ABSTRACT
The use of MCDA frameworks can be difficult for users who aren’t experts in the MCDA methods. There aren’t 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 don’t provide a large number of available MCDA methods to use and don’t 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.

1. INTRODUCTION
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. However, there aren’t many frameworks that provide many MCDA methods or don’t have the possibility to add new methods to the framework. Also, some frameworks have poor usability and user experience. Furthermore, some of these frameworks don’t provide enough information about the methods and how can they be executed successfully. Some are hard to understand at first and aren’t easy to use (require experience or manuals). In consequence, users may not use the framework often or they might try to find another framework that suits them. 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 and features to improve the usage of the methods and its interface.

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. It’s important to note that after analysing the results from the evaluations made using the first version of the framework that the framework was updated.

2. RELATED WORK
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].

The following three frameworks were relevant for studying and gaining knowledge about the world of MCDA frameworks before developing a new framework. The purpose was to deeply analyse these frameworks and learn about their ad-
vantages, 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.

2.1 MCDA-Ulaval

The MCDA-Ulaval\(^1\) is a free tool programmed in Java that supports MCDA methods from the ELECTRE family algorithms. The user interface is simple and basic in terms of visuals. It’s also well organized by showing a list of current projects and their corresponding data in a 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. Additionally, users can import and export data in a CSV format which is handy for editing and reusing them in other projects. Each project can have multiple sets of the data (e.g. have two lists of criteria each one with different content) and obtain their respective 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 isn’t stated where to start and the users have to explore the tool or read the manual previously. In consequence, the user experience may not be positive at the beginning, because the tool isn’t 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. Another detail to mention is the sections visualization, because when a section is open and isn’t 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 without knowing it.

The import function works well, but in the configuration section it’s not possible to understand what can be imported, if everything or only tables. 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’s possible to export the data, the tool isn’t consistent and doesn’t integrate well the export function (e.g. exporting criteria is done differently comparing with exporting the results, the interface and how to do it’s different). These two functions could be simplified by adding a generic import and export button in the tool, that would be visible to the user. After executing the method, the user cannot change the data and cannot execute it again. The user has to create a new set of data and execute or the user simply deletes all the results so that the data can be editable again. In consequence, 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.

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’s 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.

2.2 SRF

SRF is a software that is used to determine the weights of criteria in the ELECTRE family algorithms and its an implementation of the revised Simos’ procedure \[^{3}\]. The user interface of the software is “old-school” (i.e. classic 90’s software) that follows several steps to achieve the final results. 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. The disadvantages are quite noticeable when a user first uses it: although the interface is simple, it’s quite hard to use or understand how it works at first even if the users follow the steps; it’s not visible where and how to change the values; and isn’t possible to export the results (only printing them).

In general, despite the fact that the software only executes one method and presents a poor user interface, it’s 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).

2.3 Diviz

Diviz\(^2\) is a software tool for designing, executing and sharing MCDA methods, algorithms and experiments \[^{4}\] . The main idea is for users to simply arrange a workflow of methods and data by making the scenario the user wants to execute. 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. The same goes for the functionalities like managing projects, the possibility to get different results by experimenting various data and be able to compare them at the end. 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 \[^{4}\]. This also offers the opportunity to

---

\(^{1}\)MCDA-Ulaval official page, http://cersvr1.lsa.ulaval.ca/mcda/?q=en


\(^{3}\)is a free tool programmed in Java that supports MCDA methods from the ELECTRE family algorithms.

\(^{4}\)is a software tool for designing, executing and sharing MCDA methods, algorithms and experiments.
extend the list of methods, not only by developers, but also by any user that requests his own method.

Despite all of these qualities, there are still some flaws present and some points that could improve the tool. The main issue is the lack of user experience, not only it’s difficult for users (who are aware of the MCDA) to understand what each method does, but it’s also required some experience to get used to. For the first time, users may need to read the tutorials available. Additionally, the process to insert data for the methods isn’t practical and might be frustrating for users. Each type of data 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. 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. 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.

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 add new methods by anyone is a huge step and an useful one, because users who create new methods don’t need to create a new framework to use it. But also, it’s an easy way to share methods (i.e. open source web services). This makes a great idea to explore and find out an easier form to add the new methods. Furthermore, the open source MCDA methods can be reused in other frameworks which might be an interesting idea.

3. PROBLEM ANALYSIS AND SOLUTION DESIGN

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 and Use Cases

The main problems 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. Furthermore, there aren’t 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 [5]. By considering all of these aspects the user experience will increase and any user can use it and understand what is going on. 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 use case presented below (figure 1) presents the final requirements for the front-end side of the new MCDA framework.

![Use Case: Front-end](image1.png)

While the use case in figure 2 displays the final requirements for the back-end side of the new MCDA framework.

![Use Case: Back-end](image2.png)

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’s 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). They will help reduce the implementation time and will guide the developers.

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), so the new framework can use already some MCDA methods 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.

3.2 Framework Architecture and Technology

The architecture used in the final solution, MCDA Framework, is presented in figure 3. The Web Browser and Application Server (i.e. the green nodes) represent 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. 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 (management of the data required for the method); Method Execution (execute the method using the inserted data); and Results List Management (management of the obtained results).

Figure 3: Architecture of the MCDA Framework.

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’s 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 (the Administrative Manual provides more information about this subject). The Order By method uses the Local Files.json, while the Order People method uses the Database. 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’s 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. 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. Regarding the Diviz Server, Flask and Python are used and it’s a web-based server. However, the technologies used in separated servers can differ, if and only if, the communication between the MCDA Framework and the server can be done successfully.

4. SOLUTION IMPLEMENTATION

This section describes in detail the solution implementation and justifies the proposed solution. It was created a
User Manual which contains a brief description of the framework, but also displays a detailed tutorial how every function works.

4.1 Main Pages Section

The main pages of the framework are the first pages to be seen and accessed by the users. It provides some essential information 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 by displaying the main information in the center of the page.

When the sign up page is accessed, 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. The login page 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 an anonymous person to enter.

4.2 Projects Management Section

After the login is done successfully, the users go directly to the projects page (see figure 5) which displays a specific user account as example. In figure 5, the projects are displayed as a table containing some information about each project like name, method chosen, notes and dates (creation date and last update date). 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 (name and method selection are mandatory). It’s possible to cancel the new addition by clicking on the "Cancel" button. For each project, there are four types of functions available (the four colored buttons in the left): Open, Edit, Duplicate 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. 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). 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. If the user doesn’t want to change them anymore, just need to click on the "Cancel" button. Finally, 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. 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 6 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 depending on the method’s needs) and finally the execution and results section.

When a project is opened, it goes directly to the description section (see figure 6). This section displays information about the project (like name, method chosen, etc...), shows a brief explanation of the method and contains the necessary steps to execute the method 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 the whole project by having the cur-
that is dependent in other data types, the respective table to fill or select the value for these types of data. For data and respective field are displayed. The user only needs to choose from a available list, a clear section indicating the mandatory data values like adding a single value or selecting an element. If certain fills are mandatory, that element can be edited or deleted (after being successfully added, that element can be edited or deleted). These functions make it faster 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 (i.e. projects management section) or go back to the login page (i.e. logout). Furthermore, below the sections from the left vertical bar, it’s 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’s shown the respective section which displays the available options of the function chosen. In the import section, the user must choose what type of data to import, but also choose the file that will be imported (the files must be in CSV format). It’s possible to import all the data with the respective files at once. However, if the file format is incorrect or the content doesn’t correspond to the data selected, the import function for that option is canceled.

The same goes for the export section, 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. 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’s a faster way to do so.

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. If certain fills are mandatory, an alert message is displayed and the new element isn’t added until the respective fields are filled. After being successfully added, that element can be edited or deleted (after answering the confirmation message). 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. 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.

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 that uses the SRF method (mentioned in the Related Work section). The user must order the criteria according to their importance and can increment the criterion’s importance. 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). Auxiliary methods can be a huge help for specific MCDA methods, but also this shows the possibility to integrate other existing MCDA frameworks.

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. 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 and displays some information regarding the results (like name, notes and execution date). The last line of the table is where the users can execute the method and fill the results informations (they aren’t mandatory so they can be empty, but they can be edited at any time). Also, the users can see or hide the current data of the project before executing, by clicking on the "View Current Data" button. The users can only execute the method after all data sections are completed. 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).

After the execution is finished, the results are added to the results table. There are two types of results, successful and failure. The only differences between the two are: the failed execution doesn’t 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. However, when this information is shown, 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 for successful executions), Delete and Reload. The View function displays or hides the results information and the Delete function removes the results from the results table and database (after answering the confirmation message). The Export Results function shows a list of options which the user selects what he wants to export from the re-
5. FRAMEWORK EXTENSION

This section describes how the extension of new methods to the framework is done. While maintaining the interface, consistency and main functionalities. It was created an Administrative Manual which describes all the necessary steps in detail 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. Additionally, the respective requirements should be defined, as well as, the necessary auxiliary methods. Then 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).

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. These files contain the required main functions to be adapted according to the method's features. There are in total five files which three of them correspond to the three mandatory sections mentioned in the previous section (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 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 aren't implemented in the framework and the developer considers that they should be added, it's possible to join these new functions. However, if they are treated as main functions then 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 front-end. The developer adapts and makes the necessary changes in the example files according to the method requirements. In short, the developer must: implement the functions needed to call the server functions or to execute the method directly in the framework; adapt each data section to the required data; 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 and exporting according to the available options and data. In addition, the developer must update the select all and select none functions from both in order to update the status of the options from the list. Some examples are provided by the Javascript file example which is where these functions will be implemented.

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, 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 is implemented in Python and Flask. 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 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. Also, the database is the easiest and fastest way to interact and communicate between servers, but most importantly it doesn't contain any security restrictions like when using local files. However, using the database to get and manipulate the data isn't recommended for frameworks that contain many types of data that are constantly changing. So it's recommended to use local files, if the methods are executed directly in the framework.

6. EVALUATION

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:

- Planning and Preparations: 15 minutes
- Implementation: 20 minutes
- Evaluation: 5 minutes

Overall, the user feedback was positive, with some suggestions for improvement in the user interface and data management.
• 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 (not mandatory);
• Setup: after receiving the files, 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, user experience, utility of the framework and obtain their feedback.

6.1 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. The users followed a specific order to execute the tasks which started with the simplest tasks (i.e. basic functions plus using the basic “Order By” method) and the level of difficulty increases when going to the next task. After completing the tasks using the “Order By” method, the users then used an advanced MCDA method (i.e. Electre Tri-C method [6]), but also explored other advanced functions. 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.

6.2 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, while other evaluations were made in person at INESC-ID (Alameda). These choices allowed a closer observation of the user’s movements, better feedback, note the eventual mistakes and time taken to complete the tasks. Regarding the setup for the evaluation, the users needed a computer with Internet access and one of the following browsers Chrome or Firefox. Note that some users read or had a fast look at the User Manual for preparation, but it wasn’t mandatory (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 framework3 with the respective browser in order to start executing the tasks.

6.3 Survey
After all the tasks were completed, each user answered the respective survey (15 questions in total) which were divided in three parts: usability, user experience and 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’re when measured. Usability is concerned with the “effectiveness, efficiency

and satisfaction with which specified users achieve specified goals in particular environments” [7]. 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 [8]. To evaluate the usability it was followed the 5Es dimensions [9]: effective, efficient, engaging, error tolerant and easy to learn. 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 or misreadings they did during the tasks. For the engaging and easy to learn dimensions the users answered four questions from the survey (like ranking difficulty of the functions and level of satisfaction).

Regarding the user experience, it was evaluated by answering the seven questions from the survey to classify the 6 scales according to the UEQ [10] (i.e. User Experience Questionnaire) which are: attractiveness, perspicuity, efficiency, dependability, stimulation and novelty. However, the questions provided by the UEQ weren’t used, but instead it was used and created specific questions inspired by the SUS [11] (i.e. Measuring Usability With The System Usability Scale). Four of the seven questions are from 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 by SUS. Finally, the last four questions of the survey correspond to the user’s personal opinion which weren’t mandatory. These questions cover the negative and positive aspects of the framework, ideas to improve the framework and any last comments that they felt that needed to be expressed.

6.4 Results and Discussion
After doing the 17 evaluations, the respective analysis and discussion were made 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 weren’t comfortable with the Electre Tri-C method or MCDA (i.e. they didn’t know how the method worked) and in consequence some answers were influenced because of it.

Considering the Table 1, in general the users considered most of the available functions very easy to use with the exceptions of the export functions, 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. 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

3MCDA Framework home page (new version), http://mcdaframework.sysresearch.org/home.html
frameworks or users who have previously read the manual, finished the tasks faster than the other users. But overall the time between users wasn’t distant indicating that the framework is efficient and easy to use.

In Table 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 didn’t understand the Electre Tri-C method which causes the results to decrease in average. Overall, the satisfaction for the whole experience was almost very satisfactory between users. It’s 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 didn’t make mistakes or eventual errors, but the ones who did it in the end managed to recover the respective mistakes. In general, it can be concluded that the framework’s usability is quite good and all the 5Es dimensions were archived.

Table 1: Answers from usability question 1.

| Quest. 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 1.10 1.11 1.12 1.13 1.14 1.15 |
|--------------------|-----------------|-----------------|-----------------|-----------------|
| Med.               | D               | D               | D               | D               |
| Mod.               | D               | D               | D               | D               |


Table 2: Usability questions answers 2, 3 and 4.

| Quest. 2.1 2.2 2.3 2.4 3. 4. |
|-----------------------------|-----------------|-----------------|-----------------|-----------------|
| Aver.                       | 3.9             | 4.4             |
| Med.                        | C               | C               | C               | D               |
| Mod.                        | C               | C               | C               | D               |

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.

Table 3: User experience questions answers 5 to 11.

| Quest. 5. 6. 7. 8. 9. 10. 11. |
|-----------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Aver.                       | 3.8             | 4.4             | 4.2             | 4               | 4.4             | 4.4             | 3.8             |
| Med.                        | 4               | 5               | 4               | 4               | 4               | 5               | 4               |
| Mod.                        | 4               | 5               | 4               | 5               | 4               | 5               | 3               |

From 5. to 11.: On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree.
The suggestions provided by the users when answering the personal opinion questions were analysed carefully. Some of them can be considered for future work as they’re 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.

7. CONCLUSIONS

The use of MCDA frameworks can be difficult for users who aren’t experts in the MCDA methods. There aren’t many frameworks that focus on usability or user experience. In consequence, the frameworks can be difficult to use, not understandable and users require experience. In addition, some frameworks don’t provide many MCDA methods to use and don’t have the possibility to add more methods. 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 add new methods while adding useful and good ideas and even reused or integrated other existing frameworks to 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 (who have some knowledge about MCDA). The framework provides useful functionalities (like project management, auxiliary methods, exporting, etc...) and maintains its consistency when using different methods. It’s possible to extend the framework by adding new methods by developers who only need to follow the steps written in the Administrative Manual and perform the required changes. The framework also overcomes some negative aspects and difficulties presented in other MCDA frameworks. But also, it offers the opportunity to reuse or integrate specific methods provided by them like Diviz or SRF. The results from the evaluations made for the first version of the framework revealed that although some users weren’t MCDA experts and the majority never used other 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 pleasing and friendly. Some aspects like stimulation and innovation were the least positive, as well as, the occurrence of some usability issues. These issues were considered relevant enough to be corrected in order to improve the usability. So the MCDA framework was updated and the second version was released. Overall, the users were satisfied with the whole experience and framework. The final solution received positive feedback and presented a good usability and user experience.

7.1 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 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 of the SRF method); 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.

8. REFERENCES