ICEUBI2011
International Conference on Engineering UBI2011
Innovation & Development

beSMART: a software tool to support the selection of decision software

Anabela Tereso, Ana Sampaio, Hugo Frade, Miguel Costa, Tiago Abreu
School of Engineering, University of Minho, Portugal
anabelat@dps.uminho.pt; anaisamp@gmail.com; emanspace@gmail.com; miguelpintodacosta@gmail.com; tiagoalvesabreu@gmail.com

Conference Topic – CT6 – Computer Science

Abstract

This paper presents a tool which goal is aiding the user to choose the best Decision Support Software between a set of Software residing in a database, according to the features desired, and using multicriteria decision methods.
This application was developed using the C# programming language and allows to save in a file all user data, such as the set of Decision Software Tools under consideration and its features to be used afterwards.

Key Words: Multicriteria Decision Methods, Decision Support Software.

Introduction

Nowadays making decisions, especially in the business world, requires a lot of responsibility, because of the consequences they may bring. To choose a software tool is not an easy task, because of the available number of alternatives and characteristics under consideration. All choices must be well thought out. Thus, it becomes increasingly necessary to consider all the alternatives on the market in order to choose the best one available, for each case under study.
The number of Decision Support Software available in the market has been increasing dramatically making the choice of the best option for a particular context a complex task. We developed an application aimed to assist users (experienced or not) to choose the best Decision Support Software for each case.
We started by considering the multicriteria methods and the calculations they involve [1][2], and some review papers on Multicriteria Decision Aid Tools and Applications [3][4] and [5]. The methods used in the software developed will be explained with an illustrative example.
Before starting to develop the application, we also explored Web-HIPRE [6], which is a software for decision analytic problem structuring, multicriteria evaluation and prioritization. There are some common features between this software and our application, but while Web-HIPRE is more general, our application is focused on support selection of decision software packages, having some characteristics that help in the selection of these applications, like the possibility to view their webpage.
Our application, called beSmart, is represented by the logo shown in figure 1.

Figure 1 - beSmart's logo

We believe that our application is innovative. Besides the MCDAS Tool [3][4][5] we didn't find in the literature any other tools dedicated to the selection of multicriteria software. beSmart was a completely new developed application, designed to help users to select decision support software, in a simple and intuitive way, helping the user to make the best possible

decision. It can also be used with other types of problems. Due to the methodology used, which is carried out in steps, the user can realise in what stage he or she is in the decision process, helping to make the decision process more reliable.

Multicriteria methods

To make comparisons between different software we used three multicriteria methods: SMART [7], AHP [8] and ValueFn, based on the Utility Theory [9][10].

SMART method: is a simple and quick technique to decide the priority between alternatives. It consists of assigning points to each alternative, where higher scores represent more important alternatives.
AHP method: is a structured technique for dealing with complex decisions. Consists in filling a table, by assigning a degree of importance to each criterion, in relation to another. This method requires the computation of the consistency rate, which is a mathematical value of reference that shows the consistency of the comparison done. According to Saaty (1980) this rate should be lower than 0.10 [8].
ValueFn method: corresponds to a function that maps directly to the evaluation of the alternatives. It can be a maximization or a minimization function, depending on the user's intention to maximize or minimize the given attribute.
These methods will be further explained bellow.

Application Development

This application was developed in three steps: requirements analysis, modelling using UML (Unified Modelling Language) [11] and, finally, implementation.

Requirements analysis

The requirements analysis was done through some meetings with the client, where it was realized what he really wanted, as described next.
The main goal of this application is to make comparisons between a set of software, where the user may select software and characteristics involved, as well as the methods he wants to use in his comparison. The beSmart's task is classifying software according to the user's preferences.
Among all the requirements asked, stand out the possibility to edit the software list, allowing managing contents of the database, which means the database should be dynamic, and also allow the user to have his own database file.
There are other important aspects required like seeing the software webpage, which allows visualizing the list of software in the database and its webpage (see figure A7 in appendix A).

Modeling using UML

In order to obtain a good code organization regarding all tasks required, we performed the problem modeling using UML language [11].
We created a Use Cases diagram (see figure 2), Sequence diagrams that describe how a given function is performed, and finally a Class diagram that describes the structure of the main classes of our software (see figure 3).

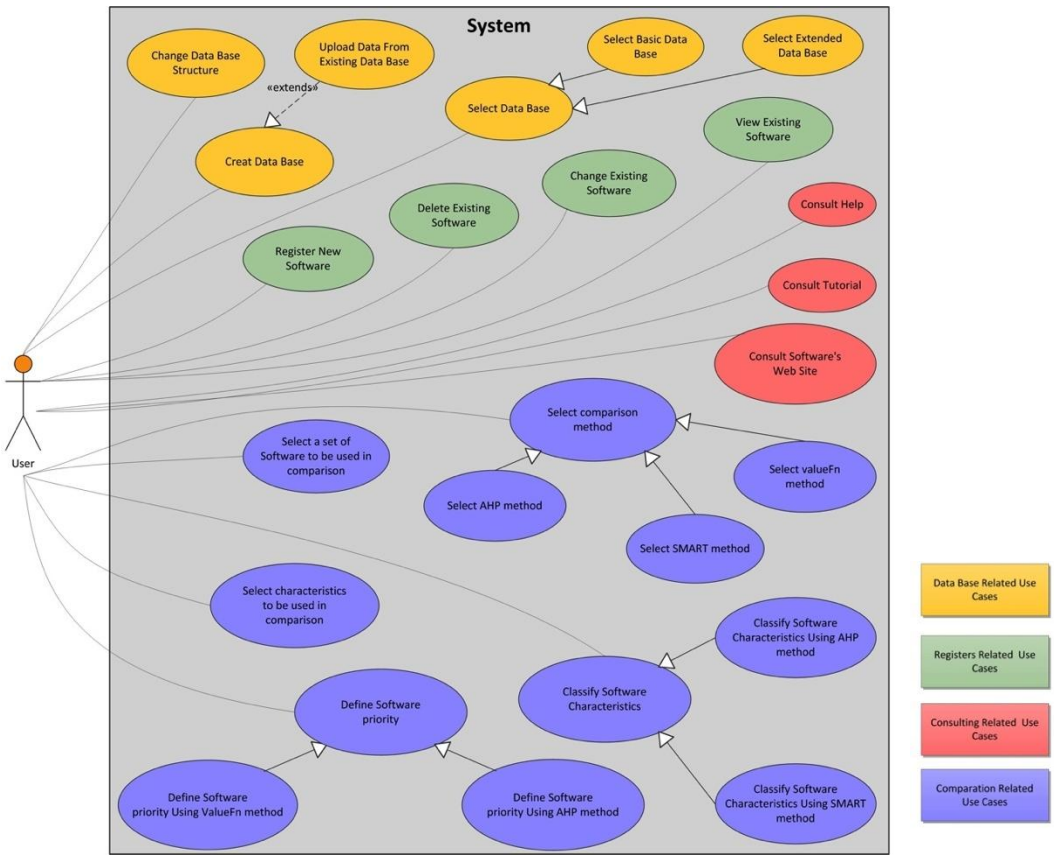

Figure 2 - Use Cases diagram

The use cases diagram (figure 2) describes all actions that a user can perform in the application. Different colors are used to represent different groups of actions.

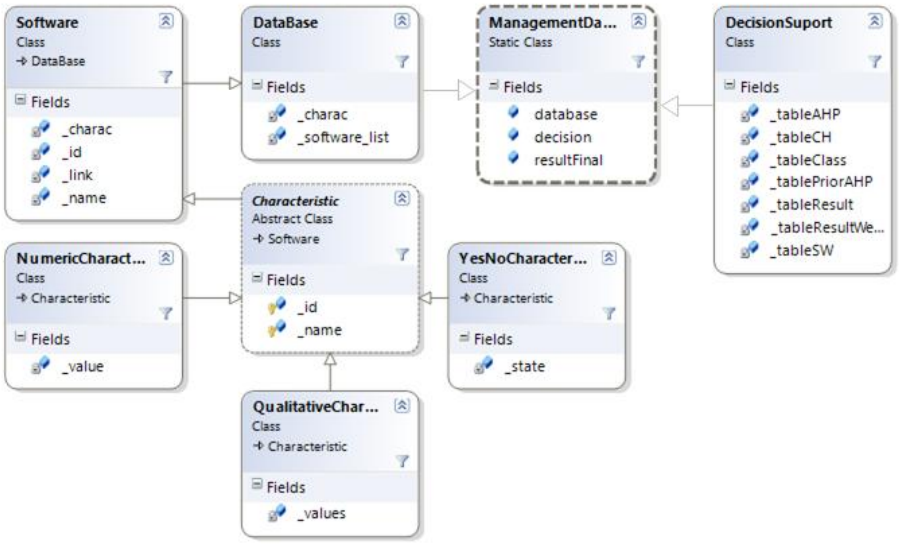

Figure 3 - Class diagram

In the class diagram, the ManagementDataBase class (ManagementDa… in figure 3) is responsible to apply the Decision Support Module to the Database Module.

Implementation

The application architecture is organized in three layers:
1. **Data**, which is responsible to provide information to the business layer.
2. **Business**, which makes all the calculations needed for the final result.
3. **Interface,** which is responsible for user interaction with the system.

The language chosen for the design of this application was C# [12][13], due to a requirement imposed by the scope where the project was integrated.

Using beSmart

The first interaction with the software is the welcome window, which has a welcome message to the user.

Figure 4 - Welcome Window

After the user presses the enter button, he can start using the application. A small user guide is presented in appendix A.

A comparison example is given in appendix B.

Decision Process Explanation

To start the decision process, the user must choose the set of software to be compared. Then, the user is prompted to indicate which features (criteria) he wants the program to include in the decision process. After this, the user is faced with the task to assign weights to the features chosen, using SMART or AHP.

If the SMART method is chosen, the program presents the features and the user must assign 10 points to specify the least important feature. Then, he assigns a higher number of points to other features, depending on their importance compared to the least important one. After that, it is necessary to normalize the values entered. This is computed by dividing each value by the sum of all values. The results are shown in a table, as in the following example:

Table 1 – Computing the priority of criteria using SMART

Criteria	Points	Priority
Interaction with User	10	$\frac{10}{25}=0.4$
Cost	15	$\frac{15}{25}=0.6$
Sum	25	1

If AHP method is chosen, a table is presented to the user allowing pair-wise comparison of criteria in the achievement of the overall goal (select the best software).
The main diagonal of the table is filled automatically. The user must assign the comparison values above the main diagonal, respecting the scale proposed by Saaty (1980) [8], although intermediate values may be used:

Table 2 – Saaty's scale

If *x* has/is … that/than *y*	So the preference number to be given is:
The same importance	1
A little more important	3
A much more important	5
A far more important	7
Absolutely more important	9

Values like 1/5, for example, may also be used to represent that y is much more important than x, while 5 represents that x is much more important than y (as in table 2).
For each column, the sum of all values is computed. Then the result matrix is normalized. Afterwards, the sum and the average of the values contained in each line are evaluated. The average values are the final weights of each criterion.

The next step is calculating the consistency rate. Initially, the comparison matrix is multiplied by the final weight matrix calculated above. Subsequently, each value is divided by the corresponding value of the weight matrix, resulting in a vector. The average of these values is very close to its maximum eigenvalue (λ_{max}). Then it is necessary to calculate the consistency index, which for a matrix of size N is given by the following formula:

$$\text{Consistency index} = \frac{\lambda_{max} - N}{N - 1}$$

Afterwards, Saaty's Random Index (RI) table (table 3) is used, to check the consistency rate.

Table 3 – Saaty's Random Index

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RI	0.00	0.00	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.54	1.56	1.57	1.58	1.60

Depending on the number of features to be compared (N) the RI is chosen. The maximum value that we have found was for N=16. Finally, to evaluate the consistency rate, the consistency index is divided by RI. If the consistency rate is less than 0.1, the matrix is considered consistent. If the value is greater than 0.1, an iterative method will be applied to improve the consistency rate.

If we have to iterate to improve the consistency rate, the steps used are described below:

- Multiply the final weight matrix by the matrix containing the values entered, resulting in another weight matrix.
- Normalize this matrix, and obtain a new matrix of final weights.
- Compare this new final weighted matrix with the one obtained previously.
  - If the maximum difference between each value is 0.0001, recalculate the consistency rate, the same way as before.
  - If the difference is higher, the steps are repeated. It means it is necessary to multiply the last final weight matrix by the matrix that contains the values entered, normalizing the resulting matrix and re-check the difference with the previous final weight matrix.

After this phase has been completed successfully, begins a new one that can be called Priority Setting. This phase is mainly directed to the feature. For each feature, the user can choose between these two methods: AHP or ValueFn.

If the user chooses the ValueFn method for each feature, he is prompted to choose if he wants to maximize or minimize it. For example, in the case of the cost of software feature, the user would want to minimize it. On the other hand, in the case of the quality of user interaction feature, the maximization would be more adequate.

Then, for each feature/software, one of the following formulas is used, depending on the interest to maximize or minimize it. In the case of minimization the following formula is applied:

$$y = \frac{x - Min}{Max - Min}$$

In the case of maximization, the applied formula is:

$$y = \frac{Min - x}{Max - Min}$$

Min and Max are the minimum and maximum values presented in the software chosen. The following example shows how the priorities are evaluated using the ValueFn (in a case of maximization):

Table 4 – Compute the priority of Interaction with user, maximizing the feature

Software	Interaction with User
A	1
B	3
C	5
Min	1 (A)
Max	5 (C)

	Priority	Normalized Priority
	0	0.000
	0.5	0.333
	1	0.667
Sum	1.5	1

After the priorities are calculated, they are normalized.
For a case in which it is intended to minimize the feature, the results are:

Table 5 – Compute the priority of Cost, minimizing the feature

Software	Cost
A	100
B	800
C	1000
Min	100 (A)
Max	1000 (C)

	Priority	Normalized Priority
	1	0.818
	0.22	0.182
	0	0.000
Sum	1.22	1

AHP method can also be used in this step (setting priorities to software). This works just like explained above for the criteria.

Final Results

The final ranking of the software is obtained by doing some final calculations. The priorities for each software, relative to each criterion, are multiplied by the weight of the respective criteria. Then, for each software, the previously obtained values are summed up. These values are the final software weights, considering all criteria.

The software with highest value will appear at the top of the rankings, and the rest in descending order. This result is finally presented to the user.
Several graphics and tables are showed to the user to support the interpretation of the results achieved (see figure B8 in appendix B).

beSmart is available for free download and is ready to be installed through the following link: http://code.google.com/p/besmart/ [14]. All documentation about this application can also be consulted in the software webpage.

Conclusion and future work

Although its implementation was complex and time consuming, the decision support module is totally functional, and its success makes the results provided by beSmart safe and reliable.
We consider that beSmart interface is intuitive for a user who does not know the methods used in a decision-making process. Moreover, this application is generic and can be used to make any kind of comparison, not only software but also other things like cars, computers, and so on.
As future work, it is our intention to improve the software by adding new features like sensitivity analysis, which will allow giving even more confidence to the user on the results obtained. We also plan to implement a Help Module, with a search system, able to present to the user some tips to clarify doubts during the decision process.

References

(1) Vincke, P. (1992) Multicriteria Decision-aid, New York, John Wiley & Sons.

(2) Zeleny, M. (1982) Multiple Criteria Decision Making, McGraw-Hill.

(3) Tereso, A., Seixedo, C. (2010) “Multicriteria Decision Aid: Evaluation and Comparison of the Main Tools”, EURO XXIV conference, Lisbon – Portugal, July 11-14, 2010.

(4) Tereso, A., Seixedo, C. (2010) “A Multicriteria Decision Aid Software Application for selecting MCDA Software using AHP”, 2nd International Conference on Engineering Optimization, Lisbon – Portugal, September 6 - 9, 2010.

(5) Seixedo, C. (2009) “Multicriteria Decision Aid: Evaluation and Comparison of the Main Tools”, Systems Engineering Master Dissertation (in Portuguese).

(6) Web-HIPRE, Global decision support Copyright 1998-2007 Systems Analysis Laboratory, Helsinki University of Technology,

(7) Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. Management Review, Volume 70, Issue 11(AMA FORUM), pp. 35-36.

(8) Saaty, T (1980), The analytical hierarchy process, McGraw-Hill, New York.

(9) Canada, John R. and Sullivan, William G. (1989) Economic and MultiattributeEvaluation of Advanced Manufacturing Systems, Prentice Hall College Div. Chapter 9: Multiattribute Decision Analysis: utility models.

(10) Keeney, R. and Raiffa, H. (1993), Decision with Multiple Objectives: preferences and value tradeoffs, Cambridge University Press.

(11) FOLDOC (2001). Unified Modeling Language last updated 2002-01-03. Accessed 25 July 2011, http://foldoc.org/UML

(12) C Sharp (programming language), http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29

(13) Visual C# Developer Center, http://msdn.microsoft.com/en-us/vcsharp/aa336706

(14) beSmart setup: http://code.google.com/p/besmart/

Appendix A

beSmart User Guide

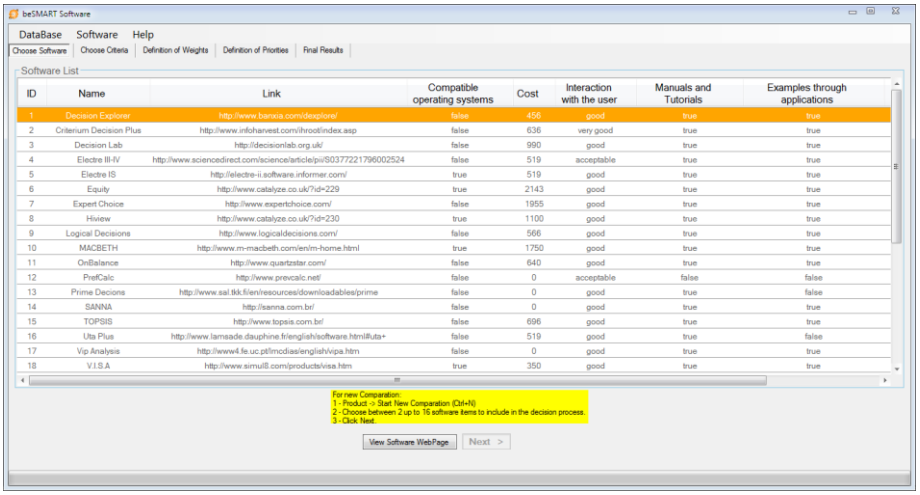

Figure A1 - Software List window

Software List Window

After the welcome window, it appears the Software List window (figure A1). Here it can be accessed the list of software available and their characteristics.

Menu Bar

The menu bar is presented in all windows except the welcome window. There are three sub-menus: DataBase, Software and Help.

DataBase sub-menu
In this menu the user can open a file with the appropriate data, “save” or “save as” the data into a file, and finally exit the application.

Software sub-menu
The main feature of this program is to compare software. So, the user starts a comparison of software by selecting the option Start a new Comparison, in this menu. When this button is pressed, it appears in the beginning of each line a check box, which will allow selecting the set of software to be compared.
The next option is Edit Software List, which allows adding or removing software as well as their characteristics.
Finally in this menu the user can see the software web page pressing the button View Software Webpage.
We will further explain these two options bellow.

Help sub-menu
The Help sub-menu presents two options: Tutorials and About.
In the Tutorial option, the user can see the information about how to use each of the multicriteria methods presented above. Figure A2 shows an example. Figure A3 shows the About beSmart window with the developers of the application.

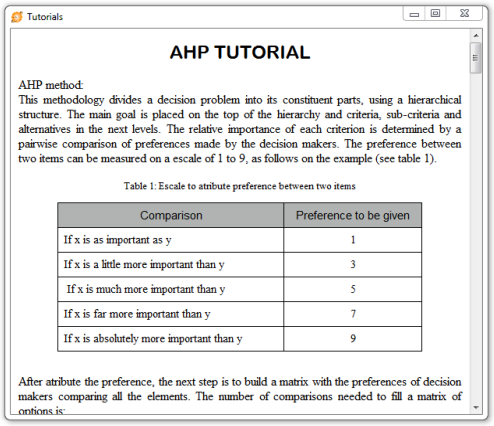

Figure A2 – AHP Tutorial: explains how to use the method

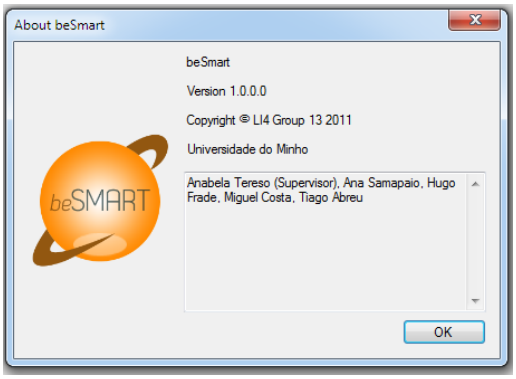

Figure A3 - About beSmart window

Edit Software List

This option gives the chance to the user to manage his software list, by performing the following actions: add new software or delete software from the list (see figure A4).
This option also allows adding, removing or editing software characteristics.

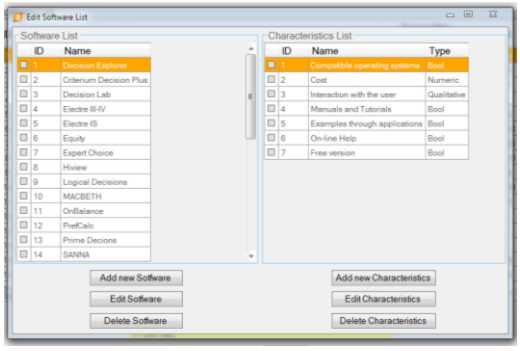

Figure A4 - Edit Software List window

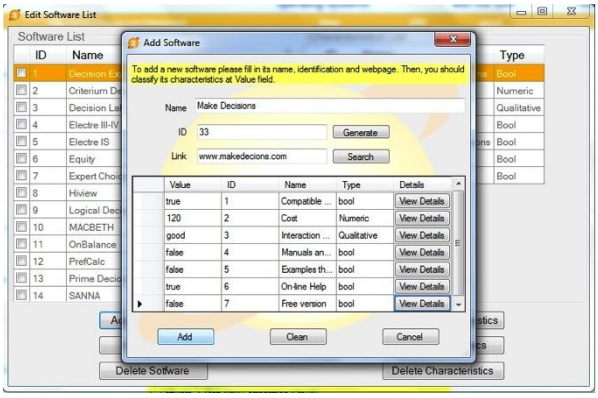

Figure A5 – Adding a new software called “Make Decisions”

Add Software
If the user wants to add new software, in addition to name, identification and webpage link, he must fill in all other characteristics (see figure A5).

Edit Software
To edit Software information, the user has to select which one he wants to edit and fill with the correct values.

Delete Software
If a software item no longer exists, or simply loses interest to the user, it can be deleted from the database.

Add Characteristics
There are three types of characteristics: numerical, qualitative and yes/no characteristics. A numerical characteristic is a number, to be applied to characteristics like cost. A qualitative characteristic is described by a scale, such as, 1-bad, 2-normal and 3-good. Finally, there are yes/no characteristics, to be applied to characteristics like on-line help.

When the user adds a new characteristic, it is added to all the software in the list, but with the null value. The user then has to complete its values with the option Edit Software (see an example in figure A6).

Edit Characteristics
This option allows the user to select which characteristics he wants to edit. After that, one by one, they will appear to be edited with new information.

Delete Characteristics
In the same way that we edit characteristics we remove them, by selecting which ones we want to remove and pressing the delete button.

View Software Website

The user can see each software website by double-clicking on the software's name (see figure A7).

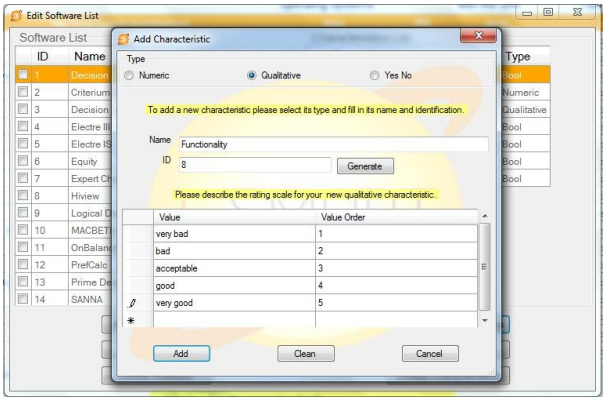

Figure A6 - Adding a new qualitative characteristic called Functionality

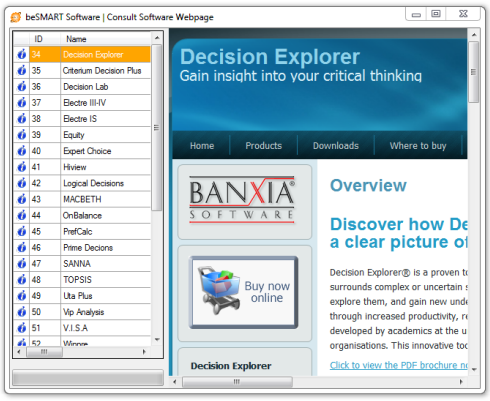

Figure A7 - Viewing the Decision Explorer Software website

Compare Software Process

As explained before, to start a new comparison we have to go to the software sub-menu, in the menu bar, and press "Start a new Comparison".
After that, the user has to select the software that he would like to compare. When the set of software is selected, the next button has to be pressed. Then, the following window appears and the stage is increased. Also, the progress bar, which is in the bottom of the page, moves forward 25% per stage, in the total of 4 stages. This bar only allows the user to realize at what stage he is, and how far it takes to finish the process.
Now the user has to select the characteristics that he would like to compare, checking the corresponding boxes.
With all the software and characteristics selected, it is time to start giving weights to criteria and priorities for the software within each criterion.
Firstly the user selects the method to define criteria weights, between Smart (figure A8) and AHP (figure A9).

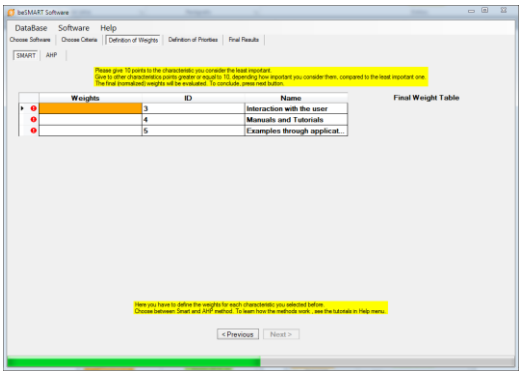

Figure A8 - Defining weights with Smart method

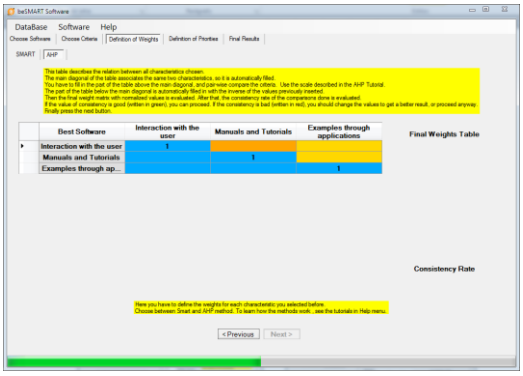

Figure A9 - Defining weights with AHP method

Each method should be filled according to the information given in the tutorial.

Then, the user proceeds to the definition of priorities between alternatives, for each criterion. Here it can be chosen ValueFn (figure A10) or AHP method (figure A11). The difference between this step and the previous one is that it can be chosen a different method for each characteristic, instead of using the same method for all.

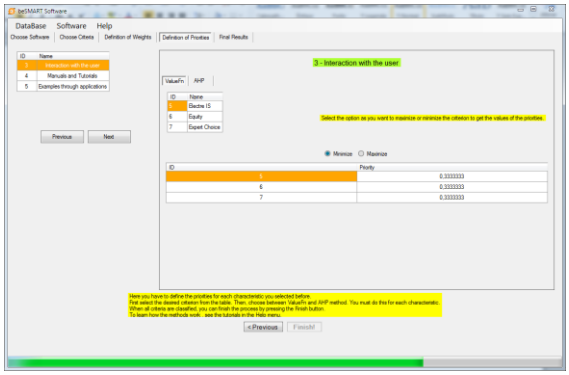

Figure A10 - Defining priorities with ValueFn method

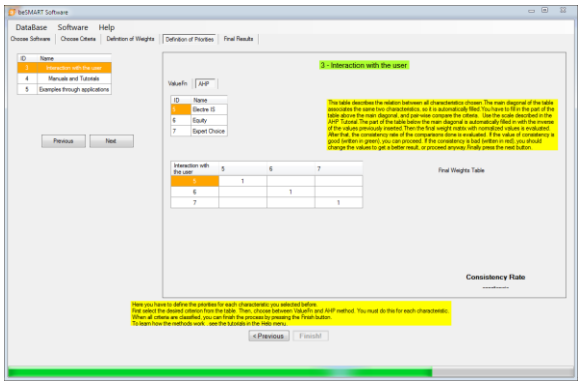

Figure A11 - Defining priorities with AHP method

At the end of this step, the process can be concluded, by pressing the Finish button. Then, the results are shown in a ranking table, including the weights given to each software (see an example in figure B8).

Appendix B

Example of a comparison done using beSmart software

After opening the application, we must go to the DataBase menu and select Open. Then, we choose a database file with beSmart extension, click Ok and the list of software in the database will be displayed. To make comparisons, we must go to the Software menu and choose Start a new comparison (or simply use CRTL+N).
Then, it will be displayed the window presented in figure B1. Here we should choose the set of software to be compared, between 2 and 16 software.

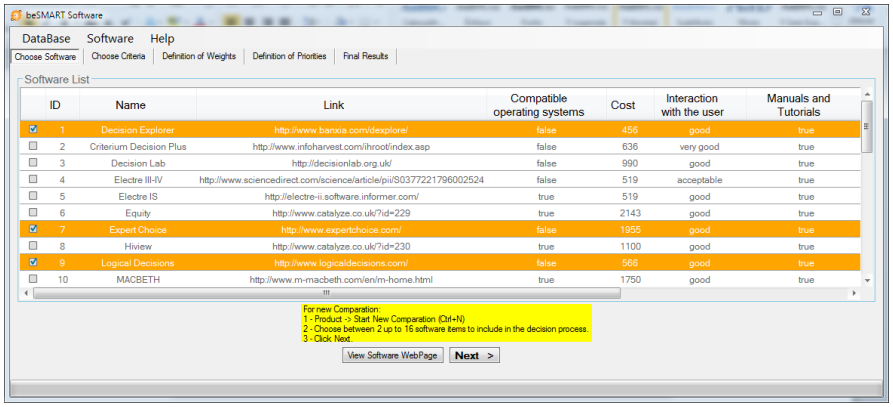

Figure B1 – Selecting software

On this example, the selection was Decision Explorer, Expert Choice and Logical Decisions.

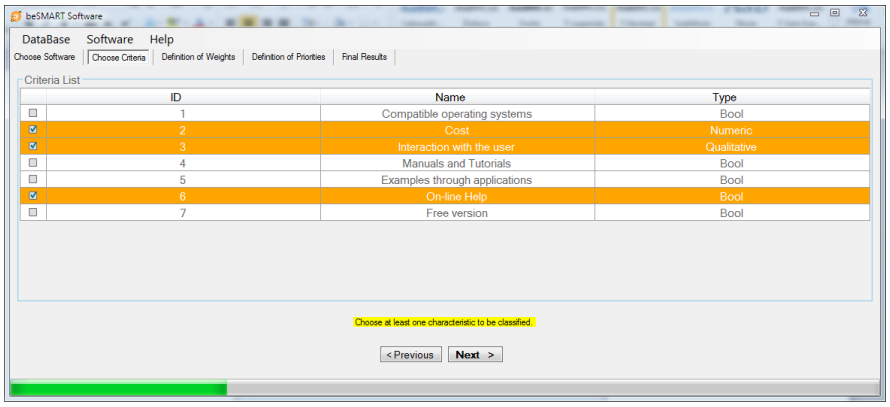

Figure B2 – Selecting characteristics

The next step is to select the characteristics considered important. In this example we selected Cost, Interaction with user and On-line Help (see figure B2).
Pressing Next button, it is time to give weights to the characteristics or criteria, specifying the relative importance of each one, compared to others. To do that, we need to choose a method, which can be Smart or AHP.

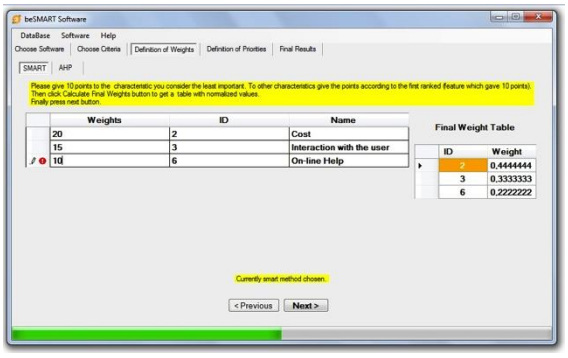

Figure B3 – Giving weights to the characteristics using smart method.

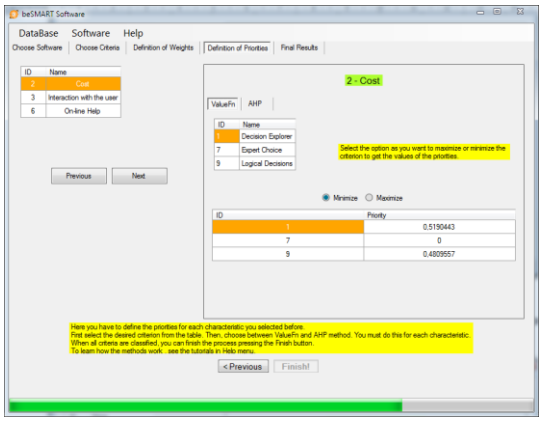

Figure B4 – Defining priorities for Cost using ValueFn method

In this example we choose the Smart method. We give 10 points to the least important characteristic (On-line Help), 20 points to the most important (Cost), and 15 to Interaction with the user (see figure B3).

Then, we move to the next step by pressing Next button. Now, we need to define the priorities of each characteristic. Here we can choose between ValueFn and AHP method.
For the first characteristic (Cost) we decided to choose the ValueFn method and minimize it (see figure B4).

Then, we select the next characteristic (Interaction with the user), by pressing Next button. For this one we choose AHP method.

As we can see in figure B5, the values we gave are not consistent. So, to have better results, we decided to change these values, until we achieved a good consistency rate (figure B6).

For the last characteristic (On-line Help) we choose ValueFn method again, but this time maximizing the characteristic (figure B7).

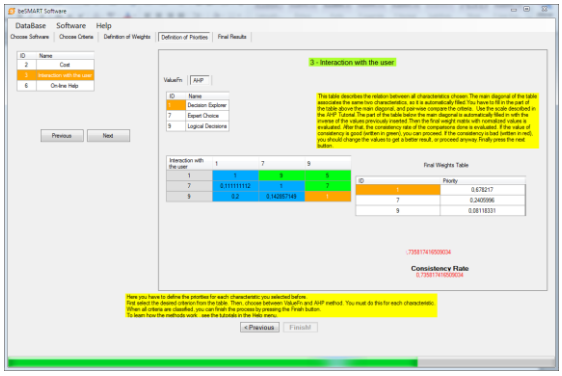

Figure B5 – Defining priorities for Interaction with user (bad consistency rate)

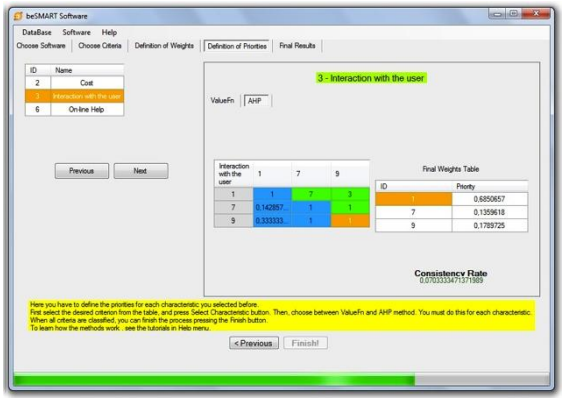

Figure B6 – Defining priorities for Interaction with user (good consistency rate)

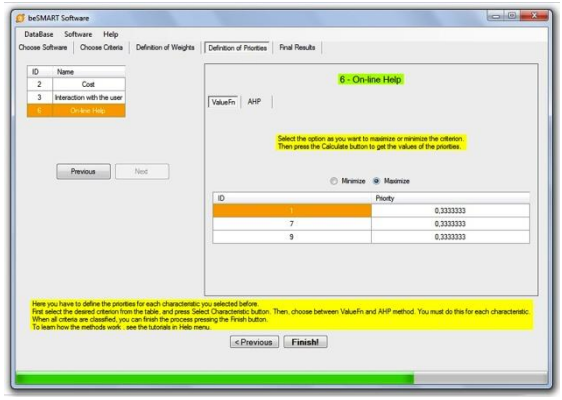

Figure B7 – Defining priorities for On-line Help

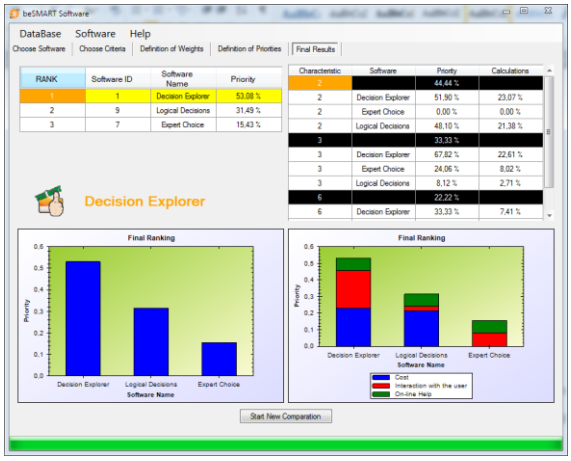

Figure B8 – Final Results

Completing this step, we finish the process and get the final results (see figure B8).

The results we got indicate that we should choose Decision Explorer, with 0.53 of priority; then Logical Decisions, with 0.35 of priority; and finally Expert Choice, with 0.12 of priority. To conclude, the software more appropriate in this case, considering the user choices, would be Decision Explorer.