DecSpace Framework

Improvements to the Catalog of Methods

Francisco Miguel Berenguer Todo Bom

Thesis to obtain the Master of Science Degree in
Information Systems and Computer Engineering

Supervisor(s): Prof. José Luís Brinquete Borbinha

Examination Committee
Chairperson: Prof. Nuno João Neves Mamede
Supervisor: Prof. José Luís Brinquete Borbinha
Member of the Committee: Prof. Pável Pereira Calado

June 2018
Acknowledgments

I would like to thank my supervisor Professor Jose Borbinha for the opportunity to work on this project and for the continued support throughout the duration of my work.

I would also like to thank Professor Jose Figueira, Ana Sara Costa, and Antonio Higgs for their help and availability throughout the development of this project.

Lastly, I would like to thank my family and friends who supported me during this last year.
Abstract

Multi-criteria decision analysis (MCDA) is a domain of knowledge that explicitly evaluates multiple criteria in decision making problems. It is an area of growing interest with a wide-array of uses in various fields. This document contains an overview of MCDA theory that explains the main concepts of decision problems, an analysis on other software platforms related to MCDA, an analysis of the state of the DecSpace platform prior to this work and finally describes the work done and how it was validated through users.

The work for this project was done on the DecSpace platform. DecSpace is a web-based client application that features a number of different implemented MCDA methods and a user-friendly interface for non-specialists. It was developed in a project prior to this one and improved through the work done on this project.

One of the more important tools related to MCDA and this project is the diviz framework, which is a software tool for constructing complex MCDA workflows through a visual interface. The diviz platform also has many methods available for anyone to use through their webservices. The main new functionality developed in this project was the integration of DecSpace with the diviz webservices in an effort to increase the number of MCDA methods available through DecSpace.

Additionally, a new local method was added, and in order to improve performance and security the database solution was changed. Finally, some small changes to the interface were made to improve usability and user-experience. A user-evaluation was done on the platform's usability and it managed to achieve the expected usability scores. As a result of this evaluation, some small improvements were made to the platform's interface.

Keywords

Multi-criteria, decision, software, interface, application
Resumo

Apoio Multicritério à Decisão (MCDA Multi-criteria decision analysis) é uma área de conhecimento que evalua explicitamente múltiplos critérios em problemas de decisão. Trata-se de uma área de cada vez mais interesse e com aplicabilidade em vários domínios do conhecimento. Este documento contém uma visão geral sobre teoria de MCDA que explica os conceitos de problemas de decisão, uma análise sobre outras soluções de software relacionadas com MCDA, uma análise sobre o estado prévio da plataforma DecSpace e uma descrição do trabalho desenvolvido e como foi validado através de testes com utilizadores.

O trabalho para este projecto foi feito na plataforma DecSpace. O DecSpace é uma aplicação web que oferece vários métodos MCDA e uma interface fácil de usar para não-especialistas. Foi desenvolvida num projecto anterior a este e melhorada através do trabalho feito aqui.

Uma das ferramentas mais importantes para MCDA e este projecto é a plataforma diviz, que se trata de uma plataforma de software capaz de construir grafos MCDA complexos através de uma interface visual. A plataforma diviz contém também vários métodos disponíveis através de serviços web. A maior funcionalidade desenvolvida para este projecto foi a integração do DecSpace com os serviços web do diviz com o objectivo de aumentar o número de métodos MCDA disponíveis no DecSpace.

Para além disso, um novo método local foi adicionado à plataforma e a solução de base de dados foi alterada num esforço para melhorar segurança e performance. Finalmente, várias alterações foram feitas à interface para melhor usabilidade e a experiência dos utilizadores. Foi feita uma avaliação da plataforma com utilizadores para testar usabilidade que obtiveram resultados positivos. Através destas avaliações, várias melhorias à plataforma foram desenvolvidas graças aos comentários feitos pelos utilizadores.

Palavras Chave

Multi-critério, decisão, software, interface, aplicação
Contents

1 Introduction 1
   1.1 MCDA Concepts and Foundations ........................................... 2
   1.2 Motivation and Objectives .................................................. 2
   1.3 Document Structure ......................................................... 3

2 MCDA Overview 5
   2.1 Introduction to MCDA ....................................................... 5
   2.2 The decision problem ....................................................... 6
   2.3 Types of decision problems ................................................ 7
   2.4 MCDA methods ............................................................... 7
   2.5 The decision aiding activity .............................................. 8
   2.6 Value Measurement ......................................................... 8
   2.7 Example of an MCDA Problem ............................................. 9
   2.8 Conclusions ............................................................... 9

3 Related Work 11
   3.1 OrclassWeb ............................................................. 11
      3.1.1 Structure ............................................................. 12
      3.1.2 Orclass web tool ................................................... 13
   3.2 Decerns ................................................................. 14
   3.3 MCDA-ULaval ............................................................ 15
   3.4 Framework Diviz ........................................................ 17
      3.4.1 The diviz architecture .............................................. 17
      3.4.2 Introduction and Motivation ...................................... 17
      3.4.3 Description and Features .......................................... 18
         3.4.3.A General Use .................................................... 18
         3.4.3.B Workflow Design ................................................ 18
         3.4.3.C Executing methods and results ................................ 18
         3.4.3.D Available algorithm components ............................. 19
         3.4.3.E Comparing methods ............................................ 19
         3.4.3.F Workflow sharing .............................................. 19
      3.4.4 Resources used by diviz ........................................... 19
4 Problem Analysis and Requirements

4.1 Framework Decspace

4.1.1 Requirements and Use Cases

4.1.2 Architecture

4.1.3 Technology

4.1.4 Domain Model

4.1.5 Main features of DecSpace

4.2 Problem Requirements

4.3 Conclusions

5 Solution Proposal and Development

5.1 Technology of the Solution

5.2 Adding new local methods to the platform

5.2.1 Additive Aggregation

5.3 Retrieving currently available Diviz methods

5.4 Calling Remote Methods from the Diviz Server

5.4.1 The first prototype: Calling the Rank Alternatives Values method

5.4.2 The Generic Diviz method

5.5 Moving the Database to a Local Solution

5.6 Improvements to the Decspace Interface

5.6.1 Scalable Method Selection in the Workspace

5.6.2 Identifiable Modules in Workspace

5.6.3 Recognizing Erroneous Input by User

5.6.4 User input conditioning

5.7 Implemented Methods and Example Use

5.7.1 Additive Aggregation

5.7.2 ACUTA

5.7.3 ELECTRE

5.8 Conclusions

6 Solution Evaluation

6.1 Process

6.1.1 Test Sessions

6.1.2 Location
## List of Figures

3.1 OrclassWeb criteria definition interface .............................................. 12  
3.2 OrclassWeb preferences elicitation ..................................................... 13  
3.3 Decerns during construction of a method ............................................ 14  
3.4 ULaval criterion parameters definition .............................................. 16  
3.5 Example of a diviz workflow .............................................................. 18  
3.6 Comparing methods in a single workflow ........................................... 20  
4.1 Use Cases diagram for DecSpace ....................................................... 24  
4.2 DecSpace Architecture ........................................................................ 25  
4.3 DecSpace Domain Model ...................................................................... 27  
4.4 DecSpace Homepage ............................................................................ 28  
4.5 DecSpace Workspace ........................................................................... 29  
5.1 The Additive Aggregation Modal .......................................................... 35  
5.2 Example of an XML method description file ......................................... 36  
5.3 Highlighted search bar in the Generic Diviz method catalog .................. 36  
5.4 Partial view of the Example tab in the ACUTA method .......................... 37  
5.5 Code for processing downloaded XML method descriptions .................. 37  
5.6 Button to download the diviz method descriptions in the Administrator page 38  
5.7 Partial SOAP envelope captured by Fiddler .......................................... 39  
5.8 Code for communication with the diviz server ..................................... 40  
5.9 Rank Alternatives Values workflow .................................................... 41  
5.10 Modal for the Generic Diviz method for the ACUTA method ............... 41  
5.11 Example Output of the ACUTA method run through the Generic Diviz module 42  
5.12 Method Selection Modal .................................................................... 44  
5.13 Example workflow before and after color changes. .............................. 45  
5.14 Error in Additive Aggregation modal with blank field .......................... 45  
5.15 ACUTA method workflow in diviz ...................................................... 47  
5.16 First step of the ACUTA method workflow in DecSpace ...................... 48  
5.17 Second step of the ACUTA method workflow in DecSpace ................. 48  
5.18 ELECTRE method workflow in diviz .................................................. 49  
5.19 First step of ELECTRE workflow in DecSpace .................................... 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.20</td>
<td>Second step of ELECTRE workflow in DecSpace</td>
</tr>
<tr>
<td>A.1</td>
<td>Homepage - Main features highlighted.</td>
</tr>
<tr>
<td>A.2</td>
<td>“Sign Up” page - Steps for creating a new account</td>
</tr>
<tr>
<td>A.3</td>
<td>“Log In” page - Steps for logging in</td>
</tr>
<tr>
<td>A.4</td>
<td>“Log In” page - Forgot password highlighted</td>
</tr>
<tr>
<td>A.5</td>
<td>“Forgot Password” page</td>
</tr>
<tr>
<td>A.6</td>
<td>“Projects” page - Main features highlighted</td>
</tr>
<tr>
<td>A.7</td>
<td>“Projects” page - Create project feature highlighted</td>
</tr>
<tr>
<td>A.8</td>
<td>deleteproject</td>
</tr>
<tr>
<td>A.9</td>
<td>“Projects” page - “Delete All Projects” confirmation message.</td>
</tr>
<tr>
<td>A.10</td>
<td>“Workspace” page - Main features highlighted</td>
</tr>
<tr>
<td>A.11</td>
<td>The message that appears by clicking the “Save Workflow” button.</td>
</tr>
<tr>
<td>A.12</td>
<td>First steps for importing input files.</td>
</tr>
<tr>
<td>A.13</td>
<td>Last steps for importing input files.</td>
</tr>
<tr>
<td>A.14</td>
<td>First steps for importing a workflow</td>
</tr>
<tr>
<td>A.15</td>
<td>Last steps for importing a workflow</td>
</tr>
<tr>
<td>A.16</td>
<td>First step for inserting a method.</td>
</tr>
<tr>
<td>A.17</td>
<td>Last step for inserting the “Additive Aggregation” method</td>
</tr>
<tr>
<td>A.18</td>
<td>The archive of a certain project - Main features highlighted</td>
</tr>
<tr>
<td>A.19</td>
<td>The confirmation message that appears after clicking the “Reload Workflow” button of a certain workflow.</td>
</tr>
<tr>
<td>A.20</td>
<td>Confirmation message before deleting all workflows in the archive of a certain project.</td>
</tr>
<tr>
<td>A.21</td>
<td>An example of a workflow to be deleted.</td>
</tr>
<tr>
<td>A.22</td>
<td>Confirmation message before deleting all workflows in the archive of a certain project.</td>
</tr>
<tr>
<td>A.23</td>
<td>Link to access the “OrderBy1” module’s data highlighted</td>
</tr>
<tr>
<td>A.24</td>
<td>Necessary steps to add new criteria to the “OrderBy1” module.</td>
</tr>
<tr>
<td>A.25</td>
<td>Necessary steps to add new actions to the “OrderBy1” module.</td>
</tr>
<tr>
<td>A.26</td>
<td>Link to access the “Sort1” module’s data highlighted.</td>
</tr>
<tr>
<td>A.27</td>
<td>Necessary steps to add new objects to the “Sort1” module.</td>
</tr>
<tr>
<td>A.28</td>
<td>Necessary steps to reorder objects in the “Sort1” module.</td>
</tr>
<tr>
<td>A.29</td>
<td>Link to access the “GenericDiviz1” module’s data highlighted.</td>
</tr>
<tr>
<td>A.30</td>
<td>Necessary step to select desired method.</td>
</tr>
<tr>
<td>A.31</td>
<td>Necessary steps to import data to the “Generic Diviz” method.</td>
</tr>
<tr>
<td>A.32</td>
<td>The “X” button that closes a module.</td>
</tr>
<tr>
<td>A.33</td>
<td>Confirmation message for deleting a module.</td>
</tr>
<tr>
<td>A.34</td>
<td>Necessary steps to add a new connection.</td>
</tr>
<tr>
<td>A.35</td>
<td>Hovering over a connection.</td>
</tr>
<tr>
<td>A.36</td>
<td>Example of an “Additive Aggregation” Output.</td>
</tr>
</tbody>
</table>
A.37 Initial page of the "Method Catalog". ................................................................. A-19
A.38 Page of the method "OrderBy" in the "Method Catalog". ................................. A-19
A.39 Example view of a selected method from the Diviz Servers in the Method Catalog . A-20
A.40 The “Download Example” button for the “OrderBy” method in the “Method Catalog”. A-20
A.41 Necessary steps to successfully change your password. ................................. A-21

C.1 First part of usability survey ................................................................. C-2
C.2 Second part of usability survey .............................................................. C-2
C.3 First part of survey concerning usefulness ................................................. C-3
C.4 Second part of survey concerning usefulness .............................................. C-3
C.5 Questions regarding control over interface ................................................. C-4
C.6 Questions regarding learning difficulty ....................................................... C-4
C.7 Open ended recommendation questions .................................................... C-5

D.1 Screen showing new search bar icon and placeholder text in “Generic Diviz” method catalog. ................................................................. D-2
D.2 New "Delete all Modules" button in the Workspace. ...................................... D-2
D.3 New placeholder text and buttons for Generic Diviz modal. ........................... D-2
D.4 Multiple modules being added no longer stacks them on top of each other. ........ D-3
List of Tables

2.1 Performance Table ................................................................. 9

6.1 Expected Results for User Evaluation ........................................... 53
6.2 Average Results for User Evaluation ........................................... 54
6.3 Average score in usability survey .............................................. 55
Acronyms

MCDA  Multi Criteria Decision Analysis
DSS   Decision Support System
DecSpace Decision Space
DM    Decision Maker
PDS   Process Development State
ELECTRE ELimination Et Choix Traduisant la REalite
ORCLASS Ordinal Classification
DECERNS Decision Evaluation in Complex Risk Network Systems
MADM  Multiple Attribute Decision-Making
XMCDA MCDA XML
CEO   Chief Executive Officer
HTML  Hypertext Markup Language
MEAN  MongoDB, Express.js, AngularJS, and Node.js
CSS   Cascading Style Sheets
API   Application Programming Interface
CSV   Comma Separated Values
JSON  JavaScript Object Notation
SOAP  Simple Object Access Protocol
Multi Criteria Decision Analysis (MCDA) is a domain of knowledge for evaluating problems with multiple and sometimes conflicting criteria that serves to aid in the decision making process. It is a growing area of knowledge with a vast applicability in many different fields and large number of different methods.

The outcome for these methods will vary depending on which criteria are taken into account and their respective weight or importance, which varies by problem and user. As such, there is no optimal or ideal solution for any given problem and compromises must be found.

Some of these methods can be complicated to use even for professionals that are not familiar with the field, thus the need for software solutions often called Decision Support System (DSS). These are then used to support the Decision Maker (DM) during the decision process. These systems aim to facilitate the use of MCDA methods and provide different MCDA methods that can then be applied to the user’s specific problem.

The platform Decision Space (DecSpace) is an online platform that aims to provide non-specialist users with a way to utilize the power of MCDA through a simple interface and is the platform where development work for this thesis was done. This dissertation has the objective of describing the work done on the DecSpace platform as well as the motivation behind these changes and their objectives. The process behind this development was based on studies done on MCDA, research on the currently available tools and an analysis of what the platform needed to continue growing. Some user-evaluation test were also conducted at the final cycle of development to test out the new features and correct the flaws pointed out by users as well as implementing some of their recommendations.
1.1 MCDA Concepts and Foundations

MCDA is concerned with solving problems by making decisions based on multiple-criteria in a way that supports the decision-maker's unique and personal decision process. Typically there is no optimal solution and it is up to the decision-maker to alter the criteria to his own preference to find a solution that works for his particular problem and satisfies the relevant stakeholders.

In the case of MCDA, solving a problem is often simply finding the best alternative based on the input and criteria from a multitude of alternatives. Depending on the method chosen, there might not be a single solution chosen, but a small set of acceptable alternatives. The typical outputs for the main types of problems tackled by MCDA methods will be covered in the next chapter.

The difficulty with MDCA lies therefore in the nature of having multiple (sometimes conflicting) criteria for a given problem and because they often incorporate subjective information that can be hard to quantify. Any solution found will depend on the user's preferred criteria which must be incorporated into the method. It is therefore recommended that the decision-maker run multiple MCDA methods with varying weights given to the criteria to generate an array of results. From these multiple solutions the decision-maker can select one of them even if there will always be a "trade off" from one solution to another.

1.2 Motivation and Objectives

Although MCDA is applicable to many different areas of knowledge, its use can be complex for users without a high level of knowledge in the methods. The need for software solutions arises from this difficulty, but even some of these tools can be complicated to use by non-professionals.

When evaluating a problem, decision makers may also want to consider different MCDA methods before making their decision. However, these methods are often different in nature from each other and require a different set of criteria and inputs. Without a unified framework, using various MCDA methods for the same problem can be a very time consuming activity if one needs to format their problem and data to fit each individual MCDA method.

The solution proposed by DecSpace is a unified online framework that MCDA methods can be developed for, under a single interface, and standardized input methods. By providing a single interface it can become trivial to apply various MCDA problems to the same problem. The framework also aims to be user-friendly enough so that it is not restricted to advanced users. The user should not need to have knowledge in MCDA methods to use the framework.

With the DecSpace prototype having been developed in another project, the objective of this thesis became to extend the platform in several aspects. The main functionality added to the platform was the interoperability with the diviz web-services, which considerably extended the number of available methods on the platform. Additionally, the extensibility of the platform was tested by implementing a new local method, the database solution was changed and several other small usability changes were made to the platform.
1.3 Document Structure

This dissertation document contains a total of eight chapters and four appendixes.

Chapter 1 contained an introduction to the topic of MCDA, the problems this area of knowledge attempts to solve and how DecSpace as a platform aims to aid in solving some of those problems.

Chapter 2 will properly introduce the field of MCDA and various concepts that are fundamental to its understanding. This section will also cover the various types of problems, the different types of methods, an overview of what goes into the decision aiding process, and concludes with a practical example of applying an MCDA method to a real world problem.

In chapter 3 the various state of the art software tools that deal with MCDA problems will be presented. These tools each present their own set of distinct characteristics including their strengths and weaknesses. This chapter aims to describe these platforms to then critically analyze each of them in terms of technology and usability in an effort to discern what characteristics DecSpace should have.

Chapter 4 will describe the original Decspace prototype in detail since it will serve as a foundation for this project’s work. This section will cover the design and architecture of the platform and conclude by lifting the solution requirements by identifying what the original prototype is lacking and what functionalities would improve its usability and functionality.

Chapter 5 will explain all the work done on the platform in-depth in terms of design and technology. It will also explain the reasoning behind why these changes were considered important to the platform. The final section will explain some of the more prominent methods available in for the platform and why these were considered important to highlight.

In chapter 6 the evaluation process that DecSpace was submitted to will be explained, as well as an analysis of the results and what the consequences of these evaluations mean for the future of the platform.

Chapter 7 will present the final conclusions of the dissertation as well as what future work should be done on the platform in order to improve it further.

The appendixes contain information that supported the user evaluation process. Namely the user manual, evaluation guide, evaluation survey and the final interface improvements.
This section will be looking at the theory behind MCDA problems and solutions, as well as the process of decision aiding, to give context to the importance of these methods. The different types of decisions problems and MCDA methods will also be explained. The final section of the chapter will give a practical example of a MCDA problem and a suggested solution for it.

2.1 Introduction to MCDA

MCDA is a branch of Operational Research that uses mathematical techniques to aid in decision problems. These methods can be applied to various domains of knowledge such as healthcare[1][2], environmental management[3][4], risk management[5][6], governance[7], finance[8], among several others.

The decision aiding process usually involves two main actors. The first is the decision maker who is the one the aid must be provided. The second is the analyst, who is in charge of facilitating the modeling of the decision problem. This process is done with the decision maker in order to include his personal preferences and objectives for the given problem.

Generally, the decision aiding process can be broken down into the following segments [12]:

- **Problem representation:** The problem situation must be explicitly defined, including who has the problem, why it’s a problem, and who is responsible for it. This usually involved identifying all the relevant stakeholders for the situation.

- **Problem formulation:** Formalizing the problem using decision aiding language in an effort to be as unambiguous as possible. This includes the identification of the problem criteria and possible actions to solve it.
• **Evaluation model:** Where the information gathered must be organized in order to retrieve a formal answer to the defined problem.

• **Final recommendation:** The final step is where the output from the chosen model is evaluated and understood to see if it’s a suitable solution.

### 2.2 The decision problem

In MCDA, a decision problem is a complex problem with various characteristics that must be solved in a way that the relevant stakeholders consider optimal. To help solve the problem, a decision maker will attempt to frame the problem with attention to what criteria the stakeholders consider important but oftentimes there is no "optimal" solution to a problem [9].

When faced with a problem the analyst starts off by gathering information on preceding studies and on previous critical points in the decision problem. With the help of the decision-maker, the analyst tries to "frame" the problem by describing the Process Development State (PDS). The PDS is the set of facts and conditions that characterize the problem and is based on previous history and constraints regarding the problem. In this context, "actions" are potential solutions to a problem. When analyzing an MCDA problem, there are four levels that should be considered [10]:

• **Level I:** Object of the decision and spirit of recommendation or participation - In this level we are concerned with how the decision will be modeled and how actions will be differentiated. In this level the analyst suggest a "recommendation" on how the problem should be approached as scientifically as possible.

• **Level II:** Analyzing consequences and developing criteria - This level concerns itself with how the modeling of the problem will affect the process. The analyst must above all construct appropriate criteria and analyze their potential to form the basis for insightful comparisons among the actions.

• **Level III:** Modeling comprehensive preferences and operationally aggregating performances - In this level the analyst aims to define which criteria best capture the essence of the consequences for decision aiding. It also concerns itself with how an action's performance measures on the various criteria and how that translates to a good or bad performance. At this level it is very hard to be objective and not be affected by one’s value system.

• **Level IV:** Investigating and developing the recommendation - At this level a need for formal procedures arise as a way to acquire and process information that will lead to solutions of the problems. These procedures are varied and we will distinguish different types in the following section.

It should be noted that these levels should not be considered to occur in series but instead as separate aspects to be considered. The choices made in later levels can and often do influence choices made
in previous levels. These processes should be defined by the analyst, along with the stakeholders, including the decision maker.

2.3 Types of decision problems

On any given day people face a several different decision problems. However, there have been four main types of decision problems identified [10]:

- **The choice problem:** Where the goal is to select the single best option or reduce the options to a subset comprised of “good” options.

- **The sorting problem:** Where the goal is to sort the various options into pre-defined ordered groups called categories. The aim of these problems is to aggregate options that share similar characteristics for descriptive, organizational or predictive reasons.

- **The ranking problem:** Where the aim is to sort the various options form best to worst by scoring them or comparing them to each other in pairs. These rankings can vary with the different criteria.

- **The description problem:** Where the goal is to describe the options and their consequences. The objective being that describing the options is the first step to understanding them more fully to solve the decision problem.

The MCDA community has also proposed two more problem types:

- **Elimination problem:** A particular branch of the sorting problem where to goal is to eliminate options that are not appropriate.

- **Design problem:** Where the objective is to identify a new possible options that meets the goals of the decision maker.

2.4 MCDA methods

Choosing the appropriate MCDA method can often be an arduous task. None of the methods are perfect nor can they be applied to all the problems. The three major types of MCDA methods are described below:

- **Full aggregation approach:** Each criterion is given a score and these are then combined into a single global score. This approach has the particularity of compensable scores, where a bad score can bring down a good one and vice-versa. (i.e. Analytic Hierarchy Process, Analytic Network Process, Multi-Attribute Utility Theory, Measuring Attractiveness by a Categorical Based Evaluation.)
• **Outranking approach:** In this approach a bad score cannot be compensated by a good score. It also accepts the notion of incomparability between options, which results in a partial order of options. With this approach two separate options may have the same score but different behaviors and characteristics and are therefore incomparable. (i.e. Preference Ranking Organization Method for Enriched Evaluation, Elimination Et Choix Traduisant la Réalité©)

• **Goal, aspiration or reference level approach:** With this approach a goal is defined for each criterion. The option that comes closest to completing all the selected goals is then chosen. (i.e. Technique of Order Preference Similarity to the Ideal Solution, Goal Programming, Data Envelopment Analysis)

2.5 **The decision aiding activity**

The decision aiding activity is based on three fundamental pillars [12].

1. **Actions:** Formal definitions of the possible actions or alternatives for a given problem.

2. **Consequences:** Aspects, attributes or characteristics of the actions that allow them to be comparable to each other.

3. **Modeling of a preference system:** This consists of an implicit or explicit process, that for each pair of actions envisioned, assigns one of the following possibilities: Indifference, preference, or incomparability.

Based on these pillars the analyst should attempt to obtain a coherent structured set of results to guide the decision process and follow an approach that aims to produce knowledge from a certain hypotheses. This approach should be based on models discussed with the decision maker.

2.6 **Value Measurement**

The objective to this approach is to construct a means of associating a number with each alternative, in an effort to create a preference order that’s consistent with the decision maker’s preferences [19]. A preference order must comply with the following rules:

- Preferences are complete - For any pair of alternatives, either one is preferred or there is indifference between them.

- Preferences and indifferences are transitive.

It is also generally advisable standardize the value functions in a well-define manner. This is more easily done for criteria associated with measurable attributes, but can also be done qualitatively.
Table 2.1: Performance Table

<table>
<thead>
<tr>
<th></th>
<th>g1[min]</th>
<th>g2[min]</th>
<th>g3[avg]</th>
<th>g4[avg]</th>
<th>g5[avg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>13000K</td>
<td>3000K</td>
<td>Average</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td>b</td>
<td>15000K</td>
<td>2500K</td>
<td>Good</td>
<td>Bad</td>
<td>Very Good</td>
</tr>
<tr>
<td>c</td>
<td>10900K</td>
<td>3400K</td>
<td>Good</td>
<td>Good</td>
<td>Very Bad</td>
</tr>
<tr>
<td>d</td>
<td>15500</td>
<td>3500</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>e</td>
<td>15000</td>
<td>2600K</td>
<td>Good</td>
<td>Very Bad</td>
<td>Bad</td>
</tr>
<tr>
<td>kj</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

2.7 Example of an MCDA Problem

Let's say a company Chief Executive Officer (CEO) is faced with a decision aiding situation and has to make a decision. They should get the help of an analyst and the CEO will assume the role of decision maker.

In this example the decision maker needs to decide on the new location of a hotel from five different options according to five criteria. The five criteria identified are:

- g1 (Investment costs)
- g2 (Annual costs)
- g3 (Recruitment)
- g4 (Image)
- g5 (Access)

Where g1 and g2 are expressed in thousands of euros and g3, g4 and g5 are expressed in a qualitative scale: very bad (1), bad (2), rather bad (3), average (4), rather good (5), good (6) and very good (7). The values between parentheses are used to code these verbal statements. The weight of each criterion is represented as “kj”. The performance matrix would therefore look as seen in Table 2.1.

From the data in Table 2.1 we can now apply different MCDA methods which will likely yield different results based on method specific factors and the decision maker’s own preferences. It is important that the decision maker and analyst work together to frame the problem and preferences to the best of their abilities for more accurate results.

2.8 Conclusions

This chapter presented a brief description of the theory behind MCDA and some of its methods. The objective of MCDA is supporting the decision maker during the decision aiding process by taking into account the subjective preferences and opinions of all the stakeholders of a given problem.

MCDA can be applied to many domains of knowledge which consist of different data types. This explains the large number of diverse methods and the importance of using them to better understand a decision problem in order to achieve a solution. A simple real world example problem was also shown to illustrate.
This need for varying methods and support of different data types is one of the main driving forces behind this project, which aims to provide various methods in an easy to understand interface.
In this section we will briefly describe some of the current MCDA software solutions related to this project. These tools were chosen based on features that would be interesting to implement in the DecSpace project and their relevance in the area.

### 3.1 OrclassWeb

The Ordinal Classification (ORCLASS) method is focused in classifying multi-criteria alternatives, which are the options for solving the problem, and can be characterized according to the criteria provided [17]. The method differs from other verbal decision analysis methods because it does not aim at ordering alternatives but at classifying all possible alternatives, which are given by the Cartesian product of the criteria values defined for the problem. The method aims at categorizing the alternatives into a small number of decision classes or groups, which are pre-ordered according to the decision maker's preferences.
3.1.1 Structure

ORCLASS methodology aims at classifying the alternatives in a given set: the decision maker needs these alternatives to be categorized into a small number of decision classes or groups, usually two. The first group covers the most preferable alternatives, and the less preferable alternatives belong to the second one. However, more than two classes might be defined for a problem. The method can be divided into three stages: Problem formulation, structuring of the classification rule, and analysis of the information obtained.

- During the problem formulation stage the set of criteria, their respective values, and the decision groups are defined. The criteria values must be sorted in an ascending order of preference from most preferable to least.

- The structuring of the classification rule is done based on the decision maker’s preferences and the classification task is presented as a set of boards. Each cell is composed by a combination of values for each criterion defined to the problem, which represents a possible alternative to the problem.

- Once all the cells are filled, one can analyze the results.
3.1.2 Orclass web tool

This tool is a web-based program built in a web environment in platform Java 1.6 using JSF 2 and runs in server Tomcat 6. It aims to provide an automated way to perform the ORCLASS comparison process of alternatives. OrclassWeb was developed divided into four stages.

1. Criteria and criteria values definition
2. Alternatives definition
3. Construction of the classification rule
4. Presentation of results obtained

![OrclassWeb preferences elicitation](image)

**Figure 3.2:** OrclassWeb preferences elicitation

The ORCLASS system increases its complexity exponentially with the number of criteria and criteria values for each. With the OrclassWeb tool, the user can apply ORCLASS with a greater number of criteria and criteria values.
When using the OrclassWeb tool, the user will go through three stages. At the beginning of the application the user will need to define the criteria and criteria values associated that the alternatives will be compared against. This stage is called “Problem Formulation”. The second stage, called “Construction of the Classification Rule”, involves defining the alternatives and characterizing them according to the criteria values. The next step is the elicitation of preferences stage where the decision maker will answer some questions in order to obtain the scale of preferences. Given these answer, OrclassWeb will divide the alternatives into the classes/groups. As the final step, OrclassWeb presents an interface composed by the groups and the related alternatives preferred for each one in order to allow a complete and more detailed analysis of the results.

3.2 Decerns

Decision Evaluation in Complex Risk Network Systems (DECERNS) is a framework for MCDA or spatial and non-spatial alternatives that uses scalable decision support systems. There are two application described in this project, DecernsSDSS and DecernsMCDA.

The main objective of this project was to develop models and computer tools for aiding in decision making for land-use planning problems. This platform contains different MCDA methods and
DecernsSDSS is a web application built according to Java EE 6 specifications with a three-tier architecture: presentation, application processing, and data management.

- The presentation tier implements the user-interface. It serves to facilitate all the operation connected with SDSS project creation or loading, implementation of MCDA functions, methods and tools, preparation of the output forms and saving the projects for later modification or documentation.

- The application tier coordinates the applications, process commands, and performs calculations. It's also the link between the two surrounding tiers. The main components of this tier are the GIS module/subsystem which include the GIS functions, the MCDA model/subsystem which includes the various MCDA methods and the Models Manager system which includes the models integration components.

- The Information/Data tier contains data in database and XML files and the functions necessary the access these files. The XML files are used to store information on MCDA models, map legends and spatial alternatives.

DecernsMCDA is a scalable DSS for analysis of multi-criteria problems. The main differentiator between DecernsMCDA and other similar systems is the inclusion of several popular Multiple Attribute Decision-Making (MADM) methods in a single framework which is useful for uncertainty treatment. It has been shown to be an effective tool for decision support regarding environmental problems, remediation of contaminated sites and risk based land-use planning.

### 3.3 MCDA-ULaval

MCDA-ULaval is a free single-layered desktop application tool programmed in Java that implements multiple criteria decision analysis algorithms developed by the University Laval. It is a stand-alone desktop application with no connection to other systems or services. Supported decision methods are Electre II, III, Tri-B, Tri-C, Tri-rC and Tri-nC [18].

The system is based on the concept of “project”. Each project has a set of actions and criteria from which subsets are derived. It can also contain many performance tables and decision configurations. It also features the option to choose between cardinal and ordinal levels of measurement for each criterion, as well as maximization or minimization of criteria. Also included is sensitivity analysis of a decision’s parameter giving the lower and upper bond of the interval more stability. Performance tables are imported and exported via CSV files and there is also the option to export as CSV the concordance, discordance, credibility and outranking matrices of a result.

In terms of visuals output, the performance tables an also be displayed through spider charts and rankings in Electre II and III can also be represented in graphs. Other general output such as results and rankings can be displayed in graphs as well, e.g. pie charts.
Figure 3.4: ULaval criterion parameters definition
3.4 Framework Diviz

The DecSpace project borrows heavily from the diviz framework and as such diviz will be described in detail in this section. The diviz software is a tool for modeling, processing, and sharing MCDA techniques [11]. The target audience are users experienced in MCDA methods, ranging from researchers, teachers, and students. The main features of diviz are its modular nature and ease of comparison of results produced by different methods for the same problem.

3.4.1 The diviz architecture

The diviz tool is a classical 3-tier application made up of the client, a component that accesses the MCDA XML (XMCDA) web-services (described in detail later) and a server. The server side concerns itself with executing the submitted workflows, whereas the client side is where the workflows are designed by the user. The web-services are used to communicate between these two sides.

3.4.2 Introduction and Motivation

Research activities in the area of MCDA methods are ever developing, specially in more recent times thanks to the availability of computing power. Many such methods are often available as software programs. However, if a user wants to test out several different MCDA methods, he is often faced with multiple issues.

1. Different techniques will be implemented in different software programs with different interfaces.

2. Different software applications will often require different specific data formats, making it difficult to test multiple MCDA algorithms.

3. Many of these programs are not easily usable or even available.

4. Several programs are not free to use nor open source.

In order to overcome these problems, the Decision Deck project was born in 2009, its objective being to collaboratively develop open source tools implementing MCDA techniques. The target audience for these tools would be practitioners who require MCDA methods in their field of work, teachers who present these methods in courses, and researchers who want to test, share and compare algorithms or to develop new ones.

One of the initiatives of the Decision Deck Project was diviz which is, as mentioned earlier, an open-source software tool made to design complex workflows of MCDA algorithms in an intuitive manner, execute said methods, and share them. The diviz tool also makes use of two other outcomes of the Decision Deck Project:

- XMCDA: A standardized CML recommendation used to represent objects and data structures from the field of MCDA. It’s used to allow different methods to interact with each other by sharing the same standard.

- XMCDA web-services: distributed open-source computational MCDA resources.
3.4.3 Description and Features

This section will focus on describing the diviz software and the resources it utilizes. It will cover the main usage and features, the work methodologies that arise from this tool and finally the external resources on which it relies.

3.4.3.A General Use

The diviz tool’s purpose it to build, execute, and share complex MCDA algorithms in the form of workflows. It allows the construction of these workflows, also referred to as methods, by combining various modular calculation components. This methodology provides several advantages:

- Removes the black box effect of certain methods.
- Provides a better understanding of the methods and their similarities or lack thereof.
- Allows the user to easily test variations of methods by replacing certain components.

Succinctly, it allows users to better understand, interact, and alter know MCDA methods.

3.4.3.B Workflow Design

MCDA workflows are designed through an intuitive graphical interface. Different algorithms are represented by boxes which can be linked together or with other supplementary elements or data files through connectors. As such, diviz does not require any sort of programming knowledge, although it does require a basic understanding of the various modules and how to combine them effectively. The construction of workflows is done by dragging and dropping modules from a list of available calculation elements into the workplace. The relevant data files must also be included in the workflow and connected accordingly.

3.4.3.C Executing methods and results

Once a workflow is completed, the user can execute the method. The calculations needed are not done locally but instead performed on high performance computing servers through XMCDA web-services, mentioned earlier and described in greater detail later. As such, diviz naturally requires an Internet connection to function. The user can check the method’s output and also the results of
intermediate calculations done during the method’s individual steps. This functionality facilitates the fine tuning of the method’s parameters.

3.4.3.D Available algorithm components

Throughout this chapter there have been mentions of the modular algorithm components that make up diviz workflows. These elements are available through web-services and can be divided into four main categories:

1. Calculation components containing aggregation operators, disaggregation techniques, post-analysis elements, etc.
2. methods containing full MCDA methods.
3. visualization components that allow certain input and output data elements to be viewed graphically.
4. reporting components that create aggregated reports of multiple output data pieces.

These elements are documented on the diviz website and their requirements (such as inputs) are also described. These programs allow diviz to reconstruct classical MCDA methods such as ELECTRE, PROMETHEE and UTA-like techniques, as well as some inverse analysis techniques.

3.4.3.E Comparing methods

As mentioned earlier, comparing different MCDA methods is a challenging task since there is no standard for their data formats or single platform where they can be run. Since the diviz tool allows for many different methods to be constructed and has a standardized data format in XMCDA, it becomes a very convenient tool to compare the outputs of various methods. For example, in a single workspace a user can connect the same data source to various methods and compare their results.

3.4.3.F Workflow sharing

The diviz software allows for easy exporting and importing of workflow configurations. This means that a user can create their own method and share their particular algorithmic treatment of data with other users who could then replicate it by importing the workflow configuration on their own set of data.

3.4.4 Resources used by diviz

The diviz software makes use of two other outcomes of the Decision Deck Project, namely the XMCDA data standard and the XMCDA web-services. In this section we will explore what advantages these tools bring to the table.

As stated earlier, one of the main difficulties of using many different MCDA methods is their heterogeneity in how they are programmed and how they process data. The lack of standardization of
Figure 3.6: Comparing methods in a single workflow
these aspects did not allow for the combining of different tools easily. As such, resolving complex
decision problems that required multiple MCDA methods was a very time consuming task.

In order to overcome this problem and be able to run the same problem with the same data through
multiple methods, the XMCDA data standard was defined, which can be adopted by many programs
to make them inter-operable. The XMCDA markup language is written in XML and a set of syntax
rules define its structure. The latest XML schema for XMCDA is available via the XMCDA website.

The root tag contains sub-tags described by five main general categories:

1. Description of the current decision aid project or description of the XMCDA file.

2. Description of the MCDA concept attributes, criteria, alternatives and categories.

3. The performance table

4. The preferences related to the criteria, alternatives, attributes or categories, which can be pro-
vided by the decision maker or as a result of a separate algorithm.

5. Input/Output messages from methods or algorithms such as log messages or input parameters.

It should be noted that these tags are not mandatory, as none are other than the root tag.

The second product of the Decision Deck Project used by diviz are the XMCDA web-services,
which enable easy access to remote MCDA algorithms. Having the algorithms available through web-
services allows for any user with an Internet connection to access them. These web-services have
the following properties:

- Their inputs and outputs are formatted according to the XMCDA standard, which allows them to
  inter-operate.

- They are asynchronous, which allows for a user to submit requests that require a long time to
  compute and access the results at a later time, without the need to stay connected.

- They can be programmed in any programming language (that can run in a Linux machine).

- They are released under an open-source license.

MCDA methods are often constructed by combining a sequence of modular algorithms, as ex-
plained earlier, although this is not always clear. The XMCDA web-services provide the functionality
for these elementary modules. This also works to eliminate the repetition of certain operations. If
many methods can be created from a few elementary algorithms, then the same code for these algo-
rithms can be reused for the various methods.

3.5 Conclusions

This section analyzed some of the software solution to MCDA problems available today. These
platforms were all different to each other and each provided their own set of advantages and disad-
vantages in relation to each other.
MCDA-Ulaval has an interesting project management aspect which is useful for comparing results and has specific tools to help introduce data into the program. However, it’s an offline application and has a strict interface that doesn’t allow multiple methods to be run off each other’s results.

Decerns has the advantage of having a wide array of visualization techniques and also provides many MCDA methods. It suffers by not being able to compare results from different projects directly, as well as not having a data importation feature.

The diviz software boasts many qualities such as its flexibility, usability and the inter-interoperability of its methods thanks to the way workflows are designed. These qualities are something Decspace aims to replicate and were the motivation behind the workspace.
Problem Analysis and Requirements

This section aims to explain what tasks were considered important to the platform and the reasoning behind those decisions. The first section will describe the DecSpace platform in detail in the state it was before development started on this thesis. From there, the requirements for improving the platform were lifted.

4.1 Framework Decspace

DecSpace is a web-based software solution that aims to provide a user-friendly interface for using various MCDA methods for non-specialists. It is a prototype developed in another project [14], which in turn was an improvement over another MCDA framework prototype [15]. The work done on this prototype will serve as a foundation for the development done during this project. This chapter will describe the original DecSpace platform in detail.

4.1.1 Requirements and Use Cases

The main requirement for DecSPace was that it needed to be a web-based platform. This aspect made the application platform independent and more easily accessible by users, since all you need is a device with a modern web browser and an Internet connection. There is no need to install the application and work can be done on separate devices since all your data will be stored in the database.

Another advantage of a web-based solution is more simplicity on the development side. The application becomes more scalable since different aspects of the program can be separated by server and client side. Development complexity should also be kept to a minimum, which is the reason behind the choices made for the technology used, explained in detail in section 5.
DecSpace should also be open-source, to allow other developers and researchers to eventually extend the platform through plugins.

DecSpace should provide MCDA methods with simple to use interfaces and be easily expandable with minimum coding.

![Use Cases diagram for DecSpace](image)

**Figure 4.1:** Use Cases diagram for DecSpace

### 4.1.2 Architecture

DecSpace shares its basic architecture with other MCDA tools such as diviz, Decerns and the original MCDA framework it was originally based on. DecSpace is designed in a classic three-tier architecture comprised of a data tier, an application tier and a client tier. This three tiered solution has the advantage of compartmentalizing certain aspects of the software, such as keeping the method implementations separate from the interface and data sources. The three tiers of DecSpace are as follows:

- **Client tier** - This consists of the user interface that interacts with the system. It is through this tier that a user can send requests to the application tier. It is mostly made up of the Hypertext Markup Language (HTML) pages that serve as an interface and some client-side javascript code that controls the pages' behavior and is therefore located entirely in the web-browser.

- **Application tier** - This is where most of the computational work is done. It communicates with the users through the client tier and connects with the data tier and any other external services
when needed. It is deployed in the application server.

- **Data tier** - This is where the information is stored. This tier communicates with the application tier when it needs to answer data requests.

![DecSpace Architecture](image)

**Figure 4.2: DecSpace Architecture**

### 4.1.3 Technology

In terms of technology, the DecSpace solution is built on the MongoDB, Express.js, AngularJS, and Node.js (MEAN) stack approach. This stack defines itself as an "opinionated full stack Javascript framework" that simplifies and accelerates the development of web applications. The stack is comprised of the following tools:

- **MongoDB** - MongoDB is a free open-source cross-platform database document oriented database program. Unlike SQL, it uses JSON-like documents with schemas, which means the databases can hold different documents with different fields. One of the key features of MongoDB is its scalability in terms of cluster, performance, and data.

- **Node.js** - Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side design to build scalable network applications. It has few I/O functions and can therefore handle many concurrent connections.

- **Express** - Express is a web application framework for Node.js and is considered the standard server framework for Node.js. It provides a robust set of features for web and mobile applications through its API that has many HTTP utility methods and middleware.

- **AngularJS** - AngularJS is the application framework for the front-end development of the application. It is an open-source Javascript based framework that is extensible and interacts well with other libraries.

Since Decspace is a web application, it also uses HTML for formatting and designing the structure of the web pages that make up the application. Cascading Style Sheets (CSS) is also used for visually styling the pages. In addition, JQuery is used to navigate the document and selecting the various
elements to transform them client-side. To help design the web-pages, BootStrap, an open-source framework for building reactive web-pages, was also used.

On top of this technology stack, there are several other libraries and Application Programming Interface (API)s that are used for various purposes in DecSpace. On the front end, DecSpace uses Google Web Fonts 8 for the use of highly customizable fonts; Font Awesome 9 (version 4.7.0) for using expressive icons; InteractJS 10 (version 1.2.6) for dragging and dropping the modules around in the workspace; MathJax 11 (version 2.7.1) for displaying mathematical equations in the CAT-SD method; JSZip 12 (version 3.1.3) used in the features of importation and exportation of the workflows; FileSaver (version 1.3.2) for saving the imported data files; and ngDraggable in order to move the objects in the “Sort” method and the cards in the “SRF” method.

For the back-end side of DecSpace, the most relevant packages are: "client-sessions" for implementing the concept of sessions and cookies; "nodemailer" for sending emails from the official DecSpace email address (which is "decspace2017@gmail.com"); "expr-eval" for evaluating mathematical expressions (used by the CAT-SD method) and "mongojs" to perform the connections to the database.

The data used in the application, including user-data and data relating to the MCDA methods is stored in an online MongoDB service called mLab.

4.1.4 Domain Model

There are a few concepts that should be explained along with their relationships in order to better understand DecSpace.

A user is identified in the database by its email address (which must be unique) and a password, which are used to log in. They also have a name and privacy setting associated with their account. Other details such as the date they first registered and the date of the last login are also stored. A user can have multiple projects, defined by a unique identifier.

Projects also contain information such as their name, user, date of creation and date of the last update. A single project can have several workflows archived by the user and has a unique identifier. Projects are associated to the user that created them and can be either private or public. Private projects are only accessible to the original creator whereas public projects are accessible by anyone.

A workflow is composed of various modules such as input files, method modules, and output modules. These modules can be added to the DecSpace workspace and connected to each other to form executable workflows. There are different types of method modules, one for each available method on the platform.

These concepts are illustrated by figure 4.3.
4.1.5 Main features of DecSpace

This section will explain the main features of the DecSpace platform. For each of these, the relevant webpages to these functionalities will be shown and explained.

4.1.5.A The homepage and basic usability features

The homepage makes a general introduction to the DecSpace platform and is designed to be simple, clean and to captivate the attention of new users. From here, the user can access most of the basic pages where various information is available. The homepage is also where the user is first introduced to the navigation bar, which will be consistent throughout all the pages on the platform, providing a quick and easy way to navigate. The following pages are accessible from the homepage and available to all users:
4.1.5.B User accounts and projects

To the right of the navigation bar there are options for users of the platform to sign-up or login if they have already created an account. Having registered users be able to log in is an important feature that allows each user to have control over their own private projects but also allows them to share their projects with other users through the “Public Projects” page if they so wish. Once logged in, the navigation bar will include some extra links to pages reserved for registered users. These are as follows:

- "My Projects" page: This pages allows a user to view and manage his existing projects, as well as create new ones. The page lists projects according to their names, creation dates, last update, and privacy settings. The projects can be viewed in a table displaying all the information or sorted in a list view according to one of these parameters. From this page, a user can create, open, duplicate, or delete any given project. Each feature is available through a dedicated button next to each project. A button to delete all user projects is also available and all deletion functionalities require a confirmation by the user to avoid accidental deletions. When creating a project, a user must choose the name and privacy settings, which will determine whether the project will be private and only viewable by the creator or public and available to everyone using the platform. Public projects will be viewable in the “Public Projects” page explained below.

- "About" Page: Provides basic information about the platform, including a description, its objectives, characteristics, and most relevant features.

- “Method Catalog” pages: This page contains a list of available methods and their respective documentation. For each method, there is a brief description, as well as an example of input data and the respective output by the method. The user can also download example files in JSON or CSV formats.

- “FAQ” page: This page contains the answers to frequently asked questions about the platform.

- “Contact Us” page: On this page users can find contact information about the developers if they want to ask questions or make a suggestion.
• "Public Projects" page: This page is available to all logged in users and contains information on all created projects that were define as "public" by their creators. The page displays the projects in a similar manner to a user’s projects page but varies in what actions a user can take. A user can open and view any public project but cannot make any changes to them. If a user wants to work on and change a certain project, he can clone a copy of the project to his own projects page.

4.1.5.C Workspace

The workspace is the most important page in the DecSpace platform because it is where users can use MCDA methods and build their own workflows. A user is redirected to this page when he opens or creates a new project. There are various functionalities available in this page to build a workflow. Method modules can be added to the workspace through a dedicated button and the user can also import CSV and JSON files into the workspace to use them in methods. Alternitavely, a user can click on a method module, which in turn will open that method’s specific input modal where the user can insert data manually. Workflows can be saved at any point and reloaded at a later time through the “Archive” button.

Once a user has setup all the methods and input parameters, the workspace can run the methods which will produce the outcome in a method-specific output module.

4.1.5.D Administrative features

The administrative functions of DecSpace are available through a dedicated webpage available only to the administrator account. In this page the administrator can manage all users and projects currently registered. For the users, the following information is shown: email address, name, number of projects, date of the last login, sign up date and whether or not the user is currently logged in. For
each project it describes the name, user, creation date, date of last update and privacy setting. From here, the administrator can delete the data on any user or project.

4.2 Problem Requirements

The main objective of the work done throughout this thesis on the DecSpace platform was to expand the platform’s usability and functionalities. The code base from DecSpace was mostly kept and changes were only implemented when necessary. The objective was to add new functionalities and have them seamlessly integrate into the existing framework.

Most of the changes made were mostly back-end with only a few minor changes made to the user-interface and a few bug-fixes. The main objectives can be summarized as:

- **Add a new local method:** This was done to test whether the platform was extensible when it comes to the local methods implemented.

- **Import XML type files:** The previous version only accepted Comma Separated Values (CSV) and JavaScript Object Notation (JSON) file types so this new functionality had to be added in order to have input files that follow the XMCDA standard.

- **Implement a method calculated remotely:** A method where all the calculations are done in a remote server was implemented to test the extensibility of the platform to use remote methods.

- **Import the diviz method catalog:** Once the remote method had been tested, a new functionality to import the diviz catalog in real time was implemented. The user can now browse all available methods through the diviz webservices.

- **Implement the generic diviz method:** Through this method the user can pick any of the remote methods available through the diviz webservices and call on them through DecSpace.

- **Change the database to a local solution:** With the growth of the platform it became important to have more control over the database. DecSpace used an online database service which was good enough for the prototype but having a local database increases the speed, security, and control over the data.

- **Interface and usability changes:** Various small changes were made to the user interface to deal with the new functionalities as well as some slight improvements suggested by users of the platform.

How these changes were implemented and a more detailed reasoning for their inclusion is explained in section 5.
4.3 Conclusions

This section started by explaining the architecture, technology and the concepts behind the DecSpace platform. When developing the new functionalities, the code base was reused and an attempt was made to make them blend into the existing framework with minimal changes.

The main actors in the DecSpace platform remain the same: anonymous user, registered user, developer and administrator. Although the database solution was changed, the platform maintains the three-tier architecture of client, application server, and database.

This version served as the base for the development of this project and as such had a great influence over the technologies that were used and which functionalities were considered to be lacking. By analyzing this framework in detail it was decided which aspects needed improving and how the new functionalities could be implemented as seamlessly as possible into the existing framework without major changes.

From here, the requirements for improving the solution were lifted and described. The details on how each new functionality was implemented are described in detail in chapter 5.
Solution Proposal and Development

This section will cover the work done on the Decspace platform during this thesis and go into
detail on why certain choices were made. The first section describes the technology used in the
development and the following sections will go into detail on all the functionalities that were added.
Each section will cover a different area that was added or improved in DecSpace.

The development for this thesis followed an “Evolutionary Prototyping” method where new func-
tionalities were added iteratively.

5.1 Technology of the Solution

The technological architecture of the platform remained unchanged from the prototype on which
it was built. The only major change was database solution which was previously an online solution
provided by mLab and is now a local solution. Details on this change and why it was done are covered
in section 5.5.

Since the main changes to the platform were done on the server side, the changes in technology
are limited to new NodeJS libraries. The libraries used were the following:

- **request**: Used to simplify http calls and used to make Simple Object Access Protocol (SOAP)
  requests.
- **xml2js**: Simple XML to Javascript object converter used to parse downloaded XML files.
- **html-entities**: Used to encode and decode the SOAP envelopes as required.
5.2 Adding new local methods to the platform

One of the tasks performed during this thesis was adding more methods to the DecSpace platform. To do so, the procedure of adding new methods to the platform defined by the previous project developer was followed. Some of the methods added during this work had special requirements in terms of input files and ways to interact with the data and as such special cases had to be made during the coding for these methods. This will be explained in detail in the relevant sections. Adding a method to the platform that complies with how other methods are implemented involves a series of steps outlined below:

1. Adding the new method to the methods list modal
2. Create the module of the new method
3. Redefine the "createModule" function
4. Redefine the "createModuleData" function
5. Redefine the "reloadModule" function
6. Create the method service
7. Redefine the "executeMethod" function
8. Define the modal box of the new module
9. Create the controller for the new method
10. Redefine the Data deletion functions
11. Complement the "selectCurrentModule" function
12. Create the Output module
13. Add the method to the Method catalog

Details on what needs to be changed in all of these steps can be found in the report for the original DecSpace prototype [15]. As seen above, adding a new method to the platform involves the developer to change a significant number of files and functions. Although extensible, the platform could be improved by making the methods more modular and separated from the main code-base.

5.2.1 Additive Aggregation

The additive aggregation method was the first method implemented in DecSpace for this project for its simplicity and relevance.

This method takes in two sets of data, one which defines the criteria for evaluation and their respective weight, and another with the various options and their defined scores for each criterion. Both of these sets must be defined in standardized scales for the method to be performed correctly.
although the method does not impose any limitations on the scale used; that is left to the user. The data can be entered manually by the user through a modal that allows this or, alternatively, input files with the relevant data in json or CSV format can be provided. The method is run through a service like all other DecSpace implemented methods and returns a table with the results, ordered from best to worst. The data required for the method to run can be introduced either manually through the method module modal or through input files in the CSV or JSON formats.

This method was added to test the extensibility of the DecSpace platform when it comes to implementing new methods that are processed locally. Although successful, the addition of this method also helped show some of the problems currently facing the platform in terms of extensibility. There are several aspects to adding new methods, explained in the steps needed in the previous subsection, that suffer from being method specific and not a generic solution. This means that although extensible, special cases often have to be made to accommodate new methods which will be detrimental in the long run in case the platform wants to have a significant amount of local methods.

Figure 5.1: The Additive Aggregation Modal

5.3 Retrieving currently available Diviz methods

In order to have up-to-date versions of what methods are currently available through the diviz servers, a new functionality for retrieving them was developed for Decspace. It works by automatically retrieving XML description files, shown in figure 5.2, from the Diviz website which are available publicly and processing these files to retrieve the relevant information. These description files have a common internet address and a unique numerical identifier and are therefore straightforward to cycle through until all method descriptions have been downloaded. Once downloaded, Decspace will process the data, convert the XML descriptions into Javascript objects and store them as JSON files in the Decspace database. This way, it becomes much faster to access and read this data whenever the user needs it without the need to connect to the Diviz servers each time we need to access this information.
All communications with the diviz servers and the Decspace database are done on the server side of the application through NodeJS. All the processing is done server side and only the relevant information is sent back to the client. The server calls by the client browser are made through Express.js.

**Figure 5.2:** Example of an XML method description file

```
provider="UTAA" name="ACUTA" displayname="ACUTA" version="2.0"/>
<documentation>
<description>
 Computes ACUTA method - analytic center for UTA - which provides a set of additive value functions that have a central pos-

</description>
</documentation>
</input>
<delta name="delta" displayname="delta" inoptional="1">  
<description>
 Optional delta value for UTA - delta is the utility gap between two successive alternatives in the preference ranking.
</description>
</delta>
</methodParameters>
</methodDescription>
</vfcds>
```

Users can view the method descriptions through the method catalog. Since there is a significant number of methods available, the user can search for a specific method through the search bar shown in figure 5.3. Once a method is selected, the page will automatically show the method description as well as the expected inputs and outputs. This information is derived from the XML description files and generated automatically. All information contained in these pages was written by the method’s creator.

**Figure 5.3:** Highlighted search bar in the Generic Diviz method catalog
The process of downloading the XML method descriptions is not supposed to be executed frequently for various reasons. For starters, the diviz methods are not updated very frequently so there is no advantage to downloading and processing the same files repeatedly. Another reason is that the process is slow, dependent on network speed and users should not be subjected to wait for it whenever they want to access the documentation for these methods, which is why the data is stored in the DecSpace database. Lastly, this process requires the Diviz servers to be available. While they usually are, it would still become an unnecessary dependency.

```javascript
if (typeof result.program_description !== 'undefined') {
  var methodParams = {
    name: '' + result.program_description.program.$.name,
    version: '' + result.program_description.program.$.version,
    provider: '' + result.program_description.program.$.provider,
    description: '' + result.program_description.documentation.description,
    contact: '' + result.program_description.documentation.contact,
    inputs: [],
    outputs: []
  };
  var tempInput = [];
  tempInput = tempInput.concat(result.program_description.parameters.input);
  tempInput.forEach(function (el) {
    var new_input = {};
    new_input.name = '' + el.$.name;
    new_input.isOptional = '' + el.$.isOptional;
    new_input.description = '' + el.documentation.description;
    new_input.xmoda = '' + el.xmoda;
    methodParams.inputs.push(new_input);
  });
}
```

**Figure 5.4:** Partial view of the Example tab in the ACUTA method

**Figure 5.5:** Code for processing downloaded XML method descriptions
The process of downloading and processing the diviz method documentations is therefore exclusively an administrator privilege and can be done through a button on the administrator page. Users can still access the most current method descriptions which will always be stored on the diviz database.

![Administrator]

**Figure 5.6:** Button to download the diviz method descriptions in the Administrator page

This functionality was added in preparation for the Generic Diviz method, which is described in section 5.3.2.

### 5.4 Calling Remote Methods from the Diviz Server

One of the main features developed for the Decspace platform for this thesis was the ability to call remote methods from the diviz servers which are made available through public web-services. Adding this functionality was considered crucial because it would add a great number of methods to the platform, making it more versatile and robust as a result.

A basic proof of concept of this feature had been developed in the original prototype [14] but it came with several limitations. Only a single method was available and to call it, a python script made by the diviz team had to be used. In order to use this script, a web-framework running on a server in python had to be used to run the code (flask), as well as a file-management system to both feed the input files and read the response file. The system was therefore changed as to not rely on a separate python server and file-system.

The communications with the Diviz server are all done through standard SOAP requests, where all the relevant information is contained within the envelope itself. There are no input or output files being transferred, the input files for any given method are opened and written into the SOAP envelope when requesting a method and the same is done for the results, where all the relevant data is contained within the response envelope. The solution was then to write an algorithm that builds custom SOAP envelopes with all the relevant data and then sends them to the appropriate web-service address. This is done in NodeJS and the data can come from either an input XML file or from data from within the application. With this solution we remove the need for the flask framework, the file-system, and the need to rely on the provided python script, allowing for a finer control of how the methods are called and what is being sent.
In order to figure out the exact format the SOAP envelopes needed to follow to be accepted, the python script provided by the diviz team was analyzed in depth. The software tool Fiddler was also used to track what exact packets were being sent and how they were encoded in order to program DecSpace to build the envelopes to those specifications.

![Figure 5.7: Partial SOAP envelope captured by Fiddler](image)

All the coding dealing with outside communications is done on the NodeJS server, as illustrated in figure 5.8. The client browser sends the relevant information to our server, which in turn processes it into a properly formatted SOAP request according to the specifications of the diviz server and sends the request. The NodeJS server then receives the response and processes it before sending it back to the client browser. This was done so that the client side of the application is unencumbered by the execution of the communicating methods and in order to use necessary libraries made available to NodeJS. The communication between the client browser and our NodeJS server is done through Express.
5.4.1 The first prototype: Calling the Rank Alternatives Values method

The first remote method implemented is called the "Rank Alternative Values" method that exists on the diviz servers. It's a simple method used for demonstration purposes that simply orders a set of "alternatives" based on their "overall values". It essentially constructs an ordered list based on the inputs.

This particular method works with raw XML data that follows the XMCDA format, provided by the user, and constructs a properly formatted SOAP envelope with the request to the Diviz server. The user can alter the data, but to do so they must work directly on the raw XML text, which can be confusing and not user-friendly. The recommended way to use this method is to import the data directly from a file. If the user is familiar with XML data formats, they will be able to change values directly in DecSpace after importing the raw XML data but this functionality is not recommended as it’s not user-friendly and is there for testing purposes only. Although this is a remote method, a user of the platform will interact with it in the same way he would any locally programmed method, as shown by the workflow in figure 5.9.
This method was programmed as a first prototype to test the connection between the DecSpace platform and the diviz servers and as such has limited functionality and interactivity.

5.4.2 The Generic Diviz method

This new method allows the user to call any method currently available and listed in the Diviz webservices website. It can be used through the workspace like all other methods but has a few key differences. Since this method does not represent one single method and instead can be used to call upon a large number of them it has a dynamic interface modal for user input. When the input modal is open, the user can search for the method they want to use through the search bar and the relevant necessary inputs will appear once the user has selected a method. The user may then fill the form with the relevant data by hand but since the files necessary to run diviz methods have to follow the XMCDA standard a better approach would be import the files directly, which can be done through the import files button.

Once the user has input all the relevant data, they can run the method through the workspace like they would any other. The communication with the diviz server is then done through the NodeJS.
server and the output is produced in the output box in the workspace. The output contains a message from the server which informs the user if the method was run successfully, as well as the answer body. If the input data is incorrectly formed or if the server is busy the message will inform the user. The answer body will contain the relevant data, which varies by method, and also follow the XMCDA data format so it might not be easy to read for novice users. However, since the use of this generic method requires the user to have a basic understanding of the XMCDA format to submit the input data, it is assumed they will be able to interpret the results. An example output is shown in figure 5.11.

![XML Response](image)

**Figure 5.11:** Example Output of the ACUTA method run through the Generic Diviz module

### 5.5 Moving the Database to a Local Solution

The previous version of Decspace used the website mLab.com as a database service for all functionalities including user login data, project data, and data related to the Inquiry method. This service functions as a cloud-based MongoDB database.

This solution has the problem of relying on an external resource outside our control which comes with a few disadvantages. The main problem is relying on the current state of this external service. If the service is busy or offline then the application will not function. The second problem was the difficulty in having multiple instances of the database attributed to different developers. The solution was tied to a single account in mLab.com and a single database, meaning that every local distribution of Decspace was accessing the same database. A third problem was that the execution time for various functionalities was negatively influenced by having to wait on this service, sometimes for several seconds for even the most basic functionalities. Finally, having a remote solution also meant less control over the files and presented some security concerns although we won’t address them for now.
The solution to this problem was to create a local MongoDB database in one of the servers provided by IST. To so so, the 3.6 version of MongoDB was installed on the Linux server. The new database has authentication control enabled to protect it from anyone other than the DecSpace developers from accessing it. The connection to the server is done automatically by the DecSpace platform and it's possible to define which database the build will access.

Several instances can be created and access to them can be distributed to the different developers and users of the platform. This way multiple instances of the platform could run independently of each other and the development of the platform would be kept separate from the current deployed version. The main advantages to this solution are access speeds, finer control, and potentially better security if the need arises.

5.6 Improvements to the Decspace Interface

This section will briefly explain various small changes made to the Decspace interface in order to improve the user experience or add some new non-critical functionalities to the platform. The need for these changes were identified by Decspace users and implemented throughout the project development as the need arose.

5.6.1 Scalable Method Selection in the Workspace

As the number of available methods grew it became apparent that the current solution for selecting them (using a simple drop-down menu) would be insufficient. As such, the drop-down menu was replaced by a button that brings up the method selection modal, a new component that displays all available methods in a dynamic table and split into two groups: the local methods and the remote methods (shown in figure 5.12). When expanded, the user can see a small description of the selected method, along with all the necessary inputs for it to work. There is also a link that will redirect the user to the method catalog, which opens in a new tab as to not lose the workspace page, where a more detailed description of the method and an example can be found.
5.6.2 Identifiable Modules in Workspace

I order to better separate the different types of modules from each other in the workspace, it was decided to change their overall appearance based on their type. In figure 5.13 we can see the same method workflow before and after the changes to see that it is now easier to identify what modules are what type at a glance. The input files are now a light blue whereas the output files are a light green color. The method modules remained the same gray color.
5.6.3 Recognizing Erroneous Input by User

In order to avoid having the users fill out the method input data incorrectly, when inputting them manually through the modals, some safeguards were implemented to make sure the user knows that there are mistakes in the input data. This will show by highlighting the relevant field in red as seen in figure 5.14 when a user attempts to leave a required field blank.

5.6.4 User input conditioning

Although some measures were taken to notify users when the method inputs are incorrect, some methods also condition the user to make it impossible to enter invalid data manually. For example, in the Additive Aggregation method, it's not possible to input non-numerical values into the "weight" and "score" fields. This was done to minimize user error.

If the user inputs data directly from a file there is no guarantee the relevant fields will be properly filled and as a result the method may not run or return incorrect results.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value-Score</th>
<th>Processor-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neuro</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.14: Error in Additive Aggregation modal with blank field
5.7 Implemented Methods and Example Use

This section will go over some MCDA methods chosen as the most relevant for the DecSpace platform.

The first is a local method developed to test the extensibility of the platform.

The remote methods are all available through the Generic Diviz method so long as the user inputs the data in the correct format. The method calculations are done in a remote server and the answer is sent back as an XML file, viewable in the DecSpace platform. Example input files for all available diviz methods are available to download as an effort to make the use of these methods simpler, since users will have to adhere to the XMCDA data format which is not always intuitive to write.

It should also be noted that many of the services available through Diviz are not full MCDA methods but instead modular algorithms that perform specific tasks. By linking these modules together the user can create a full MCDA method as seen in figure 5.15. This was done to make the extensibility of the platform easier since simple modules can be used in various methods. The example methods in this section will show how to construct these more complex workflows through DecSpace.

5.7.1 Additive Aggregation

The Additive Aggregation method is important because it is one of the most basic forms of an MCDA method and serves as a basis for understanding more complex methods.

MCDA problems have a set of alternatives as results (which may or may not be a discrete list), and a set of criteria through which these alternatives will be compared. To do so, one must construct some sort of "model", which describes an MCDA method. These should incorporate two primary components:

- Preferences for each criterion that describe their relative importance.
- An aggregation model that allows comparisons and to combine preferences.

This aspect of modeling the problem will be influenced by the decision maker's judgments and preferences, which is not always easy since they often have difficulties in understanding what they want to do.

Some criterion can be easily converted into an objective numerical scale. For ease of discussion we will assume that the criteria in the following example follow this rule and that increasing values are preferred. An important distinction between objective measures and their respective value must be made, however. Some measures will be non-linear and their "objective value" cannot be taken as truth, since preferences very rarely align linearly with the objective values. A clear example of this are "preference thresholds", where certain values are unacceptable and an objective number might not be able to transmit this notion. Some criterion do not lend themselves to be easily classified numerically since they are more subjective in nature.

An additive aggregation is a way to demonstrate how a basic MCDA method will work that can be easily explained and understood by decision-makers from a wide variety of backgrounds. This can be
constructed in the form:

\[ V(a) = \sum_{i=1}^{m} w_i v_i(a) \]

Where:
- \( V \) = score
- \( a \) = alternative
- \( m \) = number of criteria
- \( w \) = weight
- \( v \) = value

Succinctly, to perform the additive aggregation method the decision maker needs to attribute a score for each criterion on a scale that is the same for all other criteria. A weight is then given to each criterion, also on a scale equivalent to all other weights, and the alternative’s total score is calculated by adding the product of each criterion’s score and weight. It is the simplest MCDA method and crucial for understanding how subjective to the decision maker MCDA methods can be. By slightly altering scores and weights one can produce many different results.

### 5.7.2 ACUTA

The Analytic Center for UTA (ACUTA) provides a set of additive value functions that have a central position in the polyhedron of admissible value functions, given some preference information (preference relation) [22]. This method determines value functions based on a ranking of the alternatives given by the decision maker. The ACUTA method workflow as seen in diviz is shown in figure 5.15.

![Figure 5.15: ACUTA method workflow in diviz](image)

To replicate the workflow in the DecSpace framework, we start by adding the ACUTA method through the “Generic Diviz” module, filling in the relevant data through the modal and executing the workflow as seen in figure 5.16.
Once that method has run, we can open the Output module to get the "ValueFunction" output and use it as an input for the "Compute Normalized Performance Table" method. This method also takes in some other inputs used in the ACUTE method module. After filling in the relevant data, the user should remove the first "Generic Diviz" module to avoid it running again and execute the workflow, as seen in figure 5.17 to obtain the result.

Finally, the user can use the output from the "Compute Normalized Performance Table" module as input to compute the "General Weighted Sum" method. Unfortunately, DecSpace does not currently feature any method to visualize the data by plotting it like the diviz software.

5.7.3 ELECTRE

There are two main parts to an ELimination Et Choix Traduisant la REalite (ELECTRE) application: first, the construction of one or several outranking relations, which aims at comparing in a comprehensive way each pair of actions; second, an exploitation procedure that elaborates on the recommendations obtained in the first phase[21]. The nature of the recommendation depends on the problem being addressed: choosing, ranking or sorting.

The Electre Methods are usually used to discard unacceptable alternatives to the problem after which another MCDA to select the best one is used. The Advantage of using the Electre Methods before is that it saves time by eliminating unacceptable alternatives.

In this section we will looking at an example method workflow constructed from various modular methods from the diviz webservices. We will use the "Generic Diviz" method to reconstruct the workflow depicted in 6.4 in Decspace.
We start by creating two generic diviz modules for the "ElectreConcordance" and "ElectreDiscordance" methods. We fill in the data for these methods and run the workflow, which will produce two output modules, as seen in figure 5.19. From here, we create a new "Generic Diviz" module to call the "ElectreOutranking" method, and use the data from our respective output files as input data. To do so, we must copy the relevant data from the output module and paste it into a file to be imported in the "ElectreOutranking" module, or we can simply paste the data directly into the input modal for this module.

Once the method data is inserted, we execute the workflow again to run the "ElectreOutranking" method. It should be noted that if the previous modules used for the "ElectreConcordance" and "ElectreDiscordance" are kept in the workspace they will also be executed again. It is advisable to delete the modules once we no longer need them since we can just keep the results in the output modules, as seen in figure 5.20. Once executed, we now have the data from the "ElectreOutranking" method. If the user so chooses, he can also run the "cutRelation" method to cut a fuzzy relation at a given threshold. It should be noted that we will not be running the "plot" modules since these are used for graphical interpretations of data which DecSpace is currently unable to produce.
5.8 Conclusions

In this chapter, the main changes to the platform are described in detail.

The design philosophy behind all this project’s solution development was to extend the platform with minimal changes to the user experience and interface. The technological changes were limited to the database solution and a few NodeJS libraries that were imported to the project as needed.

The main features developed for the platform were presented as well as the process involved with coding them. The main focus of the thesis, the ability to call remote methods from the diviz server, expanded the platform’s catalog of available methods considerably but managed to integrate with the local methods. These remote methods are called by a user the same way one would call a local method. This followed the design principle of the platform of being as user-friendly as possible, although some knowledge of data formats is needed to use the “Generic Diviz” method.

A few of the minor changes to the interface aimed at improving the user-experience are explained with a few examples given.

In the final section we showed some examples of the new implemented methods on the platform. The additive Aggregation method is considered important because it serves as a baseline for MCDA methods. The ACUTA and ELECTRE methods are made available through the diviz servers and are more complex and intended for advanced users and were selected due to their popularity in the MCDA domain of knowledge. They also serve to prove that DecSpace is capable of executing methods composed of multiple modules where the input data is dependent on other modules, although the current solution is not user-friendly.
The role of evaluation is to assess the system design and test the system to ensure that it behaves as expected and meets user requirements [20]. Evaluation of a system is also something that should be done throughout the system’s development as opposed to a single evaluation once the system is complete. This was done throughout the project’s development since the development method was akin to evolutionary prototyping. This evaluation will take into account several aspects such as how easy the system is to learn, its usability and the user’s satisfaction with it.

6.1 Process

The system evaluation was carried out with the objective of identifying possible issues with the system’s usability, functionality, and user-experience. The evaluation process follows an “experimental evaluation” and was conducted on 20 participants. The following sub-sections describe the various aspects of the evaluation in detail.

6.1.1 Test Sessions

Prior to the tests, the coordinator sent the user manual and test guide to the participants. They were asked to look through the manual before beginning the tasks to get a chance to understand what Decspace is and how it works. Each test session consisted of the following phases:

- **Preparation**: This consisted of the coordinator and participant were both ready to begin the evaluation and had access to the system, evaluation guide and user manual.

- **Introduction**: The coordinator would briefly explain the purpose of the system and the basic concepts and uses behind MCDA. The participant could, during this phase, ask any questions to clarify their understanding of the topic and system.
• **Task completion**: During this phase the participant would attempt to clear all the tasks that were defined in the evaluation guide. During this phase the coordinator is silent and taking notes of the various relevant metrics.

• **User Experience Survey**: The participant would answer a quick survey about their overall experience with the system and have a chance to make suggestions for some future improvements.

### 6.1.2 Location

The tests were done in either the IST Taguspark campus or through skype.

### 6.1.3 Participants

The participants chosen to take part in this evaluation were all considered potential users of the platform that had no prior experience with Decspace. Although none of them had experience with MCDA software, they did have some knowledge of MCDA concepts and software systems. The tasks proposed involved using some simple methods and did not require any specific knowledge of either computer systems or MCDA. A total of 20 users took part in the test, 7 of which were knowledgeable in the field of MCDA.

### 6.1.4 Schedule

The test schedule varied with the availability of each participant. The evaluation was done during late March and early April of 2018.

### 6.1.5 Coordinator and Observer

The author of this dissertation and developer of Decspace (Francisco Todo Bom) acted as the coordinator and observer during these test sessions.

### 6.1.6 Help

At any time during the test sessions, the participant could consult the user manual if they were having difficulty completing a certain task and in case that proved insufficient, ask the coordinator directly for some assistance. In case either of these measures proved necessary, the coordinator/observer recorded what caused these difficulties.

### 6.1.7 Variables and Metrics

During the test sessions, the following metrics were recorded and analyzed:

• **Time to complete each Task**: Measured by simply timing how long each participant took to successfully complete each task. This metric is relevant because it is indicative of the performance of each participant in each task which helps identify which tasks are considered harder
Table 6.1: Expected Results for User Evaluation

<table>
<thead>
<tr>
<th>Task</th>
<th>Successful Completion</th>
<th>Time to Complete (minutes)</th>
<th>Non-Critical Errors</th>
<th>Critical Errors</th>
<th>Help (User Manual)</th>
<th>Help (Coordinator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>1</td>
<td>1:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Task 2</td>
<td>1</td>
<td>1:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Task 3</td>
<td>1</td>
<td>2:00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Task 4</td>
<td>1</td>
<td>4:30</td>
<td>1.5</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Task 5</td>
<td>1</td>
<td>3:30</td>
<td>2</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Task 6</td>
<td>1</td>
<td>4:00</td>
<td>2</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

and more complex. A task with consistently high complete times might warrant a change in design whereas one that is easily completed in a reasonable time frame is considered adequate.

- **Successful task completion**: Records how many of the given tasks each participant managed to successfully complete according to the success requirements defined. If a certain task is proving difficult to complete successfully for many participants then the system design is flawed.

- **User errors per task**: This metric will measure the number of mistakes a user makes when completing a task and two different types of user errors were considered. Common and non-critical errors occur when the participant took unnecessary steps during a task. An error is considered critical if a mistake that is not easily correctable was made and compromised the task completion.

- **Help per task**: If a user was unable to complete a certain task, they could resort to two types of help and both of these were measured separately. The user could get help from the user manual or, if that proved insufficient, ask the coordinator.

- **Usability and User Experience**: This metric was measured through a survey conducted at the end of the test session and aimed to map some qualitative attributes into a more objective scale. The survey can be found in Appendix C.

- **Recommendations**: This is the only open ended metric measured and allowed each participant to write some quick observations and recommendations on how they believe the system could be improved.

### 6.1.8 Objectives

The objectives for what would be considered a successful evaluation were created based on the expert opinion of the system designer and programmer, as well as the results from the previous evaluation of DecSpace. The following were considered to be an expected result for each metric:

In terms of the usability scores derived from the questionnaire, an average of 3.5 points was expected.

### 6.1.9 Evaluation Success Conditions

The system would be considered successful if 80% of the defined objectives were cleared.
Table 6.2: Average Results for User Evaluation

<table>
<thead>
<tr>
<th>Task</th>
<th>Successful Completion</th>
<th>Time to Complete (minutes)</th>
<th>Non-Critical Errors</th>
<th>Critical Errors</th>
<th>Help (User Manual)</th>
<th>Help (Coordinator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0:57</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1:15</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2:14</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4:05</td>
<td>1.37</td>
<td>0.05</td>
<td>0.05</td>
<td>0.10</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3:15</td>
<td>1.95</td>
<td>0.05</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3:38</td>
<td>1.47</td>
<td>0.55</td>
<td>0.11</td>
<td>0.45</td>
</tr>
</tbody>
</table>

6.2 Results Analysis

The user-evaluation process described above was conducted on 20 participants. The average results for each of the metrics can be seen in table 6.2.

The results shown in table 6.2 confirm that most tasks were completed, on average, slightly below the expected time with the exception of task 3. This task required the use of the “Generic Diviz” method catalog which proved to have an unintuitive interface for users. The improvements made to mitigate this problem are described in the following section.

In terms of errors made, tasks 4, 5, and 6 are the only ones with notable problems. However, these were expected since the participating users were new to DecSpace. Critical errors were kept low with the exception of task 6, which had many users commit the same mistake when attempting to fill out data for the “Generic Diviz” module. Since users were unaware of their mistake, this led to failure to complete the task and counted as a critical error. However, the coordinator would then intervene and inform the user of their mistake, after which they could successfully complete the task. The cause of this error has since been corrected, as detailed in the next section.

The manual and coordinator help was also kept to a minimum with the exception of task 6 which had users commit a mistake and not realizing it. This was due to the nature of the task, which involved using a diviz method available remotely that complied with the XMCDA data format. As such, novice users were unable to tell if the result presented was correct or not. As a result, the coordinator had to step in and explain their mistake and have them repeat the task correctly.

In terms of usability, the evaluated user answered a survey about how they felt about the learning difficulty and usefulness of the platform. These questions were multiple choice and had the users rank several usability aspects of the platform on a 5 point scale, where higher is better. Additionally, they were asked to rate how intuitive and in control they felt during the evaluation process. In terms of difficulty, every tested functionality was considered easy or very easy on average with the exceptions of the more complicated functions such as connecting modules, importing data, and using the “generic diviz” module. However, these functionalities were all rated “very useful” in terms of usefulness. In order to address the issues that caused problems and difficulties with these tasks, some interface improvements were made, detailed in the next section. In terms of intuitiveness and responsiveness, DecSpace was rated highly with an average above 4 on a 5 point scale. The survey questions can be seen in Appendix C and the average results for each question can be seen in table 6.3.

Finally, the users also submitted their own personal observations of the platform through open
Table 6.3: Average score in usability survey

<table>
<thead>
<tr>
<th>Question</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.3</td>
</tr>
<tr>
<td>2</td>
<td>4.6</td>
</tr>
<tr>
<td>3</td>
<td>4.4</td>
</tr>
<tr>
<td>4</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>3.8</td>
</tr>
<tr>
<td>7</td>
<td>4.4</td>
</tr>
<tr>
<td>8</td>
<td>4.4</td>
</tr>
</tbody>
</table>

ended questions as seen in Appendix C. These were taken into account when deciding on how to further improve the platform’s usability.

6.3 Implemented Improvements

Throughout the user-evaluations and through the survey some changes were suggested on how to improve the platform’s usability. The coordinator also took note of the most common mistakes and how to change the interface to minimize them. The following improvements to the interface were developed after the user-evaluation phase:

- **Option to delete all modules from workspace**: Suggested by users after having to manually delete all the modules to start from a fresh workspace.

- **Change initial position of modules to avoid overlap**: This change was suggested when adding multiple method modules made them overlap, and therefore not visible, in the workspace.

- **Add a representative symbol to the search bar**: Adding a magnifying glass symbol to the search bar would make it more obvious as to what it is.

- **Add a placeholder title in the "Generic Diviz" method catalog**: Users found it unintuitive to enter the page and be confronted with an empty description and example tab. They also did not think to click on the table containing the method descriptions. As such, a placeholder title and description informing the user that no method is selected was introduced.

- **Improved buttons to upload files in "Generic Diviz" module**: Inputing files requires users to first select them from their computer and importing them into the input field. The previous solution proved non-intuitive to many users and as such the interface was changed to feature a more informative button that tells users they need to import the data.

- **Erase content from "Generic Diviz" modal**: In case of error, a button should help remove the content from a certain input instead of having to manually select and erase content from the input box.

- **Placeholder text for empty field**: In order to inform users that certain fields are blank some placeholder text was placed. This way users are more likely to notice the blank fields and import the data correctly.
These changes to the interface can be seen in Appendix D.

6.4 Conclusions

This section described the process and results of the user-evaluation for DecSpace.

The User-Evaluation can be considered a success since it fulfilled the objective of clearing over 80% of the defined objectives. The main purpose of the evaluation was to test how intuitive the interface was to people unexperienced with the platform. Every task had the participants test a unique functionality. Therefore all of them required different interactions where lessons learned from previous tasks would not improve performance.

The test was conducted on both MCDA experts and novices with no significant differences in the results, which proves the concept that DecSpace is a user-friendly platform for non-experts. Although the experts had more familiarity with the concepts at play, the interface proved to be equally simple to both groups.

The constructive feedback given by users throughout the evaluation process and through the open ended survey questions also helped further improve the platform by identifying the major usability problems. These improvements were then developed for the final version of DecSpace, improving the overall usability.
Conclusions and Future Work

This chapter will go over the final conclusions taken from this dissertation. This includes the motivation behind the development of the platform, all the features that were successfully developed, and suggestions for future work to improve the framework further.

7.1 Conclusions

DecSpace was originally created as an answer to the expanding use of MCDA methods in various fields of study. It’s main objective was to become a more inclusive platform to allow non-experts the use of these methods.

The design and technology used in the platform was chosen according to a few key features that were deemed necessary: online availability, ease of use, as well as simplicity and extensibility of the platform by developers. As such, the MEAN stack was chosen as the main technological base for the platform, a choice made in the early versions of DecSpace. Since the work on this thesis aimed to build on the previous version of DecSpace, the technological solution remained consistent.

In order to decide which features should be added to the platform in this project, a study was done on the current state of the art platforms for executing MCDA methods. Although they all possessed their own strengths and weaknesses, none of them provided the ease of use and availability DecSpace aims to provide. They also suffered from having completely different interfaces and data formats from each other, whereas DecSpace aims to be a unified platform where multiple methods can be developed for and used.

During the development of this thesis the main objectives of the platform remained consistent with those of the previous version and as such the platform was extended more so than changed. The new features aimed to blend into the existing interface as an effort to keep the platform as simple as
possible for users.

The development done on DecSpace during this thesis proved its extensibility, which was a core requirement. A new local method was added, as well as access to the entire library of remote methods created by the diviz team. The list of remote methods can now be updated in real time and consulted through the method catalog. The database system was moved from a cloud service to a local infrastructure in order to reduce dependencies and improve availability, security and performance. Finally, some changes to the interface were done in order to improve usability.

The user evaluation was successful and proved that the new features for DecSpace were usable by non-experts of MCDA. The participants were able to quickly learn the main functionalities of the platform. Some recommendations on improving the platform were also suggested by users and developed for the final version.

It can be concluded that the framework was successfully enhanced and its extensibility was demonstrated by the inclusion of new features.

7.2 Future Work

The work done during this thesis was focused on improving the platform through the application back-end. This leaves a lot of work to be done on the user-experience side which can still be greatly improved.

In terms of functionality, there are a few key aspects that would greatly improve the platform.

Although extensible, adding new local methods to the platform is currently a complicated procedure that only developers familiar with the platform will be able to do. This greatly reduces the potential of the platform since simplifying this process would allow outside users to develop their own plugins and extend it. The way local methods work in DecSpace should be reworked to be more modular and generic to allow for this extensibility.

Having access to the diviz web-services greatly increased the number of methods for the platform, but using and interpreting the results form these methods is still a complicated task due to the nature of the XMCDA format. In order to be improved, DecSpace should be able to interpret all XMCDA data formats, which are available to consult in the diviz website. By having them in DecSpace's own database, the methods would become a lot more usable.

The data produced as a result from all methods in DecSpace is still limited to text, which is not optimal in MCDA methods. As such, having a way of displaying results through more graphical outputs such as graphs would be interesting to the decisions makers in order to better analyze the results.

Lastly, DecSpace would benefit from having a system that guides users to the appropriate MCDA method for their specific problem, such as a recommendation system. It's already been proven users can use a specific method if they follow instructions but being to solve their own real-world problems through the platform would be an interesting inclusion. Less experience users could follow a step by step guide that had them insert their data and the platform would then choose an appropriate method and present them with the results.
Bibliography


[18] Irã¨ne Abi-Zeid, Nicolas Couture-Grenier, William Jones Yankeu Ketchapa, Luc Lamontagne, Mohammed Mouine, and Oscar Nilo *MCDA-ULaval: Multi-Criteria Decision Aiding Tool* http://cersvr1.fsa.ulaval.ca/mcda/?q=en


A.1 Introduction

A.1.1 About the Application

DecSpace is a web-based application that aims to provide an easy-to-use and intuitive approach to using Multi-Criteria Decision Aiding (MCDA) methods. It is focused on supporting a flexible and responsive work environment for a greater user experience.

The main objective of this framework is to present an environment that allows an easy exploration and use of the most diverse MCDA methods, independently of their complexity. That exploration is made through the construction of workflows.

The complexity of those workflows is only dependent on the user, since a single workflow may contain several connected MCDA methods or one method only. Therefore, it is possible to design sophisticated decision-aiding processes, but also to keep it minimalistic.

A.1.2 About the Document

The main objective of this manual is to inform the user about the most relevant characteristics and purpose of DecSpace, as well as to demonstrate its main functionalities and how they can be correctly executed.

Beginner users are recommended to read this guide before starting to use DecSpace and while working for the first few times. When the user is more comfortable with the application, he should consult this guide when in need to clarify any doubt.

A.1.3 Required Equipment

To use DecSpace it is required to have a desktop computer or laptop with an internet connection. It can also be consulted in smaller devices (e.g. tablets and smartphones), but it is not optimized for those devices and the full experience is not guaranteed.

A web browser is also needed to access DecSpace (Google Chrome is recommended but not mandatory). The link to DecSpace's homepage is decspacedev.sysresearch.org.

A.2 Getting Started

In this section the most basic functionalities of DecSpace are described, as well as the interfaces where they are accessible.

In the initial page of DecSpace (called homepage), it is possible to access the pages to sign up (see Section 2.1), log in (see Section 2.2) and try DecSpace anonymously (see Section 2.4).
Logging out (see Section 2.3) and recovering your password (see Section 2.5) are other basic features of DecSpace, but are accessible through different pages.

### A.2.1 Signing up

The "Sign Up" feature allows the creation of a new account, in order to start using DecSpace. For that effect, the following steps have to be pursued:

1. Click the "Sign Up" button displayed in the navigation bar (outlined in Figure 1). Then, you should be redirected to the "Sign Up" page (see Figure 2).
2. Enter your first and last name.
3. Enter a valid email address.
4. Enter a password you consider safe.
5. Define the account's privacy settings. The administrator of DecSpace has the possibility of verifying which public accounts are online at any moment. Private accounts do not allow the administrator to check their status.
6. Click the "Register" button to finish this process. After the form is validated, a success message should appear.
A.2.2 Log In

In order to work in DecSpace, you have to log in first. That process can be accomplished by following these steps:

1. Click the "Log In" button displayed in the navigation bar (outlined in Figure 1). Then, you should be redirected to the "Log In" page (see Figure 3).

2. Enter your email address.

3. Enter the corresponding password.

4. Click the "Log In" button to finish this process. If the introduced data is validated (i.e. is the same as the one entered during the “Sign Up” process), you should be redirected to your list of projects.

![Log In page](image)

Figure A.3: “Log In” page - Steps for logging in

A.2.3 Log Out

You should log out, after you finish working in DecSpace. Logging out is a very easy process. You just need to click the “Log out” button available in the navigation bar (see Figure 4).

![Log Out button](image)

Figure A.4: “Log Out” button in the navigation bar.

A.2.4 Anonymous User

On one hand, the “Anonymous User” features allows you to use DecSpace without the need to sign up or log in, but on the other hand the anonymously created projects are not stored in DecSpace’s database.

All the features of DecSpace are available for anonymous users as well.

To access this feature, you only need to click the “Try It Now!” button in the homepage (outlined in Figure 1).
A.2.5 Recover Password

If you happen to forget the password of your account, there is a simple way of recovering it:

1. In the "Log In" page, click the “Forgot your password?” link (see Figure 5) and you should be redirected to the “Recover Password” page (see Figure 6).

2. Enter the email address that you used to sign up in DecSpace.

3. Click the “Recover Password” button. If the entered email is registered in DecSpace, a successful message appears on the screen and a message with the password is sent to that email address.

![Figure A.5: "Log In" page - Forgot password highlighted](image)

![Figure A.6: "Forgot Password" page](image)

A.3 Projects Page

After logging in (see Section 2.2), you are redirected to the "Projects" page. Like it is demonstrated in Figure 7, the functionalities that are available in that page are: create project (see Section 3.1), open project (see Section 3.2), duplicate project (see Section 3.3), delete project (see Section 3.4) and delete all projects (Section 3.5). The following subsections describe those features and how to use them.
A.3.1 Create Project

To create a brand new project, you simply have to:

1. Enter a name for the new project.

2. Click the "Create Project" button.

After these two steps are performed, the new project should be listed with the respective creation date and last update.

A.3.2 Open Project

To work on a certain project, you need to “open” it, i.e. get access to its data and workflows. In order to do that, you simply have to click the “Open Project” button (dark blue button listed next to each project) and then, you should be redirected to the workspace (see Section 4 for a detailed description of the workspace’s features).

A.3.3 Duplicate Project

It is possible to make a copy of any of your projects, including all of its data and methods. In order to duplicate a project, you just have to click the "Duplicate Project" button, which is the light blue button to the right of the "Open Project" button, displayed for each project.
A.3.4 Delete Project

If you wish to delete a certain project, simply follow the next steps:

1. Click the “Delete Project” button, which is the red button to the right of the “Duplicate Project” button, displayed for each project.

2. A confirmation is needed in order to delete the selected project. If you really wish to delete the project, click the “Confirm Project Deletion”, otherwise click the “Cancel Project Deletion”.

A.3.5 Delete All Projects

Deleting all of your projects is similar to deleting a single project.

1. Click the “Delete All Projects” button, which is displayed under the list of projects, and a confirmation message should pop up.

2. In the confirmation message, click the “Yes” button to delete all projects or “No” if you do not wish to delete all of your projects.

Figure A.9: “Projects” page - Delete project feature highlighted

Figure A.10: "Projects" page - "Delete All Projects" confirmation message.

A.4 Workspace

The workspace is the page where you can build your workflows. Workflows consist of various modules. Modules can be data files (also called input files), MCDA methods, or Output.

A module may have input and output points. Input points receive data from the output points and output points deliver data to input points. That relationship only happens if the two points are connected. A connection can only exist between one input point and one output point of different modules.
The various features available in this page are: execute the current workflow (see Section 4.1), save the current workflow (see Section 4.2), check the project's archive (see Section 4.6), import input files (see Section 4.3), import workflows (see Section 4.4) and add any of the available methods to the current workflow (see Section 4.5). It is also possible to manually add/check data (see Section 4.7), close modules (see Section 4.8), rearrange modules (see Section 4.9), delete modules (see Section 4.10), add and remove connections (see Section 4.11 and Section 4.12 respectively). These features are highlighted in Figure 11.

Figure A.11: "Workspace" page - Main features highlighted

A.4.1 Execute Workflow

In order to view the output of each module in a certain workflow, you need to execute the workflow, i.e. transfer data between the connected modules and perform the methods.

That can be achieved by clicking the "Execute Workflow" button (the leftmost button shown in Figure 11).

A.4.2 Save Workflow

If you wish to save the current state of the workflow, you must follow these steps:

1. Click the "Save Workflow" button, which is displayed to the right of the "Execute Workflow" button in Figure 11.

2. A message should appear (shown in Figure 12), allowing you to add a comment to the current workflow (so that it is shown in the archive - see Section 4.6), but it is optional.

3. Click the "Save" button shown at the bottom of the message and your workflow should now be saved.

Figure A.12: The message that appears by clicking the "Save Workflow" button.

---

The various features available in this page are: execute the current workflow (see Section 4.1), save the current workflow (see Section 4.2), check the project's archive (see Section 4.6), import input files (see Section 4.3), import workflows (see Section 4.4) and add any of the available methods to the current workflow (see Section 4.5). It is also possible to manually add/check data (see Section 4.7), close modules (see Section 4.8), rearrange modules (see Section 4.9), delete modules (see Section 4.10), add and remove connections (see Section 4.11 and Section 4.12 respectively). These features are highlighted in Figure 11.

Figure A.11: "Workspace" page - Main features highlighted

A.4.1 Execute Workflow

In order to view the output of each module in a certain workflow, you need to execute the workflow, i.e. transfer data between the connected modules and perform the methods.

That can be achieved by clicking the "Execute Workflow" button (the leftmost button shown in Figure 11).

A.4.2 Save Workflow

If you wish to save the current state of the workflow, you must follow these steps:

1. Click the "Save Workflow" button, which is displayed to the right of the "Execute Workflow" button in Figure 11.

2. A message should appear (shown in Figure 12), allowing you to add a comment to the current workflow (so that it is shown in the archive - see Section 4.6), but it is optional.

3. Click the "Save" button shown at the bottom of the message and your workflow should now be saved.

Figure A.12: The message that appears by clicking the "Save Workflow" button.
A.4.3 Import Input File

If you already have the necessary input data in your device and wish to use it instead of manually inputing the data then you can directly import it into the workflow. It should be noted that JSON and CSV files are used for local methods whereas XML files are used for remote methods. Perform the following steps to import an input file:

1. Click the "Import Data" button
2. Select "Import Input File".
3. A new window should appear, asking you to choose the files you wish to import. Click on "Choose Files" to open your File Explorer. You can select multiple files, but only CSV, JSON and XML files are accepted.
4. Click the "Import" button and your data will appear on the workspace as input modules.

Figure A.13: First steps for importing input files.

Figure A.14: Last steps for importing input files.

A.4.4 Import Workflow

If you wish to import a previously exported workflow into the workspace, you have to:

1. Click the "Import Data" button.
2. Select "Import Workflow", like it is shown in Figure 15.
3. A new window should appear asking you to choose the workflow you wish to import. Only the ZIP format is accepted.
4. Click the "Import" button and your workflow should be successfully imported. These last two steps are demonstrated in Figure 16.

![Figure A.15: First steps for importing a workflow](image1.png)

![Figure A.16: Last steps for importing a workflow](image2.png)

A.4.5 Add Method

Adding a new method to the workflow requires the following steps:

1. Click the "Methods" button.

2. Select the method from the modal that appears (a short summary of the method will appear, with a link to the full description below).

3. Click on the "ADD" button to the right of the selected method.

![Figure A.17: First step for inserting a method.](image3.png)
A.4.6 Archive

The archive is a collection of the saved workflows of the current project. It lists the saved workflows showing the date they were created and the comments added to them during the saving process (see Section 4.2). All of those workflows can be reloaded (see Section 4.6.1), exported (see Section 4.6.2), deleted individually (see Section 4.6.3) or deleted all at once (see Section 4.6.4). These features are illustrated in Figure 19.

The "Archive" button (the rightmost button displayed in Figure 11) gives access to all of the features described above.
A.4.6.A  Reload Workflow

All of the listed workflows in the archive can be reloaded back to the workspace. That can be accomplished by executing these points:

1. Click the “Reload Workflow” button (leftmost button displayed for each workflow, as it is shown in Figure 18).

2. A message asking you if you want to save the current workflow before reloading the new one will appear. That confirmation message is shown in Figure 19. Click “Yes” or “No” to reload (or not) the workflow.

![Confirmation message](figure.png)

Figure A.20: The confirmation message that appears after clicking the “Reload Workflow” button of a certain workflow.

A.4.6.B  Export Workflow

The workflows presented in the archive can also be downloaded to your device in a ZIP file. To do that, you just have to click the “Export Workflow” button located in the right side of the “Reload Workflow” button for each workflow (see Figure 19).

A.4.6.C  Delete Workflow

Any workflow listed in the archive can be deleted, which is carried out by:

1. Clicking the “Delete Workflow” button (which is the rightmost button displayed for each project, just like it is demonstrated in Figure 19).

2. After that, the deletion of the selected workflow has to be confirmed, which is done by clicking the “Confirm Workflow Deletion” button or canceled by clicking the “Cancel Workflow Deletion” button (an example of this step is shown in Figure 21).
A.4.6.D Delete All Workflows

To delete all workflows listed in the archive of a certain project, you have to:

1. Click the "Delete All Workflows" button displayed at the bottom of the archive screen (see Figure 19).

2. After the last step, a confirmation message should appear (shown in Figure 21). Click the "Yes" or "No" button according to your desire.

![Confirmation message](image)

Figure A.22: Confirmation message before deleting all workflows in the archive of a certain project.

A.4.7 Delete All Modules

In order to delete all modules from the workspace, simply click the red "Delete all Modules" button.

A.4.8 Add and Check Data

As mentioned before, a module can be of two different types: data or method.

The input data of method modules can be manually added or changed, while data modules contain static data and cannot be changed.

In the following sections (Section 4.7.1 and Section 4.7.2), it is exemplified how to add data to "OrderBy" and "Sort" method modules.
A.4.8.A Add Data to the "OrderBy" Method Module

For example, adding data to a "OrderBy" method module can be performed in the following way:

1. Click the name of the module you want to add data to (see Figure 23).

2. For the criteria follow these steps (as seen in Figure 24):
   - (a) Enter the name of the new criterion.
   - (b) Choose the type of the new criterion ("Text" or "Number").
   - (c) Choose the direction of the new criterion ("Ascendant" or "Descendant").
   - (d) Click the "Add Criterion" button and the new criterion should be listed above. All of the previous values can be changed or deleted at any time.
   - (e) After adding all of the criteria, choose the one you wish to order the actions by.

3. For the actions, you just have to perform this procedure (demonstrated in Figure 25):
   - (a) Enter the name of the new action.
   - (b) Enter the values for the added criteria.
   - (c) Click the "Add Action" button and the new action should be listed above. All of the previous values can be changed or deleted at any time.

Figure A.23: Link to access the "OrderBy1" module’s data highlighted

Figure A.24: Necessary steps to add new criteria to the "OrderBy1" module.

Figure A.25: Necessary steps to add new actions to the "OrderBy1" module.
A.4.8.B Add Data to the "Sort" Method Module

As another example, the data of a "Sort" method module can be added like this:

1. Click the name of the module that you want to add or change data (see Figure 26).
2. For the objects, follow these steps (as seen in Figure 27):
   (a) Enter the name of the new object.
   (b) Click the "Add Object" button and the new object should be listed above.
3. For the order, you should just have to drag the object you wish to the desired position. For example, it is demonstrated in Figure 28 the dragging of the object "Photography" from the "Least Important" position to the "Most Important" position.

A.4.8.C 4.7.3 Add data to "GenericDiviz" Method Module

Remote methods called by this generic method have to be added in a different format for processing on the diviz servers. The user can input the data manually but it is highly advisable to simply import the relevant data from an XML file that is properly formatted. To add data do the following:
1. Click on the “GenericDiviz” Module name.

2. Search for the method you wish to call manually or through the search bar.

3. Click on the method row as seen in figure 30 and the relevant input fields will appear.

4. For each input, you must input the data manually or import it from a file as seen in figure 31 by doing the following:
   (a) Click on the “Choose File” button to open up the file explorer.
   (b) Select the file for the input (only XML files are accepted).
   (c) Click on the blue “Import Data” button to import the file into the input field.
5. Once all fields are filled in, click the "OK" button to close the module.

Please note that some fields are optional and not needed to run the method. These should be left blank. In case of error, the "Delete Data" button will clear the field.

**A.4.9 Close Module**

After you have added or changed the data you wanted in a certain module and want to return to the workspace, you just have to close that module.

For that effect, simply click the "X" button presented at the top right corner of every open module. This is exemplified in Figure 32.

![Figure A.32: The "X" button that closes a module.](image)

**A.4.10 Rearrange Module**

Modules are added to the workspace in a fixed position. To be able to see them all, you need to rearrange them in the workspace, i.e. move them to another location inside the workspace.

That can be easily achieved by simply dragging the desired module to a different spot in the workspace.

**A.4.11 Delete Module**

To delete a module from the current workflow, you only have to follow these simple steps:

1. Click the "X" button that is displayed in the module you wish to delete, just as it is shown in Figure 22.

2. A confirmation message should appear. Click "Yes" or "No" to confirm or cancel the module deletion (see Figure 33).
A.4.12  Add Connection

In order to transfer data between two different modules, a connection has to be added between them. That can be done by (illustrated in Figure 34):

1. Clicking an output point (represented by a red square) of a module.
2. Clicking an input point (represented by a yellow square) of another module.

A line should appear on screen to represent the connection between the two modules. “Creating Connection” on the top right side of the screen shows a green color if an output point has already been clicked or red otherwise.

A.4.13  Remove Connection

To eliminate a certain connection, you only have to click it. When hovering over a connection, its width increases so that it is clearly identified, just like it is shown in Figure 35.
A.4.14 View Method Results

Once you execute a workflow successfully, each method will generate and "Output" file module where the method results will be displayed. The format of these results vary per method. In Figure 36 we can see some example results for the "Additive Aggregation" method.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Aggregated Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tot</td>
<td>47</td>
</tr>
<tr>
<td>Aud</td>
<td>39</td>
</tr>
<tr>
<td>BMW</td>
<td>33</td>
</tr>
</tbody>
</table>

Figure A.36: Example of an "Additive Aggregation" Output.

A.5 Method Catalog

The "Method Catalog" lists all of the methods that are available in DecSpace. For each of those methods, their description and an example of input and output data is available.

A.5.1 Access a Method’s Description and Example

To access the description and example of any of the available methods, simply:

1. Click the "Method Catalog" button displayed in the navigation bar.
2. Select the method you wish to know about. For example, in Figure 37 the "OrderBy" method is chosen.
3. The method’s description is shown by default, but it can be easily changed to the example of its input and output data by clicking the "Example" tab, much like it is shown in Figure 38.

Figure A.37: Initial page of the "Method Catalog".

Figure A.38: Page of the method "OrderBy" in the "Method Catalog".
A.5.2 View Remote Method Descriptions

To access the descriptions of remote method available through the diviz server (Figure 39) do the following:

1. Click on the “Method Catalog” button in the navigation bar.
2. Select the “GenericDiviz” method.
3. Search for the relevant method manually or use the search bar.
4. Select the method by clicking on the relevant row.
5. View the relevant information on the “Description” and “Example” tabs.

Figure A.39: Example view of a selected method from the Diviz Servers in the Method Catalog

A.5.3 DownloadExample

It is possible to download the input data of any of the available methods in the “Method Catalog”. For this effect, simply click the “Download Example” button, which is accessible in the “Example” tab (see 40).

Figure A.40: The “Download Example” button for the “OrderBy” method in the “Method Catalog”.

A.6 Settings

The “Settings” page allows you to configure different aspects of your account. This page is available after you log in. At this moment, it is only possible to reset your account’s password (see 41).
**A.6.1 Reset Password**

To reset your password, follow these simple steps:

1. Click the “Settings” link in the navigation bar.
2. Enter the current password.
3. Enter the new password.
4. Click the “Reset” button.

![Password Reset Form](image)

**Figure A.41:** Necessary steps to successfully change your password.
User Evaluation Guide
This document outlines the tasks the users should complete in order to test some of the new functionalities of DecSpace.

B.1 Setup

The following will be needed to complete this test:

- A desktop computer or laptop with Internet access.
- The DecSpace user manual.
- A web browser (preferably Google Chrome) open at decspacedev.sysresearch.org
- The online survey to fill out after completing the test.

B.2 Tasks

The following tasks must be completed in the order they are presented:

B.2.1 Task 1

Scenario: Sign up and log into DecSpace
Guidelines:

- Sign up using your first and last name. Provide a valid e-mail address. The privacy setting should remain public.
- Log in

B.2.2 Task 2

Scenario: Explore the "Method Catalog" to better understand the "Additive Aggregation" method and download the example files.
Guidelines:

- Learn how the additive aggregation method works in the "Method Catalog" by checking the description and Example tabs.
- Download the example files available in the "Example" tab of the page.
- Unzip the files in your device.

B.2.3 Task 3

Scenario: Explore the "Method Catalog" to better understand the remote "diviz" methods available.
Guidelines:
Check the "Generic Diviz" method in the method catalog.

Search for the ACUTA method using the search bar.

Read the description and example tabs.

Download the example files available in the "Example" tab of the page.

Unzip them in your device.

B.2.4 Task 4

Scenario: Create a new project and use the "Additive Aggregation" method.

Guidelines:

- Go to the "My Projects" page.
- Create a new project with the name "Test Project", and a "Public" setting.
- Open the new project.
- Add the "Additive Aggregation" method to the Workspace and move it around the workspace as you like.
- Open the method module box and add the following criteria (for a hypothetical choice between cars):
  - Name: Price, Weight: 4
  - Name: Horsepower, Weight: 10
  - Name: Consumption, Weight: 6

- Add the following Options:
  - Name: BMW, Price: 3, Horsepower: 9, Consumption: 4
  - Name: Audi, Price: 4, Horsepower: 7, Consumption: 5
  - Name: Fiat, Price: 6, Horsepower: 5, Consumption: 6

- Close the module and execute the workflow.
- Check the output generated and note what car was considered best.

B.2.5 Task 5

Scenario: Make use of the import Data functionality.

Guidelines:

- Delete all modules from the workspace
- Add a new "Additive Aggregation" method module.
• Import the "criteria.json" and "options.json" files which were previously downloaded and unzipped on your device.

• Re-arrange the modules on the workspace and check that both were imported correctly by opening their respective modules.

• Connect the "criteria.json" module ("Output" point) to the "AdditiveAggregation1" module "Criteria" point).

• Connect the "options.json" module ("Output" point) to the "AdditiveAggregation1" module "Options" point).

• Open the "Additive Aggregation" module and check that the data has been filled in.

• Close the module and execute the workflow.

• Check the output generated and note what car was considered best.

B.2.6 Task 6

**Scenario:** Run the remote method "ACUTA" from the diviz servers.

**Guidelines:**

• Delete all modules from the workspace.

• Add a "Generic Diviz" method module.

• Open the module and select the ACUTA method by using the search bar.

• Find the input files from the zip you downloaded earlier from the "Method Catalog". The ACUTA method files are under: "/ACUTA-UTAR\tests\in1"

• Add the input files to the relevant fields. Some of the optional fields with no matching file should remain blank.

• Close the module and execute the workflow.

• Wait for the method to run and then open the output module containing the answer.
User Evaluation Survey
Decspace Usability Survey

This survey aims to measure the satisfaction of participants who tested the Decspace platform. Please answer truthfully; this should not take more than 5 minutes. Thank you for your collaboration!

Rank the difficulty of using the following features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Very difficult</th>
<th>Difficult</th>
<th>Average</th>
<th>Easy</th>
<th>Very easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signing up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logging in</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigating the website</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessing the Method Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downloading examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creating projects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adding method to the workspace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filling method data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executing the workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rearranging modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleting modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Importing data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using Generic Diviz module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opening output modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using Generic Diviz search bar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connecting modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure C.1: First part of usability survey

Figure C.2: Second part of usability survey
How would you rank the following features in terms of usefulness?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Very useless</th>
<th>Useless</th>
<th>Average</th>
<th>Useful</th>
<th>Very useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signing up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logging in</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigating the website</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessing the Method Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downloading examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creating projects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adding method to the workspace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filling method data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executing the workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rearranging modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure C.3:** First part of survey concerning usefulness

<table>
<thead>
<tr>
<th>Feature</th>
<th>Very useful</th>
<th>Useful</th>
<th>Average</th>
<th>Useless</th>
<th>Very useless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executing the workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rearranging modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleting modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Importing data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using Generic Diviz module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opening output modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generic Diviz search bar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connecting modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure C.4:** Second part of survey concerning usefulness
Figure C.5: Questions regarding control over interface

1. Classify the statement: "I always knew what was going on in Decspace."

   - Strongly disagree: [ ] [ ] [ ] [ ] [ ]
   - Strongly agree: [ ] [ ] [ ] [ ] [ ]

2. Classify the statement: "I felt like Decspace was very responsive to my actions."

   - Strongly disagree: [ ] [ ] [ ] [ ] [ ]
   - Strongly agree: [ ] [ ] [ ] [ ] [ ]

3. Classify the statement: "I always felt in control navigating the different pages of Decspace."

   - Strongly disagree: [ ] [ ] [ ] [ ] [ ]
   - Strongly agree: [ ] [ ] [ ] [ ] [ ]

Figure C.6: Questions regarding learning difficulty

1. How easy was it to learn how to use Decspace?

   - Very difficult: [ ] [ ] [ ] [ ] [ ]
   - Very easy: [ ] [ ] [ ] [ ] [ ]

2. How would you classify Decspace in terms of usefulness?

   - Very useless: [ ] [ ] [ ] [ ] [ ]
   - Very useful: [ ] [ ] [ ] [ ] [ ]

3. How intuitive would you consider the interface of Decspace?

   - Very unintuitive: [ ] [ ] [ ] [ ] [ ]
   - Very intuitive: [ ] [ ] [ ] [ ] [ ]
Recommendations

What did you like the most about Decspace?
Long answer text

What did you like the least about Decspace?
Long answer text

Do you have any recommendations for the platform? (For example, new features or functionalities you felt were missing)
Long answer text

Do you have any other observations you’d like to make?
Long answer text

Figure C.7: Open ended recommendation questions
Interface Improvements
Figure D.1: Screen showing new search bar icon and placeholder text in "Generic Diviz" method catalog.

Figure D.2: New "Delete all Modules" button in the Workspace.

Figure D.3: New placeholder text and buttons for Generic Diviz modal.
Figure D.4: Multiple modules being added no longer stacks them on top of each other.