Front-end Migration and User Experience Improvements in the DecSpace MCDA Framework

Rúben André Paulino Garcia Rodrigues
Instituto Superior Técnico
Lisboa, Portugal
ruben.andre.rodrigues@tecnico.ulisboa.pt

ABSTRACT

Multiple Criteria Decision Aiding (MCDA) is a field of study dedicated to developing methods and techniques that help decision-makers in complex decision scenarios involving multiple criteria, often conflicting. DecSpace is a web-based Decision Support System (DSS) - a software tool that implements modular MCDA methods in a framework, which can be composed and connected freely inside projects. The framework allows users to upload their own data and create and save persistent projects containing workflows with multiple method modules and data modules.

The main objective of this work was the migration of the front-end of the DecSpace application into the Vue.js JavaScript framework, while simultaneously implementing several usability improvements. Some relevant existing DSS tools were analyzed in order to extract their greatest strengths, contributing to the development of a new DecSpace prototype. This new version is stable and future-proof, ready for further improvements and integrations with other MCDA projects.

After the development was finished, the resulting prototype was evaluated through tests with users, both familiar and unfamiliar with MCDA. The issues raised by the evaluation were fixed, resulting in an improved new version of the prototype.

KEYWORDS

Multiple Criteria Decision Aiding; Decision Support System; Web Application; Framework; Front-end Migration; User Experience.

1 INTRODUCTION

Making decisions is a staple of human nature, being rational beings. Multiple Criteria Decision Aiding (MCDA) is a branch of Operational Research dedicated to analyzing and solving problems characterized by multiple criteria. Its main objective is to support decision-makers during the decision process by producing algorithms that can aid them in these situations (like choice, ranking and sorting problems). An MCDA method serves a specific type of problem, appropriately processing certain input parameters and returning some output, which is supposed to help the decision-maker with the decision. As more and more of these MCDA methods are developed, several software solutions have come up. These Decision Support Systems (DSSs) help the users by running hard computations required by MCDA methods, many times with very large datasets.

DecSpace is a web application that intends to be a multipurpose DSS by providing a framework for composing and executing several MCDA methods. The project was originally developed by Master's students from Instituto Superior Técnico, with its main objective being to be a web-based, open-source, easy to use application with a large catalogue of MCDA methods as well as connections to other existing open-source software. However, the application has some problems which make it necessary to conduct changes to it on the technology level, particularly on the front-end side. This dissertation documents those problems and the work that was done to solve them and improve the application.

1.1 Problem Description

As MCDA evolves and is getting more popular, with the advent of new methods every year, so does the number of software solutions to support these processes. These DSSs exist in a wide variety of styles, serving different purposes and entities. Most of them, however, either serve a single very specific purpose (i.e., implement a single specific MCDA method) or are tremendously outdated, particularly concerning usability and user experience. None of these tools are particularly friendly to users that are not familiar with MCDA, even though some MCDA methods could be easily understood and applied in casual contexts, precisely by casual users.

DecSpace was originally developed to meet these needs, but some problems had arisen: the front-end implementation of the project was unmaintainable and several new features were required. Besides, DecSpace should have been better integrated with other ongoing MCDA projects.

1.2 Motivation and Objectives

Due to the inherent problems of the original version of DecSpace, a major task needed to be done: a full migration of the front-end technology. This implied a full rewrite of the front-end, which could be done while simultaneously implementing new features and usability requests. The new prototype should be maintainable by upcoming developers, by providing a clean and modular system to implement new methods for the framework. Also, the groundwork for future integrations with other MCDA projects should be established.

1.3 Contributions

The final solution is a robust prototype of the DecSpace application implemented with new front-end technologies. The framework is stable and many desired features were implemented into it, with a significant amount of work dedicated to making it maintainable and flexible for future implementations of new MCDA methods. A module generation system was developed, separating the logic of methods into module, modal and service, which will be crucial for the near future of the project, as multiple new developers get on

board. During the development, many new features were implemented, making the new version an improvement from the previous one. New iterations of the prototype were frequently reviewed by an MCDA specialist and potential future user, which contributed to a better application overall, catering to both experienced and inexperienced users.

Furthermore, a set of papers focused on DecSpace were accepted and published during this time, in Conferência da Associação Portuguesa de Sistemas de Informacão (CAPSI) 2018, in the 88th meeting of the EURO Working Group on MultiCriteria Decision Aiding (EWG-MCDA) and in the International Symposium on Business Modeling and Software Design (BMSD) 2019 [1–3].

1.4 Document Structure

This document is structured into several different chapters:

(1) **Introduction:** The current chapter, where a brief introduction to MCDA and the DecSpace project is given, as well as a description of this document. The motivations of this Master's thesis are exposed, directly correlated to the existing problem, and the contributions that resulted from it are listed.
(2) **MCDA Overview:** A general overview of what MCDA is is given, including the different groups of MCDA methods that exist. In addition, an analysis is made on the state-of-the-art of MCDA software tools. For that, four different tools are presented, their advantages and disadvantages. At the end of the chapter, some conclusions about the tools are drawn from this analysis.
(3) **Problem Analysis:** An overview of the DecSpace framework and its problems is presented. At the end of the chapter, the requirements that a potential solution should fulfill are presented and justified.
(4) **Solution Design and Implementation:** This chapter describes the work that was done from several different points of view, starting from the differences in architecture, domain and technologies from the previous version. A detailed technical description of the project, of methodologies used, milestones of the development and concluded objectives follows, closing with a practical demonstration of the framework.
(5) **Evaluation:** Following the solution details, Chapter 5 describes what was done to evaluate the work done on it. The rationale behind the chosen evaluation method is stated, as well as what the results were and what they might mean in the context of the project.
(6) **Conclusions and Future Work:** The final chapter wraps up this dissertation, with a summary of the work that was done and what can be taken out of it - the knowledge generated by the Master's Thesis. Finally, some notes about future work possibilities are given.

2 MCDA OVERVIEW

This chapter presents a general introduction to MCDA, its different methods and how they can be used to solve real-life problems. Afterwards, four different DSS tools are analyzed and their advantages and disadvantages are weighted and related to the DecSpace project.

2.1 Introduction to MCDA

Decisions are a heavy part of the everyday life of humans. As rational beings, we make decisions for everything, all the time: what do we want to eat for breakfast or lunch; what TV channel should we tune in to; which car brand and model should we purchase? While some of these decisions could be more head-scratching than others, none of them were particularly hard - humans are perfectly capable of solving them on their own. However, sometimes situations arise where human beings cannot properly weight the different criteria, and thus become poor decision-makers, failing to see the best options.

For instance, if one wants to purchase a house, some of the relevant criteria could be: price, number of rooms, location, safety of the neighborhood, energetic efficiency, etc. Obviously, we would want to minimize the price, but that comes at the cost of reducing the quality of most of the other criteria. Therefore, one must reach a trade-off where all of these criteria are balanced in an acceptable way. Most of the times, humans do this implicitly, with more or less thought behind it, but for complex scenarios with too much in stake, this is not always possible [4].

MCDA is a term that is used to describe a range of mathematical techniques and algorithms that help make decisions in complex scenarios [4]. These techniques can be extremely useful in situations where different stakeholders, with different criteria and objectives (which may conflict) need to reach a common agreement [5]. Along the years, several researchers have proposed methods and techniques for aiding decision makers to handle decision situations effectively and efficiently [6]. The versatility of MCDA methods makes them ideal to solve problems in multiple areas of knowledge, from business to everyday life, with different kinds of MCDA methods available [7].

2.2 Examples of MCDA Tools

Most of the times, MCDA methods cannot be processed simply by "pen and paper" - they imply heavy computations - so they must be supported by software tools that can do the heavy lifting while presenting the user with an enjoyable user interface. As of 2018, there is a myriad of software tools that implement MCDA methods, some with more features than others. Most of the existing tools, however, are traditional, desktop software applications, as opposed to the modern trend of web applications, which offer a lot more in terms of interoperability, scalability and portability [7, 8].

Four different MCDA tools were analyzed, in order to review what the existing solutions were doing right and wrong. These tools were selected for their unique approaches and relevant advances in the field. However some of these tools have usability issues, which will be highlighted to better learn from them.

Priority Estimation Tool (PriEsT) is an MCDA software tool that supports Analytic Hierarchy Process (AHP)-type decision-making, a technique that combines both quantitative and qualitative, objective and subjective types of judgment. PriEsT offers multiple equally-good solutions - you can easily create your own project, add your available options, define the criteria, and then PriEsT does all the work, leaving you to analyze the results, for which it provides a range of different methods. PriEsT offers useful features, like some data visualization and Extensible Markup Language (XML)

data exporting. Unfortunately, the last version dates to the end of 2015, the tool only represents a niche, and even though its user interface works well, it is quite outdated, with Windows XP styling.

1000minds is a commercial collection of decision-making tools by 1000minds Ltd, including also prioritization and conjoint analysis tools. 1000minds is powered by the in-house developed Potentially All Pairwise RanKings of all possible Alternatives (PAPRIKA) method and offers these tools in a modern, easy to use web application with ratings calculated via user-directed questions. 1000minds also offers solutions for conjoint analysis, where large amounts of people contribute to the decision process, effectively what is called group decision-making. Additionally, it offers plenty of variety in terms of interesting visualization options.

diviz is one of the best MCDA tools available because of the modern UI it presents as well as a reliable back-end and variety of MCDA methods supported. It is one of the few tools that aim to support as many methods from different categories as possible in a centralized, free of charge solution. The user interface is based on a workbench, with a toolbar for available methods, a tree with all workflows and, of course, the central panel where the methods, represented by boxes, are configured, set up, and executed. Using a drag-and-drop approach, the tool has a small learning curve and is very appealing to new users. One of the problems with diviz is that it is a standalone desktop program, which lacks many of the benefits of a web application. However, diviz can be considered unique in the market of MCDA tools as it is one step ahead of most of them.

Analytica is a "visual software environment for building, exploring, and sharing quantitative decision models". Developed by Lumina Decision Systems, Analytica is their flagship product - a quantitative decision-support environment that extends the capabilities of a traditional spreadsheet to help with decision-making [9]. It can even be said it is a combination of Excel and MATLAB with an extended graphical user interface [8]. One of the interesting features that differentiate Analytica from classical spreadsheet software, like Excel, is its Intelligent Arrays. The user interface is not particularly current, but it does its job well, taking advantage of high contrast and sober positioning of panels and toolbars. Analytica does not support a large variety of different visualization methods, but the ones it does support work really well, like influence diagrams and tornado diagrams.

After the analysis of the previously mentioned MCDA tools and frameworks, some conclusions can be made. One particularly obvious observation is that very few, if any, of these applications have current user interfaces, which makes it much harder to connect with modern potential users of the tool, which are expecting something totally different in regard to interfaces (especially non-experts). Only some of these tools are web applications, with most of them being desktop-based. An ideal MCDA framework should be used online, via a browser.

Overall, a newly developed tool should extract some of the best features of all the analyzed tools. Preferably, it should do it on a web application using state of the art technology in terms of user interface and back-end. One of the major conclusions from analyzing these tools is that the web interface should be able to appeal to less skilled potential users, while keeping the modernity in a clean way. Also, another thing to note is to try not to clutter the workspace too much, something some of these tools suffer a bit from.

3 PROBLEM ANALYSIS

In this section, a brief description of the original prototype of DecSpace (referred to as DecSpace 1.0) is presented. From this definition, the associated problems and limitations that motivate this Master's Thesis are drawn. Finally, a list of goals that directly answer the stated problems of DecSpace 1.0, which were the objectives of this work, are explored.

3.1 DecSpace 1.0

DecSpace is an MCDA framework that aims to make life easier for the users, providing a simple and straightforward way to consult multiple MCDA methods, create, modify and publish both simple and complex MCDA projects [10–12]. DecSpace ambitions to be that and much more, aggregating many of the features from the solutions that already exist into a single, standalone, web application. The center core of the framework is a workspace, essentially a sandbox where users can compose multiple methods and data, represented as modules, linked by connections (1). The framework provides registration and login services to the users, which are then able to save their projects and edit them later, as well as means for the developers to add additional MCDA methods easily in the future. Furthermore, one of the objectives is to have a centralized framework that is able to solve a range of different problems, with different methods.

In DecSpace version 1.0 there was the concept of local methods, which only had five different implementations, namely CATegorization by Similarity-Dissimilarity (CAT-SD), Inquiry, OrderBy, Sort, Deck Cards Method: Simos-Roy-Figueira (DCM-SRF) and AdditiveAggregation. However, DecSpace later also supported a prototypal concept of remote methods (via Decision Deck web services).

DecSpace was designed following a three-tier architecture - a type of client-server architecture where the project is split into three fundamental components: presentation tier, application tier and data tier. This is a proven methodology for software development, with many benefits, mainly related to the modularity of the solution, making partial upgrades/modifications much easier.

The presentation tier is, essentially, what we call the user interface. Its function is mainly to expose DecSpace's features and services to the user in a user-friendly way. The application tier is where the business logic is located. It acts as a server, as in

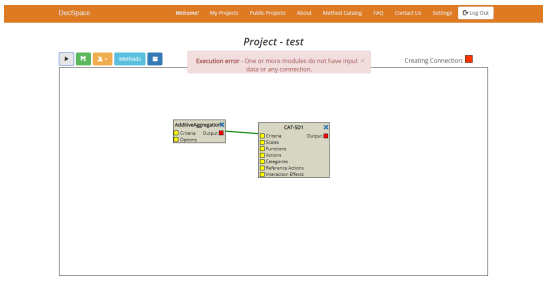


Figure 1: Workspace view of DecSpace 1.0

the client-server architecture, handling the requests received from the possibly multiple clients. For persistence, the application tier communicates with the data tier - the database, where all the information that must be persistent is stored. The database software is responsible for storing data properly sent from the application tier only, as well as retrieving any entries that are requested, also from the application tier.

The original DecSpace version 1.0 was developed on top of a MongoDB, Express, Angular and Node.js (MEAN) stack, a is a very popular JavaScript-based open-source framework for developing both websites and web applications.[1] It is called a full stack, since it includes every component necessary for creating and running a web application from the ground up - from the data layer to the client. MEAN makes life easier for developers because every component is manipulated with JavaScript, which allows them to easily build robust, cohesive applications.

The presentation tier is implemented with Angular.js, which sits on top of the application tier, composed of Node.js and Express.js. Finally, the data tier is taken care of by the MongoDB Database Management System (DBMS). These four layers constitute the MEAN stack, which is growing in popularity for its advantages mainly in scalability and performance, as opposed to classical, more conservative stacks, like the Linux, Apache, MySQL and PHP (LAMP) stack.

3.2 Requirements

As referenced in the previous section, DecSpace 1.0 front-end was developed using AngularJS technology. Unfortunately, it currently presents a range of problems, which highly affect the maintainability of the framework. AngularJS was abandoned by the developers, which instead focused their attention on a full rewrite of the framework, simply called Angular.

This triggered the discussion between the stakeholders and developers of DecSpace of whether the front-end development effort should still be directed at AngularJS, or if we should switch to a different, more modern and maintained front-end technology. From this discussion and after a good amount of research, three different technologies emerged: Angular (the new version - Angular 2+), React and Vue.js. One of the reasons these three frameworks were selected was their JavaScript-based nature, just like the original Angular. This, and their compatibility with the remainder of the MEAN stack, made them ideal candidates to replace the AngularJS front-end while keeping the back-end implementation, which had no inherent problems (permitting a partial migration instead of a full one).

After deliberation, the final choice eventually fell on Vue.js, mainly picked because of the great performance it offers, as well as its simplicity and gentle learning curve. Vue.js, or simply Vue, is an open-source, front-end JavaScript framework, just like AngularJS, which is suitable for building powerful and complex web applications and websites, especially single page ones, all while keeping a minimalist and clean design. With rising popularity over the last couple of years, Vue.js aims to be a lightweight alternative to AngularJS, while also offering very interesting features for developers, like hot-reloading, reactivity, routing, etc.

[1] http://mean.io/

The solution, then, consists heavily on the process of migrating the old front-end from AngularJS to Vue.js. Since both technologies are totally incompatible, this implies a full rewrite of DecSpace's front-end. Furthermore, since a number of reported bugs and necessary improvements already existed, these will be taken into account when developing the new user interface, resulting in a state-of-the-art, maintainable, and improved solution, relatively to DecSpace 1.0.

A list of goals was redacted by analyzing the preexisting notes on bugs and desired features and combining it with an observation of the current system. The development process was to be split into three phases. On the first phase (G1), the front-end migration to Vue would be concluded and several improvements on user experience would be implemented. Then, in the second phase (G2), focus would be on improving the management of local MCDA methods and figuring out ways to connect to more external projects. Finally, on the third phase (G3), improvements and bug fixes on already implemented methods would take place, and new local MCDA methods would be implemented from scratch. The detailed list of final goals can be consulted below.

- **G1: Infrastructure and User Experience Improvements**
  - **G1.1: Front-end migration to Vue.js:** The first major goal of the development process will be to finish the migration from AngularJS to Vue.js, which is already in motion. This implies a full rewrite of the framework's interface, effectively starting from nothing. The old interface must be reverse-engineered and each module must be reimplemented in Vue.js.
  - **G1.2: Usability Improvements:** Implement the possibility of copying and pasting the modules inside the workspace, improving the usability of similar occurrences of the same methods. Implement drag-and-dropping of input files for direct upload. Improve the workspace by using a color scheme that clearly distinguishes between input modules, output modules and method modules. Allow the user to rectangle select multiple boxes, to copy, drag, or delete them.
  - **G1.3: Error Reporting:** Improve the error messages presented to the user. The error messages that occur when running a workflow should be much more specific, helping the user understand what exactly is wrong and how to fix it.
  - **G1.4: Privacy Settings:** Allow the users to change the privacy setting of a project (public/private) at any time. Implement the possibility of sharing projects with custom groups of users.
  - **G1.5: Help and Tutorials:** Implement tutorials of some kind, either static videos, interactive ones, or useful tooltips while filling in the parameters of a method. These should provide a higher sense of security to the user.
  - **G1.6: Continuous Generic Minor Bug Fixing:** The development of the new version in Vue will undoubtedly spawn several minor bugs along the way. These generic small bugs should be continuously fixed as a process, during the development process, to avoid generating bigger problems in the future.

- **G2: Catalog and method management improvements**
  - **G2.1: Management of Local Methods:** DecSpace should support more local methods to provide a cohesive and powerful framework for the users. Therefore, new MCDA methods should be implemented from scratch locally on the DecSpace framework. This includes theoretically implementing the method itself and implementing an easy to use, intuitive, interactive interface for it. This process must be repeated for each method that is to be added. Thus, this goal refers to the local method management process, how to optimize it and make it clearer.
  - **G2.2: Management of External Methods:** Additionally to the already supported diviz Extensible Multi-Criteria Decision Analysis (XMCDA) remote methods, DecSpace has the potential to be extended on this field. Some more remote methods should be set up, and additional external connections may be realized, particularly to the other Decision Deck projects. For instance, "The R initiative" is a package that uses the R statistical environment to enrich the MCDA process - it could be a candidate to link with DecSpace[2].
  - **G2.3: Methods Information:** Provide more information to the user for each method. For instance, more references, classification of the method and better descriptions, description of the necessary data, etc.
- **G3: Local Methods Implementation and Improvements**
  - **G3.1: DCM-SRF - Revisions and Improvements**
    - **G3.1.1: DCM-SRF - Ambiguity:** Resolve a situation where there is a certain degree of ambiguity in the DCM-SRF method. The situation where between criterion A and criterion B there are two rows with one blank card each is equivalent to the situation where there is a single row with two blank cards. This may confuse the users. Either the user should be warned about it or it should not possible at all.
    - **G3.1.2: DCM-SRF - Images on Cards:** Possibility to add images to the cards, making the identification of the criteria much faster. It should support a range of preselected images as well as user-uploaded custom ones. Alternatively, the cards could be filled with a color chosen by the user.
    - **G3.1.3: DCM-SRF - Visualization:** Implement alternative ways to visualize the results with the weights of the criteria, like histograms, bar charts, pie charts, etc. This can also be applied in different ways to other methods.
  - **G3.2: CAT-SD Tables:** Merge both "Criteria" and "Scales" tables for higher cohesion, since both have information about criteria.
  - **G3.3: Implementation of Choquet Integral Method:** Implementation of a new local MCDA method that has been requested called Choquet Integral Method [13], using the newly implemented foundations that makes adding new methods easier.

[2]https://www.decision-deck.org/project/

These goals were to be accomplished during the development phase of the Master's Thesis and were planned accordingly. After the development phase, the end product should be a stable prototype of the DecSpace service incorporating these features and potential others that would come up.

4 SOLUTION DESIGN AND IMPLEMENTATION

This chapter refers to the execution of the objectives formalized in the previous chapter, essentially a detailed description of the work developed across six months. The new DecSpace prototype (DecSpace 2.0), a result of this work, is conceptually described to better understand the improvements made after this time, what was accomplished, how it was done, and why certain design decisions were made. A technical summary of the software that was developed is also given, highlighting used programming languages and libraries, and how they correspond to the end product, specifically certain elements of the user interface.

4.1 Solution Design: DecSpace 2.0

From the work developed during the period of this Master's Thesis resulted a new prototype of DecSpace, which will be referred to as "DecSpace 2.0" from here on. This new prototype mainly maintains the back-end implementation of DecSpace 1.0 (with some adjustments for the new features), but has a brand-new front-end implementation in Vue.js.

For the new implementation some concepts were changed, which in turn prompted changes to the preexisting diagrams. Thus, in the following subsections are the newly-made diagrams adapted for DecSpace 2.0, with several corrections overall.

4.1.1 Use Cases.

The UML use case diagram in Figure 2 models the DecSpace 2.0 use cases.

A New User is anyone that has just accessed the platform - the user can log in if already registered, sign-up with a new account or stay anonymous. The Anonymous User is a user that has not logged in - he/she is still able to see public projects, but can only work on temporary projects without logging in. The Registered User is a user that has logged in with his/her DecSpace account and has access to all the features of the Anonymous User, plus can now also

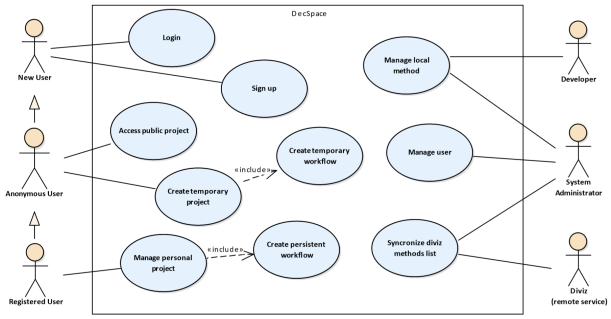


Figure 2: Use case diagram of the DecSpace 2.0 service

save his/her work persistently as a private project. The Developer can add newly-developed local MCDA methods to the DecSpace system to be made available. The Administrator can manage all the DecSpace users, public or private projects, as well as all methods (including the remote methods with diviz).

Each use case is briefly described in Table 1.

4.1.2 *Domain Model.*

The domain model of DecSpace 2.0 that incorporates its behaviour and data is shown illustrated in Figure 3.

The first two main objects represented are User and Project. The User has a valid unique email address and a username that can be changed - the first one is used by the system to identify it while the second serves as an author name for projects - storing as well the encrypted password for login authentication. Every user object also has recorded the date it has created at, and the date of its last update. A user can own multiple projects, and consult multiple public projects.

Table 1: Description table of DecSpace 2.0 use cases

Login	The new user introduces the email and password to log into his/her DecSpace account.
Sign-up	A new user inputs the necessary information for the system to register a new DecSpace account in the system and create a new user.
Access public project	The anonymous user consults the list of public projects, seeing the project name, its creator, creation date and date of the last update.
Create temporary project	The anonymous user creates, edits, removes or publishes his/her temporary projects.
Create temporary workflow	Initiates a temporary workspace where the anonymous user is able to create methods, connections between methods, as well as input and output data.
Manage personal project	The registered user consults, creates, edits, removes or publishes his/her own private projects as public projects.
Create persistent workflow	Builds the workspace where the registered user is able to create and edit methods, connections between methods, as well as input and output data.
Manage local method	Both the developer and the system administrator can manage local methods, meaning adding or removing an MCDA method from the catalog, making it available, or not available to be used by the users.
Synchronize diviz methods list	The system administrator can synchronize the methods available with the ones that diviz offers - this includes fetching the most recent list of XMCDA methods available on the diviz framework.
Manage user	The system administrator is able to manage a user in the system, meaning the user may delete or consult the information about any of the users.

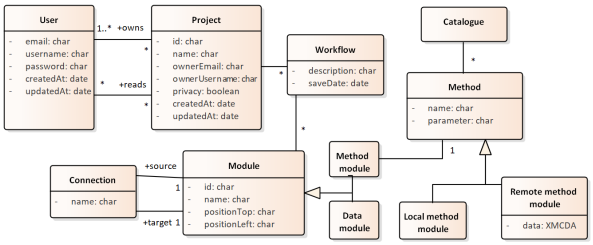


Figure 3: UML domain model diagram of the DecSpace 2.0 service

The Project has a unique ID (used to identify it), the name of the project itself and the name and email of the project's owner. The creation date and the date of the last update are both also recorded, as well as the privacy state of the project. Each Project can have multiple Workflows which in turn can have multiple Modules.

The Workflow contains its save date and may also contain a description produced by the user. The Module entity (which can be either a Method Module or Data Module) has a unique id, used to identify it, a name, and attributes for top and left relative position on the workspace, which are recorded to position it spatially. Modules can be linked by Connections (source to target), which are identified by a unique name. A Data module consists either of data imported by the user or generated by the Method modules, while a Method Module contains one, and only one Method.

The Catalogue entity contains all the available methods that can be chosen by the user to be added and used. Each Method entity has a name and parameters, which are specific for each implementation of the different methods. Methods can be of two categories: Local method module which contains its own implementation and algorithms of execution; or Remote method module which contains a link to an XMCDA method from Decision Deck (called using SOAP requests).

4.1.3 *Architecture and Technology.*

DecSpace 2.0 maintains the three-tier architecture described in the previous chapter, since the changes were made over the front-end. Due to the flexible nature of this architecture, it is possible to easily migrate the technology of one of the layers. For instance, one could also easily replace the data tier from MongoDB to any other DBMS (MySQL, for example). Nevertheless, some corrections were made to the original deployment diagram, which can be seen updated in Figure 4.

The technology stack is now a MongoDB, Express, Vue.js and Node.js (MEVN) stack, in which the front-end technology was migrated from Angular.js to Vue.js. Both stacks are very similar, being based on JavaScript, which made the migration possible.

4.2 Implementation and Completed Goals

Development of the solution started in September 2018. At that stage, the project was passed over from the previous developer of the DecSpace project, in the context of his Master's Thesis [12],

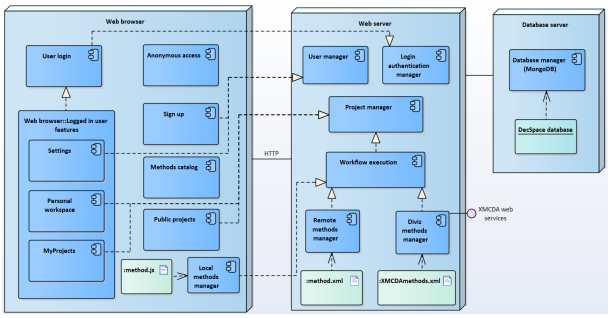


Figure 4: Deployment diagram of the DecSpace 2.0 service

which describes the development state of DecSpace at that time. His responsibilities involved starting the migration process from AngularJS to Vue.js, and so the bare-bones of the project were already coded in the new technology. The extent of the migration included the informational, support pages like the homepage, Frequently Asked Questions (FAQ), method catalogue and settings. Furthermore, rudimentary systems were put into place for user and project management, allowing for basic login and sign-up, as well as project creation. Finally, a very basic workspace was implemented, only for visual demonstrations. It only included a non-executable version of the Sort method, simply to demonstrate the drag-and-drop nature of the workspace.

The immediate direction of the development was extending on what was already done to bring the project to the state it was before the migration, while at the same time implementing new features as needed and from the list of goals stated in Chapter 4. The different challenges faced during the development phase and what was accomplished are explained in the following sections, mainly focusing on the workspace (where most of the development time was spent).

The effective development endeavor ended up with some slight deviations from the original planning. One of the main reasons for these differences was the estimation of the state of the project in September 2018. In reality, the workspace was basically just a mockup at this point, thus a considerable amount of development time was spent at the migration process. In this section, the differences between planning and what was done are presented and explained.

Referring to the goals listed in Section 3.2, they can be split into three categories: fully done, partially done, and not done. In Table 2 are all the proposed goals sorted into the respective categories.

The development was very focused on the migration process, stability and maintainability of the platform, which still allowed for continuous usability improvements. However, this prevented the implementation of some other desired Quality Of Life (QOL) improvements planned in G1.2 (Usability Improvements), like module copy-pasting and rectangle selection. For the same reason, G1.5 (Help and Tutorials) was not fully completed, because very little work was done in the method catalogue. However, a user guide was created as well as a couple of demo videos. Also, the migration itself improved the application with some help features, like tooltips on buttons with additional information to guide the user.

Table 2: Degree of completeness of goals

Fully done	G1.1, G1.3, G1.4, G1.6, G2.1, G3.1.1, G3.1.2 and G3.2
Partially done	G1.2 and G1.5
Not done	G2.2, G2.3, G3.1.3 and G3.3

G2.2 and G2.3, which were originally planned and refer to external method listing and executing (particularly the Decision Deck web services) were scrapped at an early stage when the true extent of the migration progress was understood. Since the priority was the stability and flexibility for future changes on the platform, at some point it was agreed that the remote method execution would be done last. Eventually, this feature was moved to the roadmap of the project, to be picked up by the next developers.

Likewise, G3.1.3 (DCM-SRF - Visualization) and G3.3 (Implementation of the Choquet Integral Method) were also scrapped due to low priority. The platform was, however, developed taking into account multiple stakeholders (including method developers). As a consequence of this, the project is now (at the start of 2019), for the first time, welcoming more than one concurrent developer (1 platform developer and 2 method developers). So it made sense to push G3.3 to the new method developers and focus more on the platform itself. G3.1.3 would also be encapsulated into a larger, more generic task - a dedicated visualization module that would work with all the modules, not just DCM-SRF.

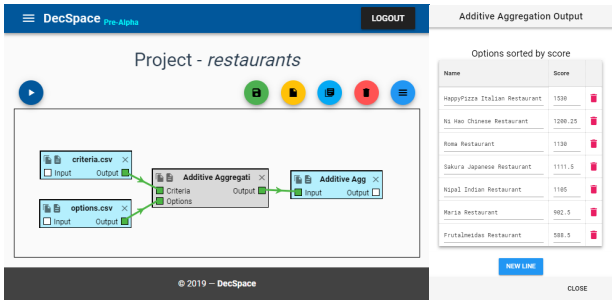
Overall, the completed goals were integrated into a robust prototype (referred to as pre-alpha version) that can be built upon from now on. The migration process to Vue.js is complete (G1.1), as it now has the same features as the original Angular version and more (along with the same methods).

Along with the migration, better error reporting was implemented, using a coherent system at the bottom of the screen, across all the pages of the application. This was implemented in a reusable way, making it possible to transmit messages to the user from any place (including from methods). For instance, one of the complaints from the previous prototype was the lack of specificity with error when executing workflows. Now, DecSpace tells the user that there is an error, states in which of the modules the error occurs, and what is actually missing from it that prevents the execution. The developers can take advantage of this system by color-coding their messages (red for error, blue for info and green for success).

Additionally, a method-specific error reporting system was implemented for the CAT-SD method, which can be extended to other applicable methods in the future. CAT-SD is a particular method implementation, since its parameters are very connected between each other. For instance, categories registered in the "Reference Actions" parameter must also appear on the "Weights" parameter. The method, thus, always verifies its input and warns the user if there is anything missing from any of the parameters.

4.3 Demonstration

To better exhibit the capabilities of DecSpace and what a typical use scenario for a regular user would be, in this section an example is provided. A potential user of DecSpace could either already have his/her own prefilled data files, or take advantage of the interface



(a) Workspace view **(b) Output data**

Figure 5: The restaurants' example in DecSpace

provided by DecSpace to fill in his/her data manually. For large data sets the best option is to use prefilled files and upload them to DecSpace. In our example, the user wants to make a decision regarding a list of restaurants. As input data, he/she has access to a list of criteria for restaurants and the respective weights that should be taken into account when deciding. Also, there is a list of options containing several restaurants and their score for each of the criteria.

One of the simplest MCDA methods is Additive Aggregation, which is essentially a weighted sum [14]. For very basic decision-making and taking into account the fact that the user already has defined the weights of the criteria, Additive Aggregation is enough to judge the restaurants in a very primitive way. Thus, the user creates a new project in DecSpace and uploads his/her two data files, generating two data modules. From the method catalogue, the user then selects the Additive Aggregation and connects the data modules to the newly generated method module. DecSpace will warn the user if there is anything missing or if the data is not in the correct format, otherwise the input endpoints will be colored green.

Finally, the user can execute the workflow by clicking the Execute button, which will result in a scenario similar to that of Figure 5a. A new data module is generated, containing the output (i.e. the result) of the executed Additive Aggregation method. The output data can be seen in Figure 5b - it displays the aggregated score of the options, sorted in descendant order. *HappyPizza* restaurant has the highest score (1530) based on the provided criteria and their weights. Therefore, the "best" restaurant, given the available data, is *HappyPizza* restaurant by a considerably large margin (with a score approximately 27% higher than the second highest).

The user is then able to save the project to DecSpace to review it later on any machine with access to the internet, after logging in to his/her account. The project can also be made public for anyone to see and make copies. If the user needs the data on a local computer file, the workflow can be exported in the form of a `.zip` file containing `.csv` files with the data modules' content. These files can be imported into, for example, Microsoft Excel for further processing.

5 EVALUATION

After the development work was finished and a stable prototype existed, a user guide was created, using as an example the DCM-SRF method. This guide was needed for an experimental use case of DecSpace - to be used in a university course with students, to explore the DCM-SRF method. After briefly presenting the application, it goes through the basics, like setting up a user account, managing user projects and using the workspace. The general rules of the workspace are explained (method modules, data modules, connections, etc.), followed by a detailed step-by-step guide of the DCM-SRF method, along with a real example. The guide includes many screenshots of the steps and relevant links where applicable.

To validate the work that was done, some sort of evaluation had to be conducted. An adequate way of doing this is with usability tests, with potential users of the application [15]. Since one of the core objectives of DecSpace is to be easy to use by both experienced MCDA users and non-experts as well, it would be interesting to direct the tests towards both groups. Therefore, it was decided that the tests would be done with users with basic knowledge of English and computers, with half of them familiarized with MCDA and the other half not.

5.1 Preparation

To prepare the tests, a user evaluation guide had to be created. Since the developed work was mostly on the workspace (the support pages of the web application had already undergone usability tests when the platform was first developed, whose results can be consulted in [12]), the new tests should be focused on its features. Thus, a list of all the major features present in the workspace was made to begin with:

- Generating new methods;
- Drag-and-dropping of modules;
- Editing module name and description;
- Duplicating a module;
- Deleting a module;
- Manually editing module data;
- Deleting a connection;
- Deleting workflow;
- Executing workflow;
- Uploading data files;
- Connecting modules;
- Importing and exporting workflow.

The ideal tests would go through all these features so that the users could experience them and give feedback afterwards. Thus, a set of three tasks was produced. The first task is about the general mechanics of the workspace, serving as first contact for the user. The second task is more advanced, with a scenario for the Order By method which is properly executed. The third and final task includes user-uploaded data modules and exporting and importing of the workflow. All the tasks are done via the anonymous workspace, which can be accessed with one click from the home page, making the testing process all about the workspace and avoiding further problems.

The three tasks were compiled into an evaluation guide. This was the guide that was supplied to the users when conducting the usability tests. The guide also includes a brief introduction and

setup instructions. The user manual was made available through the evaluation guide, instructing the users to use it as they wished (they could read it all, skim through it, or not read it at all).

After doing what was stated in the evaluation guide, users were asked to fill a quick survey, rating their experience in multiple features as well as their overall enjoyment of the application and some other interesting metrics. The survey was deployed by using Google Forms. [3] Since some of the referenced users with experience in MCDA were located all over the world, the tests could not be made in person. So, all the resources and instructions were sent to the users, which conducted the usability tests themselves.

Due to the nature of the application, some care was taken to ensure some of the participants were familiar with MCDA, while others were not (about half of each, of a total of 20 participants). To deem the tests a success, some objectives were previously set:

- Maximum time to complete the test (task 1 to 3): 30 minutes;
- Minimum average for each of the other metrics, (from 1 to 5): 3.

5.2 Results

By analyzing the results, one can see that the tests can be considered a tremendous success. All the features tested had scores between 4 and 5 (on a scale of 1 to 5). The average time of completion of the test was about 13 minutes, with the longest test taking 20 minutes. The usability questions were also very positive, with results in the same range.

The survey also included space for optional, additional feedback. When asked what they liked the most about DecSpace, 8 users reported that they really appreciated the simplicity of it, which is one of the main objectives of the project. Some other users also talked about the intuitiveness and user-friendliness of the framework.

When asked the opposite (what they liked the least), some interesting matters were brought up. A user complained that the workspace was "too empty at the start" - this is a direct consequence of the sandbox approach. Two more users referred that they really enjoyed the notification system implemented, and that it could be used more extensively, namely to notify the user when a new module is generated. This was quickly implemented, resulting in more frequent user feedback messages.

Another user mentioned some difficulties when connecting modules, due to the small size of the connector boxes - unfortunately, this characteristic is inherent of the chosen approach. A consistency issue with "OK" and "Close" buttons in different windows was discovered by one of the users, which was promptly corrected. The buttons from the workspace toolbar caused some confusion (the chosen icons), but users were able to quickly understand them because of their tooltips (an improvement from previous versions). Some other sporadic suggestions were: more methods, an undo button in the workspace (would require the implementation of a history tracker) and visualization features to better analyze the results, which are already in the roadmap for the near future (not in the scope of this Master's thesis, however).

[3] https://www.google.com/forms/about/

5.3 Conclusions

Overall, the users were very satisfied with the DecSpace experience. Since during the development the application was continuously tested, with weekly instantaneous feedback from a potential user (very experienced with MCDA), there was a continuous process of bug-fixing and constant improvements. This resulted in very successful tests, with few bugs discovered (and few immediate improvements to make). Thus, most of the suggestions of the users were about larger features that are in the roadmap of the project, but were already outside the scope of this thesis.

The users seem to notice the simplicity of the framework, which is indeed one of its main objectives. Also, the test results had no correlation with the degree of MCDA familiarity of the users (the tests did not require MCDA knowledge - only basic methods were used), which means that both experienced and inexperienced users seem to experience the DecSpace framework similarly - there are not many obstacles for non-experts.

6 CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions drawn after the work developed for the duration of this Master's Thesis. A retrospective of what was done is given, along with the inherent relevant contributions for the MCDA community. Afterwards there are some notes on the possible future of the DecSpace project: what the next steps should be, what features could be implemented to improve it, and what direction should be followed to keep the project relevant and raise its weight in the community.

6.1 Conclusions

The importance of DSS is unquestionable, both at the academic level and at the inexperienced users level. These tools are what will make the field grow, by supporting both proven existing methods and new methods that will come up, which only benefit from having software support. Inexperienced users can also be captivated via satisfying and easy to use user interfaces, which will introduce them to MCDA.

DecSpace was presented in this dissertation as a solution to the previously stated problems, although it itself had its own challenges, mainly because of the maintainability of the front-end technologies. A new prototype of the application was developed using Vue.js and cutting-edge JavaScript technologies, which can be considered an improvement from the previous version. A considerable amount of effort was directed into making it maintainable and future-proof, simplifying method implementation so that the project could be scaled up in the near future, allowing the participation of multiple developers at the same time. The application development also took into account the perspective of inexperienced users, verified in the evaluation conducted which surveyed both experts and non-experts.

Several features and usability improvements were implemented, iteratively reviewed by an MCDA specialist. A user interface for an MCDA method (DCM-SRF) was tailor-made directly for its creators, resulting in an ideal and simple card-sorting process - which can be easily understood by non-experts. The user evaluation was conducted with 20 participants of different MCDA backgrounds

and was extremely useful for gathering users' perspective and suggestions, and fixing some small bugs.

6.2 Future work

The developed work is at a stable position and will be immediately continued by other students. The roadmap of the project is well-defined, as new important features will be continuously developed until a public release is possible. Some opportunities are already coming up for testing the framework on a large scale, namely academic partnerships with professors all over the world who want to use the experimental version of DecSpace with their students.

When talking about the biggest steps to take next, XMCDA integration is the obvious one. After the 88th EWG-MCDA, it was agreed that DecSpace would be integrated with XMCDA with close support from the Decision Deck team, which is very interested in new perspectives towards their open-source technology, currently only interfaced via diviz's desktop application. By implementing XMCDA support, DecSpace will instantly gain access to more than a hundred web-services made available by Decision Deck, becoming a much more interesting framework. On the other hand, dedicated methods should still be developed exclusively for DecSpace so that it does not depend on the XMCDA web-services. The new MCDA developers are already working on new method implementations (e.g., ELimination Et Choix Traduisant la REalité (ELECTRE)) to increase the number of available methods, taking advantage of the new modular method system.

In addition, the backlog of the project contains many interesting features and usability improvements that will need to be added to the project. Error validation should be perfected even more, to help users and avoid as many mistakes as possible, paired with robust user feedback notifications (one of the most referenced features during the usability tests). Data visualization is another crucial feature to implement, providing the user with alternative visualization forms besides the existing tables. This should be done with D3.js[4], a data visualization library which will easily integrate with the JavaScript-based DecSpace project.

Finally, before releasing the application to the public, it is essential to implement and improve the documentation of the project on the user side. This means standardizing the method catalogue and finding ways to make it as simple as possible, while also maintaining technical rigor (references to bibliography should be included, as well as practical use scenarios with examples of input, parameters and output. This can be paired with numerous usability and QOL improvements on the roadmap to make a solid and simple application.

A final note to the MCDA Method Selection Tool[5], a tool that helps select MCDA methods tailored to the specific decision problem [16]. This could be an excellent addition to DecSpace for non-experts that do not know which method is right for them.

REFERENCES

[1] João Amador, Ana Sara Costa, Rúben Rodrigues, José Rui Figueira, and José Borbinha. Exploring MCDA methods in DecSpace. *Atas da 18ª Conferência da Associação Portuguesa de Sistemas de Informação 2018: A Indústria 4.0 e os Sistemas de Informação (Proceedings of the CAPSI 2018)*, pages 139–149, 2018. https://aisel.aisnet.org/capsi2018/14/.

[2] João Amador, Ana Sara Costa, Rúben Rodrigues, José Rui Figueira, and José Borbinha. DecSpace: A user-friendly web-based platform to explore MCDA methods. *The 88th meeting of EURO Working Group on Multicriteria Decision Aiding*, September 2018.

[3] Ana Sara Costa, Rúben Rodrigues, André Xiang, José Rui Figueira, and José Borbinha. Supporting the use of decision aiding methods by non-specialists. In B. Shishkov, editor, *Lecture Notes in Business Information Processing. Business Modeling and Software Design*. Cham, Switzerland: Springer-Verlag, 2019.

[4] Bernard Roy. *Multicriteria Methodology for Decision Aiding*. Springer US, 1st edition, 1996.

[5] Alessio Ishizaka and Philippe Nemery. *Multi-Criteria Decision Analysis: Methods and Software*. John Wiley & Sons, 06 2013.

[6] Denis Bouyssou, Thierry Marchant, Marc Pirlot, Alexis Tsoukiàs, and Philippe Vincke. *Evaluation and Decision Models with Multiple Criteria: Stepping Stones for the Analyst*, volume 86. Springer US, 01 2006.

[7] Salvatore Greco, Matthias Ehrgott, and José Rui Figueira. *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 233. Springer-Verlag New York, 2nd edition, 2016.

[8] Jyri Mustajoki and Mika Marttunen. Comparison of multi-criteria decision analytical software for supporting environmental planning processes. *Environ. Model. Softw.*, 93:78–91, July 2017.

[9] Granger Morgan and Max Henrion. Analytica: A Software Tool for Uncertainty Analysis and Model Communication. In *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, 1998.

[10] Cristina Verdasca. A Framework to Explore MCDA Methods. Master's thesis, Instituto Superior Técnico, 2016.

[11] André Barbosa. Decspace: A Multi-Criteria Decision Analysis Framework. Master's thesis, Instituto Superior Técnico, 2017.

[12] João Amador. Decspace: User Interface Challenges in an MCDA Framework. Master's thesis, Instituto Superior Técnico, 2018.

[13] M. Bottero, V. Ferretti, J.R. Figueira, S. Greco, and B. Roy. On the Choquet multiple criteria preference aggregation model: Theoretical and practical insights from a real-world application. *European Journal of Operational Research*, 2018.

[14] Luis C. Dias and João N. Clímaco. Additive aggregation with variable interdependent parameters: the VIP Analysis software. *Journal of the Operational Research Society*, 51, 2000.

[15] Manuel J. Fonseca, Pedro Campos, and Daniel Gonçalves. *Introdução ao Design de Interfaces*. FCA - Editora de Informática, 2nd edition, 2012.

[16] Jarosław Wątróbski, Jarosław Jankowski, Paweł Ziemba, Artur Karczmarczyk, and Magdalena Zioło. Generalised framework for multi-criteria method selection. *Omega*, 86:107 – 124, 2019.

[4] https://d3js.org/
[5] http://mcda.it/