Abstract The objective of the Decision Deck project is to collaboratively develop open-source software tools implementing Multiple Criteria Decision Aid (MCDA) techniques. This chapter presents three selected initiatives from this project, namely the XMCDA data standard, the XMCDA web-services and the diviz software, which altogether enable to test and combine various MCDA algorithms via a unified data model, homogeneous software pieces and a user-friendly experimentation platform.

Keywords: MCDA software, data standard, web-services, algorithmic workflows
Selected initiatives of the Decision Deck project
Sébastien Bigaret and Patrick Meyer

Contents

1 Introduction 8

2 The XMCDA data standard 10
   2.1 Technical aspects and choices for XMCDA 11
   2.2 Overview of XMCDA 12
   2.2.1 Rough structure 14
   2.2.2 Generic XMCDA types 15
   2.2.3 Method and project specific data 17
   2.2.4 Definition of alternatives, criteria, attributes and categories 18
   2.2.5 Advanced information on alternatives, criteria, attributes and categories 19
   2.3 Example 24
   2.4 Concluding remarks and perspectives for XMCDA 27

3 XMCDA web-services 27
   3.1 The general approach 28
   3.2 The web-services 30
   3.3 Software architecture 31
   3.4 How to access the XMCDA web-services 33
   3.5 Concluding remarks and perspectives for the XMCDA web-services 34

4 diviz 34
   4.1 Description and properties of diviz 34
   4.2 A quick look at the architecture 38
   4.3 Application 38
   4.4 Concluding remarks and perspectives for diviz 46

5 Conclusion and call for participation 46

Acknowledgments 47
1. Introduction

Research activities in and around the field of Multiple Criteria Decision Aid (MCDA) have developed quite rapidly over the past years, and they 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 other scientific research fields, as, e.g., statistics or data mining, there exist software platforms which allow to easily compare different analysis methods and to test them on a given dataset inside a common framework. Among the most famous ones, one can cite platforms such as the GNU R statistical system by the R Development Core Team (2005) or the Weka suite of machine learning software by Hall et al. (2009). Both of these suites are open-source and OS independent, which has certainly contributed to their large dissemination and acceptance among many researchers and users.

In order to overcome the earlier mentioned difficulties linked to the software situation in the field of MCDA, a group of researchers has got together to create the Decision Deck project. Its objective is to collaboratively develop open-source software tools implementing MCDA techniques. As such, its purpose is to provide effective tools for at least three types of users:

- practitioners who use MCDA tools to support actual decision makers involved in real world decision problems;


**Introduction**

- teachers who present MCDA methods in courses, for didactic purposes;
- researchers who want to test and compare methods or to develop new ones.

From a practical point of view, the members of the Decision Deck project work on developing multiple software resources that are able to interact. Consequently, several complementary efforts focusing on different aspects contribute to Decision Deck’s various goals. At the time of writing, 5 scientific initiatives can be identified inside the project:

- **XMCDA**: a standardized XML recommendation to represent objects and data structures issued from the field of MCDA. Its main objectives are to allow different MCDA algorithms to interact and problem instances stored in XMCDA to be analyzed by various MCDA algorithms;
- **XMCDA** web-services: distributed open-source computational MCDA resources;
- **diviz**: an open-source Java client and server for algorithmic MCDA workflow design, execution and sharing (via XMCDA web-services);
- **d2**: a rich Java client offering a few MCDA methods;
- **d3**: a rich Internet application prototype for XMCDA web-services calling.

The coordination of the Decision Deck project is done by the Decision Deck Consortium. It is a French non-profit association which steers and manages the project along the lines of the project’s manifesto (see Decision Deck Consortium (2009)) and is headed by an administration board.

To stimulate the research activities of and around the project, the Decision Deck Consortium organizes bi-annual workshops which enable researchers to present their latest contributions and to promote the results outside the members of the project.

In this chapter, we focus on three of the initiatives of the Decision Deck project: the **XMCDA** data standard, the **XMCDA** web-services and the **diviz** software. They represent a typical sample of the various scientific research efforts of the project and underline the complementary efforts leading to usable MCDA tools. The chapter is structured as follows: first, in Section 2 we introduce the **XMCDA** data standard and its main purpose and give some general ideas about its use in practice. Then, in Section 3
we present the ideas behind the XMCDA web-services and their current implementation in Decision Deck’s web-services framework. We then continue by introducing the diviz software and its main characteristics in Section 4, before concluding in Section 5.

2. The XMCDA data standard

As mentioned in Section 1, one of the major difficulties linked to the use of the tools from the MCDA research domain is the heterogeneity of the available software programs and their input and output data formats. Unfortunately, this lack of standardization of the data does not allow to combine the existing tools and to create treatment chains which would involve 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 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.

In order to overcome these difficulties, and in particular 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 section we present XMCDA and explain its construction and motivate the choices which have been made during the elaboration process. The objective of this section is not to give a detailed technical description of XMCDA, but rather to present a quick overview which should help a beginner to better understand its philosophy and to ease the adoption of the standard. In Section 2.1 we first discuss the general principles of XMCDA and the course which has lead to the current version. Then, in Section 2.2 we give details on the elements which set up XMCDA and show how to encode classical concepts issued from MCDA. Finally, in Section 2.3 we present a detailed description of the XMCDA encoding of a classical example from the literature before drawing some conclusions in Section 2.4.
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 \url{http://www.decision-deck.org/xmcd}. 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 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 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

\(^1\)\url{http://www.w3.org/XML/}
\(^2\)\url{http://www.w3.org/XML/Schema}
\(^3\)\url{http://www.w3.org/Style/XSL}
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 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 gave birth in Spring 2008 to 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.

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 adding of a new subtag in an XMCDA type;
- change from XMCDA a.b.c to XMCDA a.(b+1).0 for more substantial modifications on the standard, like, e.g., the adding of a new tag (or XMCDA type) under the root element;
- 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 type.

Any modification request on the standard should be submitted to the specification committee, which then discusses this proposal and either approves it or suggests an alternative solution to the raised issue.

2.2 Overview of XMCDA

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 and recipes on how to store MCDA data in XMCDA. The reader who wishes to have a look at the implementation of a classical MCDA problem in XMCDA can skip this section for now and head directly to Section 2.3.

In order to avoid misunderstandings, let us first briefly introduce a few conventions used in this section.
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, etc.);

- The term **XMCDA type** stands for an XML structure that was created for the purpose of XMCDA and it also represents the corresponding XMCDA tag.

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 tag names `<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 attributes can be found in any of the main data 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 attributes has a particular purpose in XMCDA:

- The **id** attribute allows to identify an object with a **machine readable** code or identifier. As an illustration consider the following set of two alternatives “a03” and “a04” which is identified by “set1”.

  ```xml
  <alternativesSet id="set1">
    <element>
      <alternativeID>a03</alternativeID>
    </element>
    <element>
      <alternativeID>a04</alternativeID>
    </element>
  </alternativesSet>
  ```

- The **name** attribute allows to give a **human-readable** name to a particular object. As an illustration consider the following code, which shows how a named parameter, specifying the number of iterations of an algorithm, could be passed to an MCDA method.

  ```xml
  <parameter id="numIt" name="number of iterations">
    <value>
      <integer>3</integer>
    </value>
  </parameter>
  ```
The `mcdaConcept` attribute allows to identify the MCDA concept linked to a particular instance of an XMCDA type. To illustrate this, consider the following example, where the set of alternatives `{a03, a04}` is considered as a kernel of a graph.

```xml
<alternativesSet mcdaConcept="kernel" name="acceptable cars">
  <element>
    <alternativeID>a03</alternativeID>
  </element>
  <element>
    <alternativeID>a04</alternativeID>
  </element>
</alternativesSet>
```

It is up to each user of XMCDA to determine if, and how he wants to name a certain MCDA concept. As a consequence, XMCDA is very flexible when it comes to adapting to various methodological vocabularies.

To give a correct and intuitive presentation of XMCDA, in the following sections we first present the general outline of an XMCDA file. Then, we introduce a few generic XML structures which can be considered as the atoms of the XMCDA standard, before detailing the many MCDA concepts which can be represented in XMCDA.

### 2.2.1 Rough structure.

As XMCDA is written in XML, it contains several tags under the root tag `<XMCDA/>`. They describe various data related to a decision aid problem. To summarize, we put them in five general categories:

- information related to the current decision aid project or the description of the XMCDA file: `<projectReference/>`;
- output messages from methods or algorithms (log or error messages) and input information for methods or algorithms (parameters): `<methodMessages/>` and `<methodParameters/>`;
- description of major MCDA concepts as attributes, criteria, alternatives, categories: `<attributes/>`, `<criteria/>`, `<alternatives/>`, ...;
- the performance table: `<performanceTable/>`;
- preferences related to criteria, alternatives, attributes or categories: `<criteriaValues/>`, `<criteriaLinearConstraints/>`, ...
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.

2.2.2 Generic XMCDA types. In this section we present the basic XML structures on which most of the general XMCDA types are built.

Values. In most of the XMCDA types, one needs to be able to store values containing information related to some MCDA-linked concepts. The XMCDA type <value/> is used to store such data, and can represent an integer, a real number (float), an interval, a rational, a nominal value, an ordinal value, a not available value or a binary64 string. The following example presents a list of values containing 4 elements of different types.

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

Note that there also exists an XMCDA type called numericValue which restricts value to numeric values.

Intervals. The type <value/> can represent an interval of numeric or ordinal values. The following example presents the interval of integers [4, 8].

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

Points. Some more complex XMCDA types, as, e.g., <function/>, require the concept of point. The abscissa as well as the ordinate are of the type <value/>. The following example shows a point whose coordinates are (2.71, 23).
Scales. XMCDA allows to store the definition of evaluation scales, which may be quantitative, qualitative or nominal. This `<scale/>` type 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.

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

Functions. 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.

```xml
<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>
```

Description. Each tag defined in XMCDA owns an optional description which allows to precisely specify its content. A short example is given hereafter for a list of alternatives.

```xml
<alternatives>
  <description>
    <title>The list of cars.</title>
    <comment>Only European cars are considered.</comment>
  </description>
  [...]
</alternatives>
```
The `<description/>` tag is used to add some comments on a value, a criterion, etc., or to specify, e.g., the author of a piece of information. All the tags are optional.

2.2.3 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. It is made of tags from the `<description/>` type which was presented earlier. The following code gives a short example of such a description.

```xml
<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.

```xml
<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.

```xml
<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.
In the following sections we present the XMCDA types which allow to store various information related to the field of MCDA.

2.2.4 Definition of alternatives, criteria, attributes and categories.

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

```xml
<alternatives mcdaConcept="decision actions">
  <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">
    <type>fictive</type>
  </alternative>
</alternatives>
```

Note that sets of alternatives can be defined via the `<alternativesSets/>` tag (see Section 2.2.5 for further details).

Criteria and attributes. Criteria are defined and described under the `<criteria/>` tag. For each criterion an 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].

```xml
<criteria>
  <criterion id="g1" name="horsepower">
    <description>
      <comment>Power in horsepower</comment>
    </description>
    <scale>
      <preferenceDirection>max</preferenceDirection>
      <minimum><real>50</real></minimum>
      <maximum><real>200</real></maximum>
    </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 under the `<criteriaSets/>` tag (see Section 2.2.5 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. The following example defines two categories of students, the second one being currently inactive.

```xml
<categories>
  <category id="g" name="goodStudents">
    <active>true</active>
  </category>
  <category id="m" name="mediumStudents">
    <active>false</active>
  </category>
</categories>
```

Note that sets of categories can be defined by the `<categoriesSets/>` tag (see Section 2.2.5 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.

```xml
<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>
```

2.2.5 **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.

To simplify the presentation of the XMCDA format here, we will focus on a few generic structures which are adapted for alternatives, criteria, attributes and categories in XMCDA. To avoid some redundant explanations and notation, we write xSet for the generic structure related to the XMCDA types `<alternativesSet/>`, `<criteriaSet/>`, `<attributesSet/>` and `<categoriesSet/>`. The same convention is used for the xValue, xLinearConstraint, xComparisons and xMatrix types, described hereafter.

**xSet.** An xSet is a set of elements of type x. Each of the elements, as well as the whole set, can be given a value (which allows to simply represent ordered sets). The following code represents a set of alternatives, where one alternative is valued (e.g., 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.

```xml
<alternativesSet id="good1" mcdaConcept="goodChoice">
  <element>
    <alternativeID>a03</alternativeID>
  </element>
  <element>
    <alternativeID>a04</alternativeID>
    <value><real>0.88</real></value>
  </element>
  <values name="qualities">
    <value name="similarity"><real>0.2</real></value>
    <value name="dissimilarity"><real>0.7</real></value>
  </values>
</alternativesSet>
```

**xValue.** An xValue 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”.

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

<criterionValue>
  <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
The **XMCDA** data standard that set explicitly in the `<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

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

```xml
<criteriaLinearConstraints mcdaConcept="weight">
  <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 (≥)**.

**xComparisons.** xComparisons allow to represent valued binary relations on criteria, alternatives, categories and attributes. The `<valuation/>` sub-tag of **XMCDA** type **scale** 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.

```xml
<alternativesComparisons mcdaConcept="outrankingRelation">
  <valuation>[..]</valuation>
  <relationType>outranking</relationType>
  <pairs>
    <pair>
      <initial><alternativeID>a01</alternativeID></initial>
      <terminal><alternativeID>a01</alternativeID></terminal>
      <value><real>0.00</real></value>
    </pair>
    <pair>
      <initial><alternativeID>a01</alternativeID></initial>
      <terminal><alternativeID>a02</alternativeID></terminal>
      <value><real>1.00</real></value>
    </pair>
  </pairs>
</alternativesComparisons>
```

---

4 We use here the syntax from the **LaTeX** markup language.
xMatrix. An xMatrix allows to represent matrixes of values on criteria, alternatives, attributes and categories. The following example presents a short example of a correlation matrix between criteria.

```xml
<criteriaMatrix mdaConcept="correlationMatrix">
  <row>
    <criterionID>g01</criterionID>
    <column>
      <criterionID>g01</criterionID>
      <value>
        <real>1.00</real>
      </value>
    </column>
    <column>
      <criterionID>g02</criterionID>
      <value>
        <real>-0.33</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”.

```xml
<categoriesProfiles>
  <categoryProfile>
    <alternativeID>alt3</alternativeID>
    <central>
      <categoryID>cat4</categoryID>
      <value><real>0.354</real></value>
    </central>
  </categoryProfile>
  <categoryProfile>
    <alternativeID>alt1</alternativeID>
    <limits>
      <lowerCategory>
        <categoryID>medium</categoryID>
        <value><real>1.00</real></value>
      </lowerCategory>
      <upperCategory>
        <categoryID>good</categoryID>
        <value><real>0.678</real></value>
      </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">0.89</value>
      </element>
      ...
    </alternativesSet>
  </categoryContent>
</categoriesContents>
```

Affectations 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>
</alternativesAffectations>
```
Specifying a hierarchy of concepts. Finally, to specify a hierarchy of concepts (criteria, alternatives, attributes and categories), XMCD A 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.

```xml
<hierarchy>
  <description>
    <comment>A hierarchy of criteria</comment>
  </description>
  <node>
    <criterionID>economical</criterionID>
    <node>
      <criterionID>price</criterionID>
    </node>
    <node>
      <criterionID>maintenance</criterionID>
    </node>
  </node>
  <node>
    <criterionID>ecological</criterionID>
    <node>
      <criterionID>CO2</criterionID>
    </node>
    <node>
      <criterionID>Cx</criterionID>
    </node>
  </node>
[...]
</hierarchy>
```

2.3 Example

In order to illustrate the technical discourse of Section 2.2, we choose to present the XMCD A 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). Let us 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 $g_1$), performance of the engine (criteria $g_2$ and $g_3$) and safety (criteria $g_4$ and $g_5$). 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).
The XMCDA data standard

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

Table 1. Data for Thierry’s car selection problem

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:

P309-16 > Sunny > Galant > Escort > R21t.

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

```xml
<alternatives>
    <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:

```xml
<criteria>
    <criterion name="Cost" id="g1"/>
    <criterion name="Road-hold" id="g5"/>
</criteria>
```
The evaluations of the cars on the criteria are stored in the following performance table:

<table>
<thead>
<tr>
<th>Alternative ID</th>
<th>Criterion ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a11</td>
<td>g1</td>
<td>17537</td>
</tr>
<tr>
<td>a14</td>
<td>g1</td>
<td>21334</td>
</tr>
<tr>
<td>a11</td>
<td>g5</td>
<td>2.75</td>
</tr>
<tr>
<td>a14</td>
<td>g5</td>
<td>2.25</td>
</tr>
</tbody>
</table>

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

In Section 4 we present various analyses of this example via the diviz software and XMCDA web-services, both relying on XMCDA files.
2.4 Concluding remarks and perspectives for XMCDA

At the time of writing, the official version 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 very 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 authors for any question on this matter.

Up to now, XMCDA is used by software pieces like diviz and the XMCDA web-services (which will be presented hereafter) and MCDA calculation libraries like ws-RXMCDA by Bigaret and Meyer (2010b), Rubis by Bisdorff (2007) or J-MCDA by Cailloux (2010).

A few serialisation / deserialisation libraries are also available. They allow to conveniently read and write XMCDA files in various programming languages. Currently, one can use the RXMCDA (Bigaret and Meyer (2010a)) library for the R statistical software, the PyXMCDa library for Python (Veneziano (2010a)) and the J-XMCDA library for Java (Cailloux (2010)). The interested reader should refer to the XMCDA website for regular updates on these libraries.

3. XMCDA web-services

In Section 2.1 we have presented the application domain of XMCDA, namely the algorithms used in an MCDA procedure. Here, we focus on these algorithms, which allow to manipulate, analyze and transform MCDA data through automated steps. Note that we do not focus on the graphical user interface (GUI) aspects: an example of such a GUI is discussed in Section 4.

The XMCDA web-services are the initiative of the Decision Deck project which enables an easy access to MCDA algorithms. From a general point of view, a web-service is an application which can be accessed via the Internet and is executed on a remote system. One of the great advantages of such online programs is their availability to anyone at any time and any place and on any computer which is connected to the Internet.

Before diving into the details on the XMCDA web-services, we first present the elements that drove us to choose this particular technology in Section 3.1. Then, in Section 3.2 we define the properties of XMCDA web-services and present the existing services, before introducing the big picture of the general software architecture in Section 3.3. Finally,
we detail how they can be accessed in Section 3.4 before drawing some conclusions in Section 3.5.

3.1 The general approach

The Decision Deck project is not only about producing open-source software: it is also about sharing software that can be widely reused by anyone. Consequently, very quickly, the researchers of the project sought for a solution which would meet this requirement.

The web-services option was chosen for a certain number of reasons, which are based on the following requirements:

1. Facilitate inter-operability between programs: use XMCDA;
2. Capitalize on the long-term;
3. Maximize the potential submitters / minimize the contribution effort;
4. Produce small components, instead of big black-boxes;
5. Ease the accessibility of the softwares.

Let us now detail each of these points in order to better understand the choices that were made for this Decision Deck initiative.

Interoperability We have already stressed the importance of inter-operability in the previous sections, in order to facilitate the MCDA process. As a result of this requirement, the XMCDA standard was born, and it should therefore be no surprise that the algorithmic components developed and proposed by the members of the Decision Deck project are based on this standard.

However, this is not meant to be exclusive: a given program may accept several formats, as long as XMCDA is one of them.

Capitalization A lot of implementations of algorithms have a very short life-cycle because the bigger software they are bundled into is disrupted. In such a case, most of the time, these implementations also get lost, because their extraction from the bigger software is time-consuming or very difficult. Consequently, very often, these algorithms are reimplemented in newer softwares, which is definitely not very sustainable.

Capitalization and reutilization are fairly common topics in software engineering; succeeding in addressing them is since the beginning of the project in the very center of all of Decision Deck’s initiatives.
Maximizing the potential contributors  In the MCDA field, like in any research domain, the people that are likely to implement specialized algorithms are researchers, which are not necessarily computer scientists. From this point of view, software engineering is not an activity per se; the computer is simply a toolbox, in which some of the tools (or languages) are used as a support for the research effort. As a consequence, when it comes to participating to a general programming effort, an imposed programming language is perceived as a hard constraint. Consequently, the necessity to learn such a new language is often considered as not being worth the effort or that it is not appropriate for the given research field.

In order to satisfy the requirement of maximization of the potential contributors, the key point is not to impose a software technology on anyone: any languages should be welcome.

Proposing elementary components  So-called MCDA methods are very often sequences of various more elementary algorithms. In most of the implemented MCDA tools, this is generally poorly visible. In order to clear things up, such elementary modules should be available as separate software piece, which, if properly chained, would rebuild the original methods.

This vision leads to at least four improvements:

- Remove the black box effect of certain softwares;
- Better understand the heart of the methods and their similarities / dissimilarities;
- Avoid repeated and unnecessary reimplementations of the same algorithms;
- Easily test variations of methods by replacing one of its components by another one.

Making software accessible  MCDA research suffers from a lack of availability of the published methods and algorithms. In a project where every software component shares the same environment, this can easily be solved. However, as already mentioned earlier, to maximize the number of potential contributors, the programming language should not be imposed.

Allowing anyone to participate with its favorite language has therefore its counterpart: there must exists a mean to put all elements together and propose them under a unified, easy to access interface.
3.2 The web-services

Web-services solve the issues raised by the requirements mentioned in Section 3.1: by providing all algorithms as web-services, anyone can use them exactly in the same manner, through a unique interface.

For short, an XMCDA Decision Deck service has the following properties:

- it “speaks” XMCDA: its inputs and outputs are formatted using this standard. Consequently, all web-services are guaranteed to be able to inter-operate;
- it can be made of (almost) any programming language (see Section 3.3 for further details);
- It is available anywhere, and it can be run simply, without having to take care of the specific environment needed by every program (see Section 3.4 for ways of accessing the web-services).

Obviously, web-services are not the answer for everything. Let’s take an example: suppose that one wants to develop an application focusing on a specific method, where some changes in the parameters dynamically update a graphical result (think of the ranking of the alternatives, with respect to the weights of the criteria). In this case, using web-services is probably not appropriate, since it may introduce unpleasant latencies. However, and since many algorithmic components already exist in the catalog of the XMCDA web-services, it is likely that one will be able to pick up the source code of one of them and re-use it in one’s own software. On the other hand, if a component does not exist in this catalog yet, or if it exists but is coded in the wrong language for one’s needs, one may choose to implement it and add it in the web-services collection in addition to integrating it in one’s own software. Doing this ensures that anyone with a similar need will be able to find, and re-use that specific component.

Consequently, the general approach behind XMCDA web-services goes beyond the existence of web-services: gathering individual algorithmic components fulfills the need of capitalization in the MCDA domain. As such, there exists a single place within the Decision Deck project where these components are identified and stored.

At the time of writing, about 60 XMCDA web-services are available. Some of them are built on functions from the kappalab library by Grabisch et al. (2008) which can be used for Choquet integral-based MCDA. Others contain disaggregation techniques based on outranking relations (PyXMCDA library, by Veneziano (2010b)) and additive value functions (using the UTAR library by Leistedt (2010)). Others propose outranking
relations-based techniques (based on the J-MCDA library by Cailloux (2010) and on the digraph module by Bisdorff (2007)). Various data manipulation and visualization elements are available via the ws-RXMCDA library by Bigaret and Meyer (2010b).

For an exhaustive list of existing XMCDA web-services, please refer to http://www.decision-deck.org/ws. On this website, each of the available calculation components is documented and details are given on the requirements for the inputs.

From a practical point of view, the currently available services allow to rebuild some classical methods when properly chained. Among others, one can cite the ELECTRE series by Roy (1968) and the PROMETHEE series by Brans and Vincke (1985), as well as some UTA techniques by Jacquet-Lagrèze and Siskos (1982).

3.3 Software architecture

The general architecture supporting the XMCDA web-services is presented in Figure 1. Let us now briefly detail its main characteristics.
Asynchronous processing  Some web-services finish their processing within seconds, while others may take hours. In order to handle this, an asynchronous approach is implemented, which works as follows.

- A user submits a problem (parameters) to a web-service;
- the request is stored in the general job queue, waiting to be handled;
- the server returns a ticket number to the user. It uniquely identifies the job that was submitted, and it will be used to get the results back.
- A dedicated spooler\(^5\) manages the general queue. When a new job is submitted, it transmits it to the appropriate dedicated spooler—in fact, each web-service has its own job queue\(^6\).
- Those dedicated spoolers (one per web-service) are in charge of executing the jobs. They set up the environment required by their underlying programs, then launch the execution. This mechanism is the one which permits the execution of programs in every possible languages, by assigning a specific environment (including the required libraries e.g.) to each of them.
- When the program is finished, its results are stored in yet another area.
- The user requests the result by showing the ticket number that was returned at submission time.

The unique interface  All web-services are accessible through a unique interface, composed of three methods:

- **hello**: a method with no parameters, simply returning a string with the web-service’s author and version. This method can be used to “ping” the service, i.e. to test whether it is up and running.
- **submitProblem**: submits a new jobs. Its parameters depend on the called service; some of them may be optional (each of them is

---

\(^5\)A spooler is a program running in the background, handling a temporary storage area for requests and asynchronously processing these requests.

\(^6\)The general spooler may appear to be superfluous: jobs might be submitted directly to the dedicated queue instead. For technical and maintenance reasons, we needed a simple way to stop the handling of new problems, not stopping the execution of already submitted problems and not disrupting the submission service: just stopping the general spooler achieves this simply, new submission being paused until it is restarted.
documented in the web-services page of the Decision Deck project). It returns a string containing the ticket number attached to the job.

- requestSolution: given a ticket number, returns the results that have been produced. Here again, their number depends on the service, and they are documented just as input parameters.

It is worth noting that every input and output parameters use the XMCDA standard.

3.4 How to access the XMCDA web-services

A very convenient and user-friendly way to call the services is via the diviz software, discussed in Section 4. It is worth mentioning that it can be used not only to call, but also to chain several web-services.

However, there exist other ways of accessing them. At the time of writing, they are accessible through SOAP requests\(^7\). There exist SOAP implementations for almost all possible languages. The URLs for accessing the web-services look as follows:

http://ddeck.lgi.ecp.fr/cgi-bin/SERVICE.py,

where SERVICE is to be replaced by the name of a service. Consequently, calling the web-services is just a matter of finding the right SOAP library, submitting a job with the appropriate parameters, getting back the job’s ticket number and then polling the server regularly until a solution is available. Polling for results should be done reasonably: if the submitted job is likely to run for hours, it is no use asking every 30 seconds for a solution. As a general rule, we recommend waiting at least 20 to 30 seconds between calls, to avoid overwhelming the server with requests, and eventually be temporarily black-listed.

As an example, the Decision Deck project proposes a generic python script which allows to call any web-service. Full details are available on the page http://www.decision-deck.org/ws/accessing.html.

Please also note that alternate ways of accessing the web-services are likely to appear in the future – for example, a REST architecture may be proposed by the end of 2010. For updates, and for any additional technical details, please refer to the XMCDA web-services home page\(^8\).

\(^7\)http://www.w3.org/TR/soap12-part1/
\(^8\)http://www.decision-deck.org/ws/
3.5 Concluding remarks and perspectives for the XMCDA web-services

In this section we have shown how and why the web-services are a response to a few basic requirements of the MCDA research domain. The implementation chosen by the Decision Deck project is quite innovative and appealing, as it allows the integration of programs developed in nearly any programming language. Furthermore, the web-services solution for the MCDA algorithms allows a very easy reuse of the existing programs in other projects, even outside Decision Deck.

Any interested person is welcome to contact the authors of this chapter if he / she wishes to participate in this effort. It is also worth noting that the web-services represent a convenient way of disseminating latest algorithmic advances in MCDA.

4. diviz

The diviz software is another of the initiatives of the Decision Deck project and is developed at the LUSSI department of Télécom Bretagne in France. diviz is an open-source software which allows to design complex workflows of MCDA algorithms in a very intuitive way and to execute them. One of the key characteristics of these executions is that they rely on the XMCDA web-services presented in Section 3 on distant servers.

This section is structured as follows. We first present diviz, its properties and its latest developments in Section 4.1. We then present the architecture of diviz for a better understanding of its functionalities in Section 4.2. In Section 4.3 we detail its use on the example presented in Section 2.3, and finally, we conclude and present the perspectives of diviz in Section 4.4.

4.1 Description and properties of diviz

The diviz tool is an easy to use software to build, execute and share complex workflows of MCDA algorithms. In the literature, such workflows are often called methods (consider, e.g., the ELECTRE method by Roy (1968), the UTA method by Jacquet-Lagrèze and Siskos (1982), etc). diviz enables to conveniently combine programs implementing MCDA algorithms, which can originate from various methodological schools, researchers and programmers.

By (algorithmic) workflow we mean a sequence of connected calculation steps.
The main feature of diviz is to allow to build such MCDA methods by combining various elementary calculation components\(^{10}\). Furthermore, it can be used as a research and dissemination tool for methods and experiments. We detail these features hereafter.

**Workflow design** The design of the MCDA workflows is performed via an intuitive graphical user interface, where each algorithm is represented by a box which can be linked to data files or supplementary calculation elements by using connectors (see Figure 2 for an example). Thus, the design of complex algorithmic workflows does not require any programming skills, but only necessitates to understand the functioning of each calculation module (each of which is documented in details on the website of diviz\(^{11}\)).

The inputs and outputs of these elementary components can be manifold and are typed according to the XMCDA data types. To illustrate this, consider the following example.

**Example** diviz allows to use a component called **weightedSum**. This element calculates the weighted sum of alternatives’ performances with respect to a set of weights associated with a list of criteria. Consequently, **weightedSum** requires four inputs: the description of the criteria (at least their ids) via the tag `<criteria/>`, the description of the alternatives (at least their ids) via the tag `<alternatives/>`, the performance table containing the numerical evaluation of each alternative on each of the criteria via the tag `<performanceTable/>`, and the numerical weights associated with the criteria via the tag `<criteriaValues/>`. The main output of this component is a list of alternatives’ ids associated with the weighted sum of their performances. Note that we focus later on data typing in diviz.

To construct a new MCDA workflow, the user chooses one or more modules from a list of available calculation elements which he can drag and drop in a dedicated workspace. Then he adds data files to the workspace and connects them appropriately to the inputs of the elements. Finally he connects the inputs and outputs of the components by connectors to define the structure of the workflow.

**Execution and results** Once the design of the MCDA workflow is finished, it is possible to execute it in order to obtain the possibly mul-

\(^{10}\)Note that in the sequel we will unthinkingly use the words *component, program or module* to describe the algorithms which can be combined in diviz.

\(^{11}\)http://www.decision-deck.org/diviz
multiple outputs of the algorithms. As already mentioned, these calculations are performed on computing servers through the use of the \textsc{XMCDA} web-services presented in Section 3. As a consequence, \textit{diviz} does not physically contain any calculation modules, but requires a connection to the Internet to access these resources.

After the execution of the workflow, the outputs of each of the components can be viewed and analyzed by the user. Some of these outputs might represent results of intermediate calculation steps of the workflow. Consequently, the user is given a detailed vision of the algorithm and he can quite easily tune the parameters of the algorithms. We show these features in the following example.

\textbf{Example} Consider the following workflow (typically a UTA-like disaggregation method): a first module determines piecewise linear value functions on basis of a ranking of alternatives provided by the user; a second module transforms a performance table by applying these value functions on the performances of the alternatives; a third module calculates the sum of these performances for each of the alternatives; a fourth module draws a ranking of the alternatives on basis of the overall values previously computed. The intermediary results are the value functions, the transformed performance table and the overall values of the alternatives. As each of these elements is explicitly available for the user, first he can gain a deeper understanding of the decision aid method which he has constructed, and second the fine-tuning of the input parameters (here, the number of segments of the value functions to be constructed, the ranking provided by the user, etc.) is facilitated.

In \textit{diviz} the history of the past executions is kept in the software and can at any moment be viewed by the user. More precisely, if a workflow is modified, the former executions’ results and their associated workflows are still available. This also contributes to the good understanding of the constructed chain of algorithms and helps calibrating the parameters of the workflow’s elementary components.

\textbf{Available algorithmic components} As already mentioned earlier, the algorithmic elements are available in \textit{diviz} via the \textsc{XMCDA} web-services. At the time of writing, about 60 components are available in \textit{diviz}, which which can be divided into 4 main categories:

1 \textit{calculation components} containing aggregation operators, disaggregation techniques, post-analysis elements, etc.;

2 \textit{methods} containing full MCDA methods;
3 visualization components containing modules allowing to represent graphically certain input and output data elements;

4 reporting components containing techniques to create aggregated reports of multiple output data pieces.

Each of the available calculation components is documented on diviz’s website and details are given on the requirements for the inputs.

Currently, the members of the Decision Deck project try to focus on the implementation of classical MCDA techniques in view of publishing them as XMCDA web-services which can then be accessed from diviz. In particular, classical methods like the ELECTRE series by Roy (1968) and the PROMETHEE series by Brans and Vincke (1985) have recently been adapted. This effort should lead to increase the interest in the diviz software and the web-services offered by Decision Deck.

XMCDA: the key to interactions and data visualization The diviz tool gives the possibility to easily combine various MCDA algorithms originating from heterogeneous researchers in complex workflows. To facilitate this interoperability, diviz calls the XMCDA web-services which uses the XMCDA standard to store the inputs and outputs of the components.

In diviz this has three direct consequences:

- first, all components available in diviz can interoperate, just because XMCDA web-services do. Typically, the output of one algorithm can be injected into other elementary modules without requiring data transformations;

- second, the inputs and the outputs of the elementary components in diviz are typed with respect to the different data types defined in XMCDA. This facilitates the creation of complex combinations of components;

- third, diviz takes advantage of the powerful feature of XSLT transformations to convert XMCDA documents into HTML pages for the visualization of their contents in a web browser integrated in diviz.

These features clearly connect diviz to the previously presented initiatives of the Decision Deck project.

“Methods” comparison Next to designing and executing MCDA workflows, diviz can also be a convenient tool to compare the outputs of various methods on the same input data.
Up to recently, such a task has been far from easy, as no unified software platform for MCDA techniques existed. However, with diviz and its possibility to construct complex workflows, it is easy to connect a set of data to various workflows in a single workspace, each representing a different MCDA method, and to compare their outputs. This is clearly a very simple way to check the robustness of the output recommendation of an analysis with respect to the choice of the decision aid technique. We present this feature in further details in Section 4.3 via an application.

**Workflow sharing and dissemination** The diviz software enables to export any workflow, with or without the data, as an archive. The latter can then be shared with any other diviz user, who can then import it (by loading the archive) into his software and continue the development of the workflow or execute it on the original data.

Consequently, diviz can be used as a dissemination tool in combination with a research article. Indeed, the authors of a new MCDA technique or an experiment could propose the corresponding diviz workflow together with an appropriate data set as supplementary electronic material with their article. This would certainly contribute to a larger dissemination of new algorithms and facilitate their acceptance among many researchers.

In this context, the example which is presented in Section 4.3 is available as a downloadable archive from the diviz website, and can be tested by any interested reader.

### 4.2 A quick look at the architecture

Technically speaking, diviz is a classical 3-tier application made of: the client which has been described in this section, a component accessing the Decision Deck web-services, and a server. The diviz server’s main task consists in planning and controlling the execution of the submitted workflows, making it possible for different components in the workflow to be executed in parallel, when appropriate.

Diviz users download the client only; the latter connects to the server which takes care of distributing the computations to the dedicated web-services; it gathers all intermediary and final results which are ultimately sent back to the user.

### 4.3 Application

The data used for this application is described in Section 2.3 and is based on the works from Bouyssou et al. (2000, 2006). Recall that the underlying problem pertains to the choice of a car. Therefore, we will determine a ranking on the available cars, to aid Thierry to select the one
which seems the most appropriate in view of his preferences. In order to show the potential of diviz, we will extend this example as follows:

- In a first step, Thierry expresses some preferences on the ranking of a subset of available cars as follows (as done by Bouyssou et al. (2006)):

\[
P309-16 \succ Sunny \succ Galant \succ Escort \succ R21t;
\]

- Later, in a second step, after a discussion with an MCDA expert, he changes his mind, and is no longer confident in the ranking he provided. He rather prefers to indicate preferences on the importance and on discrimination thresholds (indifference and preference) for each of the criteria. This information is summarized in Table 2. Note that Thierry is aware that these preferences might not be compatible with the ranking that he provided earlier, but he wishes to compare both recommendations.

The goal of this application is to illustrate the features detailed in Section 4.1 from a practical point of view and to underline the usefulness of the software.

In order to determine the two recommendations and compare them, the following workflow is constructed in diviz:

- A first part of the workflow represents a variant of the UTA method (Jacquet-Lagrèze and Siskos (1982)) named ACUTA (Bous et al. (2010)) in order to determine a ranking of the alternatives on basis of Thierry’s initial ranking (dashed box 1 on Figure 2);

- A second part of the workflow represents the PROMETHEE method (Brans and Vincke (1985)) in order to determine a ranking of the alternatives on basis of the intra- and intercriteria preferences (dashed box 2 on Figure 2);

- Finally, a third part of the workflow is used to compare the outputs of the two methods (dashed box 3 on Figure 2).

<table>
<thead>
<tr>
<th></th>
<th>cost ((g1, C))</th>
<th>accel. ((g2, s))</th>
<th>pick up ((g3, s))</th>
<th>brakes ((g4))</th>
<th>road-hold ((g5))</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight (%)</td>
<td>40</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>indifference</td>
<td>500</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>preference</td>
<td>2000</td>
<td>1.5</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2. Thierry’s intra- and intercriteria preferences
Let us now present each of these parts in further details.

Part 1: ACUTA  A zoom on the ACUTA part is given on Figure 3. This sub-workflow contains 7 elementary components, including the one implementing the ACUTA method. For short, ACUTA determines piecewise linear partial value functions on basis of an input ranking of a subset of alternatives, compatible with an additive value model with piecewise linear value functions. Consequently, the ACUTA module requires as inputs the description of the alternatives and the criteria, a performance table evaluating each alternative on the criteria, a ranking of a subset of alternatives, the preference directions on the values of the criteria and the number of segments of each piecewise linear value function. One of the outputs of this component is a description of a set of compatible value functions with the input ranking.
These value functions are then used by \texttt{computeNormalisedPerformanceTable} to transform the original performance table into a version, where all the evaluations of the alternatives are in the real unit interval. On Figure 4 these value functions are plotted via the \texttt{plotValueFunctions} component. Clearly, the cost criterion \((g_1)\) has the biggest influence in his choice and Thierry has a preference for cars which cost less than 18000 EUR.

The \texttt{generalWeightedSum} module is used to calculate the aggregated value of each of the alternatives of the performance table by a simple sum. This output is then plotted in a bar chart via the \texttt{plotAlternativesValues} component (see Figure 5) and as a ranking via the \texttt{plotAlternativesValuesPreorder} module. We can easily observe that alternative \(a_{11}\) (P309-16) obtains the highest overall value.

Finally, the \texttt{rankAlternativesValues} calculation element is used to obtain the ranks of the alternatives according to their overall values.

Let us now take advantage of the flexibility of diviz in order to compare these results with the output of the second MCDA technique, namely the PROMETHEE method.

\textbf{Part 2: PROMETHEE} A zoom on the PROMETHEE part is given on Figure 6. It is made of 7 components, among which 3 are very generic and are not related to this particular method. The \texttt{PrometheePreference}
module calculates, on basis of a performance table, criteria, alternatives, discrimination thresholds and weights of criteria a preference index between all pairs of alternatives (see Table 2 for details on the input parameters). This index is then used to calculate the net flow via the PrometheeFlows component, which is used to rank the alternatives. Similarly as for the ACUTA part, the preorder of the alternatives is computed via the plotAlternativesPreorder module, a bar chart of the net flows is generated via the plotAlternativesValues element (see Figure 7), and the ranks of the alternatives are obtained through the rankAlternativesValues component.

According to the net flows obtained via the PROMETHEE method, alternative \( a_5 \) (Colt) is to be considered as the best one.

Figure 5. Bar chart representing the overall values of the alternatives

Figure 6. The PROMETHEE part of the workflow
The GAIA plane allows to see the alternatives and the criteria in a common plane, and to observe the influence of the criteria weights on the final ranking via the net flows. To draw this plane, the module `PrometheeProfiles` is used to compute a transformation of the performance table into flows decomposed for each criterion. This output is then connected to the input of the `plotGaiaPlane` module to obtain Figure 8.

The GAIA plane explains partly the ranking obtained by the PROMETHEE method, and shows that the alternatives a05 and a12 will prob-
ably remain in the first positions, unless the relative weights of $g1$ and $g3$ are dramatically lowered. We can also note that the information contained in criteria $g4$ and $g5$ is to some extent redundant for the PROMETHEE method.

Let us now proceed to the comparison of the outputs of both methods.

Part 3: Comparison A zoom on this third part is done on Figure 9. First of all Thierry is interested by the output of the PROMETHEE method on the 5 cars of his initial ranking that was used by the ACUTA method. Therefore we first start by restricting the rankings of the previous two sub-workflows to these 5 cars and compare these two new rankings.

This is done in the upper part of Figure 9. Two rankAlternativesValues components take a restricted list of those 5 alternatives as input (prototypes.xml file), as well as the two main outputs of the previous sub-workflows, to produce two new rankings. These are then compared via the alternativesValuesKendall component through Kendall’s tau (rank correlation coefficient) and a graphical comparison of both rankings via XYPlotAlternativesValues. Kendall’s tau equals 0.8, which means that for those 5 alternatives, there exists one inversion in the ranking (in this case, a09 is put before a13 in the output of PROMETHEE, instead of the inverse requirement in Thierry’s initial ranking).
Figure 10. XY plot representing both rankings (abscissa: ranks by PROMETHEE, ordinate: ranks by ACUTA)

In the lower part of Figure 9, the two output rankings of the PROMETHEE and ACUTA sub-workflows are compared via their Kendall’s tau and again the `XYPlotAlternativesValues` module. The output of the latter one is given on Figure 10. Such a shape of the XY plot means that there are quite a lot inversions between both rankings (equal rankings generate a monotonically increasing graph). Consequently, Kendall’s Tau equals 0.47, which confirms that the rankings of the complete set of cars are quite different for the two chosen methods.

The reader may wonder what solution was finally chosen by Thierry. To finish this story with a happy ending, we will suppose here that Thierry likes the fact that PROMETHEE (nearly) confirms his initial
ranking. As furthermore he is quite confident in the inter- and intracriteria preferences that he provided, he chooses to follow the recommendation given by the PROMETHEE method, and buys car a05 (Colt).

Note that the workflow described in this section can be downloaded from the website of the diviz initiative [http://www.decision-deck.org/diviz/workflow.DecisionDeckBook-BigaretMeyer.html] and can easily be imported into any diviz client\textsuperscript{12} for testing.

4.4 Concluding remarks and perspectives for diviz

In this section we have presented diviz which can be used to design and execute algorithmic MCDA workflows and to disseminate research results. The diviz software is being constantly improved, and the number of available components is quickly growing.

The very simple example detailed in Section 4.3 shows the big potential of the software for the targeted user community. Researchers can easily share and test MCDA workflows or experiments, explore robustness issues linked to the choice of the algorithm or the values of the input parameters and spread their own algorithms very conveniently. diviz is also an easy to use pedagogical tool for teachers who need to present and compare classical MCDA methods. Last but not least, diviz may also be used by consultants who wish to solve real world decision problems with a given method.

5. Conclusion and call for participation

In this chapter we have presented three initiatives of the Decision Deck project and focused on their complementary aspects. XMCDA is a data standard which allows to let algorithms and methods interact or to chain them easily, the XMCDA web-services are a convenient way to gather them in a common place and diviz is a user-friendly tool to call and combine them in complex workflows.

The work on any of these initiatives is ongoing and the reader may observe significant improvements in the future. We therefore wish to make a call for participation: if you want to publish your MCDA algorithm as a web-service and make it available in diviz, please have a look at the developers corner on the website of Decision Deck’s web-services\textsuperscript{13} and contact the authors for any help request or question. Similarly, if you think that XMCDA requires some improvements or adjustments, your

\textsuperscript{12}[http://www.decision-deck.org/diviz/tutorials.html]
\textsuperscript{13}[http://www.decision-deck.org/ws/howtos.html]
opinion is welcome, and do not hesitate to contact the authors or any other member of the Decision Deck Consortium with your desiderata and observations.

Acknowledgments

The authors wish to thank all the members of the Decision Deck Consortium for their valuable efforts to the various initiatives of the project. In particular, they wish to thank Raymond Bisdorff for the stimulating discussions on the XMCDA standard in the Specification Committee, and Thomas Veneziano for his kind availability regarding various aspects of XMCDA, the web-services and diviz.


