Outline of the talk

- Prerequisites: XMCDA;
- Decision Deck’s effort on web services;
- The web services architecture;
- A few words on GUIs;
- Requirements for programs;
- Description and documentation of your program;
- An example: a visualisation component to draw the graph of a valued relation.

XMCDA: Observations

- A standard data format did not exist before 2007 to test a same MCDA problem instance on various methods (and softwares);
- Existing MCDA methods / algorithms cannot interact.

Prerequisites

XMCDA: a very quick tour

How to parse and write XMCDA files
**XMCDA: Introduction**

- XMCDA is a particular instance of **UMCDA-ML**.

- UMCDA-ML is intended to be a universal modelling language to express MCDA concepts and generic decision aid processes.

- XMCDA focusses more particularly on MCDA concepts and data structures and is defined by an **XML schema**.

**XMCDA: Objectives**

The goals of XMCDA are to ease:

- the **interaction** of different MCDA algorithms;

- the execution of various algorithms on the **same problem** instance;

- the **visual representation** of MCDA concepts and data structures via standard tools like web browsers.

XMCDA is maintained by the specifications committee of the Decision Deck project.

**XMCDA: General conventions**

- General idea: express MCDA concepts through a few general XML structures.

- **MCDA concept**: a real or abstract construction related to the field of MCDA which needs to be stored in XMCDA; for example, **the importance of the criteria**;

- **XMCDA type**: XML structure that we created for the purpose of XMCDA; for example, **criteriaValues** to store general values related to a set of criteria.

**XMCDA: Structure outline**

Several tags under the root element **XMCDA**.

A few general categories:

- Project or file description;

- Output messages from methods (log or error messages) and input information for methods (parameters);

- Description of major MCDA concepts as attributes, criteria, alternatives, categories;

- The performance table;

- Further preferential information related to criteria, alternatives, attributes or categories.
How to read and write XMCDA

- **Either** you are very motivated and you write your own XMCDA readers and writers;
- **Or** you use an existing XMCDA parsing library.

Parse and write XMCDA

- In an *ideal world*, the publication of XMCDA 2.0.0 would have been accompanied by XMCDA parsing libraries in various programming languages;
- We do not live in such an ideal world!

  *Such parsers will appear with time, with the help of the Decision Deck programmers’ community!*
- **Today**’s situation:
  - incomplete R XMCDA library;
  - Python XMCDA library under construction;
  - ... (perhaps more to come during this workshop??)
**Decision Deck’s web-services efforts**

- Promotion of the XMCDA standard

- Re-use existing programs with minimal effort:
  → programming language independence
  
  *Nearly any command line program can be run behind the WS*
  R, java, python, C, C++, perl, ...

- Do **not** request anything more than a command-line program:
  
  - If you can write such a program, you can participate!
  - Your effort is not restricted to the web-services understructure: the same program can be distributed and used independently.

**Location(s)**

- Currently deployed at the University of Luxembourg (ernst-schroeder.uni.lu);

- Redundant duplication planned at Ecole Centrale de Paris (ddeck.lgi.ecp.fr) and at TELECOM Bretagne.

  *Would allow to use the load balancing features of diviz!*

---

**Decision Deck’s web services architecture**
**Decision Deck’s XMCDA web services**

**Properties:**
- Asynchronous
  - `submitProblem` & `requestSolution`
  - *Useful in case the calculations are time-consuming!*
- Interoperable
  - *With the help of XMCDA, the output of a WS can be reinjected into another WS.*
- Typed inputs and outputs
  - *Multiple input and output parameters for the SOAP call*
  - ⇩
  - *Multiple input and output files for your program!*

**A few words on GUIs**
The input data which is sent to the web service is either loaded from a file or entered by hand by a user;

- Separation between “file inputs” and “editable inputs”;

- You will be able to specify if your input parameter is a “file input”, an “editable input”, or both!

- Can be found in the diviz software, but is not limited to diviz!
Basic requirements for programs

1. The executable (either a binary or a script) must be available on Linux;

2. The program must understand XMCDA files;
   
   It should accept them as inputs, and be able to
   save its output(s) conforming to this standard as well!

3. The program must be callable through the command-line (at least).
   
   It should not expect any interaction at all with the
   user (all the informations needed is supposed to be
   present in the inputs supplied in the command-line).

Conforming to the XMCDA Standard

It is possible that your program does not read and/or save files in
the XMCDA format.

The alternatives:

- Adapt the program itself;
  
  For example, you will add a new option (\texttt{--xmcda})
  switching it into “XMCDA mode”.
  
  You should still be able to read and write XMCDA files
  (check whether you can use an existing library).

- Write a translator:
  
  Leave your program untouched, and write a translator
  that transforms XMCDA input files into your own format, and your own outputs back to XMCDA files.

Standardising the program (simple case)

Environment:

- Multiple input and output files (we suggest one file per main XMCDA tag);
- 2 distinct directories for the input and the output files.

Example command line:

> program -i inputDirectory -o outputDirectory

Note: Temporary files need to be written to the output directory.

Input directory:

- Contains all the input XMCDA files necessary for your program to run;
- Each file should contain at most one of the main XMCDA tags
  (to allow typing of the inputs);
- Each file can freely be named according to your parameters' names (mandatory extension: .xml).

Example:

- criteria.xml contains a set of criteria in <criteria>;
- importance.xml contains their weights in <criteriaValues>. 
Standardising the program
(simple case)

Output directory:
- Contains all the output XMCDA files produced by your program;
- We strongly recommend that your program produces at least an output file containing some messages on the execution (log, error, execution status, ...).

Example:
- messages.xml contains the log of the execution

  <methodMessages>
  <logMessage name="executionStatus">
    <text>OK</text>
  </logMessage>
  </methodMessages>

- weightsSum.xml contains the sum of the weights
  <criterionValue>.

Standardising the program:
using a standard command-line

- Real-world programs do not hard-code the filenames:

  > program −i inputDirectory −o outputDirectory

- Your program should look in the input directory (passed as a parameter to the program) for the files, load them, execute the algorithm, and write its outputs to the output directory (passed as a parameter to the program);

- The names of the input and output files should be hard-coded in your program;

- In certain situations this is enough (e.g. building a simple frontend to an existing R library)

And then?

Describe the inputs and outputs of your program and write its documentation!
**Description and documentation of your program**

- Serves to **describe** the program which you wish to integrate into Decision Deck’s web-services framework (and into the diviz software);
- Written in **XML**; syntax described by an **XML schema**;
- Specifies the inputs and outputs of your program and the editable inputs;
- Used to integrate the web-service into the diviz software,
- Used to generate a **documentation** of your program.

**Example: plotAlternativesComparisons**

- Draws the graph of a valued relation between alternatives;
- **Inputs:**
  - description of the alternatives;
  - the valued relation;
  - a cut level.
- **Outputs:**
  - messages from the execution;
  - a picture containing the graph.

- The cut level parameter should be **editable** by the user in a GUI;
- The cut level parameter should optionally also be loadable from a **file**;
- The picture produced is base64-encoded when sent in an XMCDA file
description.xml

plotAlternativesComparisons

- R script;
- Uses the graphviz library for R and the XMCDA parsing library for R;
- Command line:

```bash
> R --slave --vanilla --args "[inDirectory]" "[outDirectory]" < plotAlternativesComparisons.R
```

description.xml: General structure

```xml
<program description="DecisionDeck"
    name="plotAlternativesComparisons"
    displayName="plotAlternativesComparisons"
    version="1.0" />
<documentation>
Generates a graph representing a partial preorder on the alternatives.
</documentation>
<contact>Patrick Meyer (patrick.meyer@telecom-bretagne.eu)</contact>
</documentation>

<parameters>
<input [...]>
[...]
</input>
[...]
<output [...]>
[...]
</output>
[...]
</parameters>
</program description>
```

The cut level: cutLevel.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xmcda="http://www.decision-deck.org/2009/XMCDA-2.0.0.xml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <methodParameters>
    <parameter
        name="cutLevel"><!-- REQUIRED -->
      <value>
        <real>5</real>
      </value>
    </parameter>
    </methodParameters>
</xmcda:XML>
```

description.xml: <input>

Each of the input parameters of your program needs to be described in details:

```xml
<input id="cutLevel"
    name="cutLevel"
    displayName="Cut level"
    isoptional="1" />
<documentation>
A real value indicating above which level the valued relation (comparisons) should be considered as validated.
</documentation>
</input>
```

Note:
The name attribute of the input corresponds to the name of the XMCDA input file we are describing here!
The *xmcda* tag describes the XMCDA code which is expected for this input file of your program AND the code which is to be generated by the GUI:

```xml
<xmcda tag="methodParameters">
  <![CDATA[
    <methodParameters>
      <parameter name="cutLevel"> <!-- REQUIRED -->
        <value> <!-- REQUIRED -->
          %1
        </value>
      </parameter>
    </methodParameters>
  ]]>
</xmcda>
```

**Note:**

%1 is a placeholder for the value which is entered by the user in the GUI! Useless if this is not an editable parameter.

```
<gui status="preferGUI">
  <entry id="%1" type="float" displayName="cutLevel">
    <documentation>
      <description>A real value indicating [...]</description>
    </documentation>
    <code>
      <![CDATA[ %1 > 0 ]]>
    </code>
    <constraint>
      The value should be a positive integer.</constraint>
    <defaultValue>1</defaultValue>
  </entry>
</gui>
```

**Notes:**

- The status attribute: either preferFile, preferGUI or alwaysGUI.
- Each <entry> is related to a placeholder in the <xmcda> tag.

```
<output>
Similar syntax for the output parameters!
```

---

**plotAlternativesComparisons:**

*Summary for the cut level*

```xml
<input id="cutLevel" name="cutLevel" isoptional="i" />
```

```xml
<xmcda tag="methodParameters">
  <![CDATA[
    <methodParameters>
      <parameter name="cutLevel"> <!-- REQUIRED -->
        <value> <!-- REQUIRED -->
          %1
        </value>
      </parameter>
    </methodParameters>
  ]]>
</xmcda>
```

```xml
<gui status="preferGUI">
  <entry id="%1" type="float" displayName="cutLevel">
    <documentation>
      <description>A real value indicating [...]</description>
    </documentation>
    <code>
      <![CDATA[ %1 > 0 ]]>
    </code>
    <defaultValue>1</defaultValue>
  </entry>
</gui>
```
Tutorials for the requirements for your program and the description.xml file:

http://www.decision-deck.org/ws/howto.createAWebService.html ;

Don't hesitate to contact us:

{sebastien.bigaret; patrick.meyer}@telecom-bretagne.eu

Summary
- Adapt your program to the requirements (command line, options, XMCDA);
- Write its description;
- Send us everything with examples, installation notes, dependencies, . . .
That’s all folks

http://www.decision-deck.org/ws