A Framework to Explore MCDA Methods

Cristina Isabel Alves Verdasca

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. José Luís Brinquete Borbinha

Examination Committee
Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. José Luís Brinquete Borbinha
Member of the Committee: Prof. José Rui de Matos Figueira

November 2016
Acknowledgments

Firstly, I would like to thank Professor José Borbinha for the opportunity to carry out this work, as well as for his guidance during this year and its course. I would like to express a special thanks to Ana Sara Costa and Professor José Rui Figueira for all the help, support and effort.

Secondly, I would like to show gratitude to António Higgs and Carlos Martins for their constant help and advice given.

Last but not least, I would like to express my sincere gratitude to all my family and friends for the motivation, support and encouragement during the year. Without them none of these accomplishments would ever happen.
Resumo

A utilização de frameworks MCDA pode ser difícil para utilizadores que não são especialistas em métodos MCDA. Não há muitas frameworks que se concentram na usabilidade e user experience. Em consequência, as frameworks podem ser difíceis de usar, difíceis de perceber e os utilizadores necessitam de experiência na framework para reconhecer as funcionalidades disponíveis. Além disso, algumas frameworks não fornecem um grande número de métodos MCDA disponíveis para uso e não têm a possibilidade de alargar esse número. Isto pode ser considerado uma oportunidade perdida para o aumento da variabilidade de métodos MCDA disponíveis e crescimento contínuo.

O objectivo principal deste trabalho foi o desenho e desenvolvimento de uma nova framework MCDA focando-se especialmente na usabilidade e user experience para os utilizadores (com conhecimentos sobre MCDA) usarem, mas ter a possibilidade de adicionar novos métodos MCDA na framework por programadores. Algumas frameworks MCDA actuais foram analisadas para perceber os principais problemas e aspectos da falta de facilidade de utilização. Mas também, para obter ideias e reutilizar aspectos específicos de outras frameworks que são considerados como úteis. Foi desenvolvida uma nova web framework com uma interface simples que explora métodos MCDA e a framework pode ser estendida por programadores. A framework proposta foi avaliada por utilizadores para obter feedback da solução e ver se a framework demonstrou uma boa usabilidade e user experience. Após a avaliação positiva, foram feitas algumas melhorias o que gerou a segunda versão da nova framework MCDA.

Palavras-chave: MCDA, frameworks, métodos, usabilidade, user experience, adicionar.
Abstract

The use of MCDA frameworks can be difficult for users who are not experts in the MCDA methods. There are not many frameworks that focus in usability or user experience. In consequence, the frameworks can be difficult to use, not understandable and users require experience to acknowledge the available functionalities. In addition, some frameworks do not provide a large number of available MCDA methods to use and do not have the possibility to extend its number. It can be considered a lost opportunity to increase the variety of MCDA methods available and continuous growth.

The main objective of this work was the design and development of a new MCDA framework focusing in usability and user experience for users (with some MCDA knowledge) to use, but also have the possibility to add new methods to the framework by developers. Some MCDA frameworks were analysed in order to understand the main problems and aspects of the lack of user friendliness. Furthermore, to obtain ideas and reuse specific aspects from other frameworks which are considered to be useful. It was developed a new web-based integrated framework with a simple interface that explores MCDA methods and the framework can be extended by developers. The proposed framework was evaluated by users to obtain feedback of the solution and see if the framework presented a good usability and user experience. After the positive evaluation, it was made some improvements which generated the second version of the new MCDA framework.

Keywords: MCDA, frameworks, methods, usability, user experience, extend.
## Contents

Acknowledgments ........................................................................................................ iii
Resumo ........................................................................................................................ iv
Abstract ....................................................................................................................... v
List of Tables .............................................................................................................. vii
List of Figures ............................................................................................................. viii
Glossary ....................................................................................................................... ix

1 Introduction ........................................................................................................... 2
   1.1 Problem Description ...................................................................................... 2
   1.2 Motivation and Objectives ........................................................................... 3
   1.3 Contributions ................................................................................................. 3
   1.4 Document Structure ..................................................................................... 4

2 Related Work ......................................................................................................... 5
   2.1 Overview of MCDA methods ....................................................................... 5
   2.2 Relevant Frameworks .................................................................................. 6
      2.2.1 MCDA-Ulaval ...................................................................................... 6
      2.2.2 SRF ..................................................................................................... 9
      2.2.3 Diviz .................................................................................................. 10
   2.3 Summary ....................................................................................................... 11

3 Problem Analysis and Solution Design ................................................................ 13
   3.1 Requirements ............................................................................................... 13
      3.1.1 Front-end Use Cases .......................................................................... 14
      3.1.2 Back-end Use Cases .......................................................................... 16
   3.2 Framework Architecture and Technology ................................................. 17
   3.3 Summary ....................................................................................................... 19

4 Solution Implementation ....................................................................................... 20
   4.1 Main Pages Section .................................................................................... 20
      4.1.1 Registration and Login ....................................................................... 21
   4.2 Projects Management Section ...................................................................... 22
List of Tables

6.1 Answers from usability question 1: medians and modes. .............................................. 39
6.2 Answers from usability questions 2, 3 and 4: averages, medians and modes. ............. 39
6.3 Answers from user experience questions from 5 to 11: averages, medians and modes. . 41

E.1 Users evaluation table: time execution and answers from usability question 1. ............ 88
E.2 Users evaluation table: answers from usability questions 2, 3 and 4. .......................... 89
E.3 Users evaluation table: answers from user experience questions 5 to 11. ...................... 90
List of Figures

2.1 MCDA-Ulaval user interface. .................................................. 7
2.2 Some MCDA-Ulaval configurations. ....................................... 8
2.3 SRF user interface results. .................................................. 9
2.4 Diviz user interface results. ................................................ 10
3.1 Use Case: Front-end. ......................................................... 14
3.2 Use Case: Back-end. ......................................................... 16
3.3 Architecture of the MCDA Framework. ................................... 18
4.1 One of the home pages. ....................................................... 21
4.2 Registration page and Login Page. ........................................ 21
4.3 Projects management page. ................................................ 22
4.4 Description section. ......................................................... 23
4.5 Import section: import project’s current data. ......................... 24
4.6 Export section: export project’s current data. ......................... 25
4.7 Example of a data section: Data List. .................................... 26
4.8 Example of a data section: Parameter and Data Table. .............. 26
4.9 SRF - Determining weights of criteria method. ....................... 27
4.10 Execution and results section: Execution and project’s current data. 28
4.11 Execution and results section: How the results are displayed. .... 29
5.1 Example of an export results function from the Order By method (example). 33
D.1 Survey introduction. ..................................................... 81
D.2 Survey: question 1. ....................................................... 82
D.3 Survey: questions from 2 to 3.1. ....................................... 83
D.4 Survey: questions from 4 to 5.1. ....................................... 84
D.5 Survey: questions from 6 to 8.1. ....................................... 85
D.6 Survey: questions from 9 to 10.1. ..................................... 86
D.7 Survey: questions 11 and 11.1. ......................................... 86
D.8 Survey: questions from 12 to 14. ...................................... 87
<table>
<thead>
<tr>
<th>Glossary Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUD</td>
<td>Create, Read, Update and Delete are the four basic functions of persistent storage.</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language.</td>
</tr>
<tr>
<td>CSV</td>
<td>A Comma-Separated Values file that stores tabular data (numbers and text) in plain text.</td>
</tr>
<tr>
<td>ELECTRE</td>
<td>ELimination and Choice Expressing REality is a family of MCDA methods.</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language is the standard markup language used to create web pages.</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript is a high-level, dynamic, untyped, and interpreted programming language.</td>
</tr>
<tr>
<td>MCDA</td>
<td>Multiple-Criteria Decision Analysis is a sub-discipline of operations research that explicitly considers multiple criteria in decision-making environments.</td>
</tr>
<tr>
<td>UEQ</td>
<td>User Experience Questionnaire is a questionnaire that measures usability aspects (efficiency, perspicuity, dependability) and user experience aspects (originality, stimulation).</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The use of MCDA frameworks can be difficult for users who are not experts in the MCDA methods. Flexibility, usability and simplicity are the main attractions for a good and useful framework for real-time projects and research. However, there are not many frameworks who focus in these main attractions or have a great variety of MCDA methods available. Also, some frameworks cannot be extended (i.e. cannot add new methods). It would be interesting and useful to study and implement a framework that would do all of these details and more. Basically, developing a framework that focus specially in the user experience, usability and the ease of adding new MCDA methods as main goals. Furthermore, exploring ideas in functionalities that would be essential and helpful for such methods and for users to have.

The MCDA Framework is a web-based integrated framework that explores MCDA methods and other functionalities for users (who have at least some MCDA knowledge) to use. Its essential purpose is to focus in usability and user experience, as well as the extension of new methods at anytime. But also, overcome some negative features and difficulties presented in other current MCDA frameworks. In addition, some ideas from other frameworks are reused or integrated in order to take full advantage of their potential which may improve the new framework and might be interesting for users.

This dissertation is meant to provide the reader with a solid understanding of the inner study and work of the new MCDA framework. It is important to note that after analysing the results from the evaluations made using the first version of the framework that the framework was updated. Some changes were made, since they were considered relevant and also to correct the usability issues found. The final framework described in this dissertation is the updated version of the MCDA Framework (i.e. the second version).

1.1 Problem Description

The variety of frameworks available using MCDA methods is large and diverse which allow users to explore and use any MCDA method for their own goals and purposes. However, there are not many frameworks that provide many types of MCDA methods or do not have the possibility to add new fresh methods to the framework. Also, some frameworks have poor usability and user experience. Fur-
thermore, some of these frameworks do not provide enough information about the methods and how can they be executed successfully. Some of them are hard to understand at first and are not easy to use (require experience to get used to or tutorials are needed). In consequence, users may not use the framework often or they might try to find another framework that suits them. These frameworks, specially the MCDA methods, could be explored more in many different domains, because they might improve the results and usefulness in real-time projects and researches.

1.2 Motivation and Objectives

There are not many MCDA frameworks with great usability and user experience that have a large list of MCDA methods available. Likewise, having the possibility to add new methods into the framework at anytime.

An useful and well developed web-based framework should be simple and easy as possible to use. It should also be auto-explanatory and have a consistent interface. With these statements, it would be interesting and advantageous to study and implement a framework that would respect all of these points. As well as be accessible by any user, regardless the level of knowledge and experience in these types of frameworks. Furthermore, the exploration of new ideas, features and aspects to improve the usage of the methods and its interface. But also, the variety of functionalities and facility in using the framework.

The objective was the development of a simple and flexible web-based integrated framework that explores MCDA methods to be used by MCDA expert users (or any user who has at least some knowledge about MCDA). The main goals focus specially in usability and user experience for a better management and quality of using these types of methods. But also, developing an easy way for developers to extend the list of methods available by adding new ones at anytime (while maintaining the interface consistency).

Accordingly, the final framework must be fast, efficient, and easy to use. It must offer useful features without overwhelming the users with options. The user interface must be intuitive and have little or no learning curve. Furthermore, the addition of new methods should be as generic as possible and easy to do from the developers side.

1.3 Contributions

The final solution provides three methods as examples (one MCDA method and two basic methods) in order to evaluate the developed framework. These methods allow to present the main user interface and available functionalities. Furthermore, the main objective of these two basic methods is to explain to developers how to extend the framework (the Administrative Manual describes this in detail and can be seen in Appendix B).

The contribution resulting from this work is a web framework that can make it possible to explore multiple MCDA methods by offering:

- An intuitive user interface for the use of MCDA methods by non-expert users (assuming they have
only a generic knowledge about the method);

- The possibility to extend the current number of MCDA methods by developers using a generic template and according to the method requirements;

- Project management functionalities;

- Visualization of all the results obtained and respective management (allowing to compare results with different data);

- Integrated methods from other existing MCDA frameworks and remotely reuse some of their methods.

1.4 Document Structure

The rest of this dissertation is organized in six chapters and five appendixes.

The next chapter presents some related work to our problem, which includes an overview of MCDA methods and the analysis of three MCDA frameworks (MCDA-Ulaval, SRF and Diviz).

Chapter 3 states the problem analysis and design made, like requirements, use cases, architecture and technology used. Chapter 4, describes the framework's implementation and functionalities in great detail. Then, chapter 5 explains the extension of new methods and the necessary requirements to do it correctly.

Chapter 6 describes the evaluation method, as well as the results of the user testing and the conclusions made. The last chapter concludes the dissertation with an overall discussion of the work and proposed future work.

Finally, in the appendixes we have the administrative and user manuals of the framework, and also the guidelines, survey and results presented in the evaluation chapter.
Chapter 2

Related Work

The variety of frameworks available using MCDA methods is large which allow users to explore any MCDA method for their own goals (in some cases, specific frameworks are not easily available for everyone). Although these frameworks aim to achieve a common generic goal (i.e. to use MCDA methods), some of them are considered to be more appropriate than others.

For example, to use some specific frameworks users must have some experience in these types of frameworks along with some knowledge about the methods available. These frameworks may not be user friendly by lacking usability and user experience. On the other hand, only few frameworks focus in presenting simpler interfaces, targeting a broader user group in which untrained casual users can be included. As well as having a huge list of MCDA methods available and the possibility to be extended by developers.

2.1 Overview of MCDA methods

Multi-Criteria Decision Analysis (MCDA) aims at helping a decision-maker or a group of decision-makers during the decision aiding process, which usually is assisted by one or more decision analysts [1]. In short, the basic idea of using a MCDA method is to assess the performance of actions according to multiple criteria, taking into account the preferences of the decision-maker(s) in the decision process. All actions are simultaneously evaluated under a set of relevant, and typically heterogeneous and conflicting criteria. Furthermore, the actions must be evaluated on a coherent family of criteria and consistent methods [2]. A generic decision aiding process consists of four basic steps:

- Problem situation;
- Problem formulation;
- Evaluation Model;
- Final recommendation [3].

For example, a simple personal choice such as choosing a new car to buy. As main criteria we may consider cost, consumption, safety, and power as examples and have a list of cars that we are
interested in (i.e. alternatives). It is unusual that the cheapest car from the list is the most powerful and the safest one, but we have to keep in mind that our preferences are taking into account in the decision process. We are interested in getting high returns, but at the same time reducing our risks. This is a simple example and does not need much thinking, because there’s not much information to consider or consequences. However, it is useful to understand what are the main ideas and objectives of the MCDA methods and where to use them.

These methods would be great use in situations with different points of view and complex information that requires multiple expertise, so these methods can reach one of the best possible results. Structuring complex problems well and considering multiple criteria explicitly leads to more knowledgeable and greater decisions. Furthermore, the MCDA draws upon knowledge in different domains like mathematics, economics and software engineering.

The MCDA methods have numerous benefits not only regarding the achieve of an ideal transparent and consistent decisions, but also: helps structuring the problem; makes decision-maker(s) learn about the problem situation and also about their own and others values or judgments; makes the information presentable and easy to identify; and the process provides a review for a decision [4].

2.2 Relevant Frameworks

The following three frameworks were relevant for studying and gaining knowledge about the world of MCDA frameworks before developing a new framework. All of them are diverse in many aspects and have different goals which contribute for a better understanding and interpretation about the various MCDA frameworks out there.

The purpose was to deeply analyse these frameworks and learn about their advantages, disadvantages and points that could be improved. Furthermore, what could be useful to keep or be reused in other terms and what new functionalities would be interesting for the MCDA frameworks to have.

2.2.1 MCDA-Ulaval

The MCDA-Ulaval\(^1\) is a free tool programmed in Java that supports MCDA methods from the ELECTRE family algorithms. The figure 2.1 presents the main user interface and some features. It can be observed that the user interface is simple and basic in terms of visuals. It is also well organized by showing a list of current projects and their corresponding data in the left vertical bar, making the rest of the area for workspace.

The tool allows users to create, edit and delete projects plus their data like an excel table (sections from figure 2.2 display the different types of data and how to configure them, still presenting a simple interface like adding a category in (b)). It is even possible to change what MCDA method to use at anytime. Additionally, users can import and export data in a CSV format which is handy for editing and reusing them in other projects.

\(^1\)MCDA-Ulaval official page, http://cersvr1.fsa.ulaval.ca/mcda/?q=en
Each project can have multiple sets of the data (e.g. have two lists of criteria each one with different content). This last aspect is useful and interesting, because the user can choose which list of criteria to use in the method. The user obtains the results from a specific list and can do the same with the other lists. This allows the comparison or decision of which criteria are better for the problem and review all the results. The tool also has some extra functionalities like the option to analyse the scenario or stability of the project and can generate diverse types of graphs and charts for the data.

However, the tool itself and some of its functionalities present some flaws which can lead to a poor user experience and usability. The users might require some experience when using the framework for the first time.

It is not stated where to start and the users have to explore for themselves to find out or read the manual previously. In consequence, the user experience may not be positive at the beginning because the tool is not auto-explanatory and is not intuitive enough. Furthermore, there is lack of support to clarify the user what is going on and what is needed in order to execute the method successfully. A simple tutorial containing the steps and available functions could be added, so the user could take full advantage of the tool and display which steps are already done.

Another detail to mention is the sections visualization, because when a section is open and is not closed, it may lead to confusion after a large number of sections being displayed. This may lead to an issue where the user changes data in the wrong section (or project) without knowing it and has to be aware where he is.

The import function works well, but in the configuration section (see figure 2.2) it’s not possible to understand what can be imported, if everything or only tables (like in figure 2.2 (b)). Also, an user cannot import data (i.e. criteria, alternatives and performance table) individually, it fully imports the performance table and only the names of the criteria and alternatives are imported (the user has to fill the remaining fields from these two manually).

Although it is possible to export the data, the tool is not consistent and does not integrate well the
export function. For example, to export the data from a performance table the user just needs to right click the respective section and the export option appears (exports directly after clicking it). However, for some data it can only be exported after executing the method and the user needs to select the data from the results section. Only then it is possible to export after clicking on the "Export CSV file" button.

These two functions could be simplified and with similar features. There could be a generic import and export button in the tool, that would be visible to the user and when clicking it the user could choose which data he wants to import or export in the project. It would be useful if they could explain what went wrong or what was done while importing (i.e. if a file was rejected or the import was successful).

Finally, one of the interesting aspects of the tool was to be able to have different sets of the same data and receive the results for each case. After the execution of the method the user cannot change the data and cannot execute it again. The user has to create a new set of data (add the desired values) and execute or the user simply deletes all the results so that the data can be editable again. In consequence, some time is lost and the previous results executed cannot be seen if the data changes. It would be convenient and advantageous to save all the executed results and the data used, in order to compare results. As well as experimenting different combinations of data without the need to create new sets and the possibility to get the data from previous results at any moment.

One small note, if the tool was web-based it would favor the portability aspect so more people could use the tool anywhere without concerns. It would be great if the list of MCDA methods available could be extended.

Overall, the tool has useful features (like the project management aspect, the multiple data sets and the possibility to change method) and does what it suppose to do. However, it is more favorable for users who have experience and are familiar with these types of tools. Likewise, the tool could have a bigger number of available MCDA methods or have the opportunity to extend it.
SRF is a software, coded with Borland Delphi 3, that is used to determine the weights of criteria in the ELECTRE family algorithms and its an implementation of the revised Simos’ procedure [5].

The user interface of the software is “old-school” (i.e. classic 90’s software, see figure 2.3) that follows several steps to achieve the final results and it is written in French. It shares some functionalities from MCDA-Ulaval, like a user can save his projects, it can generate graphs of the data results, and also it is not much user friendly. The main purpose of this software was to help and improve the calculation of the weights of criteria that will be used in ELECTRE methods and other MCDA methods. Furthermore, a software that could be used by any user, has different sets in how to rank the criteria and receive the different results. Plus makes the users think about the criteria whatever their complexity.

![Figure 2.3: SRF user interface: results.](image)

The disadvantages showed in the software are quite noticeable when a user first uses it: although the interface is simple, it is quite hard to use or understand how it works at first even if the users follow the steps; it is not visible where and how to change the values; and is not possible to export the results (only printing them). If all the functionalities were a little more noticeable and had functions like importing and exporting, the software could have had more positive user experience and be more flexible.

In general, despite the fact that the software only executes one method (that does not belong to the MCDA methods) and presents a poor user interface, it is interesting and quite useful to understand the criteria chosen and receive the respective weights. This method should be available as an auxiliary method for MCDA methods that require the weights of criteria, because it would help the users to think about the criteria and review them. But also, it could improve the decisions results at the end. A small note, some positive and negative points to be considered about the software are already mentioned in MCDA-Ulaval (like the project management, showing graphs and no possible extension of methods).
2.2.3 Diviz

Diviz\(^2\) is a software tool for designing, executing and sharing MCDA methods, algorithms and experiments [6]. It is an initiative within the Decision Deck project\(^3\) which aims at collaboratively develop open source software tools implementing MCDA techniques.

The tool presents a simple interface and was made to help teachers present the MCDA methods to the students (which they would use), but also to help researches on building MCDA methods. The main idea is for users to simply arrange a workflow of methods and data by making the experiment or scenario he wants to execute. In figure 2.4, it shows an example of a method workflow (the several green and blue boxes) and the user interface of the tool.

At first, the user interface reminds the MCDA-Ulaval tool with a few differences like: the projects show all their executions made (successfully and unsuccessfully) instead of the data; it displays all the available methods in the right vertical bar; and below the workflows it shows a section to view the data files and results content with different types of format views. The same goes for the functionalities, both tools have some in common like managing projects, the possibility to get different results by experimenting various data and be able to compare them at the end.

![Diviz user interface](image)

Figure 2.4: Diviz user interface.

All available methods provided by Diviz are currently open source web services, which allows anyone who is connected to the Internet to have access to a large amount of MCDA methods without having to install them on their personal computer [6]. This also offers the opportunity to extend the list of methods, not only by developers, but also by any user that requests his own method. If everything is in order (i.e. the necessary requirements to add the method are well structured and complete), the method is accepted and it is added to the tool as another available web-service.

Despite all of these qualities, there are still some flaws present and some points that could improve

---

\(^3\)Decision Deck project, http://www.decision-deck.org/project/index.html
the tool. The main issue is the lack of user experience, not only it is difficult for users (who are aware of the MCDA) to understand what each method does, but it is also required some experience to get used to. For the first time, users may need to read the tutorials and explanations available at the Diviz official website.

Additionally, the process to insert data for the methods is not practical and might be frustrating for users. Each type of data (e.g. criteria, alternatives, etc...) must be inside in a XML file and must respect the structured design by the tool. If the structure is incorrect the method execution will result in failure and the users must check and might not understand what went wrong. The tool provides feedback in what went wrong, but some types of errors should be warned before executing the method.

However, Diviz provides methods that convert files into the right XML file format, but this process must be included in the final workflow and the users must know that these methods exist. It is recommended that users know the available methods first to know which ones are useful and what they need to create their workflows. In addition, it's not clear at first where to change or define the parameters for a specific method and respective description. The user must double-click on the method box to display these types of information.

Finally, the import and export functions could be explored more. It should be possible to import/export in other formats than the tool's format type (.dvz) or XML format. Perhaps it would be a good idea to import data from a CSV file and generate the respective XML file for the method instead of putting another method into the workflow. This would save time and be more efficient from the user side, because it would not be required to learn the file structure or add the required method to convert it every time.

Overall, the tool has a great manner to build a problem, execute the method and present the results (i.e. the workflow process). While tagging along a large number of methods available which can be extended. Having the possibility to increase the number of methods by anyone is a huge step and an useful one, because users who create new methods do not need to create a new framework to use the method. But also, it is an easy way to share methods (i.e. open source web services) and all the methods are in one place which makes the tool grow continuously. This makes a great idea to explore and find out an easier form to add the new methods. Furthermore, the open source MCDA methods available can be reused in other frameworks which might be an interesting advantage for future MCDA frameworks.

2.3 Summary

This chapter presented a comprehensive analysis of some MCDA frameworks and the understanding of several advantages and disadvantages to be taken seriously regarding the development of a new MCDA framework. Some good points and functionalities are common to have and/or are interesting ideas (i.e. import/export data and results, extend the number of methods available, be able to test different data combinations and save previous results for comparison). Although some of them should be explored more to receive a better performance and power up the framework. Or be improved for a
better consistency and usability aspects.

Additionally, there are some interesting methods observed, like the determination of weights of criteria executed by the SRF software and some web-services methods provided by Diviz. These methods might be useful as auxiliary methods or might be reused as main methods. They will help the execution of some MCDA methods and might improve the framework’s performance and usability.

On the other hand, the user experience is low and average in these frameworks, as well as, the platform selection limits the portability and the process to extend in some frameworks. Sometimes they are difficulty to understand and use at first, but also are not auto-explanatory enough. The frameworks are not intuitive and users may require some training first. These aspects should be taken carefully and should be considered a better user friendly framework for any user, but be self-explanatory and simple as possible.

This analysis and observations made will be considered during the rest of the work and development of the proper solution in order to achieve an useful and simple MCDA framework.
Chapter 3

Problem Analysis and Solution Design

As a preliminary study, three different types of MCDA frameworks were analysed carefully. Their strongest and weakest points were deeply studied in order to understand what a MCDA framework should have and not have in terms of usability and performance. Furthermore, what would be interesting to improve and what functionalities to keep and consider useful when designing a new MCDA framework.

This chapter describes the analysis and discussions made to elaborate and justify the decisions taken for the final solution of a new MCDA framework. This framework explores MCDA methods and can be used by any user (with at least some MCDA knowledge) without having a complex interface. Various aspects are observed and compared with other previous frameworks, like in terms of portability and easy access. It was decided to develop a web-based MCDA framework which allows users to access the framework anywhere using a computer with Internet access. Likewise there are not many web-based MCDA frameworks out there.

Regarding the user interface, it was decided to design a simple and consistent form, but following the ten usability heuristics for user interface design [7] and main principles [8]. As for available functionalities, it was important to decide which functions are required and are most appreciated by users to have. While comparing and examining the ones available in other existing MCDA frameworks. Also, when adding new methods to the framework, trying to figure a way to build a generic and simple guide for the new additions. It should follow the method requirements and functions, but keeping the same interface and main functionalities of the framework for any method.

Finally, after considering the different possibilities for the development of a solution taking in consideration the main objectives and available resources, it was defined the desired characteristics for a final solution design.

3.1 Requirements

The main problems and disadvantages featured in previous MCDA frameworks were the lack of user experience, not providing enough information and guidance regarding the MCDA methods. Furthermore,
the small variety of available functions and methods, plus the difficulty to use the framework for the first time. It would be recommended to gain experience first in these types of frameworks or read the respective manual or tutorials. Furthermore, there are not many frameworks that allow the addition of new MCDA methods. All these issues together might make the framework difficult to use which requires learning or experience to get used to.

A good MCDA framework should be as flexible as possible and auto-explanatory, by being user friendly and easy to use for everyone to enjoy and adopt. It is important to have everything consistent and make every function visible for the users [7]. By considering all of these aspects the user experience will increase and any user (that has at least some knowledge regarding the MCDA) can use it, understand what is going on and take full advantage of the framework itself. Another aspect to increase the framework’s attractiveness would be its portability. By developing a web-based framework would make it accessible by using a computer with Internet access and most likely be taken anywhere.

The following use cases (figures 3.1 and 3.2) present the main requirements for the new MCDA framework (both front-end and back-end sides) and respective actors.

### 3.1.1 Front-end Use Cases

The use case presented below (figure 3.1) displays the final requirements for the front-end side of the new MCDA framework.

![Figure 3.1: Use Case: Front-end.](image)

Regarding the actors, an anonymous actor is not an authenticate user and can only create an account...
and login (if an account was created). The other actors have to authenticate, but even authenticated users have different privileges. An administrator manages the front pages of the framework and also the users accounts. An user is the one who interacts completely with the framework’s capabilities and main functionalities.

First, providing project management features offers the possibility to an user to manage his own projects and resume its work where it was left on. So an user can create a project by giving it a name and choosing which method it will execute. The user can also duplicate an existing project (i.e. create a copy of an existing project that can be used separately from the original as a new project). Duplicating a project is an easy way to reuse an old project and respective method data or take advantage of what was made already.

After a project is created the user can now edit it, in other words, the user can delete the project or edit some of the project’s fields (e.g. its name). Meanwhile, after opening a project, the user first views the method’s steps (i.e. tutorial) that are required to execute the method successfully (the steps will be available and help the users understand the methods). The project is configured by the user (following his needs) and the respective method data. An user can edit the method data manually (i.e. create, edit and delete each method data element manually according to its type and/or select the values provided for it). Or simply import the method data by selecting which file set of method data to import which depends on the method chosen.

Furthermore, an extra option that is useful for users is the reloading function for the method data. With this function, an user can change the current method data into the ones that were used on a specific execution showed in the respective method results. This feature allows users to test again some previous executions or use its method data for other experiments or alternatives. It is also possible to use auxiliary methods (i.e. edit auxiliary method data) that determine specific method data elements. The results depend on the method’s objectives by completing the necessary steps to execute them successfully. The users have the possibility to export all or some method data by selecting what to export. The importing and exporting functions should be available at anytime and be visible by users. These functions are a good way to edit the method data quickly and efficiently, save a great amount of time, are easy and trivial to use, and are also essential functions for any framework to have.

Afterwards the users can execute the project (i.e. method execution) to obtain the method results if all the method steps were made in order to prevent unnecessary errors or oversights. It is considered the idea of showing the steps’ status somewhere visible in the framework to allow users to see the project state (which also gives an extra support and explanation of the methods). After the execution, the users are allowed to view the respective method results and are able to export them and/or the data used. The method results of each execution are saved and can be deleted at anytime (successfully and failure executions are both saved, in case of failure it should be described the current problem or cause of the failure).

Finally, the users can logout the framework whenever they want and the current status of the projects are saved (doing an auto-save for each action made in the projects can be beneficial in usability and performance aspects).
Regarding the methods, when a project is created regardless the method chosen and it is opened, all projects must present the same user interface and have available the main functions (like importing and exporting data). But, have the different types and features according to the method chosen. When executing a method, it should connect to the framework’s database to save the results (or saved them in local files) and update the project’s status. For some methods, the framework first calls a specific server of the respective method, executes it and then saves the obtained results in the database (a detailed description and main reason for these choices are presented in the next section).

### 3.1.2 Back-end Use Cases

The following use case (figure 3.2) presents the final requirements for the back-end side of the new MCDA framework.

![Use Case: Back-end](image)

Figure 3.2: Use Case: Back-end.

The use case only contains one actor, the developer who is responsible for the maintenance of the whole framework, its updates and the extension of new methods.

A developer can add new functionalities or features and even new methods to the framework, but can also remove them or simply correct them. The maintenance of the framework includes not only the front-end and back-end, but also the respective database and the existing servers. Furthermore, when a developer publishes a new method it should maintain and respect the same user interface (this is applied to every method) and have available the main functions (like importing and exporting data). But, it should be adapted to the method's different concepts and needs. It is important to maintain consistency when adding new methods to the framework and prepare the main functions for each new method, because of the different types of data. For this reason, some examples files will be created. These files contain the main aspects of the user interface and the main necessary functions (i.e. the required HTML, JS
and CSS files that contain the main interface basis of a project using a method). They will help reduce the implementation time and will guide the developers.

A method can be implemented in any web language, because it is a web-based framework. The developer has to decide where to save the main data and respective results for the project management functionality. There are two options for this case, the developer can save the data in local files or in the framework's database. The reason why the data can be saved in a database is to have easy access to the project's data by other servers who can execute specific methods and deliver the results by also saving them in the database. This opens the possibility to use some available methods (open source) easily in the framework and the opportunity to add methods developed in other programming languages. But the most important is to provide the new framework some functional MCDA methods that can be used for testing purposes and to demonstrate the potential and main objectives of the new MCDA framework. It might be interesting to use some MCDA methods available as open source from Diviz (one of the frameworks studied in the previous chapter), because the implementation of MCDA methods is not the main objective of the work. But instead, to study and develop a new framework that explores these methods used by users (with MCDA knowledge) and has a great usability, user experience and flexibility. With this being stated, the framework can call these servers to execute the method and then obtain the calculated results (which was mentioned in the previous section). So basically, a developer can create and remove servers that execute certain methods and also has to maintenance them.

A small note, the developer can update the database if he thinks it is necessary when implementing a new method (to include some specific data or extra parameters, because the database is not only for the servers).

### 3.2 Framework Architecture and Technology

The architecture used in the final solution, MCDA Framework, is presented in figure 3.3. The Web Browser and Application Server (i.e. the green nodes) represent the main infrastructure plus the Database. While the gray node is the Diviz Server (the only separated server at the moment) which executes specific MCDA methods provided by Diviz (see section 2.2.3). In this architecture, the user interacts with the Web Browser and performs the login in order to use the framework. After the User Authentication component approves the access, the respective user interface with the list of projects is displayed. The Projects List component contains all the projects created by the user and respective management (e.g. deleting project, editing, etc...). A Project component has three main components:

- **Data List Management**: the user defines and manages the data required for the method chosen;
- **Method Execution**: the user executes the method using the data that was defined;
- **Results List Management**: the user manages and performs other functionalities to the obtained results.

All functionalities done in the Web Browser by the user will interact with the Application Server through a HTTP/Ajax communication. Basically, the Application Server manages the users, the projects, the
project data and the project results communicating with the Database through Mongoose. Depending in the method chosen, the data and results can be managed in the database or in local files (in JSON format). Having these two possibilities has its advantages. For example, it is easier for separated servers (like the Diviz Server) to receive the respective project data from the Database and save the obtained results instead of using the local files. However, for the methods that are executed by the Application Server, the local files are the ideal option and make the framework less dependent of the Database. The framework provides three methods (one MCDA method and two basic methods) at the moment:

- **Order By method**: orders a list of data by an attribute and type of order. The data and results are saved in Local Files.json;

- **Order People method**: orders a list of people (i.e. a list with names and respective ages) by an attribute and type of order. The data and results are saved in the Database;

- **Electre Tri-C method**: the Diviz Server executes the Electre Tri-C method while communicating with the Database (i.e. obtain the data and save the results).

The main purpose of the basic methods, Order By and Order People, is to show developers how to extend the framework by following these two examples (see the Administrative Manual in Appendix B for more information). The difference between these two methods is the data and results location. The Order By method uses the Local Files.json, while the Order People method uses the Database for this subject matter. These examples allow developers understand the available possibilities when adding a new method to the framework (including the possibility to create a new separated server like the Diviz
A small note, the Application Server communicates with the Diviz Server when the Electre Tri-C method needs to be executed by giving the respective project’s information (in order to receive the right data from the Database).

The technologies used are the base of a MEAN application which consists in using MongoDB, Express and Node JS for the back-end and AngularJS for the front-end (plus HTML and CSS). In addition, it is also used Bootstrap for styling purposes, some essential libraries for the front-end (e.g. JQuery, Ajax, jsZip, etc...) and back-end (e.g. jsonwebtoken, file-system, etc...). Like mentioned previously, the Application Server contains specific local files that provide the data and results of some projects in JSON format (i.e. projects that execute the Order By method). Each project has its own folder (identified by their ID) with the respective data and results inside. Mongoose performs the communication between the Application Server and Database. It provides a straight-forward, schema-based solution to model the framework's data which is a easier way to do the management and saves lots of time in implementation.

Regarding the Diviz Server, Flask and Python are used and it’s a web-based server. However, the technologies used in separated servers (like the Diviz server) can differ (e.g. a server based in Java or PHP, etc...) if and only if, the communication between the MCDA Framework and the server can be done successfully. Basically, the MCDA Framework must provide the project’s information to the server and the server must receive the data from the Database. Then the server executes the method using the data received and finally the server must send the respective results to the MCDA Framework and save them in the Database.

3.3 Summary

This chapter describes the problem analysis and the reached final solution design after the study of some related frameworks to the work (Chapter 2). The main requirements were carefully examined and compared with previous frameworks in order to avoid mistakes, useless functionalities and poor user experience. On the other hand, some requirements from these frameworks were considered and can be used, because they are good ideas for the framework to have and might improve the user friendly experience and performance.

The main goal is not to implement MCDA methods, but to build a new framework that explores these methods that can be used by anyone (with or without experience and has at least some knowledge about MCDA). It should be user friendly, easy, flexible and it should be possible to add new methods. As a result, it was decided to use some open source MCDA methods provided by Diviz (one of the frameworks studied) to be possible to test, evaluate and see the potential of the new MCDA framework developed. After all of these aspects, the final requirements for the framework were defined.

Furthermore, the architecture and technologies chosen for the framework are also described and explained why these choices are good when developing web-based frameworks. But also, how specific requirements will be supported and done.

The next chapter describes in great detail the implementation and final version of the framework.
Chapter 4

Solution Implementation

Having previously described and discussed relevant works towards the presented problem, it was conducted a problem analysis and solution design phase in order to assess the ideal methods and requirements for the development of an appropriate solution.

In order to develop a new framework that explores MCDA methods and extends the number of these methods, while keeping a simple and flexible user interface. It was necessary to take full advantage of the technologies chosen in the previous chapter and maintain the consistency plus usability of the whole interface. In Appendix A, the User Manual of the framework is displayed which contains a brief description and necessary requirements, but also displays a detailed tutorial how every function of the framework works and how to use them.

This chapter describes in detail the solution implementation and justifies the proposed solution. Note that after analysing the results from the evaluations regarding the first version, some small changes were made to the framework (i.e. relevant improvements and correction of the usability issues found). The new version is described in this chapter.

4.1 Main Pages Section

The main pages or mostly called "home pages" of the framework are the first pages to be seen and accessed by the users. It provides some essential information about the framework (like in figure 4.1) and contact information. But most importantly is where the users create a new account and login into the framework.

These main pages can be switched through the dark blue navigator bar on top (by clicking the respective page name) which is displayed all over the framework. Doing so, maintains the consistency of the interface and helps the users get used to the interface.

There are in total five main pages: the home page, the about page, the contact page, the sign up page and finally the login page. All of these pages follow the same interface presented in figure 4.1 by displaying the main information in the center of the page and follow the same style.

The home page shows a simple description of the framework and has an extra access to the login
page. While the about page (figure 4.1) describes the framework a little more and its functionalities. Finally, the contact page provides the contact information.

![MCDA Framework](MCDA_Framework.png)

Figure 4.1: One of the home pages - About section.

### 4.1.1 Registration and Login

When the sign up page is accessed (see figure 4.2 (a)), the users can create an account by providing three basic informations: name, username and password. All of them are mandatory, if the user clicks on the “Sign Up” button and some information is missing, then an alert message appears and prevents the account to be created. So when all inputs are filled the new account is saved into the framework’s database. After that, a success message is displayed and now the user can login into the framework.

The login page (see figure 4.2 (b)) is similar to the previous page, but contains an extra access to the sign up page (i.e. the “Create account” link). The user provides his credentials, clicks on the “Login” button and accesses the framework. However, if the credentials do not correspond to any existing account in the database (security aspects), an alert message appears and the login operation is canceled not allowing the user or an anonymous person to enter.

![Registration and Login](Registration_Login.png)

(a) Create a new account.  
(b) Login access to the framework.

Figure 4.2: Registration page and Login Page.
4.2 Projects Management Section

After the login is done successfully, the users go directly to the projects page, like shown in figure 4.3 which displays a specific user account as example. The projects available presented in the page belong to the user, so each user can manage his own projects.

4.2.1 Create, Edit and Delete Projects

In figure 4.3, the projects are displayed as a table containing some information about each project like name, method chosen, notes, date when the project was created and date of the last update made in the project. In the last line of the table, the "Add New Project" button allows users to add a new project into the table and after clicking it the necessary fields are displayed in order to do the new creation. The user fills the name, notes, chooses which method to use in the project and then creates the new project. The name and select method fields must be filled or an alert message appears and the project is not created. But, it is possible to cancel the new addition by clicking on the "Cancel" button. When the project is successfully created, the respective data is saved in the framework’s database.

For each project, there are four types of functions available (the four colored buttons in the left): Open, Edit, Duplicate (it will be explained in the next section) and Delete. It can be said that the functions execute what the name describes. The "Open" button opens the project selected and it goes to the respective project section. This section can have different features and aspects depending on the method chosen and this will be described in detail later on. The "Delete" button deletes the respective project and its data in the database and/or local files, after the user answers the confirmation message (i.e. the user has the possibility to cancel this action before going further).

Finally, the "Edit" button makes it possible to change the project name and notes. The user clicks on the "Save" button to save the changes made (which updates the information saved in the database). If
the user does not want to change them anymore, just need to click on the “Cancel” button.

4.2.2 Duplicate Projects

The “Duplicate” button allows the user to copy or clone an existing project. All the data of the original project, except the results, are cloned and saved as a new created project. After the project is cloned, the user can execute any of the four available functions normally, like in the other existing projects.

The only difference between cloning a project and creating a project is that when the user opens the clone project it already contains data and it is ready to execute the method (it depends on the actual state of the original project). While creating a new project from scratch is basically empty. This functionality offers the opportunity to reuse data from new or old projects, it saves time and can be useful for testing purposes or making new experiments with the cloned data.

4.3 Project Area

Each project area interface differs by the method chosen, but maintains the same consistency and aspects for every interface. The reason behind this is because the methods are diverse and mostly require different types of data. With this being said, the number of sections might be different depending on the method (these sections are displayed on a vertical bar on the left, see figure 4.4 as example).

However, there are three main sections that every project area must have which are: the description section, one data section (this is where the methods differ in terms of number of sections, they can have one or more data sections depending on the method’s needs) and finally the execution and results section. All of these sections will be described later on.

Figure 4.4: Description section from a project that uses the Electre Tri-C method.

When a project is opened, it goes directly to the description section (see figure 4.4 from a project example). This section displays some information about the project (like name, method chosen, etc...),
shows a brief explanation of the method and contains the necessary steps for the method to be executed successfully. In addition, it shows the project’s current data (i.e. the data added in each data section). The main objective for this page is to help users understand and see the state of the whole project by having the current data and instructions in the same place, plus is a good way to start a project when is opened.

Like in the main pages, the dark blue navigator on top allows users to switch areas and in this case the users can either go back to the projects page (or the projects management section) or go back to the login page (i.e. logout). Furthermore, below the sections from the left vertical bar, it is visible the import and export functions as two blue buttons. Both correspond to the current data of the project and when one of them is clicked, it is showed the respective section which displays the available options of the function chosen (see figures 4.5 and 4.6).

In the import section (figure 4.5) the user must choose what type of data (from a list of options) to import, but also choose the file that will be imported to that option (the files must be in CSV format). It is possible to import all the data with the respective files at once. However, if the file format is incorrect or the content does not correspond to the data selected, the import function for that option is canceled and an alert message appears stating the reason why the function was not successfully executed.

The same goes for the export section (see figure 4.6), a user selects the data he wants to export and the files come in a CSV format inside a folder with the project’s name. If no option is selected and the “Export” button is clicked an alert message is displayed which cancels the function.

These functions make it faster and easier to insert data and can be used in other projects (i.e. use the exported files in other projects). Additionally, a user can select or unselect all options in both sections by clicking on the respective buttons to avoid the frustration of doing it manually and it is a faster way to do so.

Figure 4.5: Import section: import project’s current data.
4.3.1 Data Sections

Like mentioned previously, each project interface might differ the number of data sections available depending on the method chosen. The needs and requirements make these sections a little different from the data point of view, but maintain consistency and be as user friendly as possible. The main aspects and style can be seen in figures 4.7 and 4.8 as examples from the Electre Tri-C method [9], plus the different ways to deal with various data types.

There are different ways to insert data, the main one consists something similar to the projects page which is a table list with the available functions displayed as buttons on the left. The first line is used to add a new element to the data set by filling the respective fields (like in figure 4.7). If certain fills are mandatory, an alert message is displayed and the new element is not added until the respective fields are filled. After being successfully added, that element can be edited (i.e. change the fields values) or deleted (after answering the confirmation message, the same when deleting a project).

For simple and obligatory data values like adding a single value or selecting an option from a available list, a clear section indicating the data and respective field are displayed. The user only needs to fill or select the value for these types of data. There are no alerts messages, because they are mandatory and a predefined value is added (then the user changes the value for what he wants).

For data that is dependent in other data types, the respective table is only shown when the other data have been added. After that, the table is displayed with the respective fields and the users can now fill the data (this type of table is presented in figure 4.8). Not forgetting that the import function is also a way to insert the data.

However, these ways to insert data are main examples, for future methods if the available examples are not the ideal way to add the data there can be new ways to do this (i.e. different types of tables, types data, etc...). But, they must maintain the style and consistency to not confuse the users and still be easy to use and understandable (this corresponds to the worst case scenario, the main examples
are considered to be enough).

All the data and changes made are saved in the database for methods that are executed in other servers (like the Electre Tri-C method [9] example provided by the diviz server) or in local files for methods that are executed in the framework (i.e. that are provided by the framework).

4.3.2 Auxiliary Methods

The main objective of the auxiliary methods is to obtain or calculate some specific data that helps users understand better the main method and the respective data. These methods are available only in the data sections that correspond to the data that will be calculated. The interface depends in the auxiliary
method requirements, so they can have various ways to be executed and information to be presented.

For example, in the Electre Tri-C method, there is an auxiliary method to determine the weights for the criteria added (see figures 4.9 (a) and (b)) that uses the SRF method. The user must order the criteria according to their importance and can increment the criterion's importance (to highlight the difference even more). Then defines the required data and finally executes the auxiliary method. The results are automatically updated and the user can export the respective results. In addition, two types of graphs for analysis are generated and they can also be exported (in PNG or JPG format).

The SRF software mentioned in Chapter 2 is where this auxiliary method came from, which is a huge help for specific MCDA methods and demonstrates the idea of having auxiliary methods in the framework. But also, shows the possibility to integrate other existing MCDA frameworks.

Basically, these methods can be available in any data section with a brief explanation and steps required to guide the users. They help users understand why these methods are available and the benefits. The users can use them for their needs, are useful and easy to use, but they are not mandatory. The users can use them if they wish.

![Auxiliary method - data and parameters.](image1)

![Auxiliary method - results.](image2)

Figure 4.9: SRF - Determining weights of criteria method.

### 4.3.3 Method Execution and Results Section

Finally, the last section is reached (the execution and results section) which consists in executing the method and obtain the respective results. Plus the visualization of all the results saved from previous executions (see figures 4.10 and 4.11).

The section displays a table (i.e. results table) containing all the results saved from different executions. Its interface is similar to previous tables from other sections (i.e. the available functions are presented as buttons on the left) and displays some information regarding the result (like name, notes and date when the execution was made).

The last line of the table is where the users can execute the method and fill the results informations (they are not mandatory, so they can be empty). Also in the last line, the users can see or hide the current data of the project before executing, by clicking on the “View Current Data” button (see example of this section in figure 4.10 which contains some results, the execution line and below the project's current data).
The users can only execute the method after all data sections are completed (i.e. the sections, including this section, from the left vertical bar are green and not red). If specific data are missing or not inserted yet and are required, the users cannot execute the method (the execution button is disabled and the sections where the data are missing are red). The method is executed by clicking the “Execute Method” button and wait patiently for the respective results (depending on the method chosen, the waiting time can differ because some methods are executed in other servers, while others are provided by the framework which are normally faster). Like stated previously, before executing the method the users can add a name and some notes to the results, but also can edit them at anytime. After the execution is finished, the results are added to the results table.

There are two types of results, successful and failure, which correspond to the execution final status. The only differences between the two are: the failed execution does not have the export function; and the failed execution results show the possible mistakes or errors that the users made when adding the data, while the successful execution shows the obtained results (see example in figure 4.11). However, when this information is showed, both also display the data used when the method was executed.

Regarding the available functions for each result (buttons on the left) they are: View, Export Results (only available for successful execution), Delete and Reload. The View function displays or hides the results information (i.e. results or error message and data used) and the Delete function removes the results from the results table and database (like in any delete function, the user must answer the confirmation message in order to execute the function). The Export Results function is similar when the user wants to export the project’s current data. A list of options is available which the user selects what he wants to export from the results (can export results, data and result’s notes). Then, they are exported as CSV files inside a folder with the results name.

Finally, the Reload function allows users to replace the current data of the project into the data used in the obtained results, which allows users to reuse old data from any results (including the failed
executions) as they wish.

![MCDA Framework](image)

Figure 4.11: Execution and results section: How the results are displayed.

### 4.4 Summary

The main focus when implementing the design solution was to build a consistent and user friendly interface that was flexible and easy to use for users that know MCDA.

In this chapter, it describes the main interface and components of the final solution of the latest version of the MCDA framework. The details regarding the specific features and functionalities are described. But also, the description of every section from the framework. The main pages section provides some information regarding the framework and it’s where the users can create an account and login into the framework.

After the login, the users can easily manage their own projects in the project management section and each project created contains its own sections (i.e. project area) depending on the method that will be executed. These various sections from the project display many features, but the main ones are: describing and showing information about the project; displaying and managing the project's current data; presenting and managing the results and finally the execution of the method. All of these sections make the actual framework and its respective interface.

Furthermore, the User Manual of the framework is available in Appendix A containing a brief description and necessary requirements of the MCDA framework, but also presents a detailed tutorial explaining how every function provided by the framework works.

The next chapter describes the framework extension, from planning the new method features and requirements to the steps required to add it successfully to the framework. While respecting the framework’s interface and main functionalities.
Chapter 5

Framework Extension

In the last chapter, the implementation of the design solution of the new MCDA framework was described. The respective interface was presented in detail and the main functions were explained. However, it was not mentioned how the methods can be added, how the execution of methods from other servers worked and the benefits of doing so. The previous chapter focused more in the user side and not the developer’s point of view.

This chapter describes how the extension of new methods to the framework is done. While maintaining the interface, consistency and main functionalities for the new additions (which is done by the developer). In addition, it also describes the objectives and advantages of having servers which execute specific methods and provide the respective results to the framework, instead of executing them directly in the framework.

The Administrative Manual is available in Appendix B which describes all the necessary steps in detail and respective order (plus explaining with actual examples), and structure in order to add the new method successfully.

5.1 Planning and Preparations

Before starting the implementation of the new method, a brief planning should be done, as well as, the required preparations for an easier, faster and well made extension. The developer should prepare a list containing the required data for the new method, before deciding specific aspects that will be mentioned further on. Additionally, the respective requirements should be defined, as well as, the necessary auxiliary methods. If they are useful and make a good addition for helping the users and the method itself (it is not mandatory to have auxiliary methods).

After the necessary data and requirements are defined, the developer needs to decide two important details: if the method is going to be executed in another server (like in the Electre Tri-C method example executed by the Diviz server) or directly in the framework; and where the actual data plus the obtained results are going to be saved (database or local files).

If the new method is going to be executed directly in the framework, then the developer decides if he
wants to save the data in local files or database. For these cases, the database is not recommended, only if it justifies the need to do so and offers benefits, because using local files does not make the framework frequently dependent in the database.

In case of the new method being executed in another server, the developer should prepare the calls between the server and the framework. But also, how the connection to the database is going to be made (not forgetting to add the new data structures into the database so the data and results can be saved). Saving the data and results in the database is the easiest and fastest way to interact and communicate between servers, but most importantly it does not contain any security restrictions like when using local files.

Finally, to help and guide the developers regarding the front-end aspects, it was created five files that build an example of the main interface (i.e. how other methods are displayed). These files contain the required main functions to be adapted according to the method’s features.

In the last chapter, the sections that form the project area were explained and it was stated that depending on the chosen method the number of sections could differ. However, each method contains three mandatory sections which are: the description section, the data section (there can be more data sections, this is where the number of sections can differ between methods) and finally the execution and results section. For this reason, there are in total five files which three of them correspond to each section mentioned (in HTML files). The other two correspond to the interface's style sheet (CSS file) and the main functions plus where necessary functions for the method will be implemented (Javascript file). These files can make the extension of a new method (is the minimum number of files, but can be more).

The developer is responsible to adapt the main functions according to the method, changes and required additions in the back-end and the database models plus the interactions.

### 5.2 Implementation

With the planning and preparations done, the implementation can now go forward. First the back-end, the developer needs to add the new functions that will interact with the method data in the database (if the method is going to be executed on another server) or in the local files. Mostly, these functions are similar to the ones already implemented in the framework (for both options) and just need some small changes. Using local files the structure must be in JSON format and for the database the respective models of the new data must be created in order to separate each method data.

Additionally, if certain functions are not implemented in the framework and the developer considers that they should be added, it is possible to join these new functions. However, if they are treated as main functions (i.e. all methods, interface, can execute these functions) all the existing methods of the framework should be updated with the new function.

When the back-end is completed, the developer then copies the example files and add them to the framework to start the adaptation regarding the frond-end (which are the base files).
5.2.1 Method Features and Data

After the example files have been added to the framework, the developer now adapts and makes the necessary changes according to the method requirements. The developer implements the functions needed to call the server functions (like get the data or results, updates, get project's information, etc... anything related to the server side).

Some examples are provided by the Javascript file example which is where these functions will be implemented. In this file there are also the required main functions (as examples) that should be adapted. In addition, it is where the auxiliary methods are going to be implemented, plus the execution of the method (which the developer is responsible for all the respective implementations).

Regarding the interface of each section, the developer has to follow the files structure (the three HTML files mentioned previously) and change the parts that need to be changed. Like the left navigator bar which lets the users switch sections, it only needs to add the respective sections that the new method will have. In the description section, the developer only needs to set the required steps for the method to be executed, a brief definition, and display the project's current data. Depending on the method, there can be one or more data sections (all of them follow the same structure) and the developer has to put the respective data by following the available examples from the HTML file and must include CRUD functions for the users to manage the data.

Finally, in the execution and results section the developer modifies the data and results displayed for the ones that correspond to the method. Also, it should be indicated the function that will be executed when the users want to start the execution of the method. The style and aspects of the interface are already defined in the style sheet (CSS file) and it can also be extended. The style sheet provides the same pattern to every section of the framework, so the consistency of the framework is maintained between methods (the developers should be careful not to stay out of the main interface).

5.2.2 Framework Main Functions

There are a few main functions that every method should execute and are essential to the framework to have. Some are already done and do not need any change like the logout function or going back to the projects section function.

The developer first needs to update the switching sections function to the corresponding sections of the new method. Then the only main functions remaining are the import and export functions. There are examples of these two functions in the respective file example, the developer needs to modify them according to the data and results of the method (both functions have a similar structure). There is a list with several options that represent the data or the results plus data used (see figure 5.1). Then it is selected what it is going to be imported or exported which it is possible to select all options or none by clicking on the available buttons or do it manually. Then the import/export button is clicked in order to execute the respective function (alert messages can occur if there are problems in the files to be imported or regarding the options selection).

In short, the developer must: change the options of the list with the correct information; modify the
alert messages with the respective names of the data; adapt the functions examples of importing (plus according to the file structure that is going to be given by the users) and exporting according to the available options and data.

A small note, the difference between exporting the project’s current data and exporting the results is that in the results the users can export not only the data used but also the results and notes. While the other option only exports the data. In addition, the developer must update the select all and select none functions (the other buttons available mentioned previously, see figure 5.1) from both in order to update the status of the options from the list.

5.2.3 Method’s Server

In case of the new method being executed in another server, the developer is totally responsible for the creation of the respective server (architecture, technology, implementation, etc.). However, the server must connect to the framework’s database in order to obtain the project’s current data and to save all the obtained results. But also, it must be possible to communicate to the framework, so it knows when to execute the method and what is the corresponding project.

For example, the Diviz server that executes MCDA methods from the ELECTRE family (provided as open source by the Diviz framework described in chapter 2) is implemented in Python and Flask. The framework calls the Diviz server in the execution and results section and waits until it obtains the response that the method was executed (successful or failure). This response is normally sent when the results have been completely saved in the database.

Comparing these servers to the methods that are executed directly in the framework, the only difference when adding a new method is where the execution of the method is done. Basically, the same steps are done regarding the interface, main functions, auxiliary methods and the use of the example files. But
the method’s execution is different (in the execution and results section) or it waits for the execution to be done in the framework or makes the respective call to the server and waits for the response.

There are great advantages regarding using these separated servers, but also disadvantages. The developers can create new methods using different programming languages from the ones used in the framework. But also, have the opportunity to reuse other implemented methods (open source) like in the case of the Diviz framework. These advantages allow the fast expansion of the list of available methods and allow any developer to create methods in different ways.

However, using the database to get and manipulate the data is not recommended for large frameworks that contain many different types of data that are constantly changing. So it is recommended to use local files if the methods are executed directly in the framework.

5.3 Summary

This chapter describes the necessary requirements that the developers need to do for the extension of new methods to the framework. The developers should plan and make the right preparations before implementing the new method like: what are the required data for the method to be executed; what auxiliary methods would be useful to implement; where the method is going to be executed (in the actual framework or in a server); and where the data and results will be saved (in local files or in the database).

After the planning is done, the implementations can go forward and the developers start in the back-end by adding the necessary functions to interact with the data. Then, the example files are used to help and guide the developers regarding the front-end. These files make the developers follow the main interface and patterns. Furthermore, they demonstrate examples of how the main functions are done, plus the available presentations of the data which the developers only need to adapt according to the method’s needs. The main functions are stated and the respective fundamentals are explained. In addition, it also explains what is required to create a new server that executes a method if the developers decide to do so. Plus the advantages and disadvantages of having separated servers.

The available Administrative Manual describes all the necessary steps in detail and the respective structure (using actual examples) in order to add the new method successfully which can be read in Appendix B.

The next chapter describes the user evaluation stage, from planning to the result analysis and respective conclusions.
Chapter 6

Evaluation

It was presented in the previous chapter how the extension of new methods to the framework is made and the respective requirements. As the involvement of real users is a crucial step in the development of a user-centered solution, a series of evaluations was conducted with a group of volunteer users in order to evaluate the framework solution. The main objective is to gather information about the users’ expectations, opinions and difficulties while interacting with the framework and see if the main objectives (i.e. regarding the usability and user experience) were accomplished.

It is important to note that the first version of the framework solution was evaluated by giving the volunteers a list of simple tasks to accomplish in the framework (while following their moves and taking notes) and then responding to a survey regarding their experience and the framework itself.

6.1 Process

Each user performed the same tasks when using the first version of the framework. The estimated time for the evaluation was between 30 and 40 minutes and was divided as follows:

- **Preparation**: users were given a zip file containing a folder with CSV files (to be used on specific tasks), a Guidelines document and an User Manual to be read before starting the tasks (it was not mandatory to read these documents);

- **Setup**: after receiving the necessary files and documents, the users did the required setup displayed in the Guidelines document;

- **Tasks**: with the setup done, the users completed the tasks presented in the Guidelines document;

- **Survey**: after finishing all the tasks, users were then asked to fill a survey to measure the level of usability and utility of the framework. But also, to measure the user experience and obtain their feedback (regarding the negative and positive aspects, new ideas for the framework and final comments). This survey can be found in Appendix D.

The User Manual (Appendix A) explains the whole framework and shows the users how to perform the respective functions. This manual could be read before starting the tasks as preparation and was
available for consulting during the evaluation. The Guidelines document (presented in Appendix C) contains a brief introduction regarding the framework and evaluation, the setup required and all the tasks to be performed in the first version of the framework.

6.1.1 Tasks

The proposed tasks consisted in exploring the first version of the framework and using all its available functions in order to cover the whole framework and receive the respective feedback in the end. The users followed a specific order to execute the tasks which started with the basic and simplest tasks (i.e. basic functions) and the level of difficulty increases when going to the next task. The “Order By” method (a method example to guide developers how to add a new method in the framework described in the Administrative Manual, see Appendix B) is the first method to be explored, because of its simplicity which allows users to get used to the main interface and explore the main functions.

After completing the tasks regarding the basic method, the users then used an advanced method (i.e. the Electre Tri-C method [9]) which is a MCDA method, but also explored other advanced functions available. These tasks consisted in showing the users the differences between methods, use an actual MCDA method with an example scenario (the analysis of a list of cars according to the specific criteria), show the major role of an auxiliary method and how to use it. The last task consisted in answering the survey containing the respective link.

All the tasks are presented in the Guidelines document which can be found in Appendix C.

6.1.2 Participants and Setup

In total 17 users participated in the evaluation and all of them had at least a bachelor’s degree plus some knowledge about MCDA (some more expert than others). The majority of the evaluations were made online using the Skype application with the screen sharing option activated (i.e. see the user’s screen), while other evaluations were made in person at INESC-ID (Alameda). These choices allowed a closer observation of the user’s movements, note the eventual mistakes, register the time taken to complete the tasks and obtain better feedback.

Regarding the setup for the evaluation, the users needed a computer with Internet access, one of the following browsers Chrome or Firefox and the Skype application in case the evaluation was made online. As previously mentioned, each user received a zip file containing all the necessary documents. Before advancing to the actual tasks the users explored the “Import files” folder to know the existing files which will be used in specific tasks during the evaluation.

Note that some users read or had a fast look at the User Manual (which can be found in Appendix A) for preparation, but it was not mandatory to do so (this makes an interesting feature to see if the framework is still easy to use having read or not the User Manual).

After that, the users opened the first version of the framework\(^1\) with the respective browser in order to start executing the tasks. The setup is described in the Guidelines document (see Appendix C).

\(^1\)MCDA Framework home page (new version). http://mcdaframework.sysresearch.org/home.html
6.1.3 Survey

After all the tasks were completed, each user answered the respective survey questions which were divided in three parts as follows:

- Usability;
- User Experience;
- Personal Opinion.

Before creating the survey some research was done in order to understand the differences between usability and user experience, because how similar they are when measured. But to find the ideal way to evaluate them in the framework. Usability is concerned with the “effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments” [10]. While the user experience is a consequence of brand image, presentation, functionality, system performance, interactive behavior and assertive capabilities of the interactive system, the user’s internal and physical state resulting from prior experiences, attitudes, skills and personality, and the context of use [11]. Although there is no fundamental difference between measures of usability and measures of user experience at a particular point in time, the difference in emphasis between task performance and pleasure leads to different concerns during development [12].

To evaluate the usability it was followed the 5Es dimensions [13] which are the following:

- **Effective**: How completely and accurately the work or experience is completed or goals reached;
- **Efficient**: How quickly this work can be completed;
- **Engaging**: How well the interface draws the user into the interaction and how pleasant and satisfying it is to use;
- **Error Tolerant**: How well the product prevents errors and can help the user recover from mistakes that do occur;
- **Easy to Learn**: How well the product supports both the initial orientation and continued learning throughout the complete lifetime of use.

These dimensions are evaluated by the users and are registered while observing them. Basically, the efficiency is determined by the average time that the users took to complete all the tasks. The effectiveness and error tolerance are analysed and registered if the users completed the tasks successfully and count how many mistakes they did during the tasks or if misleadings occurred. The Engaging and Easy to Learn dimensions the users answered the four questions presented in the survey (like ranking the level of difficulty of the available functions and level of satisfaction regarding the whole experience done in the framework).

Regarding the user experience, it was evaluated by answering the seven questions presented in the survey to classify the 6 scales according to the UEQ (i.e. User Experience Questionnaire, a questionnaire that measures the UX) [14] which are the following:
• **Attractiveness**: Overall impression of the product. Do users like or dislike it?

• **Perspicuity**: Is it easy to get familiar with the product?

• **Efficiency**: Can users solve their tasks with the product without unnecessary effort?

• **Dependability**: Does the user feel in control of the interaction?

• **Stimulation**: Is it exciting and motivating to use the product?

• **Novelty**: Is the product innovative and creative?

However, the questions provided by the UEQ were not used, but instead it was used and created specific questions inspired by the SUS (i.e. Measuring Usability With The System Usability Scale). The SUS, presents the items to participants as 5-point scales numbered from 1 (anchored with “Strongly disagree”) to 5 (anchored with “Strongly agree”) [15]. Using this question format, four of the seven questions are from the SUS which are used to answer four scales (i.e. Efficiency, Perspicuity, Stimulation and Dependability). While the Novelty and Attractiveness scales were created specific questions respecting the same format used on the other questions.

Finally, the last four questions of the survey correspond to the user's personal opinion which were not mandatory or the users could respond them in person. These questions cover the negative and positive aspects of the framework, ideas to how to improve the framework and any last comments or observations that they felt that needed to be expressed.

The survey had a total of 15 questions (i.e. 4 questions for usability, 7 for user experience and 4 for personal opinion) not including the comment questions and can be seen in Appendix D.

### 6.2 Results and Discussion

After doing the 17 evaluations, the respective analysis and discussion were made with the obtained results regarding the first version of the framework. All the users completed the requested tasks and answered the respective survey.

Note that at least five users were not comfortable with the Electre Tri-C method or MCDA in general (i.e. they did not know how the method worked or they did not know much about multi-criteria) and in consequence some answers were influenced because of it.

The complete results tables from the survey can be seen in Appendix E.

#### 6.2.1 Usability

During the evaluations, it was registered the time that the users took to accomplish all the tasks, the errors or mistakes made during specific tasks and if they managed to recover from those mistakes. While in the survey, the users classify the perceived difficulty of the functions done during the tasks as well as the SRF method (i.e. the available auxiliary method), using a 4 point scale (A)Very Difficult, B)Difficult, C)Easy, D)Very Easy).
Likewise, it was also measured the usefulness of specific functions and aspects provided by the framework (A) Should not be available, B) Useless, C) Useful, D) Very useful). In addition, using a 5 point scale (from 1 to 5, 1 being strongly disagree and 5 strongly agree) the users also classify their understanding (i.e. if they knew what was happening on the framework while following the tasks) and satisfaction regarding the whole experience (from 1 to 5, 1 being very unsatisfied and 5 very satisfied). The medians and modes for each question were determined, as well as the average of the execution time for statistic purposes and global analysis.

Considering the tables 6.1 and E.1, in general the users considered most of the available functions very easy to use with the exceptions of the export functions (for the method data and results), the edit method data functions and the SRF method which were classified as easy. It can be observed that using the auxiliary method (i.e. SRF method) was the least easy to use because of the slightly different interface. But the majority of the answers were influenced by the lack of knowledge of the SRF method and why it is necessary to follow the steps presented (see explanation in [5]).

<table>
<thead>
<tr>
<th>Quest.</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
<th>1.5</th>
<th>1.6</th>
<th>1.7</th>
<th>1.8</th>
<th>1.9</th>
<th>1.10</th>
<th>1.11</th>
<th>1.12</th>
<th>1.13</th>
<th>1.14</th>
<th>1.15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med.</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Mod.</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C,D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From 1.1 to 1.15: A) Very Difficult, B) Difficult, C) Easy, D) Very Easy.

Table 6.1: Answers from usability question 1: medians and modes.

The average time that users took to execute all the tasks was 30.5 minutes which indicates that most users did mostly well and performed the evaluation inside the respective duration. It was observed that the users with experience in these types of frameworks or users who have previously read the manual, finished the tasks faster than the other users. But overall the time between users was not distant indicating that the framework is efficient and easy to use.

In tables 6.2 and E.2 the values for the understanding and satisfaction were good. The users had almost a good understanding of what was happening in the framework, but some got influenced by the lack of knowledge of MCDA and did not understand the Electre Tri-C method which causes the results to decrease in average.

<table>
<thead>
<tr>
<th>Quest.</th>
<th>2.1</th>
<th>2.2</th>
<th>2.3</th>
<th>2.4</th>
<th>3.</th>
<th>4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aver.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.9</td>
<td>4.4</td>
</tr>
<tr>
<td>Med.</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Mod.</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Subtitles: Aver.= Average. From 2.1 to 2.4: A) Should not be available, B) Useless, C) Useful, D) Very useful.
3. On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree.
4. On a scale of 1 to 5, 1 being very unsatisfied and 5 very satisfied.

Table 6.2: Answers from usability questions 2, 3 and 4: averages, medians and modes.

Overall, the satisfaction for the whole experience was almost very satisfactory between users. It is also visible in the table that users rank the available functions as useful regarding the project duplication, the reloading of data from previous results and having auxiliary methods to determine specific data.
Having green or red sections indicating what is done and not was ranked as very useful, because users felt like they were progressing and knew what was missing before executing the project.

All users completed every single task successfully. The majority of the users did not make mistakes or eventual errors, but the ones who did it in the end managed to recover or undo the respective mistakes. In general, it can be concluded that the framework's usability is quite good and all the 5Es dimensions were archived.

6.2.2 Usability Issues

However, even with the positive results some usability issues were registered from the mistakes made by the users during the tasks.

The first one spotted was when the users needed to create the first project. The tendency was to click on the “Create Project” button before filling the respective fields on the right which causes an alert message and the project was not created. Normally people tend to visualize from left to right, in consequence they saw the button first and had the tendency to click right away before seeing the rest. The users who read the manual did not make this mistake, but some also gave the same opinion regarding this aspect. A way to improve this was to put the button on right or have a new button that gives access to this aspect (i.e. the “Create Project” button and fields) and makes the users see everything before advancing.

When users tried to export the obtained results, some users did not see or understood where the “Export” button was, because it is seen as a section list and not a button. In consequence, the respective users lost time in order to find it and one user required assistance. To avoid these events, the style of the respective button should be more visible, obvious and it should not be simple text.

The final issue observed was indicated to all the delete functions available. Every user managed to use these functions with no problems, but there was no way to recover from a delete function. Users suggested that there should be a confirmation section or message before deleting something. Doing so will avoid accidental mistakes, the actions can be undone and improves the framework’s usability.

These issues were considered important and relevant enough to be corrected (plus the addition of a brief explanation of the method in the description section), because they will improve the framework’s usability. So the MCDA framework was updated and the second version was released.

6.2.3 User Experience

In the survey, the users answered seven questions classifying each question using a 5 point scale (from 1 to 5, 1 being strongly disagree and 5 strongly agree). The averages, medians and modes for each question were determined for statistical purposes and global analysis.

Considering the tables 6.3 and E.3, in general it can be observed that the user experience was good based on the results average (with values between 3.8 and 4.4). Overall, users found the framework to be easy to use with a pleasing and friendly interface. They considered that the majority of the functions
were well integrated and the framework’s efficiency was good, but also felt very confident when using the framework (even the users who did not understand the Electre Tri-C method).

However, the following two questions regarding if the users would like to use the framework in the future and if they thought it was innovative and creative were the least positive. The majority would like to use the framework again, but the ones that were not interested in using MCDA methods were not. This leads to mixed results and some got influenced by not knowing much about MCDA methods.

Likewise, the innovation and creativity question got the same treatment, but in this case half of the users did not have the knowledge or never used other MCDA frameworks besides this one. So they did not know how to answer properly. On the other hand, the other half responded very positively to this question and stated the framework’s simplicity and easy usage when comparing to other frameworks.

In general, the 6 scales (i.e. Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation and Novelty used by the UEQ) to be evaluated were archived with a good note, but the Stimulation (i.e. users would like to use it) and Novelty (i.e. innovative and creative) scales were the least positive. These two depend mostly in the users objectives and knowledge about MCDA methods. However, it should be considered in the future to improve these aspects and trying to attract more users regardless of their experiences and background.

### 6.2.4 Opinions and Observations

The users provided their feedback and opinions personally or by answering the respective survey questions. Like mentioned previously, users with good knowledge of MCDA methods and have used other MCDA frameworks (some which are mentioned in Chapter 2) stated the positivity of having a new framework more flexible, simple and easy to use at the first try. Furthermore, users shown interest in the future for new available methods and the possibility to request their own methods to be added in the framework.

The rest of the users liked the framework in general despite some small details (which differ between users and are mostly personal customizations) and stated that there are some aspects that can be improved.

Many suggestions for new improvements were received which some of them are interesting like: the addition of new methods and information regarding the methods; incorporate other existing MCDA frameworks (like it was done for the SRF method); the possibility to recognize a list of files and match them to the list of options from the import function; export other types of formats besides CSV; and finally add a help section containing a brief explanation of the framework and functions or parts of the
Some suggestions can be considered for future work as they are good ideas to keep in mind and can improve the framework. The positive feedback received regarding the whole framework shows the framework’s potential and future growth.

6.3 Summary

This chapter presents the evaluation of the developed solution. Each of the 17 participants completed the tasks given by using the first version of the framework. Then, answered a survey regarding their experience during the tasks and the framework itself, plus personal opinions and suggestions to improve the framework.

In general terms, participants were able to perform all the tasks given and the majority did them faster than the time suggested. All functions were classified as very easy to use with some small exceptions like using the auxiliary method. The users understood what was happening in the framework and in the end, users were satisfied with this whole experience.

Overall, the users found the framework to be easy to use with a pleasing and friendly interface, but also efficient. They felt confident when using it and might consider using it in the future (users who work with MCDA methods). In addition, specific aspects and functions like duplicating projects or having auxiliary methods were considered to be useful which can be interpreted as good ideas.

In particular, some answers of specific questions were influenced by the small knowledge of MCDA methods by some users which decreased the average results (e.g. like the questions regarding innovation and creativity or if users would like to use the framework frequently).

Despite having registered some usability issues (i.e. mistakes occurred during specific tasks), in general the usability and user experience were good which indicated that the developed solution archived its main objectives regarding the user side and interface.

However, it was considered relevant to correct the usability issues because they would improve the framework’s usability. So the MCDA framework was updated to a new version (second version) according to the suggestions stated in section 6.2.2.

The next chapter will finalize this dissertation by wrapping up what has been said until here and state what can be done in the future.
Chapter 7

Conclusions and Future Work

The use of MCDA frameworks can be difficult for users who are not experts in the MCDA methods. There are not many frameworks that focus on usability or user experience. In consequence, the frameworks can be difficult to use, not understandable and users require experience to acknowledge the available functionalities. In addition, some frameworks do not provide a large number of available MCDA methods to use and do not have the possibility to extend its number. It can be considered a lost opportunity to increase the variety of MCDA methods available and continuous growth.

7.1 Conclusions

In order to design a solution to these problems, existing MCDA frameworks were analysed in order to identify the major problems, advantages and disadvantages of current frameworks. With this study, it was concluded what was essential for a MCDA framework to have and what to avoid in order to create a framework with good usability and flexibility. But also, the ideal way to extend the list of methods. In addition, some useful and good ideas from other frameworks could be reused or integrated in the new design solution which increased the variety of functions available, performance and attractiveness.

The designed and developed solution was a new web-based integrated framework that explores MCDA methods for users (assuming they have only a generic knowledge about the methods and MCDA) to use. The framework provides diverse and useful functionalities (like project management, auxiliary methods, cloning existing projects, etc...) and maintains its consistency when using different methods. It is possible to extend the framework by adding new methods at anytime. The developers only need to follow the steps written in the Administrative Manual and perform the changes plus requirements needed. The framework also overcomes some negative aspects and difficulties presented in other current MCDA frameworks. But also, it offers the opportunity to reuse or integrate specific methods provided by other MCDA frameworks like Diviz or SRF.

Analysing the results of the evaluations made for the first version of the framework revealed that although some users did not have much knowledge about MCDA and majority never used other existing MCDA frameworks, they were able to accomplish all the tasks given. In general, they thought that
the framework was easy to use, efficient, all functions were well integrated and the interface was very friendly. Some aspects like stimulation and innovation were the least positive features, as well as, the occurrence of some usability issues during the tasks. But overall, the users were satisfied with the whole experience and framework. The final solution received positive feedback and presented a good usability, flexibility and user experience, despite half of the users were not considered to be MCDA experts.

Considering all the above, a solution that explores MCDA methods was developed which can be used by any user (with at least some MCDA knowledge). But also, has the possibility to extend the number of MCDA methods by developers at anytime. The evaluation phase revealed that users (regardless the level of knowledge of MCDA methods) were able to use all the available functions of the framework with satisfying results regarding the usability features and user experience and completed all the tasks given. In addition, it was considered important to correct the usability issues found in the evaluation, because they would improve the framework’s usability. So the required changes and updates were made and the new version of the MCDA Framework was released (i.e. the second version), plus the User Manual was also updated with the new version.

It is acknowledged that some aspects of the framework also need improvement, but some suggestions provided by users are good ideas to add into the framework in the future.

### 7.2 Future Work

After measuring the results of the evaluations and analysing all the received suggestions, it was concluded that some suggestions are considered to be interesting and are good ideas to implement into the framework. These ideas might improve the framework in general and increase the variety of available functions. Like the addition of new methods and more information explaining the respective methods (for now only a brief definition and the necessary steps are presented). Incorporate other existing MCDA frameworks (like the integration done of the method provided by the SRF software). The possibility to recognize a list of files and match them to the list of options from the import function, instead of selecting each file to the right option manually. The opportunity to export files in other formats besides CSV. Lastly, the addition of a separated help section containing a brief explanation of the whole framework and functions or show specific parts of the user manual.

Besides what was stated previously, it should be considered the development of an easier and faster way to add new methods to the framework, but instead this addition be done by developers it should be done by users. This might increase the number of methods available and attract more users to use the framework, but also have a generic way to extend the framework. Furthermore, add more security features in order to protect private data of some projects (i.e. projects that contain confidential data) and to give privacy to the users. Provide the functionality to share projects between users, this allows a group of users to edit directly on the same project with the current data and makes it easier to communicate between users. Add new auxiliary functions like graph generation, data analysis or new functions to create new projects. Finally, explore new existing MCDA frameworks to obtain other ideas that could improve the framework and its interface.
Bibliography


Appendix A

User Manual

User Manual

MCDA Framework

Version 2.0

Instituto Superior Técnico
October, 2016
# Table of contents

1. Introduction  
   1.1 Scope and Purpose  
   1.2 Process Overview  

2. Getting Started  
   2.1 Interface Overview  
   2.2 Register  
   2.3 Login/Logout  

3. Create and Edit Project Metadata  
   3.1 Create Project  
      i) Create manually  
      ii) Duplicate an existing project  
   3.2 Edit Project  
      i) Save changes  
      ii) Cancel changes  
   3.3 Delete Project  

4. Execute Project  
   4.1 Back to projects page  
   4.2 View method steps  
   4.3 Edit the method data  
      i) Edit manually  
         a) Add  
         b) Edit  
         c) Delete  
         ii) Reload the method data used from the obtained method results  
   4.4 Auxiliary methods  
   4.5 Import and export the method data  
   4.6 Execute project (method execution)  
   4.7 View the method results  
   4.8 Export the method results  
   4.9 Delete the method results
1. Introduction

1.1 Scope and Purpose

MCDA Framework is a web-based integrated framework that explores Multi-Criteria Decision Aiding (MCDA) methods. The framework focuses specially in usability and flexibility for a better user experience and quality, as well as, the extension of the methods list by adding new methods at anytime. In addition, it provides project management functionalities for the users to create and manage their own projects and provides some useful functionalities for a faster and easy usage of the framework.

The main goal of this user manual is to guide and help users regarding the whole framework and its main functionalities, plus the respective steps to execute each available function successfully. Basic functionalities like register, login and logout will be covered, but also the main functionalities (e.g. project management functions, project execution, import and export functions and more).

This framework is recommended for users who have at least some knowledge of MCDA in order to understand the available methods given by the framework. The MCDA Framework operates on Chrome (recommended) and Firefox browsers.

1.2 Process Overview

In order to use the MCDA Framework, it is required a computer with internet access using a recommended browser. The main pages of the framework provide some basic information, the possibility to create a new account and login into the framework. To get inside the framework’s main interface the authentication must be done and after that the projects page is opened. Here it is possible to create, edit, delete and clone a project. A project executes the method chosen when created and it is where the necessary method data are managed (i.e. where the data are created, deleted, edited, imported, reloaded and exported), but also provides some auxiliary methods to fill specific data fields. After all the required steps are done, the method can be executed and the respective results are saved which can be deleted or exported. When exporting, it comes all in CSV format and inside a folder (zip format). It is possible to see the method steps, but also the current method data which is in two different sections in the project. Furthermore, there are alert messages in some cases like creating or adding or importing data, when they are not done correctly or something is missing.
2. Getting Started

This section will give you a starting overview of the framework’s interface and, afterwards, a more detailed description of the main starting functions of the framework.

2.1 Interface Overview

The main interface of the framework is presented in the figure below. However, the main pages and projects page don’t display the left vertical bar (i.e. sections navigation and import/export buttons) increasing the content area.

![Framework interface](image)

Figure 1: Framework interface.

2.2 Register

First thing to do is to create a new account or register into the framework in order to have access to all main functionalities and a personal working space.

In order to register just:

1. Click on “Sign up” button to go to the registration page;

![Sign Up button](image)

Figure 2: “Sign Up” button.
2. Complete the form and submit.

![Register form](image3.png)

Figure 3: Register form.

After the form is sent and validated, the registration is complete by presenting a message below the form. An alert message is displayed if the form was not totally filled.

2.3 Login/Logout

After registering, it is now possible to login into the framework, if and only if the register was done successfully and the right credentials are provided when login. An alert message is presented if the credentials submitted are incorrect.

In order to login:

1. Click on “Login” button to go to the login page;

![Login button](image4.png)

Figure 4: “Login” button.

2. Complete the form and submit.

![Login form](image5.png)

Figure 5: Login form.
The logout function is available everywhere after the login, so at anytime it is possible to logout from the framework which redirects to the login page.
In order to logout just:
1. Click on “Logout” button (available in any page or section).

Figure 6: “Logout” button.

3. Create and Edit Project Metadata

After the login, it directly goes to the projects page which displays the current projects of the respective user and it is where the projects are managed and configured. The following functions from this page are described below. Note that the majority of these functions are only available after a project is created.

3.1 Create Project

There are two ways to create a new project.

i) Create manually

A new project can be create manually:
1. Click on “Add New Project” button;

2. Complete the form;
3. Click on “Create Project” button.

After the new project is validate, it is added to the projects list, but in case the “Cancel” button is clicked the project is not created and the form disappears. An alert message is displayed if the name is not provided and/or the method is not select.

ii) Duplicate an existing project

A new project can be created by duplicating (i.e. cloning) an existing project. To do this it is only necessary to:
1. Click on “Duplicate” button on the desired project to be cloned.

![Figure 9: “Duplicate” button to clone an existing project.](image)

After the new project is validate, it is added to the projects list with a similar name to the original project (original project name plus “_2”).

### 3.2 Edit Project

After a project is created it is possible to change the project name and notes. These changes can saved or canceled.

i) Save changes

To save the changes:

1. Click on “Edit” button on the desired project;

![Figure 10: “Edit” button from an existing project.](image)

2. Change the project fields:
   a. Change name;
   b. Change notes;

3. Click on “Save” button.

![Figure 11: Save the new name and notes.](image)

After the framework validates the information the fields are updated.

ii) Cancel changes

After clicking the “Edit” button and the changes made, to cancel them just:

1. Click on “Cancel” button to cancel the changes made.

![Figure 12: Cancel changes button.](image)

The changes are not validated by the framework and the original fields are maintained.
3.3 Delete Project

Also, after a project is created it is possible to delete all its information from the list and framework. To do so it is only necessary to:

1. Click on “Delete” button on the project to be deleted;
2. Answer the confirmation message that appears (it is possible to cancel the action).

4. Execute Project

The project interface consists of several sections in order to execute the opened project and obtain the method results from the data added. Depending on the method chosen, the sections information can differ and also the total number of available sections. This interface can only be accessed if an existing project is created and opened. The following main functions provided by the project interface are explained below.

4.1 Back to projects page

In every section from the project interface it is possible to return to the projects page just:

1. Click on “Back to Projects” button.

4.2 View method steps

Every project interface contains a description section which provides information regarding the project like its fields and current method data, but also a brief explanation of the method and the necessary steps needed in order to execute the method successfully.

To view these steps and information just:

1. Open the project’s description section by clicking the section displayed in the left vertical bar.
4.3 Edit the method data

Like stated previously, the project interface sections can differ not only by the types of information presented, but also the available number of sections displayed in the left vertical bar (a project interface can have 1 or more data sections). However, the following functions are provided by every data section. Note that some method data can only be managed manually, because they are one value only or provide an options list to be selected for the value.

i) Edit manually

The method data can be managed manually. Some can be managed by providing the right value which already contains a configured value or by selecting an option from the list available. While others can be managed by the following functions presented below.

![Figure 16: Example of a data section: add, edit and delete functions.](image)

a) Add

To add a new method data element:
1. Open the respective data section;
2. Complete the form;
3. Click on “Add Data” button or “Add <name_of_the_data>” button (e.g. “Add Criterion” button, if the data element to be added is a criterion).

The framework then validates the form, if all mandatory fields are filled the respective method data element is added successfully, but in case of an empty mandatory field an alert message is shown.

b) Edit

To edit an existing method data element (i.e. change the respective fields):
1. Open the respective data section;
2. Click on “Edit” button on the desired method data element;
3. Change the data element fields;
4. Click on “Save” button to save the changes or click on “Cancel” button to cancel the changes.
The changes are validated by the framework and updated in case the “Save” button was clicked. If the “Cancel” button was clicked the changes are canceled and the original values of the fields are maintained.

c) Delete
To delete an existing method data element and remove it from the list displayed in the data section, it is only needed to:
1. Click on “Delete” button on the respective method data element to be deleted;
2. Answer the confirmation message that appears (it is possible to cancel the action).

ii) Reload the method data used from the obtained method results
It is possible to reload or get all the method data used in a certain execution which is available in the execution and results section. In other words, it can be reused old method data from previous executions by doing the following:
1. Open the execution and results section;
2. Click on “Reload” button on the method results which have the desired method data.

Figure 17: Execution and results section from the left vertical bar.

Figure 18: “Reload” button from an existing method results.

All current method data of the project are replaced with the data from the chosen results.
4.4 Auxiliary methods

Some projects provide auxiliary methods and these depend on the method chosen. They exist in order to help and facilitate certain fields from a method data to be determined or calculated. These methods follow the same interface aspects and state the necessary steps in order to execute them successfully. To use these auxiliary methods just (the images below demonstrate an example of an auxiliary method: SRF method which determines the weights of criteria for the Electre Tri-C method):

1. Open the data section that has an auxiliary method;
2. Do the required steps presented by the auxiliary method (normally they consist in providing some types of information or selecting certain options);
3. Click on “Execute” button.

After the “Execute” button is clicked, the framework executes the auxiliary method to calculate the values of the respective fields and then updates their values according to the obtained results. In addition, a
successful message is presented after the updates were made. In case some steps are not done or some information is not defined, an alert message is displayed. In addition, these auxiliary methods provide help regarding the steps or the auxiliary method in general, like shown in the figure below.

![Help sections provided by the auxiliary method - SRF method.](image)

In figure 21, right next to the “Logout” button there is a “Help” button that displays a gray section with some general information regarding the auxiliary method. Putting the mouse over a step (e.g. over “Step 1”) it displays a small gray section with a brief explanation about the step and when the mouse is moved away the section disappears.

4.5 Import and export the method data

It is possible to import the method data by providing a file with the right structured (and in CSV format) instead of adding the method data manually.

In order to do so just:

1. In any available section, click on “Import Data” button (presented below the sections in the left vertical bar):

   ![“Import Data” button.](image)

   a. Select which method data to import (check the available options);
   b. Click on “Select file” to provide the CSV files to each option checked;
   c. Click on “Import” button;
The framework validates the given files and updates the respective method data (if the files are with the right structured), plus displays a message that the importing was done successfully. By “right structured” it means that the method data fields provided by the file must respect the fields required in the data section. An alert message is displayed if the file or files are not in the right format and structured, but also if nothing was selected to import.

At anytime, it is possible to export the method data by:

1. In any available section, click on “Export Data” button (presented below the “Import Data” button in the left vertical bar);
After the “Export” button is clicked a zip file is automatically generated and it is being downloaded by the browser. The zip file contains a folder with the project name and inside all the method data selected, each in their respective CSV files. In case none of the available method data were selected to export, an alert message is presented. Furthermore, the import and export functions have available two buttons, the “Select All” button and the “Select None” button. These buttons allow to select all or none options at once from the available list. Basically, instead of selecting every single option the same manually, these buttons check or uncheck all options at once.

### 4.6 Execute project (method execution)

In order to execute a project, all the method steps described in the description section must be done. These steps are completed if and only if all the sections from the left vertical bar are green and not red (except the description section which is always a dark gray color). Basically, to execute a project:

1. Complete all necessary steps (i.e. complete all data sections);
2. Then, open the execution and results section by clicking the respective section displayed in the left vertical bar;
3. Add a result name and notes;
4. Click on “Execute Method” button.

![Figure 26: Execute the project method.](image-url)
The framework executes the project method and saves the obtained method results. It adds the results to the displayed list, plus presents a message that the execution was complete. However, if some required steps were not done correctly or some of the method data have wrong types or information, the method results are not obtained and in replace it is saved a list of possible aspects why these results were not obtained successfully (but still contains all the same functions from successful results, except the export results function). The time of execution can be different between methods. If the “Execute Method” button is disabled (i.e. cannot be clicked), it means there are some steps that are not completed (some sections are red).

4.7 View the method results

To view the obtained method results from an execution or other previous method results just:

1. Open the execution and results section;
2. Click on “View” button on the respective method results.

![Figure 27: View or hide the method results.](image)

The method results and respective method data used are displayed below the line where the button was clicked. In order to hide the method results just click on the “Hide” button. Furthermore, the results name and notes can be edited directly in the respective box.

4.8 Export the method results

To export the results and/or respective method data used just:

1. Open the execution and results section;
2. Click on “Export Results” button on the respective method results;
   a. Select what to export (results, notes and method data used);
Click on “Export” button.

This function is similar to the previous export function of the current method data. A zip file is generated containing the selected options inside a folder with the result name (all files are in CSV format, except the notes which are in txt format). In case nothing was selected to be exported, an alert message is presented and the zip file is not generated.

4.9 Delete the method results

To delete a specific method results from the framework and list just:

1. Open the execution and results section;
2. Click on “Delete” button on the respective method results;
3. Answer the confirmation message that appears (it is possible to cancel the action).
Appendix B

Administrative Manual

Administrative Manual

MCDA Framework

Version 2.0

Instituto Superior Técnico
October, 2016
Table of contents

1. Introduction
   1.1 Scope and Purpose
   1.2 Process Overview
2. Framework Overview
   2.1 Architecture
   2.2 Technology
   2.3 Structure
3. Framework Extension
   3.1 Files Examples and Basic Methods Overview
   3.2 Planning the new method
   3.3 Method using Local Files
   3.4 Method using Database
   3.5 Auxiliary Methods
   3.6 Servers
   3.7 Testing
4. Maintenance of the Framework
1. Introduction

1.1 Scope and Purpose

MCDA Framework is a web-based integrated framework that explores Multi-Criteria Decision Aiding (MCDA) methods. The framework focuses specially in usability and flexibility for a better user experience, but also the extension of the framework by adding new MCDA methods at anytime. In addition, it provides project management functionalities for the users to create and manage their own projects and also provides some useful functionalities for a faster and easy usage of the framework.

The main goal of this administrative manual is to guide and help developers regarding the whole framework features, architecture and technologies, plus the respective steps to add a new method successfully. Furthermore, the maintenance of the framework and the addition of extra servers that execute methods separately from the framework. This framework is recommended for developers who have experience in web technologies and especially in MEAN applications, but also have some knowledge of MCDA in order to extend the list of available methods. The MCDA Framework operates on Chrome (recommended) and Firefox browsers.

1.2 Process Overview

The MCDA Framework is a web-based framework developed in MEAN technologies (i.e. MongoDB, Express, AngularJS and Node JS) which communicates with other servers to obtain the results of a specific method or executes the method locally to get the results. In order to use the MCDA Framework, it is required a computer with internet access using a recommended browser. The main pages of the framework provide some basic information, the possibility to create a new account and login into the framework. After the authentication is done, the project management area is opened and here it is possible to create, edit, delete and clone a project. A project executes the method chosen when created and it is where the necessary method data are managed (i.e. where the data are created, deleted, edited, imported, reloaded and exported), but also provides some auxiliary methods to fill specific data fields. After all the required steps are done, the method can be executed and the respective results are saved which can be deleted or exported. When exporting, it comes all in CSV format and inside a folder (zip format). It is possible to see the method steps, but also the current method data which is in two different sections in the project. Furthermore, there are alert messages in some cases like creating or adding or importing data, when they are not done correctly or something is missing. A new method can be added to the framework at anytime by following the required steps that will be described later on. The new method can be executed in the framework (using local files or database to manage the data) or be executed in a separated server (using database to manage the data), which is decided by the developer regarding the type of method that will be implemented and which one is the best way to do it.
2. Framework Overview

This section will give you a starting overview of the framework’s architecture and structure, but also the technologies used in the framework.

2.1 Architecture

The architecture used in the final solution is presented in figure 1.

![Architecture of the MCDA Framework](image)

**Figure 1: Architecture of the MCDA Framework.**

The Web Browser and Application Server (i.e. the green nodes) represent the main infrastructure plus the Database. While the gray node is the Diviz Server (the only separated server at the moment) which executes specific MCDA methods provided by the Diviz framework. In this architecture, the user interacts with the Web Browser and performs the login in order to use the framework. After the User Authentication component approves the access, the respective user interface with the list of projects is displayed. The Projects List component contains all the projects created by the user and respective management (e.g. deleting project, editing, etc...). A Project component has three main components:

- **Data List Management**: the user defines and manages the data required for the method chosen;
- **Method Execution**: the user executes the method using the data that was defined;
- **Results List Management**: the user manages and performs other functionalities to the obtained results.

All functionalities done in the Web Browser by the user will interact with the Application Server through a HTTP/Ajax communication. Basically, the Application Server manages the users, the projects, the project...
data and the project results communicating with the Database through Mongoose. Depending on the
method chosen, the data and results can be managed in the database or in local files (in JSON format).
Depending on the server platform, the data and results can be saved in Local Files.json, which are
separated in a project's folder (identified by their ID) with the respective data and results inside. Mongoose performs the communication between the Application Server and Database. It provides a straight-forward, schema-based solution to model the framework's data which is a easier way to do the management and saves lots of time in implementation. Regarding the Diviz Server, Flask and Python are used and it's a web-based server. However, the technologies used in separated servers (like the Diviz server) can differ (e.g. a server based in Java or PHP, etc...) if and only if, the communication between the MCDA Framework and the server can be done successfully. Basically, the MCDA Framework must provide the project's information to the server and the server must get the data from the Database. Then the server executes the method using the data received and finally the server must send the respective results to the MCDA Framework and save them in the Database.

Regarding the available libraries, they are the following (the libraries provided by the MEAN application are not displayed):

- Application Server:
  - Jsonwebtoken (for security authentication);
  - File-system (to manage local files);
  - Fs-extra (extra utilities to manage local files);
  - Lodash (to work easily with arrays, numbers, objects, etc...);
  - Morgan (a HTTP request logger middleware for node js).

The basic methods, Order By and Order People, are examples regarding the extension of the framework and will be described later on. The difference between these two methods is the data and results location. The Order By method uses the Local Files.json, while the Order People method uses the Database for this subject matter. These examples allow to understand the available possibilities when adding a new method to the framework (including the possibility to create a new separated server like the Diviz Server). A small note, the Application Server communicates with the Diviz Server when the Electre Tri-C method needs to be executed by giving the respective project's information (in order to get the right data from the Database).

2.2 Technology

The technologies used are the base of a MEAN application which consists in using MongoDB, Express and Node JS for the back-end and AngularJS for the front-end (plus HTML and CSS). In addition, it is also used Bootstrap for styling purposes and some essential libraries. Like mentioned previously, the Application Server contains specific local files that provide the data and results of some projects in JSON format (i.e. projects that execute the Order By method). Each project has its own folder (identified by their ID) with the respective data and results inside. Mongoose performs the communication between the Application Server and Database. It provides a straight-forward, schema-based solution to model the framework's data which is a easier way to do the management and saves lots of time in implementation. Regarding the Diviz Server, Flask and Python are used and it's a web-based server. However, the technologies used in separated servers (like the Diviz server) can differ (e.g. a server based in Java or PHP, etc...) if and only if, the communication between the MCDA Framework and the server can be done successfully. Basically, the MCDA Framework must provide the project's information to the server and the server must get the data from the Database. Then the server executes the method using the data received and finally the server must send the respective results to the MCDA Framework and save them in the Database.
2.3 Structure

The framework file structure is displayed in figure 2. The structure figure indicates the 5 major parts of the framework.

First the client folder (blue square) which represents the front-end and contains various folders for the respecting files: CSS, img (i.e. images), js (javascript + angularJS) and libs (i.e. libraries). The js folder contains: 2 libraries (amchart_3 and sortable.js mentioned previously); a controllers folder that contains the controller for the framework pages (i.e. javascript + angularjs functionalities); app.js (i.e. angular application) a set up to use all the components created; and finally appRoutes.js (i.e. angular routes) that puts together which controllers belong to which view. The HTML files are also inside the client folder which represent the the various pages of the framework (not all files are displayed and in replace it’s written “More HTML files” in figure 2).

The config folder (green square) indicates the path of the database to be used (db.js) and the path of the ports to be used (app.js). The server folder (orange square) represents the back-end that has a models folder that contains all the models to be used (schema structures for the database) and a controllers folder which contains the necessary controllers (i.e. functions) between mongoose and node js. The files in the yellow square can contain functions regarding the interaction between the server and the local files (like importData.js), plus some function examples to interact with the database and to create certain data in the database (like createUserProject.js). Finally, the server.js (red square) is the main file of the framework which set up the whole framework together. About the other displayed folders and files: the node_modules folder contains all the libraries necessary for the back-end; the Projects folder is where the local files of each project are maintained; the bower.json tells bower which files are needed (it is possible to use Bower freely) and the package.json tells npm which packages are needed (both files were used when creating the framework file structure for the first time); and finally the README.txt file that displays a brief explanation of the framework file structure.
3. Framework Extension

This section gives you an overview of some aspects of the framework, plus it describes how the new methods are added to the framework and the necessary steps to do it successfully.

3.1 Files Examples and Basic Methods Overview

The framework provides 5 files as example of how the front-end should be done with all the main functionalities and aspects of a project interface (i.e. project method), but also represent the mandatory files that every new method should have at least (i.e. depending on the requirements of the new method the number of can be more than 5 containing the mandatory files). These files will be necessary when adding the new method and their names and description are the following:

- **example_description.html**: how the description section should look like - displaying the project's fields, current method data, a brief explanation about the method and steps needed in order to execute the method successfully;
- **example_data_set.html**: a data set section containing various examples how the data can be displayed (there can be more than 1 data set section if you think it is necessary to separate the new method's data);
- **example_results.html**: the execution and results section where the project should be executed and the saved results displayed, plus presenting the current method data;
- **example-controller.js**: contains examples of all the main functionalities like importing, exporting, changing section, etc… depending on the data storage (i.e. if the method uses the database or local files to save the project's data and results). It is suggested that each section contains its own controller file in case of a big method, in other words, for methods that require many functions for each section it is best to separate the functions according to each section, but still containing the main functionalities (e.g. importing or exporting).
- **example.css**: style sheet containing the necessary main styling of every section.

Like mentioned previously, the MCDA framework has at the moment 3 methods available: 1 MCDA method which is the Electre Tri-C method that is executed by the Diviz server; and 2 really basic methods (that order specific data) that are executed directly in the framework. The purpose of these 2 basic methods is to show an example of a implemented method, that uses local files for the data and results (Order By) and another one that uses the database to save and manage the data and results (Order People). These 2 methods will be mentioned later in the manual when describing the necessary steps to add the new method, as examples to follow each step. Note that for using the database the data fields should be well defined and the data should follow that structure (i.e. you can change the models of the database, but when saving the data it must correspond to the model created and not add new fields or different types of data), while using local files this aspect is indifferent.

3.2 Planning the new method

Before starting the implementation of the new method, it is recommended to plan it first in order to follow all steps in one row and to avoid revisiting previous steps, because something is missing. But also, it might become more difficult to know where the mistakes are or where the missing data should be and it takes much more time to finish implementing the method. This will help you organize, add the new method faster and you won’t forget important data or content. Basically, you should answered these questions before starting the implementation:
MCDA Framework

- What are the main objectives of the new method?
- What are the required data in order to execute the method?
- Is the method going to be executed in a separate server or directly in the framework?
- Should the method use local files or use the database to save the data and results?
- Should there be any auxiliary methods for specific data of the method or are there other auxiliary methods from other implemented methods that can be reused?
- How many sections will the method have besides the 3 main sections (i.e. description section, data set section and execution and results section)?

A small note, it is recommended to save the method data and results in the database, in case the new method is going to be executed in a separate server, because using local files is a more flexible and faster way to manage data, plus it makes the framework less dependent in the database. The database is more useful and easier for separated servers to get the method data and save the respective results comparing to local files. After answering these question, you are ready to start the implementation. There are 2 available tutorials to follow, one is for method that use local files and the other one is for methods that use the database. Jump to the respective section that suits your new method (section 3.3 or 3.4).

3.3 Method using Local Files

The following steps say respect to the methods that will use local files to save and manage the method data and results. Each step describes the implementation of the “Order By” method as an example. Note that you can add new libraries to the framework for both front-end and back-end. Also, most of the code can be copy-paste from other implemented methods that just need some adapting according to the new method.

Planning

The “Order By” method orders a list of data by its attributes which are defined by users and correspond to the required data. The users choose which attribute to order (i.e. choose an attribute from the data given) and type of order (i.e. ascending or descending). It is only necessary 3 sections (description, data set, execution and results) and the method is executed directly from the framework. The method data and results are saved in local files and it is not necessary any auxiliary method.

Implementation

- Back-end (server folder)

1. Create a new folder to save the method data files and results files or used the existing “Projects” folder to save all the necessary local files;
   - The existing “Projects” folder will be used to save all local files (see figure 3). When a project is created, it should be create a new folder with the project ID inside the “Projects” folder. The project folder will contain 2 other folder, “data” and “results” where the method data files and results files will be saved respectively.

2. Create a new controller file containing the required functions between the server and the local files;
   - Create the importData.js file containing the necessary functions between the server and the local files (see example in figure 4).

3. In models/project.js file add single attributes to the project model that you think they don’t compensate be in local files and also add them to the results structure;
• The “Order People” method added 2 attributes, type of attribute and type of order (see step 2 from section 3.4), which the “Order By” method will also need, so they will be reuse. It doesn’t compensate to save these 2 attributes on a local file.

4. In controllers/projects-controller.js file add the local files in the duplicating project (i.e. cloning a project) and delete project functions, but also create a new functions for reloading method data and save/delete results;
   • In projects-controller.js, in the delete function add the code to delete the respective project folder and files. Also, in the duplicate function, add the code to copy the respective projects data files (the results files are not necessary) and put them in a new folder for the clone project (i.e. new project folder with the same folder structure as the original project, like in figure 5). Create new functions (reloadProjectFileMethods function, example in figure 6 and saveResults/deleteResults functions) for reloading method data and to save or delete results.

5. Add the new functions from the controller file (from step 2) in the server.js file;
   • Add all the functions from importData.js file to server.js file (like in figure 7).

![Figure 4: Some ImportData.js functions.](image)

![Figure 5: Clone folders code.](image)

![Figure 6: Reload function for “Order By”.](image)
- Front-end (client folder)

6. Copy the files examples to the respective places in the client folder and change their names;
   - Copy the 5 files example (from 3.1 section) and instead of having “example” on the name files, replace it with the method name (i.e. “OrderBy” or similar): orderBy.css, orderBy-controller.js, description_orderBy.html, data_orderBy.html and results_orderBy.html.

7. Edit these new files according to the method data, local files structure, requirements and auxiliary methods (view other implemented methods as examples);
   - Update the files example according to the “Order By” method. Add new styles in the css file or change the respective ids. In the controller file (i.e. js file) add the necessary methods like the execution of the method, edit the import, export, save/delete results, manage method data and get method data according to the “Order By” method. Update the navigation between sections in the html files and controller file, plus the select all and select none buttons. Regarding the html files:
     - **Description section**: update the method definition and necessary steps according to the “Order By” method; show the correct current method data; update import/export sections.
     - **Data Set section**: display the list of data with the main CRUD table; add the options to create a new data set with the right number of attributes; add the selection of the attribute and type of order for the users to select; update import/export sections.
     - **Execution and Results section**: display the respective results; show the correct current method data; the execution button should execute the “Order By” method; update import/export sections.

8. Finally, in js/controllers/projects-controller.js file add the new method to the list of available methods of the framework and the respective path when the project is opened.
   - Add the “Order By” method to the list of available methods (i.e in $scope.data, like in figure 8) and add the path to the respective description section in openProject function (see figure 9).
3.4 Method using Database

The following steps say respect to methods that will use the database to save and manage the method data and results. Each step describes the implementation of the “Order People” method as an example. Note that you can add new libraries to the framework for both front-end and back-end. Also, most of the code can be copy-paste from other implemented methods that just need some adapting according to the new method.

Planning
The “Order People” method orders a list of people by their names or age (which correspond to the necessary data). The users choose which attribute to order (i.e. name or age) and type of order (i.e. ascending or descending). It is only necessary 3 sections (description, people set, execution and results) and the method is executed directly from the framework. The method data and results are saved in the database and it is not necessary any auxiliary method.

Implementation
- Back-end (server folder)
  1. Create the necessary method data models for the database (i.e. schema structures);
     - Create a person model (person.js) with name and age as fields with the respective types inside the models folder (see figure 10).

  2. In models/project.js file associate the new models (and other single attributes that you think they don’t require a model) to the project model and add the structure of the results that will be saved (i.e. associate the data and results to a project);
     - Associate the person model to the project, plus the attribute and order type that will be defined by the users (like in figure 11). Also, the structure of the results that the method
will generate which is a list of names and ages ordered by the respective attribute and order type (see figure 12).

3. Create the new controller file(s) (inside the respective folder) containing the required functions between the server and the database;
   - Create the people-controller.js file containing the necessary functions between the server and the database (see example in figure 4).

4. In controllers/projects-controller.js file add the new models in the duplicating project (i.e. cloning a project) and delete project functions, but also create a new function for the reloading method data;
   - In projects-controller.js, follow other methods examples and adapt the person model to the delete function (i.e. remove all data from person model associated to the project when is deleted, like in figure 13) and duplicate function (i.e. copy all data from person model associated to the project when is duplicated, see figures 14 and 15). Create a new function (reloadProjectOrderPeople function, see the example in figure 6) for reloading the method data from previous results.

5. Add the new functions from the controller files (from step 3) in the server.js file (like in figure 7);
   - Add all the functions from people-controller.js file to server.js file.

- Front-end (client folder)

6. Copy the files examples to the respective places in the client folder and change their names;
   - Copy the 5 files example (from 3.1 section) and instead of having “example” on the name files, replace it with the method name (i.e. “OrderPeople” or similar): orderPeople.css, people-controller.js, description_orderPeople.html, data_orderPeople.html and results_orderPeople.html.
7. Edit these new files according to the method data, database models, requirements and auxiliary methods (view other implemented methods as examples);
   - Update the files example according to the "Order People" method. Add new styles in the css file or change the respective ids. In the controller file (i.e. js file) add the necessary methods like the execution of the method and manage the method data. Also edit the import, export and save/delete results and get the method data according to the "Order People" method. Update the navigation between sections in the html files and controller file, plus the select all and select none buttons. Regarding the html files:
     - **Description section**: update the method definition and necessary steps according to the "Order People" method; show the correct current method data; update import/export sections.
     - **People Set section**: display the list of people with the main CRUD table; add the selection of the attribute and type of order for the users to select; update import/export sections.
     - **Execution and Results section**: display the respective results; show the correct current method data; the execution button should execute the "Order People" method; update import/export sections.

8. Finally, in js/controllers/projects-controller.js file add the new method to the list of available methods of the framework and the respective path when the project is opened.
   - Add the "Order People" method to the list of available methods (i.e. in $scope.data, see figure 8) and add the path to the respective description section in openProject function (like in figure 9).
3.5 Auxiliary Methods

The auxiliary methods are extra methods inside the project. Their objective is to determine specific types of data required from the method which help users obtaining these specific data, if they are considered to be useful. These methods are just additional code added to the data set sections and should be executed directly in the framework. In order to develop an auxiliary method, the interface and styling should follow the main framework, like in the SRF method (in the criteria set section) in projects using the Electre Tri-C method which determines the weights of criteria. The SRF can be used as a good example of what a auxiliary method should look like. Basically, everything that is required in order to execute this method should be displayed in steps with a brief explanation and in the end a button to execute which automatically updates the method data with the results obtained. If specific data is mandatory to have on the execution, it should be displayed an alert message in case the data is not defined or filled. Also, it is possible to generate graphs to be download using the amcharts_3 library to give a different view of the obtained results. Furthermore, other auxiliary methods used on other methods can be reused by just copying the respective code and paste it on the right data set section of the new method and add the necessary functions to save the obtained results on the specific data (i.e. database or local files).

3.6 Servers

If a method is going to be executed on a separated server, the previous steps regarding the implementation of a method using the database should be followed with the exception of how the execution of the method is done (which should be an interaction between the framework and the server and executed in the server). The server is mostly up to you (i.e. you development the whole server with the technologies you want), but you can see the Diviz server as an example how the interactions between the framework and server are being done. There are various ways to do this, just remember that to execute the method in a separated server, this server must received the respective project data that will be used when executing the method and the obtained results should be sent or saved in the framework’s database of the respective project.
3.7 Testing
You can test the whole framework locally before updating to the actual server by:

- Running the following terminal command inside the MCDA folder: "node server.js";
- Then open the respective link [http://localhost:8080/home.html](http://localhost:8080/home.html) to visualize the framework.

It is a good way for testing the new methods and changes made, but also to prevent mistakes before updating to the main server.

4. Maintenance of the Framework
The maintenance and update of the framework must be done in the server which the framework and separated servers (e.g. Diviz server) are in. The new main functionalities and features to the framework and interface can be added, but if they are meant for all projects, don’t forget that every project method should have that new functions or features. If it is for specific methods only, then the update is done only for the respective method. In addition, the framework must be restarted in order to update the new changes, just go to [http://vps288667.ovh.net:9000/#/hosts/localhost](http://vps288667.ovh.net:9000/#/hosts/localhost). If the updates only affect the front-end, it’s not necessary to restart.

The following links represent the MCDA framework and Diviz server respective version control repository (Github).

- Diviz server: [https://github.com/Verdasca/Diviz_server.git](https://github.com/Verdasca/Diviz_server.git)
Appendix C

User Evaluation - Guidelines

MCDA Framework Evaluation

The MCDA Framework is a web-based integrated framework that explores Multi-Criteria Decision Aiding (MCDA) methods and focuses specially in usability and flexibility for a better user experience and quality, as well as, the extension of new methods. The evaluation main goal is to evaluate the framework’s usability and interface, but also evaluate the user experience. It consists on several tasks to be completed after the main setup is done and finally a questionnaire/survey to be answered. The evaluation should take between 30-40 minutes.

Note: the framework only has one MCDA method available (Electre Tri-C) and two really basic methods (Order by and Order People) at the moment. The basic methods have the objective to help users get used to the framework interface and main functions, but also are main examples to guide developers how to add new methods.

Thank you for your collaboration and time.

Setup

Before starting the tasks, the following setup should be done:

- Have a computer with Internet access;
- Explore the “Import files” folder that was given (contains CSV files that will be needed for some tasks);
- Open a browser: Chrome or Firefox;
- Go to: http://mcdaframework.sysresearch.org/home.html;

Tasks

After doing all the setup requirements, you can now start doing the tasks below on the framework following the order displayed. The User Manual given is just for consulting, in case you don’t know how to do a step from the task. These tasks should be done individually and in the end the respective questionnaire should be answered.

Scenario: Sign up a new account to login into the framework.

Task A

1. Register on the framework (credentials of your choice);
2. Login.
**Scenario:** Create a new project using the Order By method, then explore the main functions by doing the required steps to execute the project method.

**Task B**

1. Create a new project using the Order By method, with the name of “Project 1” and with the notes “First project”;
2. Open the respective project;
3. Have a quick view of the 3 available sections;
4. Read the necessary steps presented on the description section;
5. Edit or define the method data as a list with information from people (data set section):
   (a) Add 3 attributes;
   (b) Named the respective attributes: Name, Age and Height. Also before creating the data set, fill the new data element with a person information (e.g. Ana, 16, 1.65);
   (c) Create the new data set;
   (d) Add 2 more people (i.e. data elements) with all the respective attribute values (your choice, or e.g. Maria, 24, 1.54; Rui, 48, 1.72).
6. Edit the configurations (with attribute type: Age; and type of order: Ascending);
7. Before executing the project (method execution), the results name should be “First results” and then execute the method (all sections must be green, except the description section).

**Scenario:** View the obtained results and explore the export functions on the project created.

**Task C**

1. Export data (i.e. current method data);
2. View the results obtained from Task B (i.e. “First results”);
3. Export “First results” results (only the results).

**Scenario:** On the same project, explore the import function and then execute the project.

**Task D**

1. Import method data using the file “people.csv” inside the “Import files” folder;
2. Edit the previous configurations (with attribute type: Height; and type of order: Descending);
3. Execute the project again with the results name “Second results” and view the results obtained.

**Scenario:** Continue exploring other main functions using the same project from previous tasks.

**Task E**

1. Delete “Second results” (i.e. the results obtained from Task D);
2. Reload the method data from the first results obtained (i.e. “First results” obtained on Task B);
3. Again execute the project (with results name: “Third results”; and notes: “Final execution”) and view the results obtained;
4. Logout from the framework.

**Scenario:** Clone the first project created and execute the clone project.

**Task F**
1. Login;
2. Duplicate the first project created (i.e. clone “Project 1” project);
3. Edit the clone project name to “Duplicated project” and also the notes to “Clone of Project 1”;
4. Save changes;
5. Delete the old project (this is the first project created - “Project 1”);
6. Open the “Duplicated project”, view each section (you can edit the method data if you want), execute the project and view the obtained results;
7. Go back to the projects page (i.e. the page with the list of projects).

Scenario: Create a new project using the Electre Tri-C method, do the necessary steps to execute the project, then view and export the obtained results.

Task G
1. Create a new project with the method Electre Tri-C, with the name of “Electre project”;
2. Open the respective project;
3. Import method data using all the files in “Cars” folder inside the “Import files” folder (each file name corresponds to an available option of data to import, e.g. Criteria -\texttt{criteria.csv}). Import only the 5 options for the respective 5 files (i.e. from all available options, only 5 options are to be selected);
4. Execute the project with the results name of “Cars results”;
5. View the results;
6. Export the results (export everything).

Scenario: Explore other functions provided by the Electre Tri-C method.

Task H
1. Go to the criteria set section and use the available auxiliary method (i.e. SRF method to calculate the weights of criteria):
   (a) Open the SRF method on the criteria set section (i.e. Weight);
   (b) Do the 3 steps required by the method:
      i. Add or drag a white card for the “Price” criterion;
      ii. Choose one ratio $Z$ with the value 7.5;
      iii. Define two decimal places.
   (c) Execute the SRF method;
   (d) Save or download as PNG the first displayed graph;
2. Explore the project a little more (i.e. edit the method data from any section you want, execute the project and view the results);
3. Logout.

Scenario: After finishing all previous tasks, it is time to answer the survey.

Task I
1. Answer the survey.
   Go to: https://docs.google.com/forms/d/e/1FAIpQLSf4JrHC-nW3SQOpZyyeAt18018-agq0NfoN_0-5sgwd7Ym0-g/viewform.
Appendix D

User Evaluation - Survey

Survey - MCDA Framework

MCDA Framework is a web-based integrated framework to explore Multi-Criteria Decision Aiding (MCDA) methods. The framework focuses specially in usability and flexibility for a better user experience and quality, as well as, the extension of the methods list by adding new methods at anytime.

The survey should only take 5-10 minutes, and your responses are completely anonymous.

Thank you for your time and participation :) 

*Obrigatório

Usability

The following questions regard the usability of the framework.

Figure D.1: Survey's introduction and usability section.
1. Rank the degree of difficulty for: *

<table>
<thead>
<tr>
<th></th>
<th>A) Very Difficult</th>
<th>B) Difficult</th>
<th>C) Easy</th>
<th>D) Very Easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Register on the framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 Login</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3 Create a Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4 Edit a Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5 Delete a Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6 Go back to the projects page</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7 Edit method data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8 Import method data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9 Export method data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.10 Execute project (method execution)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.11 View method results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.12 Export method results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.13 Delete method results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.14 Logout</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.15 Use the SRF method (i.e. auxiliary method)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure D.2: Survey question 1.
2. Rank the usefulness of: *
Do you think that these functionalities and features provided by the framework are useful?

<table>
<thead>
<tr>
<th></th>
<th>A) Should not be available</th>
<th>B) Useless</th>
<th>C) Useful</th>
<th>D) Very useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Duplicating projects (i.e. clone projects)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.2 Using auxiliary methods</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.3 Reloading method data used in previous project executions (i.e. previous method results)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2.4 Sections indicating what it’s done and not (i.e. green or red color display of the sections)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

3. At any time, I knew what was happening on the framework.*
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

3.1 Comments
A sua resposta

Figure D.3: Survey questions 2, 3 and 3.1.
4. How satisfied are you regarding the whole experience done on the framework.*
On a scale of 1 to 5, 1 very unsatisfied disagree and 5 totally satisfied, how would you classify the sentence.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Very Unsatisfied</strong></td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td><strong>Totally Satisfied</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1 Comments

A sua resposta

---

**User Experience**

The following questions regard the user experience.

5. I think that I would like to use this framework frequently.*
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strongly Disagree</strong></td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td><strong>Strongly Agree</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.1 Comments

A sua resposta

---

Figure D.4: Survey questions 4, 4.1, 5 and 5.1, plus user experience section.
6. I thought the framework was easy to use. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.1 Comments
A sua resposta

7. I thought that the framework is very efficient. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1 Comments
A sua resposta

8. I felt very confident using the framework. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly Agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.1 Comments
A sua resposta

Figure D.5: Survey questions 6, 6.1, 7, 7.1, 8 and 8.1.
9. I found the various functions in this framework were well integrated. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9.1 Comments
A sua resposta

10. I found the framework and its interface pleasing and friendly. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10.1 Comments
A sua resposta

11. I think that the framework interface is innovative and creative. *
On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree, how would you classify the sentence.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.1 Comments
A sua resposta

Figure D.6: Survey questions 9, 9.1, 10 and 10.1.

Figure D.7: Survey questions 11 and 11.1.
Personal Opinion

The following questions regard your personal opinion about the framework. These questions are not mandatory, but it would be appreciated an opinion regarding the negative and positive aspects of the framework.

12. Negative aspects of the framework?
A sua resposta

13. Positive aspects of the framework?
A sua resposta

14. Do you have any thoughts or ways to improve the framework?
A sua resposta

15. Any last observation or comment regarding the whole framework?
A sua resposta

Figure D.8: Survey's personal opinion section and respective questions (12 to 14).
## Appendix E

### User Evaluation - Results

<table>
<thead>
<tr>
<th>User</th>
<th>Time</th>
<th>Question 1. Rank the degree of difficulty for:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1.1</td>
</tr>
<tr>
<td>1</td>
<td>27 m</td>
<td>D</td>
</tr>
<tr>
<td>2</td>
<td>33 m</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>30 m</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>35 m</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>30 m</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>25 m</td>
<td>D</td>
</tr>
<tr>
<td>7</td>
<td>35 m</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>32 m</td>
<td>D</td>
</tr>
<tr>
<td>9</td>
<td>25 m</td>
<td>D</td>
</tr>
<tr>
<td>10</td>
<td>55 m</td>
<td>D</td>
</tr>
<tr>
<td>11</td>
<td>28 m</td>
<td>D</td>
</tr>
<tr>
<td>12</td>
<td>29 m</td>
<td>D</td>
</tr>
<tr>
<td>13</td>
<td>20 m</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>31 m</td>
<td>D</td>
</tr>
<tr>
<td>15</td>
<td>29 m</td>
<td>D</td>
</tr>
<tr>
<td>16</td>
<td>33 m</td>
<td>C</td>
</tr>
<tr>
<td>17</td>
<td>21 m</td>
<td>D</td>
</tr>
<tr>
<td>Med.</td>
<td>30 m</td>
<td>D</td>
</tr>
<tr>
<td>Mod.</td>
<td>-</td>
<td>D</td>
</tr>
</tbody>
</table>


Table E.1: Users evaluation table: time execution and answers from usability question 1.
<table>
<thead>
<tr>
<th>User</th>
<th>2. Rank the usefulness of:</th>
<th>3.</th>
<th>4.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>9</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>13</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>D</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>15</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Aver.</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Med.</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Mod.</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

2. A) Should not be available, B) Useless, C) Useful, D) Very useful.
3. On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree.
4. On a scale of 1 to 5, 1 being very unsatisfied and 5 very satisfied.

Table E.2: Users evaluation table: answers from usability questions 2, 3 and 4.
### Table E.3: Users evaluation table: answers from user experience questions 5 to 11.

<table>
<thead>
<tr>
<th>User</th>
<th>5.</th>
<th>6.</th>
<th>7.</th>
<th>8.</th>
<th>9.</th>
<th>10.</th>
<th>11.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Aver.</td>
<td>3.8</td>
<td>4.4</td>
<td>4.2</td>
<td>4</td>
<td>4.4</td>
<td>4.4</td>
<td>3.8</td>
</tr>
<tr>
<td>Med.</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Mod.</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>


5. to 11. On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree.

Table E.3: Users evaluation table: answers from user experience questions 5 to 11.