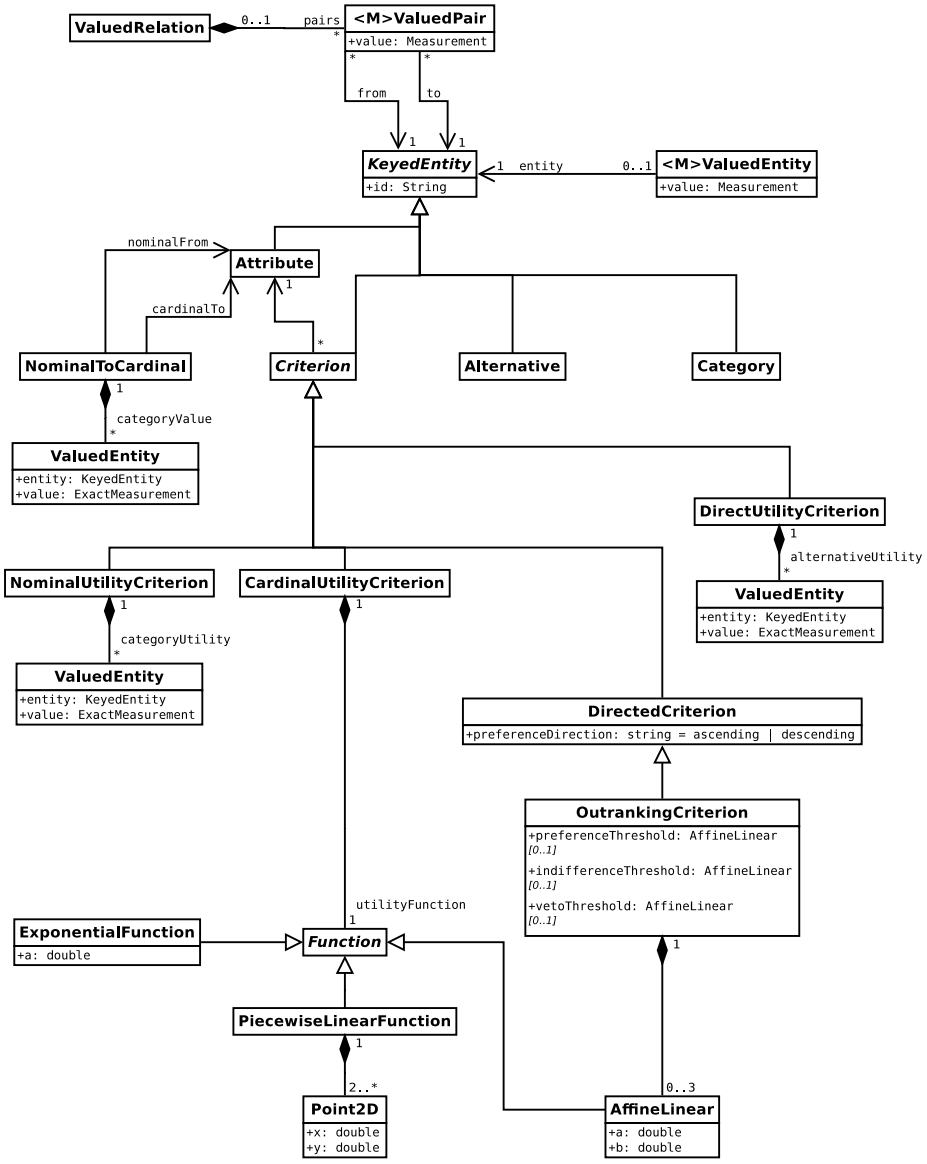
Tervonen T, Figueira JR, Lahdelma R, Almeida Dias J, Salminen P (2009) A stochastic method for robustness analysis in sorting problems. European Journal of Operational Research 192(1):236–242, DOI 10.1016/j.ejor.2007.09.008

van Valkenhoef G, Tervonen T, Zwinkels T, de Brock B, Hillege H (2013) ADDIS: a decision support system for evidence-based medicine. Decision Support Systems 55(2):459–475, DOI 10.1016/j.dss.2012.10.005

Wallenius J, Dyer JS, Fishburn PC, Steuer RE, Zionts S, Deb K (2008) Multiple criteria decision making, multiattribute utility theory: recent accomplishments and what lies ahead. Management Science 54(7):1336–1349

Zopounidis C, Doumpos M (2000) PREFDIS: a multicriteria decision support system for sorting decision problems. Computers & Operations Research 27(7-8):779–797, DOI 10.1016/S0305-0548(99)00118-5

Appendix A: Data model[1]



[1] This representation omits the measurement hierarchy (displayed in Figure 2) and set types.