XMCDA: AN XML ENCODING OF MCDA DATA

Patrick Meyer and Sébastien Bigaret
Institut Télécom, Télécom Bretagne
UMR CNRS 3192 Lab-STICC
Technopôle Brest-Iroise CS 83818, F-29238 Brest Cedex 3
Université européenne de Bretagne
{patrick.meyer,sebastien.bigaret}@telecom-bretagne.eu

Abstract Up to recently, the analysis of a decision problem via numerous Multiple Criteria Decision Aid (MCDA) algorithms required to use various data formats. This lack of a unified standard for their input and output data has probably contributed to the poor adoption of MCDA software by users. This chapter presents `XMCDA`, an initiative of the DECISION DECK project, which is a unified data model enabling to encode classical concepts from MCDA in XML. Among other things it eases the analysis of a problem instance by various MCDA methods compatible with `XMCDA` without requiring data conversions and simplifies the chaining of MCDA algorithms for the resolution of complex decision problems.

Keywords: MCDA data, standardization, XML

Contents

1. Introduction

Research activities in and around the field of Multiple Criteria Decision Aid (MCDA) have developed quite rapidly over the past years, and have resulted in various streams of thought and methodological formulations for the resolution of complex decision problems. In particular, many so-called MCDA *methods* have been proposed in the literature and are very often available as software programs.

Unfortunately, at least four major difficulties arise when it comes to using these programs in practice:

1 different techniques are generally implemented in separate software products, with *heterogeneous* user interfaces;

2 testing multiple MCDA algorithms on one problem instance is not easy, because of the various input *data formats* required by the software applications;

3 a lot of MCDA algorithms which are presented and published in scientific articles are not easily *available* and consequently often only used by their authors;

4 several MCDA software products are not *free* (neither from a financial, nor from an open-source point of view), which can be considered as a weakness for their large dissemination.

In order to overcome these difficulties, a group of researchers has got together to create the DECISION DECK project (see Decision Deck Consortium, 2009). Its objective is to collaboratively develop open-source software tools implementing MCDA techniques.

Up to recently, the problem of the heterogeneous input and output data formats in MCDA software, prevented users from combining existing MCDA tools in order to create treatment chains involving multiple software pieces. Consequently, in such a situation, the resolution of a complex decision problem comes generally down to testing only one algorithm and using various supplementary tools to analyze the results of the resolution. This can be frustrating for a lot of MCDA analysts and practitioners who might like to test various methods on a given problem, without having to recode the instance in various data formats.

To allow running a problem instance through multiple techniques or methods and to allow the chaining of various MCDA algorithms, researchers of the DECISION DECK project have suggested to define a data standard, called `XMCDA`, which could be adopted by various programs to make them interoperable.

In this chapter, we present XMCDA, explain its construction and motivate the choices which have been made during the elaboration process. Furthermore, we detail its use via two initiatives of the DECISION DECK project, namely the XMCDA web-services and the diviz software. The chapter is structured as follows: first, in Section 2 we present the history of XMCDA and explain the general ideas behind the structure of XMCDA. Then, in Section 3 we present how a lot of common MCDA related concepts can be encoded in XMCDA. Finally, in Section 4 we present the use of XMCDA in practice.

2. A first cup of XMCDA

The objective of this section is to ease the adoption of the standard by presenting a quick overview of its purpose and the philosophy which guided its construction. We therefore start by discussing the general principles of XMCDA and the course which has lead to the current version. Then, we present some general conventions that should guide the reader of the sequel. Finally, we present some atomic elements of XMCDA which underly many more general structures presented in Section 3.

2.1 Technical aspects and choices for XMCDA

The XMCDA markup language is written in XML[1], a general-purpose syntax for defining markup languages. XML's purpose is to aid information systems in sharing structured data, especially via the Internet and to encode documents. XMCDA is defined via an XML Schema[2], a set of syntax rules (together with a set of constraints) which define its structure. An XML document that complies with the XMCDA Schema is said to be a *valid* XMCDA document. The XML Schema of the latest version of XMCDA is available via the XMCDA website at http://www.decision-deck.org/xmcda. At the time of writing, the official version approved by the DECISION DECK CONSORTIUM is 2.1.0.

A powerful feature of XML-based markup languages is the possibility to easily transform documents from one format into another. XSLT[3] is a language for such transformations and allows us to convert XMCDA documents into HTML pages for a convenient visualization of their content in any web browser. The website of XMCDA provides a basic XSLT file which can be adapted for various purposes. Note that this possibility to easily manipulate XMCDA documents gives the DECISION DECK project

[1] http://www.w3.org/XML/
[2] http://www.w3.org/XML/Schema
[3] http://www.w3.org/Style/XSL

the possibility to propose converters for future versions of XMCDA which will allow to transform older XMCDA documents to the newer standards (and vice versa, under certain constraints).

In order to understand the choices which have led to the current version of XMCDA and their consequences on its application domain, it is important to differentiate between two fundamental aspects of a multiple criteria decision aid procedure. First, we consider the decision aid process which consists in multiple stepping stones and the intervention of various stakeholders. This operation aims at easing a decision maker's decision and might require the use of one or more clearly identified calculation steps, often called MCDA methods. This leads to the second important aspect of a decision aid procedure, which are the algorithmic elements underlying such MCDA methods. Such series of operations may consist of various elementary calculation steps requiring and providing specific input and output data elements.

XMCDA is at present clearly aimed at this second type of procedures, and focuses on data structures and concepts originally from the field of multiple criteria decision aid methods. As such, it does not provide means of representing the key moments or the various stakeholders of the decision aid process. These aspects are however under study in the DECISION DECK project.

The origin of XMCDA goes back to fall 2007, where a group of researchers of the DECISION DECK project gathered in Paris to think about and work on a data standard which could be used by various MCDA methods. This meeting gave birth to the DECISION DECK Specification Committee whose task is, among other things, to maintain XMCDA and to propose future evolutions of the standard. This committee approved in Spring 2008 a first version of XMCDA, named 1.0, which was used mainly by two MCDA libraries, KAPPALAB (by Grabisch et al., 2008) and DIGRAPHS (by Bisdorff, 2007). Very quickly, the poor genericity of this version limited its practical use and its spreading. Therefore, one year later, in Spring 2009, the DECISION DECK CONSORTIUM approved version 2.0.0 of XMCDA, which is a lot more generic and flexible than its predecessor and currently used by multiple software pieces, like, e.g., diviz (Bigaret and Meyer, 2010a) and the XMCDA web-services and MCDA calculation libraries like ws-RXMCDA (Bigaret and Meyer, 2010b), RUBIS (Bisdorff, 2007) or J-MCDA (Cailloux, 2010).

The releases of XMCDA are versioned a.b.c, where a, b and c are integers which are increased in case of a new release, according to the following rules:

- change from XMCDA a.b.c to XMCDA a.b.(c+1) for minor modifications on the standard, like, e.g., the addition of a new subtag in an XMCDA tag;

- change from XMCDA a.b.c to XMCDA a.(b+1).0 for more substantial modifications on the standard, like, e.g., the addition of a new tag under the root tag;

- change from XMCDA a.b.c to XMCDA (a+1).0.0 for modifications on the standard which do not allow full compatibility to earlier versions, like, e.g., the renaming of a fundamental XMCDA tag.

2.2 Conventions

Now that we have presented a short history of XMCDA and the path that lead to the current version, we give further technical details on the standard. In order to avoid misunderstandings, let us first briefly introduce a few conventions used in this chapter:

- The term 'MCDA *concept*' describes a real or abstract construction related to the field of MCDA which needs to be stored in XMCDA (like, for example, an alternative, the importance of the criteria, an attribute, a criterion, etc.);

- An 'XMCDA *type*' corresponds to an XML data type defined by the XMCDA XML Schema (and which will be written xmcda:value for the value type);

- An 'XMCDA *tag*' is an XML tag which is defined by the XMCDA Schema. An XMCDA type may be instantiated as multiple XMCDA tags.

The name of an XMCDA tag starts by a lower-case letter. The rest of the name is in mixed case with the first letter of each internal word capitalized. This allows to easily read and understand the meaning of a tag. We use whole words and avoid acronyms and abbreviations. Consider for example the tagnames methodParameter (to store some input parameter of an algorithm) , performanceTable (to store the performance table) and criterionValue (to store a value related to a criterion, as, e.g., its weight). Note that objects of the same XMCDA type can in general be gathered in a compound tag, represented by a single XML tag named after the plural form of its elements (e.g., alternatives *contains* several alternative tags).

The following three XML attributes can be found in any of the main XMCDA tags: id, name and mcdaConcept. They are in general optional,

except for the `id` attribute in the description of an alternative, an attribute, a criterion or a category. Each of these three XML attributes has a particular purpose in `XMCDA`:

- The `id` XML attribute identifies an object with a *machine readable* code or identifier. As an illustration consider the following alternative "a12" which is a Peugeot 309.

```
<alternative id="a12">
	<description>
		<comment>A red Peugot 309</comment>
	</description>
</alternative>
```

- The `name` attribute allows to give a *human-readable* name to a particular MCDA object. As an illustration consider the following code, which shows a parameter of an MCDA method specifying the number of iterations, and which is named α in this specific algorithm.

```
<parameter id="numIt" name="alpha">
	<value>
		<integer>3</integer>
	</value>
</parameter>
```

- The `mcdaConcept` XML attribute allows to specify the MCDA concept linked to a particular instance of an `XMCDA` tag. Some `XMCDA` tagnames are quite general and may not be directly related to a very specific MCDA concept. This XML attribute therefore allows to indicate more precisely what information is contained in the related tag. To illustrate this, consider the following example, which presents how a ranking of alternatives could be stored by using the `XMCDA` tag `alternativesValues`. Alternative "a03" is ranked before "a11" and therefore has a lower rank.

```
<alternativesValues mcdaConcept="ranking of alternatives">
	<alternativeValue>
		<alternativeID>a03</alternativeID>
		<value mcdaConcept="rank">
			<integer>1</integer>
		</value>
	</alternativeValue>
	<alternativeValue>
		<alternativeID>a11</alternativeID>
		<value mcdaConcept="rank">
			<integer>2</integer>
		</value>
	</alternativeValue>
	[...]
</alternativesValues>
```

It is obvious that in practice, each user of XMCDA decides if he is willing to refine the XMCDA tag by a specific MCDA concept and how he wishes to do this. This makes XMCDA very flexible as it can be adapted to various methodological vocabularies. This flexibility can also be seen as a drawback, as no specific vocabulary is imposed for this XML attribute.

2.3 Three essential XMCDA types

The XML Schema which determines the structure of an XMCDA document defines among others three essential XMCDA types which appear in most of the XMCDA tags.

The first one is `xmcda:description`. It is intended to store information on the data which is stored in an XMCDA tag. This type allows, among other things, to specify an author and a date of creation, to make a comment or to specify a bibliographical reference of a piece of information. In XMCDA the `xmcda:description` type is instantiated as the `description` tag which appears in all the XMCDA tags or the `projectReference` tag which allows to describe the project related to the current XMCDA file.

Hereafter we give a short excerpt of an XMCDA file showing such an instantiation of the `xmcda:description` type for a car selection problem.

```
<alternatives>
       <description>
              <title>List of potential cars.</title>
              <author>Calvin Hobbes</author>
              <comment>Only European cars are considered.</comment>
              [...]
       </description>
       [..]
<alternatives>
```

The second essential XMCDA type is `xmcda:value`. Its main purpose is to store numerical or literal values related to MCDA data. This type allows to store an integer, a real number (float), an interval, a rational, a nominal value, an ordinal value, etc. In XMCDA, this type is mainly instantiated as the `value` tag, which appears in a large number of XMCDA tags.

Hereafter we give an excerpt of an XMCDA file showing 5 different values.

```
<value><integer>8</integer></value>

<value>
       <rankedLabel>
              <label>Good</label>
               <rank>3</rank>
       </rankedLabel>
</value>

<value>
```

```
        <rational>
                <numerator>10</numerator>
                <denominator>3</denominator>
        </rational>
</value>

<interval>
        <lowerBound><integer>4</integer></lowerBound>
        <upperBound><integer>8</integer></upperBound>
</interval>

<value><real>3.141526</real></value>
```

A third important `XMCDA` type is `xmcda:numericValue` which restricts `xmcda:value` to numeric values. This type is instantiated as many `XMCDA` tags (`minimum`, `maximum`, `constant`, `coefficient`, ...) which require to store exclusively a numeric value.

2.4 Elementary `XMCDA` tags

In this section we present some elementary tags used in many `XMCDA` tags.

`value` and `values`. As already mentioned in Section 2.3, the `value` tag is an instance of the `xmcda:value` type and appears in many `XMCDA` tags. The `values` tag is a compound tag which contains a list of `value` tags. It can be used to represent a set or a list of values.

`point` and `points`. Some more complex `XMCDA` tags, as, e.g., `function` (see hereafter), require the concept of *point*. The abscissa as well as the ordinate are of `XMCDA` type `xmcda:value`. The following example shows a point whose coordinates are $(2.71, 23)$.

```
<point>
        <abscissa><real>2.71</real></abscissa>
        <ordinate><integer>23</integer></ordinate>
</point>
```

The `points` tag is a compound tag containing a list of `point` tags.

`function`. Functions are used in complex tags related to criteria. A `function` can either be a constant, a linear, a piecewise linear function or simply a set of points. The following code shows a constant function, a linear function, and a function described by a set of points.

```
<function>
        <constant><real>456.3847</real></constant>
</function>

<function>
        <linear>
```

```
                <slope><real>4.00</real></slope>
                <intercept><real>4.00</real></intercept>
        </linear>
</function>

<function>
        <points>
                <point>[...]</point>
                [...]
        </points>
</function>
```

`scale`. XMCDA allows to store the definition of evaluation scales, which may be *quantitative*, *qualitative* or *nominal*. This `scale` tag appears in the description of criteria. The following example shows the description of a quantitative scale whose minimal value is 0 and whose maximal value is 1 (the `minimum` and `maximum` tags being of type `xmcda:numericalValue`).

```
<scale>
        <quantitative>
                <minimum><real>0.00</real></minimum>
                <maximum><real>1.00</real></maximum>
        </quantitative>
</scale>
```

`criterionID`, `alternativeID`, `categoryID`, `attributeID`, ... In most of the XMCDA tags, a reference needs to be made to a certain criterion, alternative, category or attribute. This allows to specify to which of these MCDA concepts the data stored in the tag is related to. To do so, we use tags named `criterionID`, `alternativeID`, `categoryID` or `attributeID` which contain a string specifying the `id` of a criterion, alternative, category or attribute which has been defined beforehand (see Section 3.2 on how such MCDA concepts are defined). The following example shows the XMCDA encoding of the weights of the criteria "g01" and "g02" and the way a reference is made to these criteria.

```
<criteriaValues mcdaConcept="criteria weights">
        <criterionValue>
                <criterionID>g01</criterionID>
                <value>
                        <real>0.4</real>
                </value>
        </criterionValue>
        <criterionValue>
                <criterionID>g02</criterionID>
                <value>
                        <real>0.6</real>
                </value>
        </criterionValue>
</criteriaValues>
```

In the following section we present how the main concepts from MCDA can be encoded in XMCDA.

3. XMCDA encoding of MCDA data

The root tag of XMCDA is named XMCDA and contains several sub-tags, each of them describing data related to a decision aid problem. To summarize, these tags can be put in five general categories:

- description of the current decision aid project or description of the XMCDA file;
- output messages from methods or algorithms (log or error messages) and input information for methods or algorithms (parameters);
- description the MCDA concepts attributes, criteria, alternatives and categories;
- the performance table;
- preferences related to criteria, alternatives, attributes or categories (either provided as input by a decision maker or produced as the output of an algorithm).

Note that an XMCDA file does not require that all of these categories are present to be considered as valid. A valid XMCDA file may contain only one tag under the root element.

In the following sections we describe each of these categories and the tags they contain.

3.1 Method and project specific data

Description of the current file or project. In order to describe the current project or XMCDA file, we recommend to use the projectReference tag. Its XMCDA type is xmcda:description which was presented earlier. The following code gives a short example of such a description.

```
<projectReference id="transmogrifier">
       <version>1.0</version>
       <creationDate>2010-06-02T06:14:00</creationDate>
       <author>Calvin Hobbes</author>
</projectReference>
```

Method-specific parameters. Some methods or algorithms require some specific parameters in order to guide the resolution of a decision

problem. Those parameters can be specified by the `methodParameters` tag. Notice that a parameter can be either a value or a function. The following example presents a parameter specifying the number of iterations of an algorithm.

```
<methodParameters>
	<parameter name="iterations">
		<value>
			<integer>3</integer>
		</value>
	</parameter>
</methodParameters>
```

Method-specific messages. Some algorithms might generate error or log messages. These can be stored in the `methodMessages` tag. The following example shows a log message informing the user that the execution of the algorithm was successful.

```
<methodMessages>
	<logMessage>
		<number>0</number>
		<name>OK</name>
		<message>Execution successful.</message>
	</logMessage>
</methodMessages>
```

Note that the other children of `methodMessages` are `errorMessage` and `message`, the latter one allowing to store general messages related to the algorithm.

3.2 Definition of alternatives, criteria, attributes and categories

Alternatives. Alternatives are defined and described under the `alternatives` tag via the `alternative` tag. They can be either `active` or not and either be `real` or `fictive`. In addition, they can also be flagged as `reference` alternatives (for profiles in a sorting problem, e.g.). The `id` XML attribute of an alternative is mandatory. The following piece of code defines three alternatives related to a transportation means selection problem.

```
<alternatives>
	<description>
		<title>List of transportation means.</title>
		<author>Susie Derkins</author>
		[...]
	</description>

	<alternative id="x1" name="Train"/>

	<alternative id="x2" name="Corvette">
```

```
            <type>real</type>
            <active>true</active>
            <reference>false</reference>
        </alternative>

        <alternative id="x3" name="UFO">
            <description>
                <comment>Definitely not
                    a real alternative.</comment>
                [...]
            </description>
            <type>fictive</type>
        </alternative>
</alternatives>
```

Note that sets of alternatives can be defined via the `alternativesSets` tag (see Section 3.3 for further details).

Criteria and attributes. Criteria are defined and described under the `criteria` tag. For each criterion, the XML attribute `id` has to be given. In the following example, the first criterion "g1" represents the power of a car. It is evaluated on a quantitative scale in the interval $[50, 200]$.

```
<criteria>
        <criterion id="g1" name="horsepower">
            <description>
                <comment>Power in horsepower</comment>
            </description>
            <scale>
                <quantitative>
                    <preferenceDirection>max</preferenceDirection>
                    <minimum><real>50</real></minimum>
                    <maximum><real>200</real></maximum>
                </quantitative>
            </scale>
        </criterion>

        <criterion id="g2"/>
</criteria>
```

Attributes are defined the same way as criteria under the `attributes` tag and can be linked to other criteria (or attributes) via the `criteriaReference` (or the `attributeReference` tag). It is also possible to define sets of criteria (resp. attributes) under the `criteriaSets` (resp. `attributesSets`) tag (see Section 3.3 for further details).

Categories. Sorting procedures require the use of categories which can be defined under the `categories` tag. They can be active or not and their rank can be specified. The following example defines three ordered categories of students, the second one being currently inactive.

```
<categories>
```

```
        <category id="g" name="good students">
                <active>true</active>
                <rank>1</rank>
        <category>

        <category id="m" name="medium students">
                <active>false</active>
                <rank>2</rank>
        <category>

        <category id="b" name="bad students">
                <active>true</active>
                <rank>3</rank>
        <category>
</categories>
```

Note that sets of categories can be defined by the `categoriesSets` tag (see Section 3.3 for further details).

The performance table. The performance table is defined by the tag `performanceTable`. It contains, for each alternative (given by its `id`), a list of performances, given by a criterion `id` (or attribute `id`) and a corresponding performance value. The following example shows part of such a performance table for two alternatives and two criteria.

```
<performanceTable>
        <alternativesPerformance>
                <alternativeID>alt1</alternativeID>
                <performance>
                        <criterionID>g1</criterionID>
                        <value><real>72.10</real></value>
                </performance>
                <performance>
                        <criterionID>g2</criterionID>
                        <value><real>82.62</real></value>
                </performance>
        </alternativesPerformance>

        <alternativesPerformance>
                <alternativeID>alt2</alternativeID>
                [..]
        </alternativesPerformance>
</performanceTable>
```

3.3 Advanced information on alternatives, criteria, attributes and categories

Let us now present some more advanced `XMCDA` tags which allow to represent many concepts from the field of MCDA and various types of preferences. Notice beforehand that the names of these tags are quite general in order to allow `XMCDA` to be very flexible. As a consequence, it

might be necessary to specify the content of a tag with an appropriately chosen XML attribute `mcdaConcept`.

To simplify the presentation of the `XMCDA` format here, we will first focus on two generic structures which are adapted for alternatives, criteria, attributes and categories in `XMCDA`. To avoid some redundant explanations and notation, we write `x`Set for the generic structure related to the `XMCDA` tags `alternativesSet`, `criteriaSet`, `attributesSet` and `categoriesSet`. The same convention is used for the `x`Value, `x`LinearConstraint, `x`Comparisons and `x`Matrix tags, described hereafter.

`x`Set. An `x`Set is a set of elements of type `x`. With each of the elements, as well as the whole set, one can associate a value (which allows to simply represent ordered sets). The following code represents a set of alternatives, where one alternative is valued (by the credibility of its membership to the set), and where the whole set is valued by two *quantities* representing the similarity and the dissimilarity of the alternatives.

```
<alternativesSet id="good1" mcdaConcept="best choice">
	<element>
		<alternativeID>a03</alternativeID>
	</element>

	<element>
		<alternativeID>a04</alternativeID>
		<value mcdaConcept="membership degree">
			<real>0.88</real>
		</value>
	</element>

	<values>
		<value name="similarity">
			<real>0.2</real>
		</value>
		<value name="dissimilarity">
			<real>0.7</real>
		</value>
	</values>
</alternativesSet>
```

`x`Value. An `x`Value is a value associated with an element of type `x`. The following example shows a value associated with an alternative "alt1", and one associated with a set of criteria "cs3".

```
<alternativeValue mcdaConcept="overall value">
	<alternativeID>alt1</alternativeID>
	<value>[..]</value>
</alternativeValue>

<criterionValue mcdaConcept="importance">
	<criteriaSetID>cs3</criteriaSetID>
```

```
        <value>[..]</value>
</criterionValue>
```

For the second value we assume that the set of criteria identified by "cs3" has been defined a priori. Note that it would have been possible to define that set explicitly in the tag `criterionValue` via the `criteriaSet` tag.

xLinearConstraints. XMCDA allows to represent linear constraints related to alternatives, attributes, criteria and categories. The following example gives us the representation of the constraint

$$2 \cdot \text{weight}(c_2) - 3 \cdot \text{weight}(c_4) \leq 0.5$$

```
<criteriaLinearConstraints mcdaConcept="weights constraints">
        <constraint name="a strange constraint">
                <constraintNumber>4</constraintNumber>
                <element>
                        <criterionID>c2</criterionID>
                        <coefficient><real>2.00</real></coefficient>
                </element>
                <element>
                        <criterionID>c4</criterionID>
                        <coefficient><real>-3.00</real></coefficient>
                </element>
                <operator>leq</operator>
                <rhs>0.5</rhs>
        </constraint>
<criteriaLinearConstraints>
```

The `operator` tag can either be `eq` (=), `leq` (≤) or `geq` (≥)[4].

xComparisons. xComparisons allow to represent valued binary relations on criteria, alternatives, categories and attributes. The `valuation` tag can be used to store the scale of the valuation. The tag `relationType` allows to express what kind of relation is stored (keywords like *preference*, *indifference*, *incomparability*, *outranking*, *geq*, *leq*, *eq*, *neq*, *gtr*, *less* could be used, or any personalized strings). The following example presents an excerpt of an outranking relation, where alternative "a01" outranks alternative "a02" and not "a03".

```
<alternativesComparisons mcdaConcept="outranking relation">
        <pairs>
                [...]
                <pair>
                        <initial><alternativeID>a01</alternativeID></initial>
                        <terminal><alternativeID>a02</alternativeID></terminal>
                        <value><real>1.00</real></value>
                </pair>
```

[4]We use here the syntax from the LaTeX markup language.

```
            <pair>
                  <initial><alternativeID>a01</alternativeID></initial>
                  <terminal><alternativeID>a03</alternativeID></terminal>
                  <value><real>0.00</real></value>
            </pair>
            [...]
      </pairs>
</alternativesComparisons>
```

xMatrix. An xMatrix allows to represent matrices of values on criteria, alternatives, attributes and categories. The following example presents a short example of a correlation matrix between criteria, where criterion "g01" is positively correlated with "g02" and negatively correlated with "g03".

```
<criteriaMatrix mcdaConcept="correlation matrix">
      <row>
            <criterionID>g01</criterionID>
            [...]
            <column>
                  <criterionID>g02</criterionID>
                  <value>
                        <real>0.9</real>
                  </value>
            </column>
            <column>
                  <criterionID>g03</criterionID>
                  <value>
                        <real>-0.8</real>
                  </value>
            </column>
      </row>
      [...]
</criteriaMatrix>
```

Profiles of categories. The tag `categoryProfile` is used to describe the characteristics of a category via *central* or *limit* profiles. The following piece of code shows that alternative "alt3" is a central profile for category "cat4" and "alt1" defines the limit between categories "medium" and "good".

```
<categoriesProfiles>
      <categoryProfile>
            <alternativeID>alt3</alternativeID>
            <central>
                  <categoryID>cat4</categoryID>
            </central>
      </categoryProfile>

      <categoryProfile>
            <alternativeID>alt1</alternativeID>
            <limits>
                  <lowerCategory>
```

```
                        <categoryID>medium</categoryID>
                    </lowerCategory>
                    <upperCategory>
                        <categoryID>good</categoryID>
                    </upperCategory>
                </limits>
        </categoryProfile>
        [...]
</categoriesProfiles>
```

Contents of categories. The tag `categoriesContents` allows to store the content of each category from the perspective of the categories. The following example shows the content of category "cat1" (alternative "alt3" belongs to "cat1" with a credibility of 0.89).

```
<categoriesContents>
        <categoryContent>
                <categoryID>cat1</categoryID>
                <alternativesSet>
                        <element>
                                <alternativeID>alt3</alternativeID>
                                <value mcdaConcept="credibility">
                                        <real>0.89</real>
                                </value>
                        </element>
                        <element>
                                <alternativeID>alt4</alternativeID>
                        </element>
                        [...]
                </alternativesSet>
        </categoryContent>
        [...]
</categorieContents>
```

Assignment of alternatives. The tag `alternativesAffectations` allows to detail the content of each category from the perspective of the alternatives. The following excerpt shows that alternative "alt2" is assigned to category "cat03", the set of alternatives "alts3" belongs to the set of categories "catSet13" and alternative "alt4" belongs to an interval of categories.

```
<alternativesAffectations>
        <alternativeAffectation>
                <alternativeID>alt2</alternativeID>
                <categoryID>cat03</categoryID>
        </alternativeAffectation>

        <alternativeAffectation>
                <alternativeSetID>alts3</alternativeSetID>
                <categoriesSetID>catSet13</categoriesSetID>
        </alternativeAffectation>

        <alternativeAffectation>
```

```
                <alternativeID>alt4</alternativeID>
                <categoriesInterval>
                        <lowerBound>
                                <categoryID>medium</categoryID>
                        </lowerBound>
                        <upperBound>
                                <categoryID>veryGood</categoryID>
                        </upperBound>
                </categoriesInterval>
        </alternativeAffectation>
</alternativesAffectation>
```

Specifying a hierarchy of concepts. Finally, to specify a hierarchy of concepts (criteria, alternatives, attributes and categories), `XMCDA` proposes the `hierarchy` tag. The following code shows a hierarchy of criteria. For example criterion "economical" is made of both sub-criteria "maintenance" and "price". Note that each node can contain one or more elements.

```
<hierarchy>
        <description>
                <comment>A hierarchy of criteria
                        for a car selection problem</comment>
        </description>

        <node>
                <criterionID>economical</criterionID>
                <node><criterionID>maintenance</criterionID></node>
                <node><criterionID>price</criterionID></node>
        </node>

        <node>
                <criterionID>ecological</criterionID>
                <node><criterionID>CO2</criterionID></node>
                <node><criterionID>Cx</criterionID></node>
        </node>
        [...]
</hierarchy>
```

This overview of all the main tags of `XMCDA` shows the great flexibility and versatility of this encoding. For further details on the `XMCDA` encoding, we recommend that the interested user refers to the documentation of the `XMCDA` XML Schema which can be found on `XMCDA`'s web site at http://www.decision-deck.org/xmcda.

4. Illustration of `XMCDA` in practice

In order to illustrate the technical discourse of Sections 2 and 3, we first present in this section the `XMCDA` coding of a classical MCDA problem which has been widely discussed in the literature, namely the choice of a sports car (see Bouyssou et al., 2000, chapter 6). Then we illustrate

car ID	car name	cost ($g1$, €)	accel. ($g2$, s)	pick up ($g3$, s)	brakes ($g4$)	road-hold ($g5$)
a01	Tipo	18342	30.7	37.2	2.33	3
a02	Alfa	15335	30.2	41.6	2	2.5
a03	Sunny	16973	29	34.9	2.66	2.5
a04	Mazda	15460	30.4	35.8	1.66	1.5
a05	Colt	15131	29.7	35.6	1.66	1.75
a06	Corolla	13841	30.8	36.5	1.33	2
a07	Civic	18971	28	35.6	2.33	2
a08	Astra	18319	28.9	35.3	1.66	2
a09	Escort	19800	29.4	34.7	2	1.75
a10	R19	16966	30	37.7	2.33	3.25
a11	P309-16	17537	28.3	34.8	2.33	2.75
a12	P309	15980	29.6	35.3	2.33	2.75
a13	Galant	17219	30.2	36.9	1.66	1.25
a14	R21t	21334	28.9	36.7	2	2.25

Table 1. Data for Thierry's car selection problem

its resolution via two tools using the XMCDA standard: the XMCDA web-services (http://www.decision-deck.org/ws) and diviz (Bigaret and Meyer, 2010a).

4.1 XMCDA encoding of Thierry's car selection problem

Let us first briefly recall the main characteristics of this example and the underlying data. In 1993, Thierry, a student aged 21, is passionate about sports cars and wishes to buy a middle range 4 years old car with a powerful engine. He selects five viewpoints related to cost (criterion $g1$), performance of the engine (criteria $g2$ and $g3$) and safety (criteria $g4$ and $g5$). The list of alternatives and their evaluations on these five criteria is presented in Table 1. The cost criterion (€) and the performance criteria acceleration (seconds) and pick up (seconds) have to be minimized, whereas the safety criteria brakes and road-hold have to be maximized. Note that the values of the latter two criteria are average evaluations obtained from multiple qualitative evaluations which have been re-coded as integers between 0 and 4. Further details on these data can be found in Bouyssou et al. (2000).

As done in (Bouyssou et al., 2006, chapter 7), we suppose in this section that Thierry has already some knowledge about the 14 cars, and that he is able to express the following ranking on a few of them:

$$\text{P309-16} \succ \text{Sunny} \succ \text{Galant} \succ \text{Escort} \succ \text{R21t}.$$

Let us now show some excerpts from the XMCDA encoding of this problem. First of all, the alternatives are defined as follows:

```
<alternatives name="Thierry's potential cars">
	<alternative id="a12" name="P309">
		<description>
			<comment>Peugeot 309</comment>
		</description>
	</alternative>
	[...]
	<alternative id="a14" name="R21t">
		<description>
			<comment>Renault 21</comment>
		</description>
	</alternative>
</alternatives>
```

Then, the criteria are defined by the following piece of code:

```
<criteria>
	<criterion name="Cost" id="g1">
		<description>
			<comment>Cost in Euros</comment>
		</description>
		<scale>
			<quantitative>
				<preferenceDirection>min</preferenceDirection>
			</quantitative>
		</scale>
	</criterion>
	[...]
	<criterion name="Road-hold" id="g5">
		<description>
			<comment>Road hold (0 is worst, 4 is best).</comment>
		</description>
		<scale>
			<quantitative>
				<preferenceDirection>max</preferenceDirection>
				<minimum><real>0</real></minimum>
				<maximum><real>4</real></maximum>
			</quantitative>
		</scale>
	</criterion>
</criteria>
```

The evaluations of the cars on the criteria are stored in the following performance table:

```
<performanceTable>
	<alternativePerformances>
		<alternativeID>a11</alternativeID>
		<performance>
			<criterionID>g1</criterionID>
			<value><real>17537</real></value>
		</performance>
```

```
            [...]
            <performance>
                  <criterionID>g5</criterionID>
                  <value><real>2.75</real></value>
            </performance>
      </alternativePerformances>
      [...]
      <alternativePerformances>
            <alternativeID>a14</alternativeID>
            <performance>
                  <criterionID>g1</criterionID>
                  <value><real>21334</real></value>
            </performance>
            [...]
            <performance>
                  <criterionID>g5</criterionID>
                  <value><real>2.25</real></value>
            </performance>
      </alternativePerformances>
</performanceTable>
```

Finally, the ranking provided by Thierry can be stored as follows:

```
<alternativesValues mcdaConcept="ranking of alternatives">
      <description>
            <comment>Thierry's a priori ranking of 5 cars.</comment>
      </description>
      <alternativeValue>
            <alternativeID>a11</alternativeID>
            <value>
                  <integer>1</integer>
            </value>
      </alternativeValue>
      [...]
      <alternativeValue>
            <alternativeID>a14</alternativeID>
            <value>
                  <integer>5</integer>
            </value>
      </alternativeValue>
</alternativesValues>
```

Let us now quickly present two tools from the Decision Deck project which use the XMCDA standard.

4.2 Two tools using XMCDA

Recall that the Decision Deck project aims at developing open-source software tools implementing MCDA techniques. Next to XMCDA, the project also gave birth to the so-called XMCDA web-services and the diviz software.

The XMCDA web-services are MCDA algorithms which can be accessed online and whose calculations are performed on computing servers. Roughly speaking, these web-services allow anyone who is connected to the Internet to access a large amount of MCDA algorithms without having to

install them on their personal computer. These calculation resources use the XMCDA standard as input and output data format. A documentation of the available XMCDA web-services can be found at http://www.decision-deck.org/ws.

The diviz software (Bigaret and Meyer, 2010a) is a tool for designing, executing and sharing MCDA methods. In the context of diviz, such methods are calculation workflows of MCDA algorithms, which are available via the XMCDA web-services. The diviz tool allows to easily build these workflows via a user-friendly graphical user interface and to conveniently analyze the outputs of the various calculation elements. The backbone of diviz is the XMCDA standard: it enables the various web-services to interact (the output of an algorithm can be used as the input of another one) and it eases the visualization of the XMCDA tags via an integrated web browser and XSL transformations.

Let us now use the example of Section 4.1 to present the power of XMCDA via the construction of an MCDA workflow in diviz, based on a combination of XMCDA web-services.

4.3 Thierry's preference model

The goal of this section is to elicit Thierry's preferences and his ranking on the 14 available cars via the UTASTAR method (Siskos and Yannacopoulos, 1985) implemented as an XMCDA web-service. Among all available XMCDA web-services this task requires the use of the following ones :

- UTASTAR, which, given a performance table and a ranking on some alternatives, provides value functions which represent the decision maker's preferences;
- *computeNormalisedPerformanceTable*, which transforms a given performance table via the given value functions;
- *generalWeightedSum*, which calculates the sum of the performances of the alternatives on the criteria to obtain their overall values;
- *plotValueFunctions*, which plots the value functions;
- *plotAlternativesValuesPreorder*, which plots a graph of the ranking of the alternatives, according to their overall values.

The workflow made of these calculation elements is shown on Figure 1. As one can see, the output of UTASTAR is sent both to the input of *computeNormalisedPerformanceTable* and that of *plotValueFunctions*. The output of *computeNormalisedPerformanceTable* is then transferred

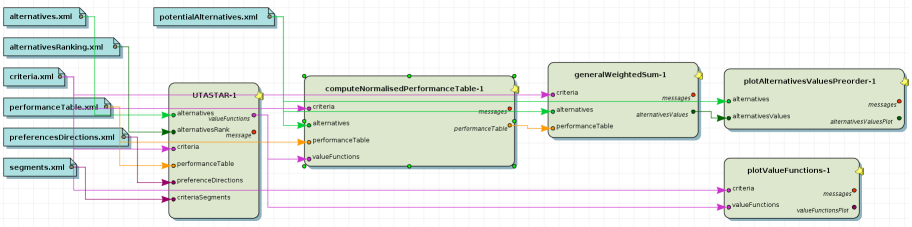

Figure 1. The workflow for the car selection problem

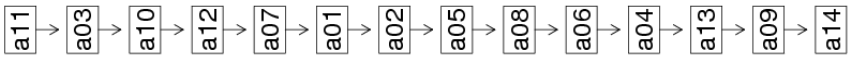

Figure 2. The ranking of the 14 cars

to the input of *generalWeightedSum*, whose output is sent to *plotAlternativesValuesPreorder* to plot the ranking of the 14 cars of Figure 2.

It is worthwhile noticing here that this chaining of the algorithms is made possible via `XMCDA` and the standardization of the inputs and the outputs of the various calculation elements.

We can see on Figure 2 that the P309-16 (a11) is considered as the best car according to this model of the preferences of Thierry and that his ranking expressed beforehand on the 5 cars is also respected.

This simple example presents well the big potential of the `XMCDA` data standard and shows how it can be used to use various algorithms without having to convert data from one format into another. Note also that this example, with all the corresponding `XMCDA` data files, can be downloaded from the diviz website at `http://www.decision-deck.org/diviz`.

5. Conclusion

At the time of writing, the official version of `XMCDA` approved by the Decision Deck Consortium is `2.1.0`. Regularly, the specifications committee receives suggestions for evolutions of `XMCDA` which can lead to a new release of the standard.

The work on `XMCDA` is clearly in an *ongoing* status. The standard is still young and might potentially evolve quickly, in case a lot of contributors show their interest in it. Note that any contribution, suggestion

or help is welcome, and we recommend contacting the author or the Decision Deck Consortium for any question on this matter.

Up to now, `XMCDA` is used by software pieces like diviz (Bigaret and Meyer, 2010a) and the `XMCDA` web-services (as shown in Section 4) and mcda calculation libraries like ws-RXMCDA (Bigaret and Meyer, 2010b), Rubis (Bisdorff, 2007), J-MCDA (Cailloux, 2010) and ws-PyXMCDA (Veneziano, 2010).

Bibliography

Bigaret, S. and Meyer, P. (2009-2010b). ws-RXMCDA. http://github.com/paterijk/ws-RXMCDA. 10, 30

Bigaret, S. and Meyer, P. (2010a). Diviz: an MCDA workflow design, execution and sharing tool. In *25th Mini-EURO Conference Uncertainty and Robustness in Planning and Decision Making (URPDM 2010)*, Coimbra. 10, 25, 28, 30

Bisdorff, R. (2007). The Python digraphs module for Rubis. http://ernst-schroeder.uni.lu/Digraph/doc/. 10, 30

Bouyssou, D., Marchant, T., Pirlot, M., Perny, P., Tsoukias, A., and Vincke, P. (2000). *Evaluation and decision models, A critical Perspective*. Kluwer's International Series. Kluwer, Massachusetts. 24, 25

Bouyssou, D., Marchant, T., Pirlot, M., Tsoukias, A., and Vincke, P. (2006). *Evaluation and decision models with multiple criteria, Stepping stones for the analyst*. Springer's International Series. Springer, New York. 25

Cailloux, O. (2010). J-MCDA. http://sourceforge.net/projects/j-mcda/. 10, 30

Decision Deck Consortium (2009). Strategic manifesto of the Decision Deck project. http://www.decision-deck.org/manifesto.html. 8

Grabisch, M., Kojadinovic, I., and Meyer, P. (2008). A review of capacity identification methods for Choquet integral based multi-attribute utility theory; Applications of the Kappalab R package. *European Journal of Operational Research*, 186(2):766–785. 10

Siskos, J. and Yannacopoulos, D. (1985). UTASTAR – An ordinal regression method for building additive value functions. *Investigaçao Operacional*, 5(1):39–53. 28

Veneziano, T. (2010). ws-PyXMCDA. http://github.com/quiewbee/ws-PyXMCDA. 30