Chapter 1
XMCDA: an XML-based encoding standard for MCDA data

Sébastien Bigaret and Patrick Meyer

Abstract Up to recently, the processing of a decision problem via several Multiple Criteria Decision Aid (MCDA) programs required to write the data in various formats. This chapter presents XMCDA, an initiative of the Decision Deck Consortium, which is a standard 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 techniques compatible with XMCDA without requiring data conversions and it simplifies the sequencing of MCDA algorithms for the resolution of complex decision problems.

1.1 Introduction

Research activities in and around the field of Multiple Criteria Decision Aid (MCDA) have developed quite rapidly over the past decades, and have resulted in various streams of thought and methodological formulations to solve complex decision problems. In particular, many so-called MCDA *methods* and have been proposed in the literature and are very often available as software programs, along with some programs implementing the algorithmic elements composing these methods.

Sébastien Bigaret
Institut Télécom, Télécom Bretagne
UMR CNRS 6285 Lab-STICC
Technopôle Brest-Iroise CS 83818, F-29238 Brest Cedex 3
Université européenne de Bretagne, e-mail: sebastien.bigaret@telecom-bretagne.eu

Patrick Meyer
Institut Télécom, Télécom Bretagne
UMR CNRS 6285 Lab-STICC
Technopôle Brest-Iroise CS 83818, F-29238 Brest Cedex 3
Université européenne de Bretagne, e-mail: patrick.meyer@telecom-bretagne.eu

Among the difficulties that arise when one wants to use these programs in practice, a major one is that they all use their own, distinct, data format to encode the MCDA problems (other difficulties exist, which are discussed in the Chapter **??**). Therefore, these data need to be re-encoded for every software application one wants to run. Moreover, this problem of the heterogeneous input and output data formats in MCDA software prevented users from combining existing algorithms, in order to create treatment chains involving multiple MCDA algorithms. Consequently, in such a situation, the resolution of a complex decision problem comes generally down to testing only one software and using various supplementary tools to analyze the results of the resolution. This can be frustrating for a lot of MCDA analysts who might like to test various algorithms on a given problem, without having to recode the instance in various data formats.

Given this situation, a group of researchers within the Decision Deck Consortium [see Decision Deck Consortium, 2009] decided to gather together to address this particular problem; the Consortium itself is presented in more details in the chapter **??**.

To allow running a problem instance through multiple methods and to allow the chaining of various MCDA algorithms, this group of researchers suggested to define a data standard, called XMCDA, which could be adopted by programs to ease their interoperability.

In this chapter, we present the latest version of XMCDA, explain its construction and motivate the choices which have been made during the elaboration process. The chapter is structured as follows: first, in Section 1.2 we present the history of XMCDA and explain the general ideas which facilitate the understanding of the structure of XMCDA. Then, in Section 1.3 we present how a lot of common MCDA related concepts can be encoded in XMCDA. Finally, in Section 1.4 we present the use of XMCDA in practice, before concluding in Section 1.5.

1.2 A first cup of XMCDA

The objective of this section is to ease the understanding 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 underlie more general structures presented in Section 1.3.

1.2.1 Technical aspects and choices for XMCDA

The XMCDA markup language is an application of XML[1], a general-purpose text format used to capture data and their structures. XML's purpose is to aid information systems in sharing structured data, especially via the Internet and to encode documents. XMCDA is defined by an XML Schema[2], a set of syntax rules (together with a set of constraints) which define its content and 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 `3.0.0`.

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 Consortium 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).

Last, the order of the elements *is* significant in an XML document ; XMCDA takes advantage of this to store the order of its elements when it matters (think values in a vector e.g.)

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 clearly intended for 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.

---

[1] `http://www.w3.org/XML/`

[2] `http://www.w3.org/XML/Schema`

[3] `http://www.w3.org/Style/XSL`

The origin of XMCDA goes back to fall 2007, where a group of researchers of the Decision Deck Consortium 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 was used by multiple software pieces like RUBIS [Bisdorff, 2007] or J-MCDA [Cailloux, 2010]. In Fall 2013, after 4 years of dedicated and distinguished service, the Consortium approved a deep-down cleaning of XMCDA which removes recurrent inconsistencies of the `2.2.0` version and consequently approved version `3.0.0`. Software using this version include diviz [Meyer and Bigaret, 2012] and the XMCDA web-services.

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.

1.2.2 Conventions

After this short history of XMCDA, we now 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 as follows: `xmcda:typeName`);

- 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 is written in medial capitals, or *camel case*, i.e. it is composed of concatenated words beginning with a capital letter, with the exception of the first letter of the name which is lower case; example: `alternativesSetsComparisons`. This allows to easily read and understand the meaning of a tag, as we use whole words and avoid acronyms and abbreviations. Other examples are `performanceTable`, storing the performance table, and `criterionValue`, storing a value related to a criterion, as, e.g., its weight. Note that objects of the same XMCDA type are 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` is the *container* of `alternative` tags).

The following three XML attributes can be found in many XMCDA tags: `id`, `name` and `mcdaConcept`. They are in general optional, except for the `id` attribute in the definition of an alternative, a set of alternatives, a criterion, a set of criteria, a category, or a set of categories. 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 Peugeot 309 from 1986</comment>
    </description>
</alternative>
```

  The `id` attribute allows to distinguish between the various alternatives and identifies them unequivocally.
- The `name` attribute allows to give a *human-readable* name to a particular MCDA object. The previous example can therefore be completed as follows:

```
<alternative id="a12" name="Peugeot 309">
    <description>
        <comment>A red Peugeot 309 from 1986</comment>
    </description>
</alternative>
```

  In a software using the XMCDA standard, this name should be displayed to the user instead of (or at least next to) the id.
- 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 kind of 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 name="Ranks of the alternatives"
                    mcdaConcept="ranks">
    <alternativeValue>
        <alternativeID>a03</alternativeID>
        <value>
            <integer>1</integer>
        </value>
    </alternativeValue>
    <alternativeValue>
        <alternativeID>a11</alternativeID>
        <value>
            <integer>2</integer>
        </value>
    </alternativeValue>
    <!-- ... -->
</alternativesValues>
```

In practice, the `mcdaConcept` should be used by algorithms to specify what content is produced or what type of data is required as input, in case of a possible ambiguity. No specific vocabulary is imposed for this XML attribute, which gives a great flexibility to XMCDA. A further example of its use is given by the following piece of code:

```
<criteriaThresholds>
    <criterionThreshold>
        <criterionID>g1</criterionID>
        <thresholds>
            <threshold id="g1i"
                       name="indifference threshold"
                       mcdaConcept="indifference">
                <constant><integer>3</integer></constant>
            </threshold>
            <threshold id="g1p"
                       name="preference threshold"
                       mcdaConcept="preference">
                <constant><integer>4</integer></constant>
            </threshold>
        </thresholds>
    </criterionThreshold>
    <!-- ... -->
</criteriaThresholds>
```

In this case, the authors of the algorithm which processes this data have specified that the discrimination thresholds have to be called "indifference" and "preference" in the `mcdaConcept` attribute so that the program can distinguish between the two `threshold` tags.
As a general rule, to leave the greatest possible flexibility to the MCDA algorithms, and in particular, to allow them to be combined with each

other, the `mcdaConcept` attribute should be used with great parsimony, especially concerning the requirements on the inputs of the algorithms.

1.2.3 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 meta-data on the information 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. In XMCDA the `xmcda:description` type is instantiated as the `description` tag which appears in all the XMCDA tags.

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>
        <author>Calvin Hobbes</author>
        <comment>Only European cars are considered.</comment>
        <keyword>cars</keyword>
        <keyword>choice</keyword>
        <creationDate>2010-06-02</creationDate>
        <!-- ... -->
    </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, a fuzzy number, 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. The bounds of an interval can be specified as open or not.

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

<value>
    <valuedLabel>
        <label>Good</label>
        <value>
            <integer>3</integer>
        </value>
    </valuedLabel>
</value>
```

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

<interval>
    <lowerBound open="true">
        <integer>4</integer>
    </lowerBound>
    <upperBound open="false">
        <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 used in many XMCDA tags (`minimum`, `maximum`, `constant`, `coefficient`, ...) which exclusively require a numeric value.

1.2.4 Elementary XMCDA tags

In this section we present some elementary tags which are used in many more complex XMCDA tags.

Value and values

As already mentioned in Section 1.2.3, 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 sequence of values. As noted in Section 1.2.1, XMCDA uses the fact that elements are naturally ordered in an XML document to store the order of its values when it matters: that order obviously matters when storing a sequence 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 type

`xmcda:value`. The following example shows a point whose coordinates are $(2.71, 23)$.

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

Function

Functions are used in complex tags related to criteria, as for example to specify a discrimination threshold or a value function. A `function` can either be a constant, an affine, a piecewise linear function or a discrete function. The following code shows a constant function, an affine function, and a discrete function described by a set of points.

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

<function>
   <affine>
      <slope><real>4.00</real></slope>
      <intercept><real>4.00</real></intercept>
   </affine>
</function>

<function>
   <discrete>
      <point>
         <abscissa><real>2.71</real></abscissa>
         <ordinate><real>23</real></ordinate>
      </point>
      <point>
         <abscissa><real>7</real></abscissa>
         <ordinate><real>45.23</real></ordinate>
      </point>
      <!-- etc. -->
   </discrete>
</function>
```

Scale

XMCDA allows to store the definition of evaluation scales, which may be *quantitative*, *qualitative* or *nominal*. The `scale` tag is in particular used to specify the evaluation scale of a criterion. The following example shows the description of a quantitative scale whose minimal value is 0 and whose max-

imal value is 100 and for which higher values are considered as better (the `minimum` and `maximum` tags are of type `xmcda:numericalValue`).

```
<scale>
   <quantitative>
      <preferenceDirection>max</preferenceDirection>
      <minimum><real>0</real></minimum>
      <maximum><real>100</real></maximum>
   </quantitative>
</scale>
```

References to criteria, alternatives, categories, ...

In many XMCDA tags, a reference has to be made to a certain criterion, a set of criteria, an alternative, a set of alternatives, a category or a set of categories. This allows to specify which of these MCDA concepts the data stored in the tag is related to. To do so, we use tags named `criterionID`, `criteriaSetID`, `alternativeID`, `alternativesSetID`, `categoryID` or `categoriesSetID` which contain a string specifying the id of a criterion, set of criteria, alternative, set of alternatives, category, or set of categories defined elsewhere (see Section 1.3.1 on how such MCDA objects 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 name="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>
```

The `active` attribute

In the tags defining criteria, alternatives or categories, an attribute named `active` (which accepts the boolean values `true` or `false`) can be used to activate or deactivate the concept it is related to. In practice, this can be very convenient if the user wishes to check the behavior of an algorithm on, e.g, a subset of criteria. Instead of deleting all the references to the unused criteria

in the XMCDA file, he simply deactivates them in their definition by putting the `active` attribute to false. The standard requires programs to ignore deactivated MCDA concepts in the general case, and that exceptions to this rule, or options allowing the user to bypass it, must be clearly documented.

*

After this preliminary presentation of some concepts and rules underlying the standard, we present in the following section how main concepts from MCDA can be encoded in XMCDA.

1.3 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 multicriteria decision aid problem. To summarize, these tags can be put in four general categories:

- definitions of MCDA concepts like criteria, sets of criteria, alternatives, sets of alternatives, categories and sets of categories;
- the performance table;
- information on preferences related to criteria, sets of criteria, alternatives, sets of alternatives, categories or sets of categories (either provided as input by a decision maker or produced as the output of an algorithm);
- parameters for programs or algorithms that do not fall in any of the previous categories, and their execution status (success or failure).

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 or even only the root tag.

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

1.3.1 Definition of alternatives, criteria, categories and performances

Alternatives / sets of alternatives

Alternatives are defined in the `alternatives` tag via the `alternative` tag. They can be either `active` or not and either be `real` or `fictive`. The XML attribute `id` 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>
    </alternative>

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

Sets of alternatives can be defined via the `alternativesSets` tag. Again, the XML attribute `id` is mandatory for each set and defines it unequivocally. The following code shows a set of two alternatives, each element of the set being characterized by a membership degree.

```
<alternativesSets>
    <alternativesSet id="set1">
        <element>
            <alternativeID>a01</alternativeID>
            <values>
                <value mcdaConcept="membership">
                    <real>0.8</real>
                </value>
            </values>
        </element>
        <element>
            <alternativeID>a02</alternativeID>
            <values>
                <value mcdaConcept="membership">
                    <real>0.75</real>
                </value>
            </values>
        </element>
    </alternativesSet>
</alternativesSets>
```

Criteria / sets of criteria

Criteria are defined and described under the `criteria` tag. For each criterion, the XML attribute `id` has to be given. A criterion can be active or not. In

the following example, the first criterion "g1" represents the power of a car. By default it is active (no `active` tag given).

```
<criteria>
   <criterion id="g1" name="horsepower">
      <description>
         <comment>Power in horsepower</comment>
      </description>
   </criterion>

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

It is also possible to define sets of criteria under the `criteriaSets` tag similarly as for sets of alternatives.

Categories / sets of categories

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

```
<categories>
   <category id="g" name="good">
      <active>true</active>
   <category>

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

   <category id="b" name="bad"/>
</categories>
```

Note that sets of categories can be defined by the `categoriesSets` tag similarly as for sets of alternatives.

The performance table

The table containing the evaluations of the alternatives on the various criteria is called performance table. It is defined by the tag `performanceTable`, and contains, for alternatives (given by a reference to its `id`), a sequence of performances, composed of a reference to a criterion `id` and a corresponding performance value. The following example shows part of such a performance table for two alternatives and two criteria.

```
<performanceTable>
   <alternativePerformances>
      <alternativeID>alt1</alternativeID>
      <performance>
```

```
            <criterionID>g1</criterionID>
            <values>
                <value>
                    <real>72.10</real>
                </value>
            </values>
        </performance>
        <performance>
            <criterionID>g2</criterionID>
            <values>
                <value>
                    <valuedLabel>
                        <label>medium</label>
                        <value>
                            <integer>3</integer>
                        </value>
                    </valuedLabel>
                </value>
            </values>
        </performance>
    </alternativePerformances>

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

*

To fully specify criteria, it might be useful to define the evaluation scale for each of them, some discrimination thresholds which might have been expressed by a decision maker, or some value functions related to them. We describe in the sequel the XMCDA tag `criteriaScales`, `criteriaThresholds` and `criteriaFunctions`, which are placeholders designed for that use.

1.3.2 Advanced information and preferences on alternatives, criteria and categories

In the previous section we have shown how the fundamental concepts from MCDA are defined in XMCDA. Here we present how supplementary data linked to these concepts is represented in the standard.

Evaluation scales of criteria

The `criteriaScale` tag associates a criterion `id` with a scale. This allows to specify on what scale the alternatives are evaluated and what preference

direction should be used. In the following example, two scales are defined for two criteria. Criterion "g1" uses a quantitative one, where the higher values are preferred by the decision maker, whereas "g2" is evaluated through a qualitative scale with 3 levels. In the latter one, the definition and the ranks of the various levels of the scale are given in the `valuedLabel` tag.

```
<criteriaScales>
   <criterionScale>
      <criterionID>g1</criterionID>
      <scales>
         <scale>
            <quantitative>
               <preferenceDirection>max</preferenceDirection>
               <minimum><real>0</real></minimum>
               <maximum><real>100</real></maximum>
            </quantitative>
         </scale>
      </scales>
   </criterionScale>

   <criterionScale>
      <criterionID>g2</criterionID>
      <scales>
         <scale>
            <qualitative>
               <preferenceDirection>min</preferenceDirection>
               <valuedLabels>
                  <valuedLabel>
                     <label>bad</label>
                     <value><integer>3</integer></value>
                  </valuedLabel>
                  <valuedLabel>
                     <label>neutral</label>
                     <value><integer>2</integer></value>
                  </valuedLabel>
                  <valuedLabel>
                     <label>good</label>
                     <value><integer>1</integer></value>
                  </valuedLabel>
               </valuedLabels>
            </qualitative>
         </scale>
      </scales>
   </criterionScale>
</criteriaScales>
```

Discrimination thresholds on criteria

In outranking methods, the decision maker may specify some discrimination thresholds on each of the criteria. To do so in XMCDA, the `criteria-`

`Thresholds` tag should be used. In the example below, for criterion "g1" two thresholds have been defined : an indifference and a preference threshold.

```
<criteriaThresholds>
   <criterionThreshold>
      <criterionID>g1</criterionID>
      <thresholds>
         <threshold id="g1i" name="indifference threshold"
                    mcdaConcept="indifference">
            <constant><integer>3</integer></constant>
         </threshold>

         <threshold id="g1p" name="preference threshold"
                    mcdaConcept="preference">
            <constant><integer>4</integer></constant>
         </threshold>
      </thresholds>
   </criterionThreshold>
</criteriaThresholds>
```

It is important to notice the importance of the `mcdaConcept` attribute here, which allows to distinguish between the two discrimination thresholds. It can be assumed that the authors of the algorithm which will use this piece of XMCDA have specified how the discrimination thresholds should be labeled in order for the program to work properly.

Value functions on criteria

In the value functions paradigm, the preferences of a decision maker can be expressed by value functions associated with the criteria. In XMCDA, this kind of preferences is stored in the `criteriaFunctions` tag. In the following piece of code, two value functions are defined for criteria "g1" and "g2". The first one is a piecewise linear function, which is represented here through a sequence of 2 segments. Each of the linear segments is defined by a head and a tail, which are points. The attribute `open` indicates whether each end is included in the interval. The second value function associated with "g2" is a linear function defined by a slope and an intercept.

```
<criteriaFunctions>
   <criterionFunction>
      <criterionID>g1</criterionID>
      <functions>
          <function>
             <piecewiseLinear>
                <segment>
                   <head>
                      <abscissa><label>bad</label></abscissa>
                      <ordinate><real>0</real></ordinate>
                   </head>
                   <tail>
                      <abscissa><label>medium</label></abscissa>
```

```
                        <ordinate><real>0.25</real></ordinate>
                    </tail>
                </segment>
                <segment>
                    <head open="true">
                        <abscissa><label>medium</label></abscissa>
                        <ordinate><real>0.25</real></ordinate>
                    </head>
                    <tail>
                        <abscissa><label>good</label></abscissa>
                        <ordinate><real>1</real></ordinate>
                    </tail>
                </segment>
            </piecewiseLinear>
        </function>
    </functions>
</criterionFunction>

<criterionFunction>
    <criterionID>g2</criterionID>
    <functions>
        <function>
            <linear>
                <slope><real>0.1</real></slope>
                <intercept><real>0</real></intercept>
            </linear>
        </function>
    </functions>
</criterionFunction>
</criteriaFunctions>
```

*

In the sequel we present generic structures which can be adapted for alternatives, criteria and categories, as well as sets of alternatives, sets of criteria and sets of categories. To avoid redundant explanations and notation, we write `x`Values for the generic structure related to the XMCDA tags `alternativesValues`, `alternativesSetsValues`, `criteriaValues`, `criteriaSetsValues`, `categoriesValues` and `categoriesSetsValues`. The same convention is used for the `x`LinearConstraints and `x`Matrix tags described hereafter.

Values associated with MCDA concepts

An `x`Value is a value associated with an element of type `x` (`x` being either alternatives, criteria or categories, or sets of them). This tag is found in compound tags called `x`Values. The following example shows a value associated with an alternative "alt1", and one associated with a set of criteria "cs3". In the first case, the stored information could be the overall value of alternative

"alt1", whereas in the second case it could be the weight of the set of criteria "cs3".

```
<alternativeValue mcdaConcept="overallValue">
    <alternativeID>alt1</alternativeID>
    <values>
        <value><real>0.8</real></value>
    </values>
</alternativeValue>

<criteriaSetValue mcdaConcept="importance">
    <criteriaSetID>cs3</criteriaSetID>
    <values>
        <value><real>0.5</real></value>
    </values>
</criteriaSetValue>
```

For both values, we assume that the alternative and the set of criteria are defined elsewhere in an `alternative` and a `criteriaSet` tag. The `mcdaConcept` attributes are again not mandatory, and they depend on the specifications of the program which uses or produces the XMCDA file.

Linear constraints related to MCDA concepts

XMCDA also allows to represent linear constraints related to alternatives, criteria and categories, or sets of them. The following example shows the representation of the constraint

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

in the standard:

```
<criteriaLinearConstraints>
    <variables>
        <variable id="epsilon"></variable>
    </variables>
    <constraint id="c1">
        <elements>
            <element mcdaConcept="weight">
                <criterionID>c2</criterionID>
                <coefficient>
                    <real>2</real>
                </coefficient>
            </element>
            <element mcdaConcept="weight">
                <criterionID>c4</criterionID>
                <coefficient>
                    <real>-3</real>
                </coefficient>
            </element>
            <element>
                <variableID>epsilon</variableID>
```

```
            <coefficient>
                <real>-1</real>
            </coefficient>
        </element>
    </elements>
    <operator>leq</operator>
    <rhs>
        <real>0</real>
    </rhs>
  </constraint>
</criteriaLinearConstraints>
```

ε is a variable which needs to be determined by the linear program. It is declared in the `variable` tag, and can then be referenced in the constraint. The `operator` tag can either be `eq` ($=$), `leq` ($\leq$) or `geq` ($\geq$). Linear constraints related to sets of `x` can similarly be represented in the `xSetsLinearConstraints` tags.

Matrices related to MCDA concepts

An `xMatrix` allows to represent matrices of values on criteria, alternatives and categories, or sets of them. The scale of the values can be specified in the `valuation` tag. 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">
  <row>
    <criterionID>g01</criterionID>
    <!-- ... -->
    <column>
      <criterionID>g02</criterionID>
      <values>
        <value><real>0.9</real></value>
      </values>
    </column>
    <column>
      <criterionID>g03</criterionID>
      <values>
        <value><real>-0.8</real><value>
      </values>
    </column>
  </row>
  <!-- ... -->
</criteriaMatrix>
```

Among other things, this structure can also represent relations or graphs between MCDA concepts (like outranking relations for example or partial preorders on alternatives).

*

In the context of sorting or clustering techniques, two further concepts need to be representable in XMCDA: the description of categories in terms of profiles, and the assignments of the alternatives to the categories.

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 the "medium" category is bounded by alternatives "p1" and "p2", whereas the "high" category has alternative "p3" as a central profile. The `categoryProfile` tags are grouped in a `categoriesProfiles` compound tag.

```
<categoriesProfiles>
   <categoryProfile>
      <categoryID>medium</categoryID>
      <bounding>
         <lowerBound>
            <alternativeID>p1</alternativeID>
         </lowerBound>
         <upperBound>
            <alternativeID>p2</alternativeID>
         </upperBound>
      </bounding>
   </categoryProfile>

   <categoryProfile>
      <categoryID>high</categoryID>
      <central>
         <alternativeID>p3</alternativeID>
      </central>
   </categoryProfile>
</categoriesProfiles>
```

Separation profiles are a special case of the bounding profiles, where the lower bound of a category corresponds to the upper bound of the category below.

Assignment of alternatives

The tag `alternativesAssignments` allows to detail to which category or categories each of the alternatives is assigned. The following excerpt shows that alternative "alt2" is assigned to category "cat03" with a credibility of 0.8, the set of alternatives "alts3" belongs to the set of categories "catSet13" and alternative "alt4" is assigned to an interval of categories.

```
<alternativesAssignments>
   <alternativeAssignment>
      <alternativeID>alt2</alternativeID>
      <categoryID>cat03</categoryID>
      <values>
```

```
            <value mcdaConcept="credibility"><real>0.8</real></value>
        </values>
    </alternativeAssignment>

    <alternativeAssignment>
        <alternativesSetID>alts3</alternativesSetID>
        <categoriesSetID>catSet13</categoriesSetID>
    </alternativeAssignment>

    <alternativeAssignment>
        <alternativeID>alt4</alternativeID>
        <categoriesInterval>
            <lowerBound>
                <categoryID>medium</categoryID>
            </lowerBound>
            <upperBound>
                <categoryID>veryGood</categoryID>
            </upperBound>
        </categoriesInterval>
    </alternativeAssignment>
</alternativesAssignments>
```

1.3.3 Program specific data

Input parameters for programs

Programs or algorithms may require specific parameters in order to guide the resolution of a decision problem. Those parameters may not necessarily be linked to MCDA concepts, and they are specified within the dedicated tag `programParameters`, which consists in a list of `programParameter` tags. The following example presents a parameter specifying the maximal number of iterations of an algorithm and a parameter specifying a minimal separation threshold between the overall values of two consecutive alternatives in a ranking.

```
<programParameters>
    <programParameter id="nb_iter_max"
                      name="maximal number of iterations">
        <values>
            <value>
                <integer>1000</integer>
            </value>
        </values>
    </programParameter>
    <programParameter id="min_separation_threshold"
                      name="minimal separation threshold">
        <values>
            <value>
                <real>0.01</real>
```

```
            </value>
        </values>
    </programParameter>
</programParameters>
```

As expected, the attribute `id` should be reserved for automatic processing. Obviously, each program must describe in its documentation the different `id`s it expects, along with the domain of validity for their respective values.

Execution results of programs

Since one of the goals of XMCDA is inter-operability, we also have a standard way for programs to communicate their return status, i.e. to success or failure.

This status is stored in the dedicated `programExecutionResult` tag; it can be either `ok`, `warning`, `error` or `terminated`. Additionally, a program may produce some human-readable messages to provide further information on its execution status. The following example shows how a program will use this tag to signal that the execution of the algorithm has failed because one of its parameter had an invalid value.

```
<programExecutionResult>
    <status>error</status>
    <messages>
        <message>
            <text>Parameter nb_iter_max: invalid value 7.3 (real)
The value must be a positive integer</text>
        </message>
    </messages>
</programExecutionResult>
```

We recommend to keep the usage of messages focused on the explanation of the execution status; in particular, they are not intended to hold logging messages produced by a program, including debugging messages.

The semantics of the status is summarized here:

- `ok`: successful execution.
- `warning`: successful execution, but the results need to be validated. For example, this can be the case when the program has made some hypothesis which cannot be automatically validated.
- `error`: the program detected a problem and stopped its execution.
- `terminated`: the program was terminated by an external cause (a crash e.g.). This status may be used by the program itself if it is capable to handle such cases, or by its execution environment.

The full details about the semantics of the return status (including what can be inferred on the validity of a program's outputs) can be found in the XMCDA reference documentation.

*

This overview 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 full documentation of the XMCDA XML Schema which can be found on XMCDA's web site at http://www.decision-deck.org/xmcda.

1.4 Illustration of XMCDA in practice

In order to illustrate the technical discourse of Sections 1.2 and 1.3, we 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]. Further illustrations of the use of XMCDA can be found in Chapter where we present the diviz workbench [Meyer and Bigaret, 2012] and the XMCDA web-services (http://www.decision-deck.org/ws) which both extensively make use of XMCDA.

1.4.1 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 21 years old student, plans to buy a middle-range, second-hand car with a powerful engine. To help with its choice, he considers five viewpoints related to cost (criterion *g1*), performance of the engine (criteria *g2* and *g3*) and safety (criteria *g4* and *g5*). Table 1.1 summarizes the alternatives (the cars) and their evaluations on the five criteria he considers.

Two of these criteria have to be maximized, namely: the road-hold and the safety criteria; the remaining three criteria have to be minimized (cost, and performances of acceleration and pickup, both of which are measured in seconds).

The reader willing to get further details on these data will refer to Bouyssou et al. [2000].

As done in [Bouyssou et al., 2006, chapter 7], Thierry also has some knowledge about the 14 cars already, and 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 illustrate how this problem can be encoded using the XMCDA data format. First of all, the alternatives are defined as follows ("..." denotes an ellipsis in the examples, so that they remain compact):

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 1.1 Data for Thierry's car selection problem

```
<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 id="g1" name="Cost">
        <description>
            <comment>Cost in Euros</comment>
        </description>
    </criterion>
    <!-- ... -->
    <criterion id="g5" name="Road-hold">
        <description>
            <comment>Road hold (0 is worst, 4 is best).</comment>
        </description>
    </criterion>
</criteria>
```

The evaluation scales of the various criteria are stored in the `criteriaScales` tag as follows:

```
<criteriaScales>
    <criterionScale>
```

```
        <criterionID>g1</criterionID>
        <scales>
            <scale>
                <quantitative>
                    <preferenceDirection>min</preferenceDirection>
                </quantitative>
            </scale>
        </scales>
    </criterionScale>
    <!-- ... -->
    <criterionScale>
        <criterionID>g5</criterionID>
        <scales>
            <scale>
                <quantitative>
                    <preferenceDirection>max</preferenceDirection>
                    <minimum><real>0</real></minimum>
                    <maximum><real>4</real></maximum>
                </quantitative>
            </scale>
        </scales>
    </criterionScale>
</criteriaScales>
```

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

```
<performanceTable>
    <alternativePerformances>
        <alternativeID>a11</alternativeID>
        <performance>
            <criterionID>g1</criterionID>
            <values>
                <value><real>17537</real></value>
            </values>
        </performance>
        <!-- ... -->
        <performance>
            <criterionID>g5</criterionID>
            <values>
                <value><real>2.75</real></value>
            </values>
        </performance>
    </alternativePerformances>
    <!-- ... -->
    <alternativePerformances>
        <alternativeID>a14</alternativeID>
        <performance>
            <criterionID>g1</criterionID>
            <values>
                <value><real>21334</real></value>
            </values>
        </performance>
        <!-- ... -->
        <performance>
```

```
            <criterionID>g5</criterionID>
            <values>
                <value><real>2.25</real></value>
            </values>
        </performance>
    </alternativePerformances>
</performanceTable>
```

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

```
<alternativesValues name="ranks">
    <description>
            <comment>Thierry's a priori ranking of 5 cars.</comment>
    </description>
    <alternativeValue>
        <alternativeID>a11</alternativeID>
        <values>
            <value><integer>1</integer></value>
        </values>
    </alternativeValue>
    <!-- ... -->
    <alternativeValue>
        <alternativeID>a14</alternativeID>
        <values>
            <value><integer>5</integer></value>
        </values>
    </alternativeValue>
</alternativesValues>
```

The interested reader will find this example in Chapter**??**, where it serves as the material on which a decision aid process is demonstrated through the use of the diviz workbench (a Decision Deck Consortium software).

1.5 Conclusion

At the time of writing, the official version of XMCDA approved by the Decision Deck Consortium is `3.0.0`. 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 but it has already proven solid, operational and stable by being used by a hundred or so web-services; however, all aspects of MCDA found in the literature are not fully covered yet; the standard evolves and integrates new concepts when they are brought to our attention. Hence, any contribution, suggestion or help are welcome, and we invite you to contact the authors or the Decision Deck Consortium for anything related to this matter.

XMCDA is used by software pieces like diviz [Meyer and Bigaret, 2012] and the XMCDA web-services and MCDA calculation libraries like ws-RXMCDA

[Bigaret and Meyer, 2009-2010], Rubis [Bisdorff, 2007], J-MCDA [Cailloux, 2010] and ws-PyXMCDA [Veneziano, 2010].

A reference implementation in Java is provided by the Consortium and is available on its website [Decision Deck Consortium, 2014]. This is a library which enables the reading and writing of XMCDA files, and the manipulation of the corresponding XMCDA objects. At the time of writing, this library is also available for Python and R; you'll find more on the current status on the XMCDA Home Page, plus tutorials and documentation on how to use these libraries.

References

Sébastien Bigaret and Patrick Meyer. ws-RXMCDA, 2009-2010. http://github.com/paterijk/ws-RXMCDA.

Raymond Bisdorff. The Python digraphs module for Rubis, 2007. http://ernst-schroeder.uni.lu/Digraph/doc/.

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

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

Olivier Cailloux. J-MCDA, 2010. http://sourceforge.net/projects/j-mcda/.

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

Decision Deck Consortium. XMCDA Home Page, 2014. http://www.decision-deck.org/xmcda.

M. Grabisch, I. Kojadinovic, and P. Meyer. 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, 2008. doi: 10.1016/j.ejor.2007.02.025.

Patrick Meyer and Sébastien Bigaret. diviz: a software for modeling, processing and sharing algorithmic workflows in MCDA. *Intelligent Decision Technologies*, 6(4):283–296, 2012. doi:10.3233/IDT-2012-0144.

Thomas Veneziano. ws-PyXMCDA, 2010. http://github.com/quiewbee/ws-PyXMCDA.