Miquel Buxons Vives

Design and implementation of data aggregation modules for the European platform Diviz

Degree Final Project

Directed by Aïda Valls Mateu

Computer science degree

Universitat Rovira i Virgili

Tarragona

July 2020

Abstract

Abstract

Computer methods to give support in decision-making are a handy tool in today's world, standing out in fields such as health-care, education, financial sector or in public government institutions. This methodology, in recent years, has experienced a great growth that has extended this discipline in many fields of knowledge.

Its strengths lie in the fact that it is based on a system without prejudice; we could say that it is an objective system. By applying a set of calculations in which we can extract a traceability of the result, we obtain an assessment of a set of alternatives considering various decision criteria, previously specified by the user. These qualities, together with the fact that there is no single method to aid decision-making, makes it a very versatile and useful tool in a wide variety of cases.

This project has carried out implementations of five modules related to the OWA family (Ordered Weights Average) for the Diviz program. This family of methods falls within the decision support methods known as *"Multi-Criteria Decision Analysis (MCDA)"*; specifically, they are based on the aggregation operators. Within the five modules developed, we could distinguish two different classes, some of them carry out the task of aggregating information and evaluating alternatives using criteria. On the other hand, there are those based on the analysis and the extraction of characteristics of the parameters necessary for the first class methods.

The Diviz platform is part of a European project and is it based on open-source code developed by various collaborating institutions. In this end-of-degree project, both general design ideas and verification tests of the correct operation of the modules developed in Diviz, have been made through close collaboration with the managers of Diviz platform (researchers from the IMT Atlantique University, Brest, France).

Resum

Els mètodes informàtics d'ajuda a la presa de decisions són una eina molt útil en el món d'avui en dia, destaquen en camps com la sanitat, educació, sector financer o en òrgans de govern públics. Aquesta metodologia, en els últims anys ha experimentat un creixement accentuat que ha estès aquesta disciplina en molts altres camps de coneixement.

Els seus punts forts rauen en el fet que es basa en un sistema sense prejudicis, podríem dir que és un sistema objectiu. En el qual, aplicant un seguit de càlculs clars, que podem extreure la traçabilitat del resultat, obté una valoració d'un conjunt d'alternatives tenint en compte diversos criteris de decisió, determinats per l'usuari. Aquestes qualitats, juntament amb el fet que no existeix un sol mètode d'ajuda a la presa de decisions, fa que sigui una eina molt polivalent i útil en una gran varietat de casos.

En aquest projecte s'han realitzat implementacions de cinc mòduls informàtics relacionats amb la família OWA (Ordered Weights Average) pel programa Diviz. Aquesta família de mètodes es troba dintre dels mètodes d'ajuda a la presa de decisions coneguts com *"Multi-Criteria Decision Analysis (MCDA)"*; concretament, es basen en operadors d'agregació. Entre aquests cinc mòduls n'hi ha de dues classes diferents, uns que realitzen la tasca d'agregació d'informació i valoració de les alternatives a partir d'uns criteris. Per altra banda, estan els basats en l'anàlisi

i extracció de característiques dels paràmetres necessaris per als primers.

La plataforma Diviz forma part d'un projecte europeu i es basa en software lliure desenvolupat per diverses institucions col·laboren. En aquest treball fi de grau, tant les idees generals del disseny com les proves per verificar el seu correcte funcionament en Diviz s'han fet en estreta col·laboració amb els encarregats de la plataforma Diviz (investigadors de la universitat IMT Atlantique, Brest, França).

Resumen

Los métodos informáticos de ayuda a la toma de decisiones son una herramienta muy útil en el mundo actual, destacando en campos como la sanidad, educación, sector financiero o en órganos de gobierno públicos. Esta metodología, en los últimos años ha experimentado un gran crecimiento que ha extendido esta disciplina en muchos ámbitos de conocimiento.

Sus puntos fuertes residen en el hecho de que se basa en un sistema sin prejuicios, podríamos decir que es un sistema objetivo. En el cual, aplicando un conjunto de cálculos claros, en que podemos extraer una trazabilidad del resultado, obtenemos una valoración de un conjunto de alternativas teniendo en cuenta diversos criterios de decisión, previamente especificados por el usuario. Estas cualidades, junto al hecho que no existe un solo método de ayuda a la toma de decisiones, hace que sea una herramienta muy polivalente y útil en una gran variedad de casos.

En este proyecto se han implementado cinco módulos relacionados con la familia OWA (Ordered Weights Average) para el programa Diviz. Esta familia de métodos se sitúa dentro de los métodos de ayuda a la toma de decisiones conocidos como *"Multi-Criteria Decision Analysis (MCDA)"*; concretamente, se basan en los operadores de agregación. Dentro de los cinco módulos desarrollados podríamos distinguir dos clases diferentes, unos realizan la tarea de agregación de la información y la valoración de las alternativas utilizando unos criterios. Por otra parte, están los basados en el análisis y extracción de características de los parámetros necesarios para los primeros.

La plataforma Diviz forma parte de un proyecto europeo y se basa en código libre desarrollado por diversas instituciones que colaboran. En este proyecto de fin de grado, tanto ideas generales del diseño como las pruebas de verificación del correcto funcionamiento de los módulos desarrollados en Diviz, se han hecho mediante una estrecha colaboración con los encargados de dicha plataforma (investigadores de la universidad IMT Atlantique, Brest, Francia).

Acknowledgements

I would like to thank my tutor, Dra. Aïda Valls, first, for allowing me to work on this project related to the field that interests me. Also, I want to thank her for all her time and dedication, guidance and support.
I am also grateful for the support and guidance received during the development of my project by the managers of the Diviz framework, Dr. Sébastien Bigaret and Dr. Patrick Meyer.
Also, I wish to thank my family for their help and support, without which I would not have been able to do this project or these studies.

Contents

1 Introduction **1**
1.1 Problem statement . . . 1
1.2 Objectives . . . 3
1.3 Document structure . . . 3

2 Multi-Criteria Decision Aiding **4**
2.1 Introduction and definition . . . 4
2.2 MCDA key concepts . . . 5
2.2.1 Alternatives . . . 5
2.2.2 Criteria . . . 6
2.2.3 Decision problematic . . . 6
2.3 Important approach to MCDA for this project: MAUT model . . . 7
2.4 Decision deck and Diviz . . . 9
2.4.1 The GIS initiative . . . 10
2.4.2 Diviz . . . 11

3 XMCDA **17**
3.1 Current version and XMCDA v3.1.1 . . . 17
3.2 XMCDA Java library . . . 18
3.3 XMCDA-R library . . . 19
3.4 Basic XSD data structures . . . 20
3.5 Projects using XMCDA . . . 21
3.6 XMCDA web services . . . 21

4 Developed modules **22**
4.1 Theoretical framework behind modules . . . 23
4.1.1 OWA (Ordered weighted Average) . . . 23
4.1.2 ULOWA (Unbalanced Linguistic Ordered Weighted Average) . . . 26
4.2 Diviz framework in detail . . . 29
4.3 Framework required to develop a module in XMCDA3 . . . 32
4.3.1 General structure of a Diviz module in XMCDA3 . . . 33
4.3.2 Documentation file . . . 37
4.3.3 License file . . . 38
4.4 Data input structures changes from v2 to v3 . . . 38
4.5 XMCDA data structure used for this implementation . . . 42
4.5.1 alternatives . . . 43
4.5.2 alternativesValues . . . 44
4.5.3 criteria . . . 45
4.5.4 criteriaValues . . . 46
4.5.5 categories . . . 47

4.5.6 categoriesValues 48
4.5.7 performanceTable 49
4.5.8 programParameters 50
4.5.9 programExecutionResult 51
4.6 Design and implementation of the new modules 52
4.6.1 OWA Module 52
4.6.2 ULOWA Module 56
4.6.3 OWADescriptors Module 62
4.6.4 Deffuzification Module 65
4.6.5 Fuzzy Labels Descriptors Module 68

5 Conclusions **72**

Appendices **76**

A Description files **77**
A.1 OWA description 77
A.2 ULOWA description 81
A.3 OWADescriptors description 85
A.4 Defuzzication description 87
A.5 fuzzyLabelsDescriptors description 89

List of Tables

4.1 Changes done in input files . . . 42
4.2 XMCDA3 structures used for each parameter . . . 42
4.3 OWA module parameters . . . 52
4.4 OWA tests cases . . . 54
4.5 Performance table for this example . . . 55
4.6 Results obtained applying Orness value . . . 55
4.7 Results obtained applying weights vector . . . 56
4.8 ULOWA module parameters . . . 57
4.9 ULOWA tests cases . . . 59
4.10 Performance table for this example . . . 60
4.11 Results obtained applying Orness value . . . 60
4.12 Results obtained applying weights vector . . . 61
4.13 OWADescriptors module parameters . . . 62
4.14 OWADescriptors tests cases . . . 64
4.15 Results . . . 64
4.16 Deffuzification module parameters . . . 65
4.17 Deffuzification tests cases . . . 67
4.18 fuzzyNumbersDescriptors module parameters . . . 69
4.19 Fuzzy numbers tests cases . . . 70

List of Figures

2.1 Utility schema . . . 8
2.2 Decision Deck initiatives . . . 10
2.3 GIS initiatiave workflow . . . 11
2.4 Opened workflows menu . . . 13
2.5 Current workflow menu . . . 13
2.6 Searchable items menu . . . 14
2.7 Diviz workflow execution result . . . 15
2.8 Module error state . . . 15
2.9 Error message . . . 15
2.10 Module correct state . . . 16
2.11 Diviz execution results examples . . . 16

3.1 XMCDA named policy . . . 18
3.2 XMCDA structured item . . . 19
3.3 Example of values method call . . . 19
3.4 R code instruction example . . . 19
3.5 xsd document - XMCDA structure example . . . 20
3.6 XMCDA website - XMCDA structure example . . . 20

4.1 Fuzzy number trapezoid . . . 26
4.2 Diviz creation workflow menus . . . 29
4.3 Blank canvas . . . 30
4.4 Module graphical connections . . . 31
4.5 Options menu of Diviz standard module . . . 31
4.6 Graphical representation of a fuzzy Numbers example . . . 32
4.7 numerical representation of a criterion example . . . 32
4.8 Necessary orders to install Linux XMCDA . . . 33
4.9 Example of the call to the load data function . . . 34
4.10 Example of the call to the input checker function . . . 34
4.11 Example of the call to the output preparation function . . . 34
4.12 Example of the call to the write data functions . . . 35
4.13 Sequence diagram of standard Diviz module functionality . . . 35
4.14 Example of test folder structure . . . 36
4.15 Script used by execute locally all test cases . . . 37
4.16 MIT License file example . . . 38
4.17 Current header for files that run in XMCDA3.1.1. structure . . . 39
4.18 Illustration of a minor change in parameter files . . . 39
4.19 Weights file current structure . . . 40
4.20 Orness file current structure . . . 40
4.21 Fuzzy numbers file old structure . . . 41

4.22 Fuzzy numbers file current structure . . . 41
4.23 XMCDA3 alternatives structure . . . 43
4.24 XMCDA3 alternativesValues structure . . . 44
4.25 XMCDA3 criteria structure . . . 45
4.26 XMCDA3 criteriaValues structure . . . 46
4.27 XMCDA3 categories structure . . . 47
4.28 XMCDA3 categoriesValues structure . . . 48
4.29 XMCDA3 performanceTable structure . . . 49
4.30 XMCDA3 programParameters structure . . . 50
4.31 XMCDA3 programExecutionResult structure . . . 51
4.32 OWA state-machine diagram . . . 53
4.33 Graphical results . . . 55
4.34 Graphical results . . . 56
4.35 ULOWA state-machine diagram . . . 57
4.36 Fuzzy numbers for the example . . . 60
4.37 Graphical results . . . 61
4.38 Graphical results . . . 61
4.39 OWADescriptors state-machine diagram . . . 63
4.40 Deffuzification state-machine diagram . . . 66
4.41 Fuzzy numbers for this example . . . 67
4.42 Alternatives for this example . . . 68
4.43 Results for this example . . . 68
4.44 fuzzyNumbersDescriptors state-machine diagram . . . 69
4.45 Fuzzy numbers for this example . . . 71
4.46 Results for this example . . . 71

Chapter 1

Introduction

1.1 Problem statement

Today computers have a large computational power and tons of data is generated daily in many different fields. Consequently, it is widely recognized the need of developing systems that help humans by means of automatic and intelligent analysis tools. For example, authorities must make a proper and extensive analysis of different options, terrain characteristics or environmental impact, when a tunnel for a road has to be constructed; otherwise, a poor data analysis can produce a faulty construction and produce a lot of victims directly or indirectly.

In this type of cases, the decision of which alternative to choose cannot be a subjective decision made by intuition; on the contrary, it must be made following a technical methodology that satisfies a set of mathematical and behavioural properties and guarantees that all the are properly evaluated. With these premises, Multi-Criteria Decision Aiding (MCDA) was born as a sub-discipline of Operations Research that evaluates different criteria for decision-making. It uses methods that come from the fields of Statistics and Economy but also includes methods from Artificial Intelligence, such as the management of uncertainty, use of symbolic calculation methods, logics, data analysis or machine learning tools.

In the previous example, we showed a decision related to public work. However, MCDA is very useful in many other fields, such as engineering, business, environmental assessment, or even pharmacology or health care, such as in the current global situation for COVID 19.

We have to imagine MCDA as discipline that establishes a general framework (i.e. conditions and properties) for systems that perform decision aiding. Under this general model, many different methods have been defined, each one is useful in different situations, depending on the needs and the type of input data we have. It is usual to apply more than one method with the same input data and later make a comparison among results.

As recognized in the literature Figueira et al. [2005], there are three main approaches to solve decision-making problems:

1. Functional Model: based on utility theory. This approach is based on, a construction of a utility function for each criterion, the scoring of the alternatives on the basis of the utility functions and the aggregation of the scores to find an overall utility degree for each alternative.

2. Relational Model: based on outranking relations. In this type of model, we compare the

performance by pairs of alternatives only one single criterion at the same time. A binary preference relation is built for each criterion, and then they are merged to find the overall preference structure among all the alternatives.

3. Logic Model: based on rules. This approach defines a set of decision rules that model the reasoning process of the decision-maker. These rules are applied to each alternative, to solve the problem.

Some of these methods have been commercialized by creating new companies, which offer the assistance to the decision-making using his own software or in the other hand, other companies commercialize their methods implementation software but with a high price. Within this context, Decision deck has been created, as an Open Source project that provides a shared implementation of these methods using a modular and interconnected components scheme. This European project has a well-represented structure with main functions distributed in four departments. In front of the administrative board, it has the *Decision Deck Consortium*, a French non-profit association that was officially born on November, 2008. In front of the current active initiatives and research groups coordination, it has the *Software Resources Management Groups*, initiatives as Diviz (a workbench to design, create and share complex MCDA algorithms). The *Specifications committee* is in charge of the development and maintenance of XMCDA (is a standardized XML recommendation to represent objects and data from MCDA) Decision deck project comprises different tools, but we will focus on Diviz and XMCDA. Finally, the management of official web-sites and all communicational aspects is made by the *Communication Dissemination Committee*. Consortium et al. [2009]

The URV's ITAKA research group has been collaborating with the Decision Deck project since 2010, when they began to develop modules for Diviz. They implemented MCDA methods based on the theory of utility (MAUT), corresponding to the techniques that ITAKA had been using in other projects. In particular, they work is related to techniques for handling linguistic data and uncertainty using Fuzzy Sets and aggregation operators. These operators allow more or less strict policies to be defined when merging values using ordinal weight vectors. The modules implemented worked with XMCDA1 language version.

In 2019, ITAKA group was contacted because the XMCDA language was being redefined and their collaboration in the new version war required. In the new version that Diviz has entered with XMCDA3 version, the previous implementation that ITAKA group had developed is not functional A redefinition of its input/output formats was required. In addition, looking at the period of these last ten years, during the different updates of Diviz, the programmers involved have moved in the direction towards modules using the R language which provides a more optimal environment for the development of these modules. These two issues motivated the definition of this Bachelor Thesis, with the aim of having new implementation of the ITAKA modules and the new design of the part of the language that deals with Fuzzy Sets and the ordinal weights.

As explained, the main objective of this project came at request of the Decision Deck project. From the beginning, communication with the project managers was very fluid, both on my part as the developer of the program and with the head of the ITAKA group, Dr. Aïda Valls. This constant communication ensures that the work presented in this Thesis fits with the new Diviz version and it will be integrated in the new platform when launched. Moreover, this communication allowed that all the doubts were solved quickly and improved the coherence substantially with the new version.

1.2 Objectives

The objectives that this project want to achieve are:

- Study the methods developed at ITAKA in version v1. Identify the key components and the input data that is specific to these methods.
- Study the structures of XMCDA3 available in this new version. Identify the changes in the language structure from version v2 to v3.
- Propose a new data structure specific of ITAKA modules (fuzzy and ordered weights) according to the XMCDA3 guidelines.
- Implement the modules compatible with XMCDA3 in R language.
- Generate a user guide for the new implementation and requisites.
- Develop a new functional module part for Diviz.

1.3 Document structure

The chosen structure for this project is as follows. In Chapter 2, we will find a first approximation to MCDA and MAUT theory and an introduction to Decision Deck and its functional tools, for example Diviz. Chapter 3 is a complete description of XMCDA language looking for the basic structure and capacities. It also contains, a comparison of the main differences between XMCDA version 2 and version 3.

In Chapter 4, the new modules are explained in detail. They are: OWA, ULOWA, Deffuzification, weights and fuzzy labels descriptors. With the explanation all design decisions, structure of the code, implementation and testing done during their development. otherwise, all modules context as their licence, documentation file and input-output parameters current structure are explained too.

Finally, Chapter 5 is the section devoted to the personal assessment of all I have learned during the development of the project, the conclusions of the work done and the definition of future lines of work.

Chapter 2

Multi-Criteria Decision Aiding

In this chapter, we explain the origin of the Multiple-Criteria Decision Aiding (MCDA) field. We define the concepts used in this work and make an introduction to one of the main approached in MCDA: the multi-attribute utility theory. The tools designed and implemented in this work follow this model of MCDA.

2.1 Introduction and definition

Multiple-Criteria Decision Analysis is a field of research that designs and develops decision support methods to evaluate multiple criteria on a set of alternatives with the purpose of find the best solution as part of the decision-making process. This field was born in 1960s, but we can say that the date for the MCDA popularity birth is 1979 with the publication of Stanley Zionts article "MCDM- If not a Roman Numeral, then What?", an article that increases the popularity of this term in entrepreneurial spheres.
MCDA try to structure and solve a decision and planning problems involving multiple criteria, supporting decision-makers in their jobs and analysis. Hardly ever exist a unique optimal solution, and it is necessary to rank the different alternatives exposing the advantages and disadvantages.

MCDA has some strengths compared to other decision-making tools from Artificial Intelligence. This discipline is open and explicit, and we always know all the steps that the program followed to find a solution. The methods make a systematic analysis of all the criteria for all criteria in terms of the decision maker's preferences, so they provide a different solution if the preferences change. Finally, it is a discipline very flexible, because the analyst can readjust the parameters that determine the characteristics of the analysis whenever he or she wants. Belton and Stewart [2002]

Nevertheless, if we ask ourselves "why is better a multi-criterion than a mono-criteria decision?", we can fall in the wrong point of view that if we try to achieve an objective or a purpose of analysing each criterion separated from the other with its own scale of importance punctuation it could be a better option, but we can wrongly neglect certain aspects of realism, facilitate the setting up of equivalences, the fictitious nature of which remains invisible, tend to present features of one particular value system as objective. However, the multi-criteria approach contributes to delimitate a broad point of view likely to structure the decision process concerning to the actors involved with the original meaning of the corresponding evaluations.

Since this discipline was popularized by Stanley Zionts in the decade of 1970, the scientific, economic and, in general, all the professional community found more and more uses have been found of MCDA. Currently, we can find applications of this discipline in situations as different Figueira et al. [2005]:

- Environmental Management and Energy planning:
  - Forest management.
  - Water resources planning.
  - Energy intensity evaluation.
- Finance and economics:
  - Business failure prediction.
  - Company mergers and acquisitions.
  - Country risk evaluation.
- Transport:
  - Highway planning.
  - Subway design.
- Marketing:
  - Customer satisfaction.
  - Design of market penetration strategies.
  - Retail evaluation.

2.2 MCDA key concepts

In this section, we formally define the main elements and concepts of MCDA models, which will be needed in the rest of the documentation. Figueira et al. [2005]

2.2.1 Alternatives

Could be defined as potential action or the action that is used to designate the element which constitutes the object of the decision, or that which decision aiding is directed towards. This definition as "action" does not incorporate any notion of feasibility. We understand this concept of action as a potential solution that deserves some interest within the decision aiding process. The concept of alternative corresponds to the particular case in which modelling is such that two distinct action cannot be defined with the same characteristics. Many authors suppose that potential actions are, by definition, mutually exclusive. Nevertheless, such a hypothesis is in no way compulsory.

In all cases, A will denote the set of potential actions considered at a given stage of the decision analysis process. This set is not necessarily stable; i.e., it can evolve throughout the decision aiding process.

By a, we will denote any potential action or alternative. When the number of actions is finite ($|A| = m$), we shall let:

$$A = \{a_1, a_2, \ldots, a_m\}$$

When modelling of actions can be done by referring to some variables $x_1, x_2, \ldots,$ it is possible to write:

$$a = \{x_1, x_2, \ldots\}$$

In another type of modelling, the value of each variable $x_i i = \{1, 2, \ldots, n\}$ designates a possible score on an appropriate scale Xi built for evaluating actions according to a specified point of view or criterion. In such cases, A can be viewed as a subset of the Cartesian product $X = \sum_{i=1}^{n} X_i$. This type of modelling is commonly used in multi-attribute utility theory (MAUT).

2.2.2 Criteria

A criterion is a tool constructed for evaluating and comparing potential actions according to the point of view which must be well-defined. For each action a, inside the decision making, must be evaluated regarding to multiple criteria.

The set of criteria is denoted as $C = \{c_1, c_2, \ldots, c_p\}$, which are the different aspects that will be analysed by the decision-making.

For each criterion $c_i(a_j)$ is called the performance of a according to this criterion.

It is necessary to define explicitly scale of the performance scores. When we compare two actions according to a criterion, we compare the two degrees used for evaluating their respective performances. We can distinguish various types of scales of criterion:

- Ordinal scale: scale where the different values inside indicate positions, but anything ensures us that any pair of values inside the scale have the same difference between them.
- Quantitative scale: performance scale whose degrees are defined by referring to a clear, concrete defined quantity in a way that it gives meaning, on the one hand, to the absence of quantity (zero or neutral value), and on the other hand, to the existence of a unit allowing us to interpret each degree as the addition of a given number of such units. Quantitative scales can use numerical ranges but also linguistic scales (i.e. sets of terms such as perfect, good, poor, etc.).
- Another type of scale: In MCDA, we can work with other types, for example interval scales.

2.2.3 Decision problematic

In MCDA, the word problematic refers to the way in which Decision Aiding is envisaged. This means that the problematic deals with answers to questions such as the following: in what terms should we pose the problem? What type of results should we try to obtain? How does the analyst see himself fitting into the decision process to aid in arriving at these results? What kind of procedure seems the most appropriate for guiding his/her investigation? Once this questions have been answered, the decision problematic is focused to a more practical issue, which is

related to the type of goal or solution we want to obtain. This is what is more interesting for the work of this thesis, because we will not deal about the general modelling of the decision problem, but on the use of a particular MCDA technique to solve it. In that sense, the algorithm to use may be linked to the following 3 types of problematics:

- The *choice* problematic: The aid is oriented towards and lies on a selection of a small number (as small as possible) of "good" actions in such a way that a single alternative may finally be chosen. This does not mean that the selection is necessarily oriented towards the determination of one or all the actions of A, which can be regarded as optimum. The selection procedure is based on comparisons between actions so as to justify the elimination of the greatest number of them, the subset of those actions which are selected (which can be viewed as a first choice) containing all the most satisfying actions, which remain non-comparable between one another.

- The *sorting* problematic: The aid is oriented towards and lies on an assignment of each action to one category (judged the most appropriate) among those of a family of predefined categories, which are usually ordered according to some preference consideration. This family of categories must be conceived on the basis of the diverse types of treatments that motivate the sorting. For instance, a family of four categories can be based on a comprehensive appreciation leading to distinguishing among: actions for which implementation (i) is fully justified, (ii) could be advised after only minor modifications, (iii) can only be advised after major modifications, (iv) is unadvisable.

- The *ranking* problematic: The aid is oriented towards and lies on a complete or partial preorder of alternatives in A which can be regarded as an appropriate instrument for comparing actions between one another; this preorder is the result of a classifying procedure allowing us to put together in classes actions which can be judged as indifferent, and to rank these classes (some of them may remain non-comparable).

2.3 Important approach to MCDA for this project: MAUT model

Within MCDA community, this approach is named English model. This method was introduced by John Von Neumann and Morgenstern for modeming the behaviour of economic agents in 1947, starts to be used in decision making in the 1960 decade and officially recognised in 1970 by the hand of P.C. Fishburn with "Utility theory for decision-making". It is based on the Utilitarianism philosophy. Keeney and Raiffa [1993]

The decision-maker wants to measure the global utility or global preference score of each alternative. This approach is argued to be the usual human mental mode to assess the different alternatives in decision-making. The steps of MAUT methods are (Figure 2.1):

Figure 2.1: Utility schema

1. Assign a utility score to each alternative for each criterion. This is made by defining a different utility function for each of the input variables x_i.

2. Merge the utility scores by using some aggregation operator, obtaining a final overall global utility score for each alternative.

3. The comparison of those global utility scores is used to solve the selection, ranking or classification decision problem.

4. Therefore, the concept of *Utility* is key in this approach. Gate [2015] It can be formalized as he degree of satisfaction for an alternative regarding some property, expressed as $c_i(a_j) = u_i j$.

there are many types of aggregation operations, one of the simplest is the weighted average, which is based on the weights element. This property determines the impact of a criterion on the overall evaluation, this property values are bounded between two numbers (usually between 0 and 1) and must be invariant for all the decision-making process. Principal properties of weights are:

- The multiplicity of weights has to be the same as criteria,

$$W = \{w_1, w_2, \ldots, w_p\}$$

- The result of the aggregation of all w_i must be equal 1

$$C(a_j) = \sum_{i=1}^{p} w_i u_{ij}$$

The comparison of those ratings is used to solve the selection, ranking or classification decision problem.

2.4 Decision deck and Diviz

The literature in the field of Multi-Criteria Decision Aiding (MCDA) proposes algorithms and methods that are acknowledged and used in real-world decision aiding processes. These techniques provide useful solutions to complex decision problems which involve several, often conflicting evaluation criteria. Several such methods and algorithms have been implemented and are available as separate software solutions. However, as they were developed in an uncoordinated way, they lack fundamental interoperability features.

The Decision Deck project, therefore, aims at collaboratively developing Open Source software tools implementing Multi-Criteria Decision Aiding techniques which are meant to support complex decision aiding processes. One of the main features of these software solutions is that they are interoperable in order to create a coherent ecosystem.

The Decision purpose is to provide effective tools for three types of users:

- practitioners who use MCDA tools to support actual decision-makers involved in real-world decision problems.
- 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 Decision Deck project works on developing multiple software resources that are able to interact. Consequently, several complementary efforts focusing on different aspects contribute to Decision Decks various goals. Consortium [2009a]

The project continues and expands the series of activities that have been pursued by the Decision Deck Community:

- **XMCDA:** a standardised XML recommendation to represent objects and data structures issued from the field of MCDA. Its main objective is to allow different MCDA algorithms to interact and be easily callable.
- **XMCDA web services:** distributed open-source computational MCDA resources.
- **DIVIZ:** an open source Java client and server for XMCDA web services composition, workflow management and deployment.
- **GIS (Geographical Information Systems):** The GIS initiative of Decision Deck focuses on tools that help with the integration of capturing, storing, checking, and displaying data related to positions on Earth's surface and Multi-Criteria Decision Aiding. This initiative is currently an expanding process.
- **R initiative:** proposes to use R statistical environment, coupled with the MCDA package, to support all the steps of the MCDA process.
- **D2:** a rich open source Java client, offering several MCDA methods. Currently in a frozen state.
- **D3:** an open source rich internet application for XMCDA web services management. Currently in a frozen state.

Figure 2.2: Decision Deck initiatives

In order to coordinate the various activities of the Decision Deck project, it is structured as follows Consortium et al. [2009]:

- **The Decision Deck Consortium:** French non-profit association, steers and manages the project. It is headed by an Administration Board. It was officially born on November 25, 2008, as it appears the official declaration published in the French Journal Officielle de la République Française.
- **The Software Resources Management Groups:** they are in charge of the organisation and the management of the developments of the six identified initiatives of the Decision Deck project. Each group is coordinated by a clearly identified contact person. These management groups are in charge of organising the bi-annual developer's days of the project.
- **The Specifications Committee:** under the direction of a coordinator, its role is to maintain and develop the XMCDA standard and to approve and publish suggested evolutions. The coordinator is in charge of organising the specifications meetings.
- **The Communication and Dissemination Committee:** under the direction of a coordinator, its role is to develop and maintain the websites of Decision Deck and to manage the communicational aspects of the project.

2.4.1 The GIS initiative

One definition that is very accurate comes from national geographic *"A geographic information system (GIS) is a computer system for capturing, storing, checking and displaying data related to positions on Earth's surface. By relating seemingly unrelated data, GIS can help individuals and organisations better understand spatial patterns and relationships."* Society [2017] Nowadays, the number of different types of data that can be compared or analysed are enormous, but this amount of data increase at the same time of new techniques to process those data appear. We can discompose GIS technology in different main parts:

- **Data capture:** choose which data we will want to collect and how. This last question it may seem trivial but is an important part to know which kind of data we will need to process, cartographic data or photograph that must be interpreted and sensors collected data or digital data. Other important steps are the definition of the scale and spatial relationships.

- **Generation of GIS maps:** all data collected previously can be processed and combined to produce a wide variety of individual maps where user can view the different parameters such as density, terrain composition, the possibility of damage in a flood of a certain map part, etc.

- **Uses of GIS:** GIS technology can be used for scientific investigations, resource management and development planning Marketing companies use GIS to decide where their publicity should be, governments can use it to plan their response to a hurricane or other natural disaster or even engineers can use GIS to support their design, implementation and management of communication networks.

The implemented plug-in provides an easy-to-use interface that generates a virtual environment to model and solves decision problems for the evaluation of geographical areas taking into account multiple criteria and actorsOlteanu et al. [2014]. An example of this workflow are represented in (Figure 2.3).

It has been developed for the purpose of AMSIG project involving Telecom Bretagne and the Service Hydrographique et Océanographe de la Marine (SHOM).

Figure 2.3: GIS initiatiave workflow

2.4.2 Diviz

Introduction and properties

Diviz is a tool for MCDA components workflow design, execution and deployment based on modules, Diviz allow users to combine these modules to create more complex MCDA workflows or methods. The data exchanged between algorithmic components is conformed the XMCDA standard, created using XLS (eXtensible Stylesheet Language) combined with standard CSS (Cascading Style Sheets).

Diviz is created as an Open Source platform-independent software program, that is because this software was written in Java and is therefore independent of the operating system you are using. Diviz is distributed as an open-source software, and even, some of elementary MCDA modules are also available open-source software bricks.

As the leading website about this software indicates Consortium [2009b], there are important properties, or characteristics, and objectives that define Diviz software. For example, we can define the main properties for Diviz as:

- Usage of XMCDA to make elementary components interoperable.
- Usage of XLS and XMCDA for a standardised visualisation of input and output data.
- The history of all the past executions is available (ideal for parameter tuning).
- All available MCDA components in Diviz are open source web services (XMCDA web services).

On the other hand, the principal objectives that characterised this software can be defined as:

- Software created to help researchers to construct algorithmic MCDA workflows from elementary MCDA components.
- Software created to help teachers to present MCDA methods and let the students experiment their own creations.
- Software created to help to compare results of different methods easily.
- Software created to allow to add new elementary MCDA components easily.
- Software created to avoid heavy calculations on your local computer by executing the methods on distant servers.

To ease the dissemination of new MCDA algorithms, methods and experiments. Every workflow can be created by the graphical union of different methods and every xml file (using XMCDA standard) necessary as input data for these methods, executed using internet connection via web services (each component is in fact proposed as a web service). After this execution output files are generated and are completely available to get it. Finally, every workflow created can be exported as a **".dvz"** document type and can be shared as any other file without a doubt.

Program and workflow design

Diviz allows you to create, save and distribute your own workflows but to implement for each one of these characteristics, there is necessary an intuitive interface to work with all elements that offer this program. This interface is distributed in four main parts:

- **Open workflows:** part reserved to workflows that have been created previously or imported one. For each workflow, there are two main elements, the main editable workflow, where we have to select it when we want to edit some part of our project, always in black, and below it all the executions that we have done with this project, for each execution we can see the execution date and hour that we have done it. Finally, every execution is printed in green if the execution is successful and with red if the execution failed even if it is a local fail in only one module.

Figure 2.4: Opened workflows menu

- **Central part:** as is represented in (Figure 2.5) is where users can edit their workflows introducing dragging MCDA modules on it and joining them by their outputs and inputs, besides that creators can drag a special module that permits load a *csv* and *xml* files to program and use it as input. When we are selecting an execution in "open workflows", the program allows us to see the execution process and results.

Figure 2.5: Current workflow menu

- **Search part:** In this box, the user can select and drag to central part the MCDA algorithms modules to use them into his/her workflow. The modules are distributed in calculation components (value techniques, outranking techniques, Data envelopment) and Tools (alternatives and criteria computations, performance table computations, CSV converters, Data visualisation and Reporting).

Figure 2.6: Searchable items menu

- **Toolbar:** as almost all the software programs Diviz has a toolbar that allow the creation, store and execution of any ".dvz" project, instead of that you can export and import any MCDA workflow of other user and execute them.

Workflow execution and intermediate file generation

A workflow that can be executed has different and recognizable parts that often appears in all of them. Firstly, what we have to do is to drag the modules to the workflow central panel, after that, join these modules with themselves, joining outputs of first step modules with inputs of next steps and, of course, join these first modules with inputs files, files that meet the requirements of XMCDA structures. All these unions are done graphically with coloured arrows. As we can see below in (Figure 2.7):

Figure 2.7: Diviz workflow execution result

After running the workflow, we will have two options to follow the process. If any module fails for any reason, this module will be painted in red (Figure 2.8) and it will not generate an XMCDA structure in a file as an output. Instead of that, the message pin shows us the message with the error that appears.

Figure 2.8: Module error state

Method messages

ERROR : The document is not a XMCDA@http://www.decision-deck.org/2009/XMCDA-2.1.0: document element namespace mismatch expected "http://www.decision-deck.org/2009/XMCDA-2.1.0" got "http://www.decision-deck.org/2012/XMCDA-2.2.0"

Figure 2.9: Error message

If the execution of this module is successful, the module will be painted in green (Figure 2.10) and the output XMCDA structure will be generated in an xml file and the message pin of this module show as the *"ok"* message. This output structure can be a table or a graphical structure.

Figure 2.10: Module correct state

The input files and output files generate an internal xml that can be viewed and Diviz recognize the structure and show us with more readable form, or if it is graphical structure, Diviz will show us a graphic. As we can see in following output examples (Figure 2.11).

Performance table

	c01	c02	c03	c04	c05
a01	VL	VL	P	H	VL
a02	VL	VL	H	P	P
a03	VL	VL	L	M	L
a04	VH	L	H	H	AH
a05	P	L	H	L	AH

Figure 2.11: Diviz execution results examples

Chapter 3

XMCDA

In this chapter, the language used in Diviz is explained. We present the different features and explain the changes on the versions, which motivated this work.

XMCDA is a data standard which allows representing Multi-Criteria Decisions Analysis (MCDA) data elements in XML according to a clearly defined grammar. XMCDA is an instance of UMCDA-ML (Universal Multi-Criteria Decisions Analysis Modelling Language) and one of the scientific projects of Decision Deck. XMCDA is maintained by Decision Deck project and is supported by the COST Action IC0602 Algorithmic Decision Theory.

XMCDA is structured XML. This kind of structure is very useful in this case because it allows to easily define some mark-up elements. In this aspect, XML let share structures, encode documents and serialise data. With this structure type, the main XML schema of XMCDA can be used to validate an instance of XMCDA in many XML editing tools or even we can write with correct grammar a new instance.

Another characteristic that XML structure gives to XMCDA is that it is very easy to be visualised in any web browser as HTML. Also, there is some Decision Deck project like Diviz that let create MCDA very complex algorithms using the concatenation of more than one MCDA with a very simplified structure based on visual relation between MCDA modules.

XMCDA data standard it is not static, can evolve on order, the creators encourage creators to make demands of new data structures to Decision Deck specification committee who will analyse the demand.

3.1 Current version and XMCDA v3.1.1

Nowadays, the last version for users is 2.2.3, however soon this version will be abandoned and substituted by version 3.1.1. This new version makes substantial changes to the structure and produce the incompatibility of previous version data structures. In fact, this change is the reason that URV Diviz modules need a new implementation, because with this new version, these modules can be executed properly.

Before introducing the most remarkable changes among versions, we will see some patrons in version changes. The naming policy in XMCDA, is based in this structure representation (Figure 3.1):

XMCDA a.b.c

Figure 3.1: XMCDA named policy

a, b, c are integers which are increased in case of a new release, according to the following rules.

- Change to a.b.(c+1): for minor modifications on the standard, e.g., the adding of a new sub-tag in an XMCDA type.
- Change to a.(b+1).0 for more substantial modifications on the standard, e.g., the adding of a new tag (or XMCDA type) under the root element.
- Change to (a+1).0.0 for fundamental modifications on the standard which do not allow full compatibility to earlier versions, e.g., the renaming of a fundamental XMCDA type.

Any modification can only be performed on the standard after the approval on the specifications' committee.

Currently, XMCDA version 3.1.1. is available as XML schema, but it is not available yet because a big change is needed. Nevertheless, creators have created a little roadmap to expose in which step we are currently:

1. Propose a reference implementation of XMCA v3.x:
   (a) The new implementation should be available to Java, R and Python.
   (b) The new implementation should be able to translate from one version to another, using a principle of the maximum effort.
2. Develop, experiment and validate a program wrapper, so the current web-services are available to accept and produce XMCDA v3.1.1 files
3. Diviz will switch to XMCDA3.
4. XMCDA v2 will officially be abandoned and XMCDA v3 will be fully in charge of conveying out MCDA data.

Currently XMCDA is in the step 1.b.

3.2 XMCDA Java library

The reference library to manipulate XMCDA objects, accepting the XMCDA v2.2.1 and XMCDA v3 schema.

In this library, there are all current data structures classes with their methods to add or subtract or manipulate information inside this structure type. The data structures are originally created by XML structure in file XSD. Bigaret [2019a]

One example of this Java class can be the *Categories-Values* structure class, where we can find the basic treatment methods for the structure (Figure 3.2).

Figure 3.2: XMCDA structured item

These methods are completely accessible in any language. For example, in my implementation, I can call *Categories-Values* method named *values()* in my R code without any error just with the following instruction (Figure 3.3).

```
categoriesValues$values()
```

Figure 3.3: Example of values method call

3.3 XMCDA-R library

This library contains functions to read XMCDA3 tags and transform them into R variables which are used to build MCDA algorithms written in R, also allows writing some R variables into XML files respecting the XMCDA3 standard.

This implementation has some prerequisites to be used. For example, when you want to install this library, we need to install with the java XMCDA Java .jar to work.

When you see the internal code of this library, you understand the reason, and it is that this library adds some methods extra layer to adapt most used structures of java XMCDA to some R datatypes, but the main structures and methods are always the same as Java.

```
performanceTableList <- getNumericPerformanceTableList(xmcdaData)
```

Figure 3.4: R code instruction example

In this statement (Figure 3.4), we can see that the method that is called returns an R datatype structure, but always we need to give it the pointer to our XMCDA structure. Bigaret [2017]

3.4 Basic XSD data structures

As I said previously, the XMCDA library is based on XML structure that gives much more adaptability to data types. These basic structures are saved in an xsd file.

```
<xs:complexType name="categoriesValues">
        <xs:annotation>
                <xs:documentation>A sequence of values related to categories.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
                <xs:element name="description" type="xmcda:description" minOccurs="0"/>
                <xs:element name="categoryValue" type="xmcda:categoryValue" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="xmcda:defaultAttributes"/>
</xs:complexType>
```

Figure 3.5: xsd document - XMCDA structure example

If as a developer, you want to search some type of information you can download the xsd file in the XMCDA web Consortium [2019] or also you can consult a complete datatypes information in the html documentation Bigaret [2019b].

Figure 3.6: XMCDA website - XMCDA structure example

These two images (Figure 3.5 and Figure 3.6) are based on the same data structure on the two sites that are available. For each structure, you can find a detailed diagram with the main structure and internal data types, or subtypes. You can navigate into this diagram open different subtypes since you arrive in more basic layers based on primitive types (integer, double, char, string, etc.). After that you have some explanations about the properties and attributes of this structure.

3.5 Projects using XMCDA

Currently, there are active projects that are using this library, but if you want to use it is an entirely Open Source code, creators, Dr. Patrick Mayer and Dr. Sebastien Bigaret put an extensive user guide for beginners of this library. Some examples of projects using XMCDA are:

- Diviz software.
- XMCDA web-services.
- Kappalab R library through the XMCDA-R library.
- Python diagraph module for Rubis.
- The J-MCDA project.

3.6 XMCDA web services

We understand web services as a collection of protocols and standards used by the intercommunication of different applications, programmed with the same or different code language. This type of protocols is advantageous, for example, in the current days on the Internet because it allows a complete intercommunication between any application in the *World Wide Web*, but it can be useful in any other communication net. Saffirio [2006]

The most popular and used web services protocols are:

- HTTP
- SOAP
- WSDL
- WS-Security

XMCDA web services are a distributed open-source computational MCDA resources, using the XMCDA standard. Currently, there are two different ways to access XMCDA web services:

- Standard SOAP request.
- Diviz software.

The last one allows you to build execution XMCDA web services workflows via the Graphical User Interface.

The SOAP (Simple Object Access Protocol) requests allow you to integrate the web service call into your application directly. A common definition of SOAP can help us to understand easily what is it *"SOAP communication protocol define how two objects in different procedures can communicate between the XML message exchange"* Blancarte [2017]. SOAP accept different typology of messages, but the most used are the RCP (Remote Procedure Call).

Chapter 4

Developed modules

In this chapter, I will do a complete documentation of the 5 implemented modules. First the theoretical definition of the operations done in the modules will be given. Next, second section is devoted to the general framework of Diviz and all the data structures and requirements to follow to implement a module in this platform. In this section, I explain the first contribution of the work, which is the definition of the format of 3 data structures in XMCDA for version v3. Finally, the third section details the main contributions of the work, that is, the design, implementation, examples, and test cases of each of these 5 modules:

- OWA operator
- OWA weights descriptors
- ULOWA operator
- Defuzzification
- Fuzzy labels descriptors

The source code, the test cases and the documentation generated for each module, are all uploaded on my personal Github. Buxons [2020] The design and development has been made according the requirements of the ITAKA research group, under the revision of Dr. Aïda Valls. However, it also has to be claimed the role of Sébastien Bigaret, researcher and development engineer in IMT Atlantique Bretagne Pays de la Loire, who has helped to resolve doubts and upload all the code to the web service used by Diviz, thus establishing with me a very close collaboration

In 2011, Jordi Canals made the first implementation of these methods in Java. I recommend you to have a look his documentation about the first implementation if you want to see the past updates of these modules Canals [2011].

Both implementations are based on the same theory about *Ordered Weights Average* (OWA) family modules, but they differ in their programming language and internal structure. For this new implementation some features have been improved, the internal behaviour of some modules has been changed, new functionalities have been included, and some I/O parameters use different XMCDA structures. Due to all these structural changes and the new design, the source code of the old implementation has not been used for the new version, being the code generated in this project original.

4.1 Theoretical framework behind modules

4.1.1 OWA (Ordered weighted Average)

Ordered Weights Averaging (OWA) operators are defined by R. R. Yager in Yager [1988], these operators are used to aggregate the crisp values in decision-making schemes and they obtain a single representative value from a set of data. This result is achieved by averaging the criteria product with a weights vector. The main particularity of OWA family operators is that the order in which the user introduces the values does not have vital importance as before the operations, there are a sort step. Into OWA model, the central behavioural policy of the aggregation procedure is that a set of weights are attached to the values according to their degree of performance. The values are first sorted from the best one to the worst one, and a weight is assigned to the positions of the values, ending with the aggregation of all the obtained values.

Since Prof. Yager introduced the order weighted averaging (OWA), they have been used in many different applications, such as:

- fuzzy control
- linguistic summaries
- market analysis
- image compression

Usually OWA input sets of data are values, but this methodology can accept linguistic data in its variants, LOWA and ULOWA, with the introduction of the fuzzy set element. They will be explained in further detail in the ULOWA subsection. Here we present the numerical version and its properties Isern et al. [2017]:

Definition: we define OWA as an aggregation operator, with these components:

Firstly, OWA is defined with elements commonly found in aggregation operators as:

- Set of alternatives, $A = \{a_1, a_2, \ldots, a_p\}$
- Family of criteria, $C = \{c_1, c_2, \ldots, c_n\}$
- Criteria evaluate the degree of satisfaction for an alternative concerning to some property: $c_i(a_j) = u_{ji} \epsilon S$
- Next element necessary to OWA calculation is a weights vector with dimension n, this operator permits to define aggregation policies. This component satisfies:

  $$\sum_{j=1}^{n} w_j = 1 \;\; w_j \,\epsilon\, [0, 1]$$

  Finally, a weighted average is calculated after sorting the values decreasingly $(u_{\sigma(i)} > u_{\sigma(i+1)})$

  $$OWA(u_1, u_2, \ldots, u_n) = \sum_{i=1}^{n} \; w_i u_{\sigma(i)}$$

Notice that there is not an association between w_i and $u_{((i))}$. Currently, this weight element is associated with a particular position. As mentioned previously, OWA is an average operator, accordingly, it has the following properties:

- Idempotency: $H(u, u, u, \ldots, u) = u$
- Monotonicity: $u_j > u_j \rightarrow H(u_1, \ldots, u_j, \ldots, u_p) \geq H(u_1, \ldots, u_j, \ldots, u_p)$
- Neutrality: $H(u_1, u_2, u_3, \ldots, u_p) = H(u_{(\sigma(1))}, u_{(\sigma(2))}, u_{(\sigma(3))}, \ldots, u_{(\sigma(p))})$
- Boundaries: $\bigwedge_{j=1}^{p} u_j \leq H(u_1, u_2, u_3, \ldots, u_p) \leq \bigvee_{j=1}^{p} u_j$
- Associativity: $H(u_1, u_2, u_3) = H(H(u_1, u_2), u_3)$
- Decomposability:$H(u_1, u_2) = u^{'} \rightarrow H(u_1, u_2, u_3, \ldots, u_p) = H(u^{'}, u^{'}, u_3, .., u_p)$

An OWA calculation example can be:

1. Define an alternative set
$$A = (a)$$
2. Define a criterion set
$$C = (u_1, u_2, u_3, u_4)$$
3. Create the value associated vector to each criterion for the alternative:
$$C(a) = (0.5, 0.3, 0.2, 0.8)$$
4. Define weights vector with the same dimension of criteria:
$$W = (0.3, 0.1, 0.2, 0.4)$$
5. Sort $C(a)$ in descending order:
$$C(a) = (0.8, 0.5, 0.3, 0.2)$$
6. Finally, use OWA weights to aggregate these re-ordered arguments:
$$OWA(C) = 0.8 * 0.3 + 0.5 * 0.1 + 0.3 * 0.2 + 0.2 * 0.4 = 0.43$$

Weight vector properties

Given the set of weights W that the decision making specifies for the OWA operator, the character of the aggregation policy can be formalised using different measures. The most used ones are presented below Yager [1996]:

- Orness measures the degree of replaceability between the criteria. High orness means that the decision making is satisfied if only a few of the criteria are giving high utility values. On the contrary, a low orness (or high andness) means that the decision making will be only satisfied if all of the criteria are simultaneously fulfilled with high scores.
$$\alpha(W) = \sum_{i=1}^{n} w_i(\frac{n-j}{n-1}), with\ \alpha\ \epsilon\ [0, 1]$$
Outstanding cases:

- When $\alpha \cong 1$, high orness (disjunctive).
- When $\alpha \cong 0$, high andness (conjunctive).
- When $\alpha \cong 0.5$, neutral averaging.

- The balance is another measure to evaluate the same "behaviour". When calculating the minimum, we are performing a conjunctive policy (simultaneity, andness) and when calculating the maximum, we are performing a disjunctive policy (repleaceability or orness).

$$Bal(W) = \sum_{i=1}^{n} w_i(\frac{n+1-2j}{n-1})$$

Outstanding cases:

- $Bal(W) = -1$, minimum operator.
- $Bal(W) = 1$, maximum operator.
- $Bal(W) = 0$, arithmetic averaging.

- The entropy measures how many criteria are used to obtain the final result.

$$H(W) = \sum_{i=1}^{n} w_i \; ln(w_i)$$

Outstanding cases:

- When $w_j = \frac{1}{n}$, high entropy (all values used).
- When $w_q = 1$ and $w_j = 0$ for all $j \neq q$,then $H(W) = 0$, minimal information used.

- The divergence is evaluating whether the aggregation is concentrated on the extreme values or it is smoothly considering all the utility values.

$$Div(W) = \sum_{i=1}^{n} w_i(\frac{n-j}{n-1} - \alpha(W))^2$$

Outstanding cases:

- For the optimistic and pessimistic case, $Div(W) = 0$.
- If $w_j = 1$ for some j,then $Div(W) = 0$.

Weights calculation based on Orness value

The OWA weight vector can be calculated from its quantifiers. The quantifier is used to generate an OWA weighting vector W of dimension n. This weighting vector is then used in an OWA aggregation to determine the overall evaluation for each alternative Yager [1996].
Thus the process used in quantifier guided aggregation is as follows:

- Use Q to generate a set of OWA weights, $w_1, w_2, \ldots, w_n$.
- For each alternative x in X calculate the overall evaluation

We can consider a quantifier function that depends on the Orness measure to calculate the weights. In this case the weights are generated as:

$$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n}) \quad for\ i = 1...n$$

If we consider the quantifier

$$Q(r) = r^{\alpha} \quad \alpha \geq 0$$

Where α is

$$\alpha = \frac{1}{orness - 1}$$

The non-decreasing nature of Q it follows that $w_i > 0$. Furthermore, from the regularity of Q, $Q(1) = 1$ and $Q(0) = 0$, it follows that $\sum_i w_i = 1$. Thus we see that the weights generated are an acceptable class of OWA weights.

4.1.2 ULOWA (Unbalanced Linguistic Ordered Weighted Average)

In this section, we will concentrate on the aggregation of linguistic terms, in an extension of the OWA operator for a set of different linguistic labels Isern et al. [2017].
In this operator, the values that are aggregated are linguistic terms, each of them associated to a certain meaning (i.e. semantics). The meaning of the term is represented by a mapping into a numerical scale, known as Domain of Reference. The linguistic terms are defined in terms of Fuzzy Sets. A fuzzy set is a generalization of a Boolean Set. It allows a graduated degree of membership between 0 and 1.
The semantics of each linguistic label is given by a trapezoidal or triangular membership function $\mu : X \rightarrow [0, 1]$, that is represented with a tuple $P = (p1, p2, p3, p4)$, where $p1p2p3p4$ are the points in the reference domain X which define the trapezoid (Figure 4.1).

Figure 4.1: Fuzzy number trapezoid

Some special cases can be defined:

- $p1 = p2$ and $p3 = p4$ then P corresponds to a crisp interval.
- $p2 = p3$ the fuzzy set P is triangular, otherwise it is a trapezoid.
- $p1 = p2 = p3 = p4$, then P is called a crisp real number

The set of possible linguistic labels must be defined in advance. All the labels use the same domain of reference and they satisfy the following property:

$$\sum_{j=1}^{T} \mu_j(x_i) = 1 \ \ \forall \, x_i$$

ULOWA operator is an extension of LOWA operator that is designed to deal with unbalanced linguistic terms. These two approximations are adaptations of the OWA operator for dealing with linguistic labels. They operate directly on the labels that are aggregated in pairs by using convex combination operation. The following structure corresponds to ULOWA operator:

If $A = a_1, \ldots, a_m$ is the set of labels to aggregate, the ULOWA operator is defined as:

$$ULOWA(a_1, ..., a_m) = W * B^T = C^m\{\beta_h, b_h, h = 2, ..., m\} =$$
$$b_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h, b_h, h = 2, ..., m\}$$

Where $W = (w_1, \ldots, w_m)$ is an m-dimensional weighting vector so that $w_i \, \epsilon \, [0, 1]$ and $\sum w_i = 1$; $h = 2, \ldots, m$ and B is the associated ordered label vector (each element $b_i \, \epsilon \, B$ is the i-th largest label in A).We have to take into account that ULOWA operates on fuzzy sets; that is a membership functions of the terms. So, when $m = 2$, the convex combination of the two terms $b_1 = s_j$ and $b_2 = s_i$ is calculating using membership functions:

$$C^2\{w_i, b_i, i = 1, 2\} = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i = s_k; \ such \ that$$
$$k = \underset{i \leq p \leq j}{\operatorname{argmax}}\{Sim(S_p, \delta)\}$$

In this expression, δ is a crisp number defined as $\delta = (x_k, x_k, x_k, x_k)$ with $x_k = x^*_{s_i} + w_1(x^*_{s_j} - x^*_{s_i})$, where $x^*_{s_i}$is the x-component of the centre of gravity (COG) of the fuzzy set associated to the label $s_i = (p_1, p_2, p_3, p_4)$.

We can define the centre of gravity (COG) of a trapezoidal fuzzy set A as $COG(A) = (x^*_A, y^*_A)$, which can be calculated as follows:

$$COG(s_A) = \begin{cases} y^*_A = \begin{cases} \frac{1}{6}\,(\frac{p_3 - p_2}{p_4 - p_1} + 2) & if \ p_1 \neq p_4 \\ \frac{1}{2} & if \ p_1 = p_4 \end{cases} \\ x^*_A = \frac{y^*_A(p_3 + p_2) + (p_4 + p_1)(1 - y^*_A)}{2} \end{cases}$$

In this way, the aggregation result s_k is the linguistic term comprised between s_j and s_i with the greatest similarity with the intermediate point δ. The similarity between two fuzzy sets is as follows:

$$Sim(P,Q) = \sqrt[4]{\prod_{i=1}^{4}(2 - |p_i - q_i|)} - 1$$

After explaining ULOWA methodology is important to understand its properties, there are explained below:

- Increasing monotonous: with respect to the argument values. Let $A = [a_1, \ldots, a_m]$ and $B = [b_1, \ldots, b_m]$ be two ordered argument vector such that $\forall j, a_j \geq b_j$ then: $ULOWA(A) \geq ULOWA(B)$.
- Commutativity: $ULOWA(a_1, \ldots, a_m) = ULOWA(a_1^{'}, \ldots, a_m^{'})$ where $(a_1^{'}, \ldots, a_m^{'})$ is any permutation of the elements in $(a_1, \ldots, a_m)$.
- Idempotency: in the sense that if $\forall j, a_j = a$, then $ULOWA(a_1, \ldots, a_m) = a$.
- Boundered: for any weighting vector W and ordered labels $A = [a_1, \ldots, a_m]$ $(a_1 \geq a_2 \geq ... \geq a_m)$, then: $min(a_1, \ldots, a_m) \leq ULOWA(a_1, \ldots, a_m) \leq max(a_1, \ldots, a_m)$.

Characterization of fuzzy sets

Unbalanced terms in ULOWA operator permit the definition of linguistic variables with different granularity and distribution.

The difference on the certainty of the terms should be taken into account during the aggregation process, as each label is providing a different amount of information about the evaluated alternative. In fact, if we consider a triangular and trapezoidal fuzzy set, they can be associated to the labels, then the uncertainty of the labels is not only related to their support intervals in the reference domain but also to their kernel.

In Isern et al. [2017], authors propose using a measure of the uncertainty of the linguistic labels as the order-inducing criterion for the aggregation. Thus, the arguments will be ordered by decreasing uncertainty. In this way, the contribution of precise labels is prioritized while the effect of uncertain labels is reduced.

two types of uncertainty in fuzzy sets are recognized:

- Specificity: measures the degree of truth of the sentence "containing just one element". This uncertainty type is related to the measurement of imprecision, which is based on the cardinality of the set, which can be calculated as follows:

$$Sp(A) = 1 - \frac{\text{area under } A}{b - a}$$

- Fuzziness: measures the difference from a crisp set, and the vagueness of the set as a result of having imprecise boundaries, which can be calculated as follows:

$$Fz(A) = 1 - \frac{1}{b - a}\int_a^b |2A(x) - 1|$$

4.2 Diviz framework in detail

Before explaining how the new modules for the previous operators have been designed and implemented, we need to know the Diviz framework. In particular, the way of working with Diviz modules, the general structure for implementation, the requirements of documentation and licensing.

In Diviz, the main functionality is based on modules that receive mandatory and optional inputs, then apply the calculations associated with this decision-making method and finally generating output files. As we analysed previously in Diviz related part in chapter 2, it is possible to concatenate modules if they share the same structure as output parameter for the first module, and as input parameter for the second one, thus strategy allow the creation of more complex algorithms.

The inputs and outputs are defined in single, or composed, structures defined in different files. This data structure follows the XMCDA standardizations. That input can be defined by two or more structures, it is possible, but not very frequent.

The first step for a person who wants to start to develop a new MCDA algorithm must do is to download and install the latest version of Diviz program, which can be found into the Diviz website Consortium [2009c]. In this website, the user can find many different installers with some basic information to complete the installation successfully. Besides, in some operating systems it is necessary to install the Java 8 version to run Diviz environment, but in the most common operating systems this installation will be done automatically when a user installs the program.

To start the execution of any module or set of modules, it is necessary to create a new workflow. To open the workflow menu and click on new. Choose a name for your workflow, as it is represented below in (Figure 4.2):

Figure 4.2: Diviz creation workflow menus

With the new workflow created, the central panel is a blank canvas where the user can drag the modules and the input files necessary to complete the execution.

Figure 4.3: Blank canvas

Now it is time to define our example, in the new workflow I will drag and drop two different modules that will be useful to see the different outputs that Diviz modules can generate. These modules are plotFuzzyCategoriesValues OWAWeightsDescriptors (under the category Data Visualisation and calculation components respectively). Simply select them, drag and drop them to the central panel.

To run these two modules, three input files are required. These inputs must be data files which respect the XMCDA standard:

- criteria.xml (contains a list of criterion).
- wheights.xml (contains a list of alternatives values).
- FuzzyNumbers.xml (contains a list of fuzzy numbers).

Important observation: some of these input files structures are deprecated in the new version implemented in this project, but they are good examples to illustrate different cases.

To incorporate the necessary module inputs the user needs to drag and drop a File(s) element from the right panel into the central panel. A dialog window will be open and will ask you to choose an input file. This procedure is required as many times as the input files are required.

Coming up next, connect the input files to the corresponding pin inside the modules in the Canvas, as it is represented below in (Figure 4.4):

Figure 4.4: Module graphical connections

The last step before ran our example are not applicable to all the modules, but some of them have different options or settings that can be changed before the execution. In our example we can find that case in OWAWeightsDescriptors, in this module, we, as users, can choose between the four different parameters. To make changes to these options inside the module, we right-click on the module and check the "Settings" option, then a menu will open with different changeable parameters, in our example we will change the OWA descriptor to, for example, Orness (Figure 4.5).

Figure 4.5: Options menu of Diviz standard module

Finally, we can run the example with the engine icon in the menu, then an execution workflow will be generated below and linked with our editable workflow. In this new element, we will find the same modules and input files with the result files generated by the execution, but it is not

editable.

These output files are XMCDA structures and can be used as inputs of other modules if we concatenate the existing modules with the new ones plugging the necessary inputs and output pins of these modules. Diviz has a viewer of xml files that recognizes these XMCDA structures. If we go to our example, one module has graphical output (Figure 4.6), otherwise, the second module has a data table output.

Figure 4.6: Graphical representation of a fuzzy Numbers example

And the second one (Figure 4.7),

Criterion value

-0.6666667

Figure 4.7: numerical representation of a criterion example

4.3 Framework required to develop a module in XMCDA3

To design and develop the new version of the different OWA and ULOWA modules, first I had to know the structure of the existing modules, as well as the way of implementing the new ones in the new version. In the diviz website, I found that currently the modules are implemented basically in Python and R. I have chosen R because is a language focused on mathematical calculations, which can be done with great precision.

In order to be compatible with the existing work in Diviz, and also to easily exchange information with the Diviz managers, I followed the guidelines they provide in the project.

To develop the new implementation in R, I use a Linux operating system machine, in fact, I used a completely new Ubuntu image of Virtual Machine created with the program Virtual Box. The Linux option meant a manual installation of XMCDA-R and rJava. To do that installation, I used the instructions and packages available on XMCDA-R Github Meyer [2019] authored by Patrick Mayer in 2019. In this website you can find the following Linux commands (Figure 4.8)

required to carry out the installation of all the necessary resources to run the new modules' code before put it on the Diviz.

```
In order to build the package, you have to download the jar file of the latest
version of the XMCDA java implementation. For example here :

https://gitlab.com/XMCDA-library/XMCDA-java/builds/5024404/artifacts/file/dist/XMCDA-java-2016.10-1.jar

Create a java directory in the inst directory, and copy the jar there :

mkdir -p /path/to/XMCDA-R/inst/java
cp /path/to/XMCDA-java-2016.10-1.jar /path/to/XMCDA-R/inst/java/

Then build the package with the classical R command :

R CMD build /path/to/XMCDA-R/

To install the package, run the following command (on the generated .tar.gz):

cd ..
R CMD INSTALL XMCDA3_x.x.x.tar.gz

You will have to install the rJava package. On linux machines, start with

R CMD javareconf -e
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAVA_LD_LIBRARY_PATH

Then install the rJava package from R with the

install.packages("rJava")
```

Figure 4.8: Necessary orders to install Linux XMCDA

Finally, to develop all the source code of each module, I based on the documentation of Weighted-Sum Bigaret [2017], this Sébastien Bigaret authored example is used as a reference to new implementations in XMCDA3 and it is written in R language. In the next subsection, I will explain this program's structure with more details. To write this source code I used the RStudio program, an integrated development environment for R and Python, with a console, syntax-highlighting editor that supports direct code execution, and tools for plotting, history, debugging and workspace management, which are available to download freely in the official website.

4.3.1 General structure of a Diviz module in XMCDA3

As we mentioned previously, all new implemented modules are based on the reference program structure of the Weighted-Sum module that you can find on the Github portal Bigaret [2017], authored by Sébastien Bigaret. We can decompound all modules in:

- Source code folder: into this folder there are all source code files, these elements are the core of all module, the followed reference structure distributes all functions in five different R files well differentiated between them:
  - utils.R: this file is responsible of loading all XMCDA structures from the input files in the input directory with the **loadXMCDAv3()** function.
  - inputHandler.R: it is the file containing all functions related to transforming the XMCDA standard structures to R data structures and the input error control, the

main function to do it is **checkAndExtractInputs()** function, but can contain local called sub-functions if it is necessary to develop a well-structured code.

- methodCalculation.R: into this file takes place the module real purpose calculations. This file's main function returns the is resulting in R data structures. It can contain local called sub-functions if it is necessary to develop a well-structured code.
- outputHandler.R: it is the file containing all functions related to transform the R data structures to XMCDA standard structures. The main function to do it is **convert()** function, but it can contain local called sub-functions if it is necessary to develop a well-structured code.
- ModuleNameXMCDAv3.R (the *"main"* file): this is the responsible for guiding all the execution of the particular functions. When the module is running the main script (explained below) execute XMCDAv3 file passing by parameter the input and output directories, at the same time this module loads all the others files that are used to make subtasks. Between each subtask, there is an error control that stops the execution if an error is detected. After that, this function loads the XMCDA structures from the files in the input directory with the function **loadXMCDAv3()**.

```
loadXMCDAv3(xmcdaData, inDirectory, weightsFile, mandatory = TRUE, xmcdaMessages, "criteria")
loadXMCDAv3(xmcdaData, inDirectory, weightsFile, mandatory = TRUE, xmcdaMessages, "criteriaValues")
```

Figure 4.9: Example of the call to the load data function

Then, if the input is loaded correctly, it is time to translate these input XMCDA structures to R data structures to make the calculation and data error control, this task is made by **checkAndExtractInputs()** function.

```
checkAndExtractInputs(xmcdaData, programExecutionResult)
```

Figure 4.10: Example of the call to the input checker function

In this point, the module knows that all input which is necessary to make the calculation is correct and it is time to make it with the function **methodcalculation()**. Now, with the calculation done it is time to transform the data obtained in R structure to XMCDA formal structure, this work corresponds to **convert()** function.

```
xResults = convert(results, xmcdaMessages)
```

Figure 4.11: Example of the call to the output preparation function

Finally, the last remaining job is to write this output XMCDA structure to a file and write the process execution message obtained to a file. **writeXMCDA()** and **putProgramExecutionResult()** helps us to do that job.

```
writeXMCDA(xResults[[i]], outputFilename, xmcda_v3_tag(names(xResults)[i]))
putProgramExecutionResult(xmcdaMessages, infos="")
```

Figure 4.12: Example of the call to the write data functions

This schema (Figure 4.13) developed in MagicDraw19 pretends to make more understandable this source code structure.

Figure 4.13: Sequence diagram of standard Diviz module functionality

- Test folder: it is the location where all tests are done to validate the module correctness. For each test case, it will be two different folders, one of them for input files and a Readme to define the purpose of the test case. This first folder acts as input directory and is named as **inX.v3**. The second folder is the responsible for storing the output files generated in the test cases, for the module, acts as output directory, the name for this folder is **outX.v3**, where X is the test case number, as it is illustrated in the following capture (Figure 4.15). That naming structures are a convention to make easier the execution of all test cases and

recognition to a tester.
In the analysis of each module individually we will see the complete description of the particular test cases for all modules.

Figure 4.14: Example of test folder structure

- description-wsDD.xml: this file is an xml containg all the relevant data about the module, input and output files. This information is used to build the list of available modules website with information of each module implemented, you can find this available modules list into this reference link Consortium [2009d].
- README file: it is a short guide on how to run all test tests.
- LICENCE file: it is an informative file with the current license under which the code has been published, MIT License. If you want more information, there is a short subsection explaining this issue below.
- Module.sh: this is the main script to execute the module. It has the same name of module and needs the input and output directory as a parameter to realise its task.
- tests.sh: this is the responsible for calling the previous script as many times as tests we have into the tests folders, using the corresponding convention for names. This tests script does not write into any output directory. This script is the public one and guarantees the correctness of all tests done. Its purpose is to compare the current files in output directory with the output extracted by the execution, if there are equals, the test is passed successfully.
- tests2.sh: this is the responsible of calling the main module script as many times as tests we have into the tests folders, using the corresponding convention for names. This tests script is private to the author of the modules, and I used to generate the output files to the output directory to validate the implementation correctness.

```
#! /bin/bash

CMD="./OWA.sh"

if [ $# != 0 ]; then
  echo "Usage: ${0}" >&2
  exit 1
fi

version=3

NB_TESTS=$(find tests -maxdepth 1 -type d -regex '.*/in[0-9]*\.v'"$
{version}"'$' | wc -l)

for i in $(seq 1 ${NB_TESTS}); do
    IN="tests/in${i}.v${version}"
    OUT="tests/out${i}.v${version}"
    ${CMD} "${IN}" "${OUT}"
done
```

Figure 4.15: Script used by execute locally all test cases

4.3.2 Documentation file

This is an important part of the implementation; this xml file has to be published on the website of Diviz, into the available modules section Consortium [2009d], the purpose of this file is to allow a complete and quick comprehension about the module functionality for all community user, explaining the current version, authors, possible contacts, description about the module functionality as well as all relevant aspects of the input and output files for example:

- id: the identifier of the parameter number, the standard identifier structure used are "inputX" or "outputX" where X was the parameter number.
- name: the name of the stablished for the internal parameter file name.
- displayName: the name used into the Diviz workbench. This is the name the final user will see.
- isOptional: only required for input files, boolean attribute which indicates if this input parameter is mandatory (0), or is optional (1).
- description: brief description of the parameter, where its purpose as well as useful information for the user are usually described.
- xmcda Tag: parameter used to identify the XMCDA structure used to store data.
- xmcda structure: Schematic representation of the XMCDA structure used by the parameter, it is the same as that tagged in the previous attribute. Non-necessary for output files.

This documentation will be available on the official website, but currently you can only read this xml files in my personal Github Buxons [2020], where there is one documentation file named as **description-wsDD.xml** into each module folder. Anyway, I have attached a copy of all the documentation files, in the same format so that you could find it on my GitHub in Appendix A.

4.3.3 License file

Although all code generated in this project is open-source and it is free to download, it is necessary to distinguish the available actions that an external person is allowed to do, and responsibilities assumed or not by the author. We will use the MIT license.

MIT license is a permissive Free Software License Free Software License which means that it imposes very few limitations on reuse and, therefore, it has an excellent License Compatibility. The MIT license allows reuse of software within proprietary Software. On the other hand, this license is compatible with many copy-left licenses, such as the GNU General Public License.

As an additional and interesting issue, in 2015, MIT license, according to Black Duck Software and data from GitHub, became the most popular license above the GPL license variants.

```
MIT License

Copyright (c) 2020 Aida Valls, Miquel Buxons, Universitat Rovira i Virgili

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in the
Software without restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so, subject to the
following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Figure 4.16: MIT License file example

4.4 Data input structures changes from v2 to v3

As we have seen previously, each input and output data complies with the XMCDA standard. Ergo, each one is stored, generated and recognized because it fulfils this standard. More specifically, XMCDA provides a set of structures that allow the standardization of all the procedures and data typologies that are needed to develop and share MCDA modules.

When someone wants to contribute in including new modules to Diviz, he/she must follow the XMCDA language in the inputs and outputs. However, in this case, as our modules are the only ones that use weights for positions, linguistic terms and fuzzy sets, the managers of Diviz asked to the ITAKA group to collaborate in the definition of the format of the input and output files that use linguistic data.

In this project, I took the files from the previous version and I updated all XMCDA data structures of inputs and outputs files to version 3.1.1. XMCDA format. These are not substantial changes in all cases, but more or less all of the data files changed the format since the XMCDAv2

is the current version at the time of the first implementation. We can classify these adjustments in three categories.

- File header format changes: this is the most frequent change, in fact, it is necessary updating to all input and output files. The header is an important part of an XMCDA file and it allows to all elements in Diviz to recognise the version of the standard is used. As example, the current header format to XMCDAv3.1.1. is (Figure 4.17):

```
<?xml version="1.0"?>
<xmcda:XMCDA xsi:schemaLocation="http://www.decision-deck.org/2019/XMCDA-3.1.1
http://www.decision-deck.org/xmcda/_downloads/XMCDA-3.1.1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xmcda="http://www.decision-
deck.org/2019/XMCDA-3.1.1">
```

Figure 4.17: Current header for files that run in XMCDA3.1.1. structure

- Minor data structure changes: in this case, they are motivated by the change in the structure format used during the time elapsed until the new version. Usually, these changes the previous structure, so it is easy to interpret it.
  An example of this could be showed in the case of performance, illustrated below in (Figure 4.18), a subcategory of performance table, which previously could contain a set of tagged "value", without a label that included all of them in a subcategory, as if it happens in the new version in which said structure needs a label "values" that gathers all the "value" tags.

```
- <performanceTable>
  - <alternativePerformances>
      <alternativeID>Poznanzki</alternativeID>
    - <performance>
        <criterionID>location</criterionID>
      - <values>
        - <value>
            <real>0.0</real>
          </value>
        </values>
      </performance>
```

```
- <performanceTable>
  - <alternativePerformances>
      <alternativeID>Poznanzki</alternativeID>
    - <performance>
        <criterionID>location</criterionID>
      - <value>
          <real>0.0</real>
        </value>
      </performance>
```

Figure 4.18: Illustration of a minor change in parameter files

- Proposal of new file format: this is the case in which we were required to define a new structure for the files of OWA and ULOWA operators. These changes have been made in accordance with experience after the usage of the previous structures during these years. In addition, we have made a proposal that is compatible with the new simplified version of the XML language and looking for a new implementation that would be more useful and understandable. The new structures were proposed by me, agreed with my project tutor, Dra. Aïda Valls, and with the managers of Diviz, Dr. Bigaret and Dr. Meyer. We have defined the structure of 3 data structures:

  - Weights: previously, this input was structured in the XMCDA structure of Alternatives Values. This structure allows storing all weights values as a values list. It is a very easy to work with and understandable structure, but if we focus on the syntax of this solution, it said that the weights vector is a list of different alternatives and weights in OWA family are a vector of values that permits to evaluate all criteria of an alternative. To replace this structure and, at the same time, improve the performance, the new implementation work with weights vector structured in two different

XMCDA options.
The first one is a criteria structure, basically, a list of a criterion where the fixed names of weights are defined. In this new implementation, we wanted to standardize the weights' names. Finally, we decided, that the best naming form it is posXX because the only important characteristic for each weight is its position into the vector. The second structure is properly the weights values vector, using a criteriaValues structure defined in XMCDA. In this structure, each criteria value substructure has the same identifier as the one used in the first structure and a real value that would be the weights is assigned into this position. A representative scheme of that structure would be (Figure 4.19):

Figure 4.19: Weights file current structure

– Orness: this input structure was not necessary for the first version of OWA and ULOWA modules. It is used in this project because there is a new functionality into these modules that allows the Diviz user to choose which type of input weights can use. In Orness option, the user gives a program parameter XMCDA structure, in which there are two fixed parameters; an activation attribute which is a Boolean and a Orness real value. To obtain that weights vector based on Orness value using the methodology explained in subsection (4.1.1).These formulas depends on n, that is the number of weights that we want to generate $w = [w_1, w_2, \ldots, w_n]$ A representative scheme for store this Orness value would be (Figure 4.20):

Figure 4.20: Orness file current structure

– Fuzzy Numbers: this is the structure that has changed the most, in fact, the main XMCDA is the same externally, a categoriesValues structure. In the past, internally, all fuzzy numbers were structured as only one categoryValue in which there is a list of fuzzy numbers. This structure is a bit difficult to work with but the issue that produces the change is that this fuzzy number final typology is unnecessary, take a look to the schema that represents the past fuzzy numbers old structure would be (Figure 4.21):

Figure 4.21: Fuzzy numbers file old structure

We can see that there is a huge amount of information in a fuzzy number, but after eight years of experience using this typology, the tutor asked me to simplify the schema. This simplification comes with the suppression of ordinate value, because after ages of experience, the users always work with the same ordinate values (0,1,1,0). This change made unnecessary the heavy structure of trapezoidal fuzzy numbers. Abetter option would be the same categoriesValues main structure but with as many categoryValue as fuzzy numbers we had. Into this categoryValue we will have four real number that would represent the abscissa values into this fuzzy number and we assume the ordinate values as (0,1,1,0). This is the new schema for fuzzy numbers XMCDA representation (Figure 4.22):

Figure 4.22: Fuzzy numbers file current structure

To sum up this subsection, (Table 4.1) indicates the changes and design decisions made into input XMCDA data structures for this project.

Design decisions table	Header changes	Minor changes	Complete changes
Alternatives	Yes	Yes	No
Criteria	Yes	No	No
Performance table	Yes	Yes	No
Weights	Yes	No	Yes
Fuzzy Numbers	Yes	No	Yes
Orness value	Yes	No	Yes

Table 4.1: Changes done in input files

4.5 XMCDA data structure used for this implementation

For all developed modules to Diviz, they receive parameters and generate outputs in XMCDA data structures format, the modules developed in this project there are not an exception. It is not necessary to use a structure for each type of input or output parameter, often, these parameters are structured on the same XMCDA options. This option will never cause errors if it is taken into account when developing the module.

For this development the used XMCDA structures are nine in total, represented in the following (Table 4.2):

XMCDA structure	Parameters using this structure
alternatives	Alternatives.
alternativesValues	OWA & ULOWA results, Fuzziness, Specificity, COG, COM & Ordinal.
criteria	Criteria & weights.
criteriaValues	Weights, Balance, Divergence, Entropy & Orness
performanceTable	Performance Table.
categories	Fuzzy numbers.
categoriesValues	Fuzzy numbers.
programParameters	Orness as input.
programExecutionResults	Output messages.

Table 4.2: XMCDA3 structures used for each parameter

For all these XMCDA structures I have prepared the class diagram that explains their structural tree. The following class diagrams, internally, are organized so that the main structure is placed at the top and all its subclass are below it, and thus, concurrently.

4.5.1 alternatives

Figure 4.23: XMCDA3 alternatives structure

4.5.2 alternativesValues

Figure 4.24: XMCDA3 alternativesValues structure

4.5.3 criteria

Figure 4.25: XMCDA3 criteria structure

4.5.4 criteriaValues

Figure 4.26: XMCDA3 criteriaValues structure

4.5.5 categories

Figure 4.27: XMCDA3 categories structure

4.5.6 categoriesValues

Figure 4.28: XMCDA3 categoriesValues structure

4.5.7 performanceTable

Figure 4.29: XMCDA3 performanceTable structure

4.5.8 programParameters

Figure 4.30: XMCDA3 programParameters structure

4.5.9 programExecutionResult

Figure 4.31: XMCDA3 programExecutionResult structure

4.6 Design and implementation of the new modules

In this section the 5 modules designed, developed and deployed are explained in detail. Each of the subsections follows the same structure, from the input/output data, to the operations, finishing with the use case tests and an example.

4.6.1 OWA Module

This module implements the OWA aggregation operator: The Ordered Weighted Averaging operators, commonly called OWA operators, provide a parameterized class of mean type aggregation operators.

OWA was defined by R.R. Yager in 1988. Here we provide an implementation of OWA operator. The main characteristic of this aggregation operator is the fact that the weights are associated with values instead of criteria. In this way, we can define different aggregation policies according to the importance that we associate with high and low performance scores.

Inputs and Outputs

To represent all input and output of this module I have done the following (Table 4.3) explaining the specific characteristics for each parameter: *The parameter "isOptional" is only applied on

I/O Parameters	id	name	displayName	isOptional	tag	structure
alternatives	input0	alternatives	alternatives	No	alternatives	4.2.12.1
criteria	input1	criteria	criteria	No	criteria	4.2.12.3
performanceTable	input3	performanceTable	performanceTable	No	performanceTable	4.2.12.5
weights	input2	weightsOWA	weights	Yes	criteria, criteriaValues	4.2.12.3 4.2.12.4
orness input	input4	programParameters	orness	Yes	programParameters	4.2.12.8
OWA result	output0	alternativesValues	owaResults	-	alternativesValues	4.2.12.2
message	output1	messages	messages	-	programExecutionResults	4.2.12.9

Table 4.3: OWA module parameters

input parameters and all structure column values are the subsection number into this documentation where this structure is well-specified.

Module design and implementation

In (Figure 4.32), we can see the flow of the program operation in OWA module.

Figure 4.32: OWA state-machine diagram

In this module I want to highlight the new checks applied to the weights file, inherent to the change made in its structure, these controls provide the security that these weights are provided in the correct nomenclature.
Also, characteristics are checked as if the names established in the performance table match with those mentioned in the alternatives and criteria. Among other checks on the format of each file that you can see in the testing section.

The inclusion of the new Orness input parameter stands out, the program provides the possibility of calculating the vector of weights with this Orness value. This change and the possibility to obtain those weights through traditional way, forces that these two files to be optional, both can never be provided together, but one of them is required to perform the calculation.

Tests

To verify the correctness of my new implementation of OWA module, I prepared a complete set of test cases to certify that I contemplate the maximum of errors that can be made, or limitations imposed by the method itself to be correct, for example, the range of limitations in an input parameter or non-correct input data types.

For this task, I have designed (Table 4.4) with the purpose of having all the well-established verified cases, so that you can see the number of test cases, their purpose, the output obtained and whether it is expected in each case.

Test number	Case purpose	Expected output
1	All input files correct (simple case).	correct results.
2	Complex correct case.	correct results.
3	ID in alternatives no coincident with performance table.	Error: there are some different id's in alternatives table and the alternatives in Performance table.
4	Different number of active criteria and OWA weights.	Error: different number of active weights and active criteria.
5	the performance table has some NA values.	correct results, with NaN as a OWA result for each alternative with almost one NaN value in criteria.
6	Some weights have no numeric values.	Error: The weights must be numeric values only.
7	some inactive alternatives	correct result, without the inactive alternative.
8	A criteria element with false value (number of weight coincident with number of active criterion).	correct results. The inactive criteria are unused to do the OWA calculation.
9	OWA weights table have some NA values.	Error: there is a missing value (NA) in OWA weights vector.
10	missing mandatory input.	Error: An error has occurred while loading the input files.
11	All input correct (simple case) with Orness value in program parameter file.	correct results.
12	Program parameters without the correct data type of activation parameter.	Error: Activation parameter in Orness program parameters structure must be Boolean.
13	Program parameters with wrong id, for example "divergence".	Error: Structural error in program parameters file.
14	Orness parameter with value out of the range [0:1].	Error: Orness value must be a real number inside the [0:1] range.
15	Weights table and active Orness are provided.	Error: too many weights options supplied (Weights table and Orness, only one is needed).
16	Weights table and Orness are provided, but Orness is inactive.	correct results using Weights table values.

Table 4.4: OWA tests cases

Example

As a practical case, I will propose you the following example with these input parameters:

- Four active alternatives (a1, a2, a3, a4)
- Five active criterion (c1, c2, c3, c4, c5)
- The weights table are unused for this example. The weights will be generate using the Orness value (0.5) that will generate the neutral averaging weights table (0.2, 0.2, 0.2, 0.2, 0.2).
- The performance table used will be as the following (Table 4.5):

	c1	c2	c3	c4	c5
a1	0.2	2	4	3	9
a2	3	6	3.7	0	1
a3	6.5	5	6	3	9.8
a4	1	7	14	10	0.01

Table 4.5: Performance table for this example

After running the module, we obtain the following solution represented in (Table 4.6) and graphically in (Figure 4.33):

Alternatives	Values
a1	3.64
a2	2.74
a3	6.06000000005
a4	6.402

Table 4.6: Results obtained applying Orness value

Figure 4.33: Graphical results

Or we can also generate an example with a vector of manually generated weights like: W=(0.2, 0.1, 0.0, 0.0, 0.7). obtaining the following solution represented in (Table 4.7) and graphically in (Figure 4.34):

Alternatives	Values
a1	2.34
a2	1.57
a3	4.71
a4	3.807

Table 4.7: Results obtained applying weights vector

Figure 4.34: Graphical results

This practical case has been run on the new implementation modules, and graphics were generated using the results that have been previously used and exporting them to the latest available version of Diviz which can be used as input for the module **plotAlternativesValues** developed by ITTB.

4.6.2 ULOWA Module

This module implements the ULOWA aggregation operator: Unbalanced Linguistic Ordered Weighted Average. Aggregation operators for linguistic variables usually assume uniform and symmetrical distribution of the linguistic terms that define the variable.

However, there are some problems where an unbalanced set of linguistic terms is more appropriate to describe the objects. ULOWA accepts a set of linguistic labels defined with unbalanced fuzzy sets. The fuzzy sets must define a fuzzy partition on the set of reference values. They can be defined by trapezoidal or triangular membership functions.

Inputs and Outputs

To represent all input and output of this module I have done the following (Table 4.8) explaining the specific characteristics for each parameter:

I/O Parameters	id	name	displayName	isOptional	tag	structure
alternatives	input0	alternatives	alternatives	No	alternatives	4.2.12.1
criteria	input1	criteria	criteria	No	criteria	4.2.12.3
performanceTable	input3	performanceTable	performanceTable	No	performanceTable	4.2.12.5
weights	input2	weightsOWA	weights	Yes	criteria, criteriaValues	4.2.12.3 4.2.12.4
orness input	input5	programParameters	orness	Yes	programParameters	4.2.12.8
fuzzyNumbers	input4	fuzzyNumbers	fuzzyNumbers	No	categories, categoriesValues	4.2.12.6 4.2.12.7
ULOWA result	output0	alternativesValues	ulowaResults	-	alternativesValues	4.2.12.2
message	output1	messages	messages	-	programExecutionResults	4.2.12.9

Table 4.8: ULOWA module parameters

*The parameter "isOptional" is only applied on input parameters and all structure column values are the subsection number into this documentation where this structure is well-specified.

Module design and implementation

In (Figure 4.35), we can see the flow of the program operation in ULOWA module.

Figure 4.35: ULOWA state-machine diagram

Apart from the same change made in the module like in OWA module, due to the addition of the optional Orness parameter this module stands out for the use of recursion to perform a calculation. This is because it was more efficient and understandable. This is the only functionality of the project programmed in this way.

$$ULOWA(a_1, ..., a_m) = W * B^T = C^m\{\beta_h, b_h, h = 2, ..., m\} =$$
$$b_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h, b_h, h = 2, ..., m\}$$

Tests

To verify the correctness of my new implementation of ULOWA module, I prepared a complete set of test cases to certify that I contemplate the maximum of errors that can be made, or limitations imposed by the method itself to be correct, for example, the range of limitations in an input parameter or non-correct input data types.
For this task, I have designed (Table 4.9) with the purpose of having all the well-established verified cases, so that you can see the number of test cases, their purpose, the output obtained and whether it is expected in each case.

Test number	Case purpose	Expected output
1	All input files correct.	correct results.
2	Complex correct case.	correct results.
3	ID in alternatives no coincident with performance table.	Error: there are some different id's in alternatives table and the alternatives in Performance table.
4	Different number of active criteria and ULOWA weights.	Error: different number of active weights and active criteria.
5	the performance table has some NA values.	Error: number of items to replace is not a multiple of replacement length.
6	Some weights have no numeric values.	Error: The weights must be only numeric values.
7	some inactive alternatives	correct results.
8	A criteria element with false value (number of weight no coincident with number of active criterion).	Error: different number of active weights and active criteria.
9	ULOWA weights table have some NA values.	Error: there is a missing value (NA) in ULOWA weights vector.
10	NA in the fuzzy Numbers values.	Error: There are non-numeric values or NA in almost one fuzzyNumber.
11	missing mandatory input.	Error: An error has occurred while loading the input files.
12	All input correct (simple case) with Orness value in program parameter file.	correct results.
13	Program parameters without the correct data type of activation parameter.	Error: Activation parameter in Orness program parameters structure must be Boolean.
14	Program parameters with wrong id, for example "divergence".	Error: Structural error in program parameters file.
15	Orness parameter with value out of the range [0:1].	Error: Orness value must be a real number inside the [0:1] range.
16	Weights table and active Orness are provided.	Error: too many weights options supplied (Weights table and Orness, only one is needed).
17	Weights table and Orness are provided, but Orness is inactive.	correct results using Weights table values.

Table 4.9: ULOWA tests cases

Example

As a practical case, I will propose you the following example with these input parameters:

- Four active alternatives (a1, a2, a3, a4)
- Five active criterion (c1, c2, c3, c4, c5)
- The weights table are unused for this example. The weights will be generate using the Orness value (0.5) that will generate the neutral averaging weights table (0.2, 0.2, 0.2, 0.2, 0.2).
- The fuzzy numbers used are the following seventh label table (Figure 4.36):

Figure 4.36: Fuzzy numbers for the example

- The performance table used will be as the following (Table 4.10):

	c1	c2	c3	c4	c5
a1	VL	P	AH	AH	M
a2	VL	M	AH	H	VL
a3	L	AH	P	P	VH
a4	M	VH	VL	VH	VL

Table 4.10: Performance table for this example

After running the module, we obtain the following solution represented in (Table 4.11) and graphically in (Figure 4.37):

Alternatives	Labels
a1	Medium
a2	Low
a3	Almost-High
a4	Medium

Table 4.11: Results obtained applying Orness value

Figure 4.37: Graphical results

Or we can also generate an example with a vector of manually generated weights like: W=(0.2, 0.1, 0.0, 0.0, 0.7). obtaining the following solution represented in (Table 4.12) and graphically in (Figure 4.38):

Alternatives	Labels
a1	Low
a2	Low
a3	Medium
a4	Low

Table 4.12: Results obtained applying weights vector

Figure 4.38: Graphical results

This practical case has been run on the new implementation modules, and graphics were generated using the results that have been previously used and exporting them to the latest available version of Diviz which can be used as input for the module **plotAlternativesValues** developed by ITTB.

4.6.3 OWADescriptors Module

Compute measures of weights given to the Ordered Weighted Average. The Ordered Weighted Averaging operators, commonly called OWA operators, provide a parameterized class of mean type aggregation operators.
For OWA weights exists different measures to characterise a set of weights associated to an OWA operator. In this module we implement the measures: balance, divergence, entropy and Orness.

Inputs and Outputs

To represent all input and output of this module I have done the following (Table 4.13) explaining the specific characteristics for each parameter:

I/O Parameters	id	name	displayName	isOptional	tag	structure
Weights	input0	weights	weightsOWA	No	criteria, criteriaValues	4.2.12.3 4.2.12.4
balance	output0	balance	balance	-	criteriaValues	4.2.12.4
divergence	output1	divergence	divergence	-	criteriaValues	4.2.12.4
entropy	output2	entropy	entropy	-	criteriaValues	4.2.12.4
orness	output3	orness	orness	-	criteriaValues	4.2.12.4
message	output4	messages	messages	-	programExecutionResults	4.2.12.9

Table 4.13: OWADescriptors module parameters

*The parameter "isOptional" is only applied on input parameters and all structure column values are the subsection number into this documentation where this structure is well-specified.

Module design and implementation

In (Figure 4.39), we can see the flow of the program operation in OWADescriptors module.

Figure 4.39: OWADescriptors state-machine diagram

The main difference of this module respect to its previous implementation is that, in this new design, it is only necessary to receive the weight vector as a parameter, it is not necessary to receive the criteria to execute because it do not have any functionality.

On the other hand, instead of the user have to choose which weight vector descriptor he wants to receive, the module calculates the four output parameters in the same execution and provides them to the user.

Tests

To verify the correctness of my new implementation of OWADescriptors module, I prepared a complete set of test cases to certify that I contemplate the maximum of errors that can be made, or limitations imposed by the method itself to be correct, for example, the range of limitations in an input parameter or non-correct input data types.

this task, I have designed (Table 4.14) with the purpose of having all the well-established verified cases, so that you can see the number of test cases, their purpose, the output obtained and whether it is expected in each case.

Test number	Case purpose	Expected output
1	All input files correct.	correct results.
2	Complex correct case.	correct results.
3	Weights with no numeric in inputs.	Error: The weights must be numeric values only.
4	Weight summation result >1.	Error: the summation of all active weights must be exactly 1.0.
5	More than 1 weights table supplied.	Error: More than one weights table supplied.
6	special case result in Orness.	correct results. Orness = 0.5.
7	special case result in balance.	correct results. Balance = 0.
8	special case result in entropy.	correct results. Entropy = 0.
9	Special case result in divergence	correct results. Divergence = 0
10	Weights with Na values.	Error: there is a missing value (NA) in OWA weights vector.
11	No input supplied.	Error: An error has occurred while loading the input files.

Table 4.14: OWADescriptors tests cases

Example

As a practical case, I will propose the following example with this input parameter:

- The weights table used for this example is (0.2, 0.2, 0.2, 0.2, 0.2).

After running the module, we obtain the following solution (Table 4.15):

Properties	Results
Balance	$2.77555e^{-17}$
Divergence	0.125
Entropy	1.60944
Orness	0.5

Table 4.15: Results

This practical case has been run on the new implementation modules.

4.6.4 Deffuzification Module

This module implements three different methods to defuzzificate the result of a ULOWA operation (Unbalanced Linguistic Ordered Weighted Average). Aggregation operators for linguistic variables usually assume uniform and symmetrical distribution of the linguistic terms that define the variable. However, there are some problems where an unbalanced set of linguistic terms is more appropriate to describe the objects than others. ULOWA accepts a set of linguistic labels defined with unbalanced fuzzy sets. The fuzzy sets must define a fuzzy partition on the set of reference values. They can be defined by trapezoidal or triangular membership functions.
For this method we apply three different operations for every ULOWA alternative result; COG, COM and Ordinal.

Inputs and Outputs

To represent all input and output of this module I have done the following (Table 4.16) explaining the specific characteristics for each parameter:

I/O Parameters	id	name	displayName	isOptional	tag	structure
ULOWA result	input0	alternativesValues	linguistic scores	No	alternativesValues	4.2.12.2
fuzzyNumbers	input1	fuzzyNumbers	fuzzyNumbers	No	categories, categoriesValues	4.2.12.6 4.2.12.7
COG	output0	defuzzificationCOG	COG	-	alternativesValues	4.2.12.2
COM	output1	defuzzificationCOM	COM	-	alternativesValues	4.2.12.2
ORD	output2	defuzzificationORD	Ordinal	-	alternativesValues	4.2.12.2
message	output3	messages	messages	-	programExecutionResult	4.2.12.9

Table 4.16: Deffuzification module parameters

*The parameter "isOptional" is only applied on input parameters and all structure column values are the subsection number into this documentation where this structure is well-specified.

Module design and implementation

In (Figure 4.40), we can see the flow of the program operation in Deffuzification module.

Figure 4.40: Deffuzification state-machine diagram

The main change made in this module focuses on the generation of output files, as in the previous case, instead of that the user has to choose which descriptor he wants to receive (COG, COM or Ordinal), the module calculates the three output parameters in the same execution and provides them to the user.

- COG value consists to apply the formula of center of gravity explained in (4.1.2):

$$COG(s_A) = \begin{cases} y_A^* = \begin{cases} \frac{1}{6}\,(\frac{p_3 - p_2}{p_4 - p_1} + 2) & if\ p_1 \neq p_4 \\ \frac{1}{2} & if\ p_1 = p_4 \end{cases} \\ x_A^* = \frac{y_A^*(p_3 + p_2) + (p_4 + p_1)(1 - y_A^*)}{2} \end{cases}$$

- In COM (center of Maxima), the defuzzified value is taken as the element with the highest membership values. When there are more than one element having maximum membership values, the mean value of the maxima is taken.
  Let A be a fuzzy set of label l_i with membership function $\mu_A(x)$ defined over $x \epsilon X$, where X is a universe of discourse. The defuzzified value is let say x^* of a fuzzy set and is defined as,

$$X^* = \frac{\sum x_i \epsilon M\ x_i}{|M|}$$

  Here, $M = \{x_i | \mu_A(x_i)$is equal to the height of the fuzzy set $A\}$ and $|M|$ is the cardinality of the set M.

- Ordinal descriptor consists of: let A be a fuzzy set of the i-th label in the set of labels L, the defuzzified value x^* of A will be $x^* = i$, being i the position of the label in L.

Tests

To verify the correctness of my new implementation of Deffuzification module, I prepared a complete set of test cases to certify that I contemplate the maximum of errors that can be made, or limitations imposed by the method itself to be correct, for example, the range of limitations in an input parameter or non-correct input data types.
For this task, I have designed (Table 4.17) with the purpose of having all the well-established verified cases, so that you can see the number of test cases, their purpose, the output obtained and whether it is expected in each case.

Test number	Case purpose	Expected output
1	All input files correct.	correct results.
2	Complex correct case.	correct complex results.
3	ID in alternatives table, not coincident with fuzzy number ID.	Error: Different id in categories and the id in their values tables.
4	NA in a fuzzy Number.	Error: There are non-numeric values or NA in, almost, one fuzzy Number.
5	No enough categories supplied.	Error: there are almost one alternative not expected in fuzzyNumbers table
6	NaN value in the alternatives values.	'match' requires vector.

Table 4.17: Deffuzification tests cases

Example

As a practical case, I will propose the following example with this input parameter:

- The fuzzy numbers used are the following seventh label table (Figure 4.41):

Figure 4.41: Fuzzy numbers for this example

- As alternatives values I used the output generated for ULOWA module represented in the following table (Figure 4.42):

Alternatives	Labels
a1	Medium
a2	Low
a3	Almost-High
a4	Medium

Figure 4.42: Alternatives for this example

After running the module, we obtain the following solution (4.43):

Alternatives	COG	COM	ORD
a1	0.50	0.50	3
a2	0.30	0.30	2
a3	0.60	0.60	4
a4	0.50	0.50	3

Figure 4.43: Results for this example

This practical case has been run on the new implementation modules, and graphics were generated using the results that have been previously used and exporting them to the latest available version of Diviz which can be used as input for the module **plotFuzzyCategoriesValues** developed by ITTB.

4.6.5 Fuzzy Labels Descriptors Module

The purpose of this module is to obtain the two types of uncertainty in fuzzy sets are recognized:

- Specificity, related to the measurement of imprecision, which is based on the cardinality of the set. Specificity and fuzziness refer to two different characteristics of fuzzy sets, measures the degree of truth of the sentence containing just one element.
- Fuzziness, or entropy, which measures the vagueness of the set as a result of having imprecise boundaries. Fuzziness measures the difference from a crisp set.

Inputs and Outputs

To represent all input and output of this module I have done the following (Table 4.18) explaining the specific characteristics for each parameter:

I/O Parameters	id	name	displayName	isOptional	tag	structure
fuzzyNumbers	input0	fuzzyNumbers	fuzzyNumbers	No	categories, categoriesValues	4.2.12.6 4.2.12.7
specificity	output0	specificity	specificity	-	alternativesValues	4.2.12.2
fuzziness	output1	fuzziness	fuzziness	-	alternativesValues	4.2.12.2
message	output2	messages	messages	-	programExecutionResults	4.2.12.9

Table 4.18: fuzzyNumbersDescriptors module parameters

*The parameter "isOptional" is only applied on input parameters and all structure column values are the subsection number into this documentation where this structure is well-specified.

Module design and implementation

In (Figure 4.44), we can see the flow of the program operation in fuzzyNumbersDescriptors module.

Figure 4.44: fuzzyNumbersDescriptors state-machine diagram

Tests

To verify the correctness of my new implementation of fuzzyLabelsDescriptors module, I prepared a complete set of test cases to certify that I contemplate the maximum of errors that can be made, or limitations imposed by the method itself to be correct, for example, the range of limitations in an input parameter or non-correct input data types.
For this task, I have designed (Table 4.19) with the purpose of having all the well-established verified cases, so that you can see the number of test cases, their purpose, the output obtained and whether it is expected in each case.

Test number	Case purpose	Expected output
1	All input files correct, simple case.	correct results.
2	Complex correct case.	correct results.
3	No ordered point in a fuzzy number values.	Error: There are, almost one, fuzzyNumber not ordered.
4	Out of fuzzy numbers rank (0 <= fuzzy numbers <= 1).	correct results, any problems because you can take a different rank.
5	No correct coverage of fuzzy numbers v1.	Error: Incorrect fuzzy Number, rank must be well covered.
6	No correct coverage of fuzzy numbers v2.	Error: Incorrect fuzzy Number, rank must be well covered.
7	Different category id in Categories and criteria values table.	Error: Different id in categories and their values.
8	More than 1 fuzzy numbers table supplied.	Error: Incorrect number of fuzzy number tables supplied.
9	No fuzzy numbers value table supplied.	Error: Incorrect number of fuzzy number tables supplied.
10	NA in a fuzzy number value.	Error: There are non-numeric values or NA in almost one fuzzyNumber.

Table 4.19: Fuzzy numbers tests cases

Example

As a practical case, I will propose the following example with this input parameter:

- The fuzzy numbers used are the following seventh label table (Figure 4.45):

Figure 4.45: Fuzzy numbers for this example

After running the module, we obtain the following solution (Figure 4.46):

Fuzzy Labels	Specificity	Fuzziness
Very-Low (VL)	0.85	0.05
Low (L)	0.7	0.01
Medium (M)	0.9	0.01
Almost-High (AH)	0.9	0.01
High (H)	0.825	0.075
Very-High (VH)	0.9	0.05
Perfect (P)	0.925	0.025

Figure 4.46: Results for this example

This practical case has been run on the new implementation modules, and graphics were generated using the results that have been previously used and exporting them to the latest available version of Diviz which can be used as input for the module **plotFuzzyCategoriesValues** developed by ITTB.

Chapter 5

Conclusions

The main idea for this project has been the complete development, testing and embedding of modules based on the OWA and ULOWA aggregation models and, MCDA components, in the Diviz collaborative environment. The modules developed would be fully functional and public. The objectives set at the beginning of this project have been achieved and even ideas not previously contemplated have been developed:

- Before the development of the modules, the key components associated with the theory behind them were studied as well as how they were developed in the first version.
- Subsequently, the new options that the XMCDA3 version allowed were studied. The modules were developed by considering these new options that the XMCDA library and the new programming language used allowed.
- Changes were made to some of the input and output parameters of these modules to facilitate their processing and external understanding. Even new features were implemented in modules that were not initially taken into account, such as the insertion of the Orness parameter as the input parameter of the OWA and ULOWA modules.
- Documentation in English was generated to be also used as a guide for future users and / or developers.

During the development of this job I have raised new ideas to consider for further projects about Diviz:

- The implementation of new OWA family-related modules as IOWA and IULOWA to Diviz framework.
- Create some modules that generate a graphical representation of the fuzzy sets or the output of ULOWA.

This project was an enriching experience for my person, where I could work in some non-explored disciplines during my degree. I learned and improved some essential professional skills, such as cooperative work and well-structured code construction. It made me discover what it is to be working in the Artificial intelligence field.

In the beginning I did not have a defined idea about how to approach this project. However, I was sure that I wanted to work in the Artificial intelligence field.

My tutor proposed me the idea of making an implementation of some decision-decision modules for the program Diviz, but I was completely free to choose in which programming language I could use and how I could organize my project as well as the task. The appointments with her, including virtual appointments during the COVID-19, were frequent as well as the communication via email with Dr. Sébastien Bigaret. With him, I was able to solve all the technical questions about how to develop the modules and, later, to upload them to the web services platform.

I chose the R language to program all the source code with the idea of improving with this programming language, as it is very useful in some computational camps. In addition, I chose to write it in the English language to give a coherence with my English wrote source code and comments, and to make possible to others Diviz modules developers to consult my project documentation to solve doubts or to find an example of how to develop these modules in R language.

This project has been a complete experience for me. I can develop some useful skills that, during URV Computer science degree, were not trained overly. I am convinced that it has been an experience that will help me in my future professional carrier.

Bibliography

José R. Figueira, Salvatore Greco, and Matthias Ehrgott. *Multiple criteria decision analysis: State of the Art Surveys*. Springer-Verlag New York, 2005.

Decision Deck Consortium, Software Resources Management Groups, Specifications Committee, and Communication Dissemination Committee. Strategic manifesto for the decision deck project. `https://www.decision-deck.org/project/_static/D2manifesto.pdf`, Mar 2009. Accessed on 2020-04-11.

Valerie Belton and Theodor J. Stewart. *Multiple Criteria Decision Analysis: an integrated approach*. Kluwer Academic Publishers, 2002.

Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: preferences and value trade off*. Cambridge University Press, 1993.

Research Gate. Mcdm vs madm vs modm vs maut. `https://www.researchgate.net/post/MCDM_vs_MADM_vs_MODM_vs_MAUT`, Apr 2015. Accessed on 2020-05-15.

Decision Deck Consortium. Decision deck. `https://www.decision-deck.org/project/`, 2009a. Accessed on 2020-04-27.

National Geographic Society. Gis (geographic information system). `https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/10th-grade/`, Jun 2017. Accessed on 2020-05-02.

Alexandru-Liviu Olteanu, Patrick Meyer, and Laurent Louvart. Model (multi-criteria ordinal evaluation). `https://formations.telecom-bretagne.eu/or/software/model/index.html`, Apr 2014. Accessed on 2020-05-02.

Decision Deck Consortium. Diviz-features. `https://www.diviz.org/features.html`, 2009b. Accessed on 2020-04-29.

Sébastien Bigaret. weightedsum-r. `https://gitlab.com/XMCDA-library/XMCDA-java`, 2019a. Accessed on 2020-04-15.

Sébastien Bigaret. weightedsum-r. `https://gitlab.com/XMCDA-library/weightedSum-R/`, 2017. Accessed on 2020-04-16.

Decision Deck Consortium. Xmcda v3 current state. `https://www.decision-deck.org/xmcda/xmcda_v3_current_state.html`, 2019. Accessed on 2020-04-15.

Sébastien Bigaret. Xmcda-3.1.1.xsd. `https://gitlab.com/XMCDA-library/XMCDA-java/-/blob/master/xsd/XMCDA-3.1.1.xsd`, 2019b. Accessed on 2020-04-15.

Mario Saffirio. ¿qué son los web services? `https://msaffirio.wordpress.com/2006/02/05/C2BFque-son-los-web-services/`, May 2006. Accessed on 2020-04-14.

Oscar Blancarte. Soap vs rest ¿cual es mejor? https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/, Mar 2017. Accessed on 2020-04-14.

Miquel Buxons. Developed diviz functions. https://github.com/MikiBuxons/DevelopedDivizFunctions, May 2020. Accessed on 2020-06-11.

Jordi Canals. Intelligent decision support tools in decision deck. *repositori-urv*, 2011.

Ronald R. Yager. *On ordered weighted averaging aggregation operators in multi-criteria decision making*, pages 183–190. IEEE Transactions on Systems, Man and Cybernetics, 1988.

David Isern, Lucas Marin, Aïda Valls, Antonio Moreno, and José M. Merigó. Induced unbalanced linguistic ordered weighted average and its application in multi-person decision making. 2017.

Ronald R. Yager. Quantifier guided aggregation using owa operators. *International Journal of Intelligent Systems*, 1996.

Decision Deck Consortium. Diviz-download. https://www.diviz.org/download.html, 2009c. Accessed on 2020-06-09.

Patrick Meyer. Xmcda-r. https://gitlab.com/XMCDA-library/XMCDA-R, 2019. Accessed on 2020-04-05.

Decision Deck Consortium. Diviz-list available modules. https://www.diviz.org/wsindex-1.html, 2009d. Accessed on 2020-06-10.

Appendices

Appendix A

Description files

Into this Annex there are all module documentations files, documentation written in xml language. You can find these documents into each module folder named as "description-wsDD.xml" on my personal GitHub Buxons [2020] or, soon, it will be published on the official website of the Diviz program, in the section of list of available modules.

A.1 OWA description

```xml
<?xml version="1.0" encoding="UTF-8"?>
<program_description xsi:noNamespaceSchemaLocation="http://sma.uni.lu/d2cms/ws/_downloads/description.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <program name="OWA" provider="URV" version="3.0" displayName="OWA"/>
  <documentation>
    <description>This module implements the OWA aggregation operator: The Ordered Weighted Averaging operators, commonly called OWA operators, provide a parameterized class of mean type aggregation operators.
OWA was defined by R.R. Yager 1988. Here we provide an implementation of OWA operator. The main characteristic of this aggregation operator is the fact that the weights are associated to values instead of criteria. In this way, we can define different aggregation policies according to the importance that we associate to high and low performance scores.</description>
    <contact>Aida Valls &lt;aida.valls@urv.cat&gt;</contact>
    <reference>A. Valls, The Ordered Weighted Averaging Operator, In: Proc. IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society, Barcelona, Catalonia, 2010, pp. 3063-3070.</reference>
  </documentation>
  <parameters>
    <input displayName="criteria" name="criteria" id="input1" isoptional="0">
      <documentation>
        <description>A list of criteria. Criteria can be activated or desactivated via the &lt;active&gt; tag (true or false).
By default (no &lt;active&gt; tag), criteria are considered as active.</description>
      </documentation>
      <xmcda tag="criteria">
                        <![CDATA[
                  <criteria id="...">
                      <criterion>
                          <active>[...]</active>
                      </criterion>
                      [...]
                  </criteria>
                  ]]>
                   </xmcda>
```

```
26     </input>
27
28     <input displayName="alternatives" name="alternatives" id="input0" isoptional="0">
29       <documentation>
30         <description>A list of alternatives. Alternatives can be activated or desactivated via
               the &lt;active&gt; tag (true or false).
31 By default (no &lt;active&gt; tag), alternatives are considered as active.</description>
32       </documentation>
33       <xmcda tag="alternatives">
34         <![CDATA[
35                  <alternatives id="...">
36                         <alternative>
37                              <active>[...]</active>
38                         </alternative>
39           [...]
40             </alternatives>
41         ]]>
42       </xmcda>
43     </input>
44
45     <input displayName="performanceTable" name="performanceTable" id="input3" isoptional="0">
46       <documentation>
47       <description>The performance table will contain the all criteria values, por each
             alternative. </description>
48       </documentation>
49       <xmcda tag="performanceTable">
50
51       <![CDATA[
52       <performanceTable>
53         <alternativePerformances>
54           <alternativeID>[...]</alternativeID>
55           <performance>
56             <criterionID>[...]</criterionID>
57             <value>
58               <float>[...]</float>
59             </value>
60           </performance>
61           [...]
62         </alternativePerformances>
63         [...]
64       </performanceTable>
65       ]]>
66       </xmcda>
67     </input>
68     <input displayName="weights" name="weightsOWA" id="input2" isoptional="1">
69       <documentation>
70       <description>The weights are associated to the values of the alternatives rather than to
             the criteria. In this way they can define different aggregation policies. Assuming
             that the values on the alternative will be sorted from the best to the worst, the list
              of weights must be ordered
71 according to the importance that is given to the values, from the highest to the lowest.
72 For example a list of weights as (0.5, 0.5, 0, 0, 0) is ignoring the 3 lowest values, and making
       an average of the
73 two highest ones. A list like (0, 0, 1, 0 ,0 ) is calculating the median, while (0, 0, 0, 0, 1)
      is taking the minimum.
74 Notice that the sum of weights is required to be 1. In version 3.0 the weights have two
      structures, one to control id and allowed to inactivated the weight and the other structure
      to put the values. The weights id in the two structures must be the same and we strongly
      recomment the form "posXX" (ex. pos01, pos02, etc) to standaritzate the structure.
75 </description>
76       </documentation>
```

```
77      <xmcda tag="criteria,criteriaValues">
78      <![CDATA[
79                  <criteria>
80                      <criterion
81        <!-- REQUIRED --> id="posXX">
82                          <active>[...]</active>
83                      </criterion>
84                      [...]
85                  </criteria>
86    <criteriaValues mcdaConcept="weights">
87      <criterionValue>
88        <criterionID>posXX</criterionID>
89        <values>
90          <value>
91            <real>...</real>
92          </value>
93        </values>
94      </criterionValue>
95      [...]
96    </criteriaValues>
97      ]]>
98      </xmcda>
99    </input>
100
101   <input displayName="orness" name="programParameters" id="input4" isoptional="1">
102     <documentation>
103     <description>The weights are associated to the values of the alternatives rather than to
            the criteria. In this way they can define different aggregation policies. In this
            option, the system computes the OWA weights vector from the degree of orness indicated
             as input. The orness values must be in the continuous range from 0 to 1. Notice that
            0.0 means andness, 0.5 neutrality, 1.0 corresponds to maximum orness.
104 </description>
105     </documentation>
106     <xmcda tag="programParameters">
107     <![CDATA[
108          <programParameters>
109               <parameter !-- REQUIRED --> id="active">
110                   <values>
111          <value>
112            <boolean>...</boolean>
113          </value>
114        </values>
115               </parameter>
116               <parameter !-- REQUIRED --> id="orness">
117                   <values>
118          <value>
119            <real>...</real>
120          </value>
121        </values>
122               </parameter>
123          </programParameters>
124     ]]>
125     </xmcda>
126   </input>
127
128   <output displayName="owaResults" name="alternativesValues" id="output0">
129     <documentation>
130     <description>Result obtained from the OWA aggregation on each alternative. It will be a
            numeric value.</description>
131     </documentation>
132     <xmcda tag="alternativesValues"/>
```

```
133     </output>
134
135     <output displayName="messages" name="messages" id="output1">
136       <documentation>
137       <description>A status message.</description>
138       </documentation>
139       <xmcda tag="methodMessages"/>
140     </output>
141
142   </parameters>
143 </program_description>
```

A.2 ULOWA description

```
<?xml version="1.0" encoding="UTF-8"?>
<program_description xsi:noNamespaceSchemaLocation="http://sma.uni.lu/d2cms/ws/_downloads/
    description.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <program name="ULOWA" provider="URV" version="3.0" displayName="ULOWA"/>
  <documentation>
    <description>This module implements the ULOWA aggregation operator: Unbalanced Linguistic
        Ordered Weighted Average. Aggregation operators for linguistic variables usually assume
        uniform and symmetrical distribution of the linguistic terms that define the variable.
        However, there are some problems where an unbalanced set of linguistic terms is more
        appropriate to describe the objects. ULOWA accepts a set of linguistic labels defined
        with unbalanced fuzzy sets. The fuzzy sets must define a fuzzy partition on the set of
        reference values. They can be defined by trapezoidal or triangular membership functions.
        </description>
    <contact>Aida Valls &lt;aida.valls@urv.cat&gt;</contact>
    <reference>A. Valls, The Unbalanced Linguistic Ordered Weighted Averaging Operator, In: Proc
        . IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society,
         Barcelona, Catalonia, 2010, pp. 3063-3070.</reference>
  </documentation>
  <parameters>
    <input displayName="criteria" name="criteria" id="input1" isoptional="0">
      <documentation>
        <description>A list of criteria. Criteria can be activated or desactivated via the &lt;
            active&gt; tag (true or false).
By default (no &lt;active&gt; tag), criteria are considered as active.</description>
      </documentation>
      <xmcda tag="criteria">
                          <![CDATA[
                  <criteria id="...">
                      <criterion>
                          <active>[...]</active>
                      </criterion>
                      [...]
                  </criteria>
                  ]]>
                   </xmcda>
    </input>

    <input displayName="alternatives" name="alternatives" id="input0" isoptional="0">
      <documentation>
        <description>A list of alternatives. Alternatives can be activated or desactivated via
            the &lt;active&gt; tag (true or false).
By default (no &lt;active&gt; tag), alternatives are considered as active.</description>
      </documentation>
      <xmcda tag="alternatives">
        <![CDATA[
                 <alternatives id="...">
                        <alternative>
                             <active>[...]</active>
                        </alternative>
          [...]
            </alternatives>
        ]]>
      </xmcda>
    </input>

    <input displayName="performanceTable" name="performanceTable" id="input3" isoptional="0">
      <documentation>
      <description>The performance table will contain only string values, which correspond to
          the identifiers of the labels defined in the domain of the linguistic variable.</
```

```
        description>
47      </documentation>
48      <xmcda tag="performanceTable">
49
50      <![CDATA[
51      <performanceTable>
52        <alternativePerformances>
53          <alternativeID>[...]</alternativeID>
54          <performance>
55            <criterionID>[...]</criterionID>
56            <value>
57              <label>[...]</label>
58            </value>
59          </performance>
60          [...]
61        </alternativePerformances>
62        [...]
63      </performanceTable>
64      ]]>
65      </xmcda>
66    </input>
67
68    <input displayName="weights" name="weights" id="input2" isoptional="1">
69      <documentation>
70      <description>The weights operate in the same way as the classical OWA operator. They are
          associated to the values of the alternatives rather than to the criteria. In this way
          they can define different aggregation policies. Assuming that the values on the
          alternative will be sorted from the best to the worst (e.g. perfect, good, good,
          medium, low), the list of weights must be ordered
71 according to the importance that is given to the values, from the highest to the lowest.
72 For example a list of weights as (0.5, 0.5, 0, 0, 0) is ignoring the 3 lowest values, and making
      an average of the
73 two highest ones. A list like (0, 0, 1, 0 ,0 ) is calculating the median, while (0, 0, 0, 0, 1)
     is taking the minimum.
74 Notice that the sum of weights is required to be 1. In version 3.0 the weights have two
     structures, one to control id and allowed to inactivated the weight and the other structure
     to put the values. The weights id in the two structures must be the same and we strongly
     recomment the form "posXX" (ex. pos01, pos02, etc) to standaritzate the structure.
75 </description>
76      </documentation>
77      <xmcda tag="criteria,criteriaValues">
78      <![CDATA[
79                  <criteria>
80                      <criterion
81        <!-- REQUIRED --> id="posXX">
82                          <active>[...]</active>
83                      </criterion>
84                      [...]
85                  </criteria>
86    <criteriaValues mcdaConcept="weights">
87      <criterionValue>
88        <criterionID>posXX</criterionID>
89        <values>
90          <value>
91            <real>...</real>
92          </value>
93        </values>
94      </criterionValue>
95      [...]
96    </criteriaValues>
97      ]]>
```

```
98        </xmcda>
99      </input>
100
101     <input displayName="fuzzyNumbers" name="fuzzyNumbers" id="input4" isoptional="0">
102       <documentation>
103       <description>Definition of the fuzzy sets associated to the linguistic variable used for
              all the criteria. The semantics of the linguistic labels are given by a trapezoidal
              membership function, that is represented with a tuple of 4 values (x1,x2,x3,x4). The
              value corresponds to the abscissa point x, the coordenate y will be standaritzed in 0
              for first and four value and 1 for second and third for the program, to make the
              structure always concave. If x2 is equal to x3, the fuzzy is triangular. The values
              must be ordered increasingly. For each label in the linguistic domain (
              categoriesValues list), a fuzzy set must be defined. The labels must be ordered from
              the worst to the best performance (fi. Low, Medium, High, Perfect).</description>
104       </documentation>
105       <xmcda tag="categories,categoriesValues">
106         <![CDATA[
107       <categories>
108         <category id="..." name="...">
109           <active>true</active>
110         </category>
111         [...]
112       </categories>
113       <categoriesValues>
114         <categoryValue>
115           <categoryID>VL</categoryID>
116           <values>
117             <value>
118               <real>...</real>
119             </value>
120             [...]
121           </values>
122         </categoryValue>
123         [...]
124       </categoriesValues>
125         ]]>
126       </xmcda>
127     </input>
128
129     <input displayName="orness" name="programParameters" id="input5" isoptional="1">
130       <documentation>
131       <description>The weights are associated to the values of the alternatives rather than to
              the criteria. In this way they can define different aggregation policies. In this
              option, the system computes the OWA weights vector from the degree of orness indicated
               as input. The orness values must be in the continuous range from 0 to 1. Notice that
              0.0 means andness, 0.5 neutrality, 1.0 corresponds to maximum orness.
132 </description>
133       </documentation>
134       <xmcda tag="programParameters">
135       <![CDATA[
136             <programParameters>
137                   <parameter !-- REQUIRED --> id="active">
138                       <values>
139             <value>
140               <boolean>...</boolean>
141             </value>
142           </values>
143                   </parameter>
144                   <parameter !-- REQUIRED --> id="orness">
145                       <values>
146             <value>
```

```
147             <real>...</real>
148           </value>
149         </values>
150                 </parameter>
151           </programParameters>
152     ]]>
153     </xmcda>
154   </input>
155
156   <output displayName="ulowaResults" name="alternativesValues" id="output0">
157     <documentation>
158     <description>Result obtained from the ULOWA aggregation on each alternative. It will be a
            linguistic label belonging to the domain defined in the fuzzyNumbers input file.</
            description>
159     </documentation>
160     <xmcda tag="alternativesValues"/>
161   </output>
162
163   <output displayName="messages" name="messages" id="output1">
164     <documentation>
165     <description>A status message.</description>
166     </documentation>
167     <xmcda tag="methodMessages"/>
168   </output>
169
170  </parameters>
171 </program_description>
```

A.3 OWADescriptors description

```
<?xml version="1.0" encoding="UTF-8"?>
<program_description xsi:noNamespaceSchemaLocation="http://sma.uni.lu/d2cms/ws/_downloads/
    description.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <program name="OWADescriptors" provider="URV" version="3.0" displayName="OWADescriptors"/>
  <documentation>
    <description>Compute measures of weights given to the Ordered Weighted Average. The Ordered
        Weighted Averaging operators, commonly called OWA operators, provide a parameterized
        class of mean type aggregation operators.
For OWA wights exists different measures to characterise a set of weights associated to an OWA
    operator. In this module we implement the measures: balance, divergence, entropy and ornes.<
    /description>
    <contact>Aida Valls &lt;aida.valls@urv.cat&gt;</contact>
    <reference>A. Valls, The Ordered Weighted Averaging Operator, In: Proc. IEEE International
        Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society, Barcelona, Catalonia
        , 2010, pp. 3063-3070.</reference>
  </documentation>
  <parameters>
    <input displayName="weights" name="weightsOWA" id="input0" isoptional="0">
      <documentation>
      <description>The weights are associated to the values of the alternatives rather than to
          the criteria. In this way they can define different aggregation policies. Assuming
          that the values on the alternative will be sorted from the best to the worst, the list
           of weights must be ordered
according to the importance that is given to the values, from the highest to the lowest.
For example a list of weights as (0.5, 0.5, 0, 0, 0) is ignoring the 3 lowest values, and making
     an average of the
two highest ones. A list like (0, 0, 1, 0 ,0 ) is calculating the median, while (0, 0, 0, 0, 1)
    is taking the minimum.
Notice that the sum of weights is required to be 1. In version 3.0 the weights have two
    structures, one to control id and allowed to inactivated the weight and the other structure
    to put the values. The weights id in the two structures must be the same and we strongly
    recomment the form "posXX" (ex. pos01, pos02, etc) to standaritzate the structure.
</description>
      </documentation>
      <xmcda tag="criteria,criteriaValues">
      <![CDATA[
                  <criteria>
                      <criterion
       <!-- REQUIRED --> id="posXX">
                          <active>[...]</active>
                      </criterion>
                      [...]
                  </criteria>
    <criteriaValues mcdaConcept="weights">
      <criterionValue>
        <criterionID>posXX</criterionID>
        <values>
          <value>
            <real>...</real>
          </value>
        </values>
      </criterionValue>
      [...]
    </criteriaValues>
      ]]>
      </xmcda>
    </input>

    <output displayName="balance" name="balance" id="output0">
```

```
45      <documentation>
46      <description>Result obtained apply balance calculation in weights table. It will be a
            single numeric value.</description>
47      </documentation>
48      <xmcda tag="criteriaValues"/>
49    </output>
50
51    <output displayName="divergence" name="divergence" id="output1">
52      <documentation>
53      <description>Result obtained apply divergence calculation in weights table. It will be a
            single numeric value.</description>
54      </documentation>
55      <xmcda tag="alternativesValues"/>
56    </output>
57
58    <output displayName="entropy" name="entropy" id="output2">
59      <documentation>
60      <description>Result obtained apply entropy calculation in weights table. It will be a
            single numeric value.</description>
61      </documentation>
62      <xmcda tag="alternativesValues"/>
63    </output>
64
65    <output displayName="ornes" name="ornes" id="output3">
66      <documentation>
67      <description>Result obtained apply ornes calculation in weights table. It will be a single
             numeric value.</description>
68      </documentation>
69      <xmcda tag="alternativesValues"/>
70    </output>
71
72    <output displayName="messages" name="messages" id="output4">
73      <documentation>
74      <description>A status message.</description>
75      </documentation>
76      <xmcda tag="methodMessages"/>
77    </output>
78
79  </parameters>
80 </program_description>
```

A.4 Defuzzication description

```
<?xml version="1.0" encoding="UTF-8"?>
<program_description xsi:noNamespaceSchemaLocation="http://sma.uni.lu/d2cms/ws/_downloads/
    description.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <program name="Defuzzification" provider="URV" version="3.0" displayName="Defuzzification"/>
  <documentation>
    <description>This module implements three different methods to defuzzificate the result of a
         ULOWA operation (Unbalanced Linguistic Ordered Weighted Average). Aggregation operators
         for linguistic variables usually assume uniform and symmetrical distribution of the
        linguistic terms that define the variable. However, there are some problems where an
        unbalanced set of linguistic terms is more appropriate to describe the objects. ULOWA
        accepts a set of linguistic labels defined with unbalanced fuzzy sets. The fuzzy sets
        must define a fuzzy partition on the set of reference values. They can be defined by
        trapezoidal or triangular membership functions.
For this method we apply three different operations for every ULOWA alternative result: COG, COM
     and Ordinal.
    </description>
    <contact>Aida Valls &lt;aida.valls@urv.cat&gt;</contact>
    <reference>A. Valls, The Unbalanced Linguistic Ordered Weighted Averaging Operator, In: Proc
        . IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society,
         Barcelona, Catalonia, 2010, pp. 3063-3070.</reference>
  </documentation>
  <parameters>
    <input displayName="linguistic scores" name="alternativesValues" id="input0" isoptional="0">
      <documentation>
        <description>A list of alternativesValue. Normally it will be the result of the ULOWA or
            other linguistic aggregation operation.</description>
      </documentation>
      <xmcda tag="alternativesValues">
        <![CDATA[
        <alternativesValues>
          <alternativeValue>
          <alternativeID>...</alternativeID>
          <values>
            <value>
              <label>...</label>
            </value>
          </values>
          </alternativeValue>
          [...]
        </alternativesValues>
        ]]>
      </xmcda>
    </input>
    <input displayName="fuzzyNumbers" name="fuzzyNumbers" id="input1" isoptional="0">
      <documentation>
      <description>Definition of the fuzzy sets associated to the linguistic variable used for
          all the criteria. The semantics of the linguistic labels are given by a trapezoidal
          membership function, that is represented with a tuple of 4 values (x1,x2,x3,x4). The
          value corresponds to the abscissa point x, the coordenate y will be standaritzed in 0
          for first and four value and 1 for second and third for the program, to make the
          structure always concave. If x2 is equal to x3, the fuzzy is triangular. The values
          must be ordered increasingly. For each label in the linguistic domain (
          categoriesValues list), a fuzzy set must be defined. The labels must be ordered from
          the worst to the best performance (fi. Low, Medium, High, Perfect).</description>
      </documentation>
      <xmcda tag="categories,categoriesValues">
        <![CDATA[
      <categories>
        <category id="..." name="...">
```

```
40          <active>true</active>
41        </category>
42        [...]
43      </categories>
44      <categoriesValues>
45        <categoryValue>
46          <categoryID>VL</categoryID>
47          <values>
48            <value>
49              <real>...</real>
50            </value>
51            [...]
52          </values>
53        </categoryValue>
54        [...]
55      </categoriesValues>
56        ]]>
57      </xmcda>
58    </input>
59
60    <output displayName="COG" name="defuzzificationCOG" id="output0">
61      <documentation>
62      <description>Result obtained from apply COG calculation for each alternative and their
            fuzzy label result.</description>
63      </documentation>
64      <xmcda tag="alternativesValues"/>
65    </output>
66    <output displayName="COM" name="defuzzificationCOM" id="output1">
67      <documentation>
68      <description>Result obtained from apply COM calculation for each alternative and their
            fuzzy label result.</description>
69      </documentation>
70      <xmcda tag="alternativesValues"/>
71    </output>
72    <output displayName="Ordinal" name="defuzzificationORD" id="output2">
73      <documentation>
74      <description>Result obtained from apply Ordinal calculation for each alternative and their
             fuzzy label result.</description>
75      </documentation>
76      <xmcda tag="alternativesValues"/>
77    </output>
78
79    <output displayName="messages" name="messages" id="output3">
80      <documentation>
81      <description>A status message.</description>
82      </documentation>
83      <xmcda tag="methodMessages"/>
84    </output>
85
86  </parameters>
87 </program_description>
```

A.5 fuzzyLabelsDescriptors description

```xml
<?xml version="1.0" encoding="UTF-8"?>
<program_description xsi:noNamespaceSchemaLocation="http://sma.uni.lu/d2cms/ws/_downloads/description.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <program name="FuzzyLabelsDescriptors" provider="URV" version="3.0" displayName="FuzzyLabelsDescriptors"/>
  <documentation>
    <description>Two types of uncertainty in fuzzy sets are recognized: (1) specificity, related to the measurement of imprecision, which is based on the cardinality of the set, and (2) fuzziness, or entropy, which measures the vagueness of the set as a result of having imprecise boundaries. Specificity and fuzziness refer to two different characteristics of fuzzy sets. Specificity (or its counterpart, non-specificity) measures the degree of truth of the sentence: Containing just one element. Fuzziness measures the difference from a crisp set.</description>
    <contact>Aida Valls &lt;aida.valls@urv.cat&gt;</contact>
    <reference>A. Valls, The Unbalanced Linguistic Ordered Weighted Averaging Operator, In: Proc. IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society, Barcelona, Catalonia, 2010, pp. 3063-3070.</reference>
  </documentation>
  <parameters>
    <input displayName="fuzzyNumbers" name="fuzzyNumbers" id="input0" isoptional="0">
      <documentation>
      <description>Definition of the fuzzy sets associated to the linguistic variable used for all the criteria. The semantics of the linguistic labels are given by a trapezoidal membership function, that is represented with a tuple of 4 values (x1,x2,x3,x4). The value corresponds to the abscissa point x, the coordenate y will be standaritzed in 0 for first and four value and 1 for second and third for the program, to make the structure always concave. If x2 is equal to x3, the fuzzy is triangular. The values must be ordered increasingly. For each label in the linguistic domain (categoriesValues list), a fuzzy set must be defined. The labels must be ordered from the worst to the best performance (fi. Low, Medium, High, Perfect).</description>
      </documentation>
      <xmcda tag="categories,categoriesValues">
        <![CDATA[
      <categories>
        <category id="..." name="...">
          <active>true</active>
        </category>
        [...]
      </categories>
      <categoriesValues>
        <categoryValue>
          <categoryID>VL</categoryID>
          <values>
            <value>
              <real>...</real>
            </value>
            [...]
          </values>
        </categoryValue>
        [...]
      </categoriesValues>
        ]]>
      </xmcda>
    </input>

    <output displayName="specificity" name="specificity" id="output0">
      <documentation>
      <description>Result obtained from apply Specificity calculation for each fuzzy Number from the list.</description>
```

```
41      </documentation>
42      <xmcda tag="alternativesValues"/>
43    </output>
44    <output displayName="fuzziness" name="fuzziness" id="output1">
45      <documentation>
46      <description>Result obtained from apply Fuzziness calculation for each fuzzy Number from
            the list.</description>
47      </documentation>
48      <xmcda tag="alternativesValues"/>
49    </output>
50
51    <output displayName="messages" name="messages" id="output2">
52      <documentation>
53      <description>A status message.</description>
54      </documentation>
55      <xmcda tag="methodMessages"/>
56    </output>
57
58  </parameters>
59 </program_description>
```