 doi:10.20944/preprints202004.0306.v1

Article

Dealing Hybrid Model Appling in the Selection and Prioritization of Software Requirements Using Verbal Decision Analysis

Paulo A. M. Barbosa[1,2]*, Plácido R. Pinheiro[1], Francisca R. V. Silveira[2] and Marum S. Filho[3]

[1] Program in Applied Informatics - University of Fortaleza - Fortaleza ZIP 60811-905 - Brazil; albertobmap@hotmail.com, placido@unifor.br
[2] IFCE Federal Institute of Ceará - Tianguá 62320-000 ZIP Brazil; raquel_silveira@ifce.edu.br
[3] 7 de Setembro College - Fortaleza - 60135-420 ZIP Brazil; marum@fa7.edu.br
* Correspondence: albertobmap@hotmail.com

Abstract: In the software development process, the decision-maker (DM) has a range of problems inherent to its function. Wrong choices during software planning can bring great risk to the project. Therefore, the planning of software releases to be delivered to the customer should be well done. This is not an easy task because releases are made up of many requirements that contain complex variables that must be considered, such as precedence, cost, requirement stability, among other features that make the requirements-selection process challenging. To make this process less exhaustive, DM can use tools that facilitate this work. In software engineering, we can find fields of research specialists in this context, such as Search-Based Software Engineering (SBSE). The SBSE makes use of advanced metaheuristics to search for optimal solutions or the closest to it. In this work, we try to use another field of research to solve this same problem type, the Verbal Decision Analysis (VDA). To do this, we elaborate a workflow that will use the same source data, execute two solutions using the two search fields (SBSE and VDA) and compare the results. In the end, we evaluated and commented on the results.

Keywords: Requirements Planning; Search-Based Software Engineering; Verbal Decision Analysis

1. Introduction

In software engineering, we find several solutions that seek to solve problems involving competitions for resources, support decision-making in large projects, ambiguities [1]. Among these problems, we also find those related to Software Release Planning (RP) that covers problems related to the selection and allocation of attributes for an ordered set of software releases [2]. Poorly crafted PR can lead to problems in the software development process. This is because the Releases, parts of the software that will be made available to the customer, are made up of a set of requirements that represent a part of the totality of the software.

A systematic methodology used in the development of software can be an economic strategy. Currently, the logistics of developing and delivering software on schedule and with a good level of quality is the premise for development companies [3].

The Software Releases methodology is critical because it allows customers to receive and know parts of the software in advance. Thus, this strategy allows an assignment of values to the requirements of this software. In this way, each new version contains a set of characteristics that make up software, and that will have a significant value for the client [2].

We can see that agile methodologies perform incremental software deliveries. Deliverables, or sprints, have a short interval to be elaborated and are implemented by a group through a complete software implementation cycle that ends when the software is delivered to the customer. Through negotiation between the development team and the clients, it is defined what each delivery will

contain (set of requirements) [4]. We can deduce that in large and complex software, the task of choosing which requirements should be allocated in specific releases is not simple and requires a set of knowledge to execute such assignment [5].

The decision-maker (DM) has the mission of pointing out characteristics about each of these requirements, such as precedence, risks, level of complexity, personnel available for development, cost, time to implementation, interests to these requirements by the developer as well as by a set of interested clients, level of volatility or stability, being able to consider for implementing those requirements that tend to change less during the development process, ensuring that the most stable requirements can be delivered first, among many other things inherent to this assignment.

We can see that software requirements can have many characteristics. Commonly, the software is delivered in the right parts between developer companies and customers. In this way, it is the DM's responsibility to order the requirements in order of deliverables. To a set of requirements that will be delivered together, we call Software Release. Different tools and methodologies support the DM in order to facilitate the process of choosing requirements optimizing them in the appropriate Releases according to objectives and restrictions agreed with the clients. Such constraints are, for example, the interests of a group of customers by specific requirements, the anticipated delivery of the most stable requirements and especially the limit of available resources for the project, as well as other restrictions that may be appropriate.

In this way, tools that give this special support to DM have high relevance and can be an essential pillar for the success of the project. In researches, we could find some of these support methodologies [6-8]. Such automated methods have several strategies for finding solutions considering objectives and constraints. Therefore, the company (business value, reputation in the market, investors) needs to invest in reasonable support solutions.

This work follows extensive research [9-11] that aims to add more characteristics and improvements to a new methodology proposed to support DM. This work proposes the construction of a decision support strategy through the use of a pair of methods (different from those used in previous research how in [12]) in the area of Verbal Decision Analysis (VDA), shown in session 4. In the following sessions, we will detail (SBSE) as the primary strategy to find suitable solutions to this type of problem, and finally, we will present a comparison of results obtained between VDA and SBSE with the appropriate comments and the evolution of this research.

2. Prioritize Software Requirements

The publication of [5] shows a definition of requirements that can be delivered in the upcoming Releases of the software. The author describes that the requirements may have prerequisites and, therefore, should be implemented, only after their precedents have been developed. The implementation of both in parallel can be considered. The author also emphasizes that customers may have separate relationships with the company and that this may affect the choice of requirements for relays.

Among the risks faced by companies that are developing software, we can find what it is about meeting customer expectations. These authors consider that these risks can impact the reputation of the company and the loss of good projects [13]. Thus, to alleviate this problem, a proper elicitation and allocation of requirements can be a viable solution. These tasks occur early in the software development process.

The number of requirements that are part of a Releases is generally higher than what was elicited. Thus, proper planning of requirements planning is necessary. Poorly designed projects can lead to customer dissatisfaction, non-compliance with objectives and restrictions, and implementation of releases that do not bring value to the business.

In many aspects, the selection and prioritization of software requirements should take into account the prerequisites already mentioned, customer satisfaction for that set of requirements, and the time it will take to implement each software requirement [5].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

Users always have expectations about the software they are going to receive. Therefore, taking into account the client's choices is essential and can be decisive for the excellent progress of the project. This makes requirements prioritization very relevant [14].

The volatility of the requirement is another aspect that we can consider in the prioritization of requirements. Volatile requirements are a problem that must be understood so that the right choices are made. This is not easy and should be considered during requirements elicitation. Their characteristics bring risks and can impact the progress of the project [15]. Requisites that are likely to change bring problems for the developer company if they are not identified in advance [16]. A real-life software development research was conducted to verify the impact of volatile requirements on software projects. What has been observed is that the main reasons for changes in requirements are the changes in requirements arising from new systemic methodologies adopted by the customer, the exclusion of design requirements and the change of internal requirements specifications. In this same work, it is seen that the changes are due to external causes such as changes in the market, developers' understanding of what is being developed and some other considerations such as reduction of scope and changes in internal policies [15].

Thus, in this research, we consider the stability of the requirement because it is considered essential, and we add this factor as a selection criterion (session 6).

We can find many methods that use genetic algorithms to solve maximization problems. One of these methods is EVOLVE [17], which considers the use of a pair of functions to maximize benefit while minimizing risks. This method supports decision making in a scenario where changes can occur. Stakeholders play an essential role in the company, and it can define values and levels of importance to each requirement. Prerequisites are taken into account in this search strategy. The evaluation of this method used data from a case study and had good results. We can observe the strategy of this method in figure 1 [17].

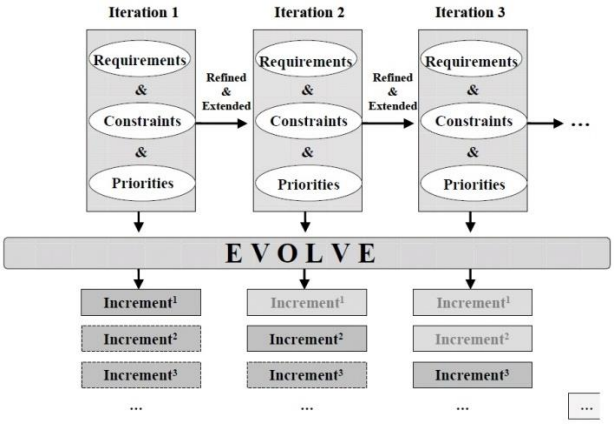


Figure 1. EVOLVE

The selection and prioritization of requirements considering stability ensure that changes in volatile requirements are realized before they are implemented. Thus, there is a reduction in costs with changes in these requirements, and those that tend not to change can be delivered first by optimizing project time and reducing delivery time.

Thus, we emphasize that the selection and qualification of requirements is a fundamental factor for success in software projects. In the next sessions, we will show known methods in the search for solutions and present the solution proposed by this research.

3. Search-Based Software Engineering (SBSE) Methodology

As regards Software Engineering, there are many problems related to the software development process, such as confusing information, a conflict between constraints, and a large number of choices and decisions that need to be made. Solving this type of problem is very difficult since there is no optimal solution [18]. When we consider that a Releases group has a set of requirements to be

 doi:10.20944/preprints202004.0306.v1

allocated, the complexity of this problem tends to increase especially, when we consider a set of goals and constraints, such as time, cost, customer satisfaction, among others.

Moreover, to solve this type of problem, a new field is known as *Search-Based Software Engineering* (SBSE) was created, which deals with the application of algorithms that use optimization to enable the search for reasonable solutions to problems inherent in software [19] promptly. In this area, problems related to Software Engineering are solved by search-based techniques, a variant of the Operational Research, and may have their problems also solved by Genetic Algorithms (metaheuristics) [1].

In this field, genetic algorithms are used as a technique to search for solutions to software engineering problems. The problems are seen here consider complex tasks performed by DM's. In this way, automatic searches are used that consider human experience. This new field was created in 1990, where the first implementations were given in the area of project management. SBSE was first cited in [19], where the authors presented the SBSE, and from then on, many research and methodologies were proposed in that field [18].

When we look for literature, we find strategies that aim to solve SBSE problems. Also, as a way to enable adaptation between the strategies adopted in this work with the existing methodologies in SBSE, we seek among the strategies that compose it, those that have a high acceptance by the scientific community and that may have the capacity of adaptability to the research of this work. Thus, both fields (SBSE and VDA) may have very similar methods both in the form of data entry and in the presentation of results, minimizing the adaptation phase of the data between the two fields of research. The methods adopted in SBSE for this work are described in sections 3.1 and 3.2.

3.1. NSGA-II Metaheuristic

The NSGA-II (Non-dominated Sorting Genetic Algorithm) [20] that works by ordering solutions through the layer in which they are on a Pareto front maintains non-dominated solutions when changing generations.

This algorithm starts with population values in P_0 and applies the strategy of *Fast Non-Dominated Sort*, which aims to find solutions close to the Pareto front. The solutions are classified according to the degree of non-dominance. This creates a front to which this solution and a set of other solutions belong. Employing special operators, a population Q_0 is created.

After creating an initial population, the goal of this metaheuristic is to create $Rt = Pt \cup Qt$, where the *Fast Non-Dominated Sort* strategy considers *Rt* as population and the boundaries are established. *F1* contains solutions not dominated and reliable candidates to be part of the front of Pareto. After establishing the values for the new front, the *crowding* runs to the last front. Consider *n_sol* a solution, and *n_pop* the number of NSGA-II individuals belonging to *n_rank*. If the solution *n_sol* + 1 belongs to *n_rank* + *1*, then the *crowding distance* calculation is not necessary, because the solutions generated until *n_pop* will already be considered for the next generation. However, if *n_sol* + *1* and *n_rank* is on similar fronts, *crowding distance* is applied to each *n_rank* solution. In this way, these *n_rank* solutions will be taken to the next generation, except those with the worst *crowding distance* values.

To achieve elitism, solutions of lower fronts are preferable at first. Thus, an overlapping distance is calculated. Soon solutions with larger *crowding distance* values are preferable.

In Figure 2 [20], we have the presentation of the strategy used in NSGA-II.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

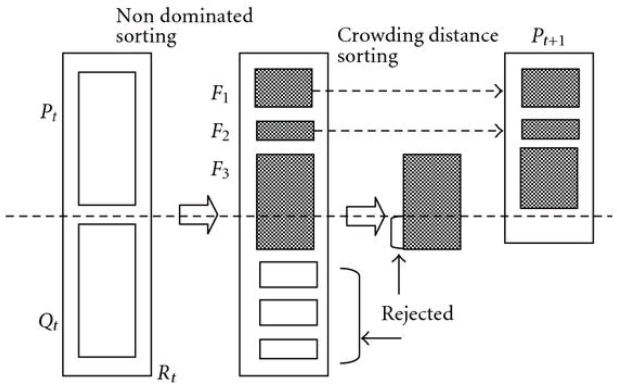


Figure 2. NSGA-II

3.2. MOCell

MOCell [21] is a metaheuristic that is structured in a Genetic Algorithm (GA). In order to increase the Pareto front that exists in this method, increasing the diversity among the generated solutions, a solution spacer is used as seen in the previous item, following the same objective. However, this method removes some solutions when this front is very filling.

Initially, we have a Pareto front with no solutions. A reproductive process is applied until the stopping conditions are satisfied. In this way, it combines individuals to obtain new offspring who will then be modified and recombined. The new individual will be evaluated and placed in an auxiliary population if he is not dominated, in front of Pareto. With each new generation, the current population is replaced by the new population through a feedback strategy. We can find more details about this algorithm [21].

4. Verbal Decision Analysis (VDA)

The Verbal Decision Analysis (VDA) field is inserted in the Operational Research and has a set of methods that classify and order alternatives where multiple criteria are considered to find solutions to problems [22]. Thus, this field of research systematically analyses actions to support decisions taking into account the verbal characteristics and appreciation of attributes qualitatively. This is the opposite of traditional methods that use quantitative factors. In Figure 3, we have a tree of methods that can sort or sort solutions.

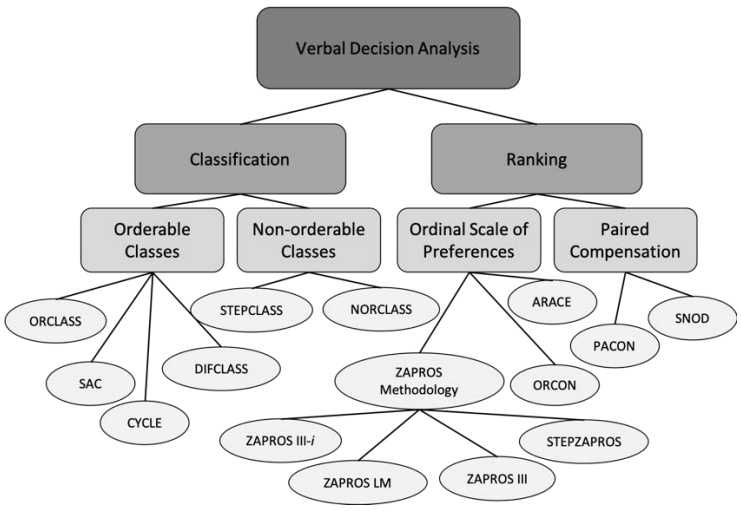


Figure 3. VDA Some methods of VDA

According to [23], we can highlight the benefits of VDA methods:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

1. VDA offers a natural language for DM;
2. Their methods have strategies to solve problems of inconsistencies;
3. For DM the VDA methods are transparent;
4. Through an analysis of the listed preferences, we can obtain explanations about the solutions that were generated;
5. In the DM view, preference elicitation methods are transparent, since VDA methods use verbal strategies.

It is interesting to analyze that VDA has excellent interaction with problems that have a high number of alternatives, but it is limited in the number of criteria and criteria values [24].

The use of these methodologies, such as ORCON (a decision aid for ordinal consistency) [25], for a given problem, can present a significant number of possible solutions. This is because many possible combinations can be generated. Therefore, we have a step called elicitation of preferences, in which the alternatives will be compared manually [25,26].

In the case of classification methods, among many, we have the SAC method that aims to classify the solutions according to the criteria informed by the DM. In section 5, we present in detail the proposal of this work that is to classify requirements using the SAC method and to order them using the ORCON method.

4.1 SAC - Overview and Structure

In the VDA field, we find a set of methods that deal with multicriteria classification problems. We can observe some of them in figure 3 [27]. The Subset Alternatives Classification (SAC) classifier [28] consists of a methodology that ordinally classifies a relatively small set of alternatives, in our case software requirements, that need to be classified only once [29]. This method has the objective of showing a small number of questions to the DM in order to reduce the time spent during the process of preferences. For this to be possible, this method has a variance to calculate the most informative vector structured in its possibility of being allocated to a class.

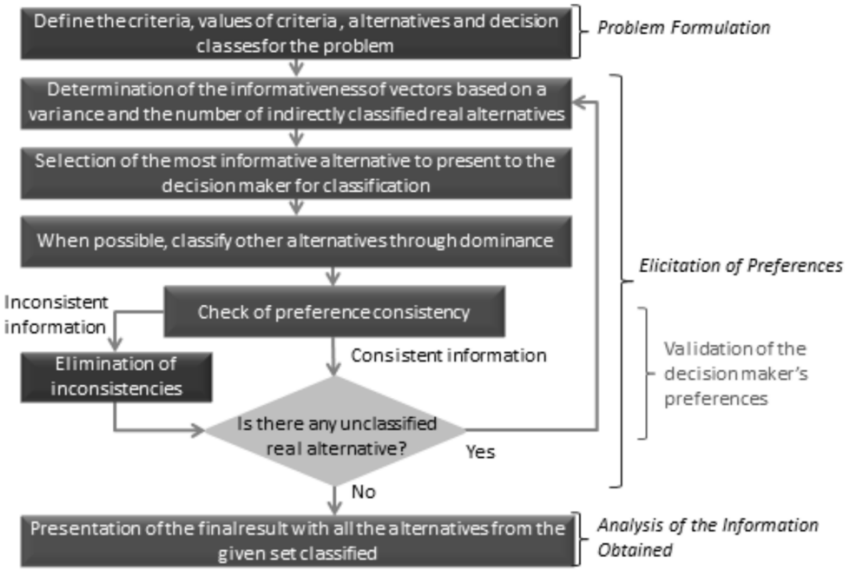


Figure 4. Subset Alternatives Classification Methodology

The formulation used by SAC in this work follows similarly to that used in [28], as follows.

1. $K = 1, 2, \ldots, N$, illustrating a set of N criteria;
2. n_p describe the number of possible values on the scale of p-th criterion, $(q \in K)$;
3. $X_p = \{x_{ip}\}$ describes a set of values to the p-th criterion, which is on this criterion scale; $|X_p| = n_p (p \in K)$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

The scale values are selected from best to worst, and this sort of order is independent of the values of other scales generated;

4. $X = Y_1 \times Y_2 \times \ldots \times Y_n$ represents a set of vectors x_i such that: $x_i = (x_{i1}; x_{i2}; \ldots; x_{iQ})$, and $x_i \in X, x_{ip} \in Y_p$ and $Q = |X|$, such that $|X| = \prod_{i=1}^{i=N} n_i$;
5. $B = \{b_i\} \subseteq X, i = 1, 2, \ldots, t$ in which the totality of t vectors illustrates the definition of the real alternatives;
6. $C = \{C_1, C_2, \ldots, C_L\}$ representing the set of ordered decision classes.

The application of the SAC method is very similar to that proposed by ORCLASS. What differs from the two methods is the fact that SAC considers the real alternatives when calculating an informativity of the vectors to be shown to the decision-maker to choose. Therefore, it is a less complicated process because it is not necessary to classify the Cartesian product of criteria values.

More details on the operation of the SAC method can be found in [12,28].

As is already known, not all requirements of a software project are implemented. For this reason, it is feasible to apply a classification before the prioritization process so that a smaller number of requirements can pass to the next stage. This already guarantees a filter that removes from the scene those requirements that would hardly be implemented. Therefore, justify the use of a classifier in this stage of the project.

In VDA there are excellent classifiers. In a previous [28], we used the ORCLASS classifier and obtained satisfactory results. In this work, we are changing the classifier and adopting SAC to verify its efficiency in solving this type of problem.

4.2 ORCON Methodology

In general, the application of VDA methods is complicated to be made in practice. They require DM expertise if there is no help or support tool. The application in areas such as the ordering of preferences consists of creating a) a matrix of comparisons, b) than assembling a Joint Ordinal Scale (JOS) with these alternatives, and finally c) performing the ordering of alternatives [25].

However, this matrix can contain problems of inconsistencies due to the complexity of the problem, the number of alternatives and the human limitation since a set of alternatives can generate many probabilities [31].

In the work of [25], a tool called ORCON is proposed that gives the user the possibility to make an ordinal matrix consistency of alternatives.

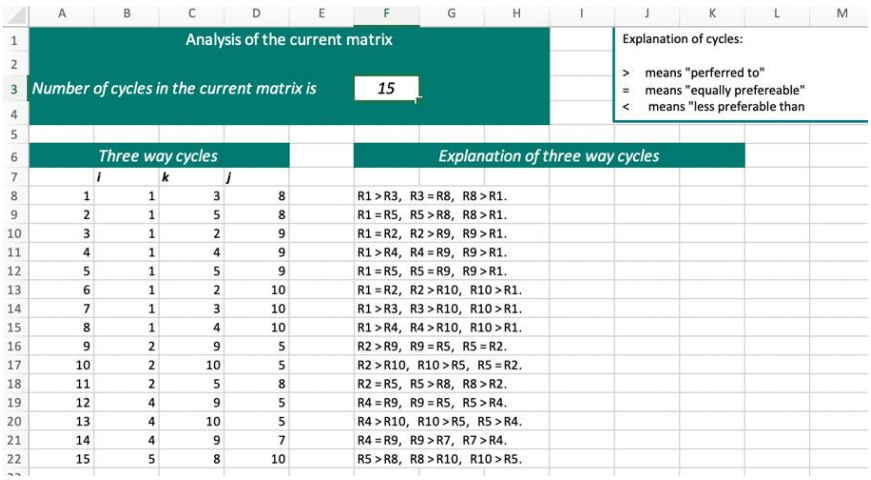
According to [25] inconsistency in the judgment of DM is structured such as transitivity. For example, they are considering that: $Req_1 > Req_2$ and $\{Req_2 > Req_3$ then $Req_1 > Req_3$. The violation of this sentence may be considered a contradiction to the preferences of the DM. For this, some authors consider calling by the term "three-way cycle" [31].

In earlier work in this line of research [9-11], we used VDA methods that used an interactive approach to correct inconsistencies. The use of the ORCON methodology is different because it works with the concepts of various methods embedded in a single user-friendly tool. This tool is free and available to both research and teaching as an aid in teaching VDA in the academic world.

In previous work [9-11] we have obtained satisfactory results using VDA as a means to select and order requirements in a software development project. In order to corroborate our results of previous studies, we look for other methods to verify the efficiency proposed by this research. In this way, we are looking for methods to adapt to this project. The chosen method, taking into account the adaptation to this research, ease of use by DM, scientific recognition and the existence of a tool of manipulation, made us choose ORCON.

In Figure 5, we can check one of the windows that make up ORCON. Further details on the operation of ORCON and obtaining this tool for use can be found in [25].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1



Analysis of the current matrix						
Number of cycles in the current matrix is				15		
Three way cycles					Explanation of three way cycles	
	i	k	j			
1	1	3	8		R1 > R3, R3 = R8, R8 > R1.	
2	1	5	8		R1 = R5, R5 > R8, R8 > R1.	
3	1	2	9		R1 = R2, R2 > R9, R9 > R1.	
4	1	4	9		R1 > R4, R4 = R9, R9 > R1.	
5	1	5	9		R1 = R5, R5 = R9, R9 > R1.	
6	1	2	10		R1 = R2, R2 > R10, R10 > R1.	
7	1	3	10		R1 > R3, R3 > R10, R10 > R1.	
8	1	4	10		R1 > R4, R4 > R10, R10 > R1.	
9	2	9	5		R2 > R9, R9 = R5, R5 = R2.	
10	2	10	5		R2 > R10, R10 > R5, R5 = R2.	
11	2	5	8		R2 = R5, R5 > R8, R8 > R2.	
12	4	9	5		R4 = R9, R9 = R5, R5 > R4.	
13	4	10	5		R4 > R10, R10 > R5, R5 > R4.	
14	4	9	7		R4 = R9, R9 > R7, R7 > R4.	
15	5	8	10		R5 > R8, R8 > R10, R10 > R5.	

Explanation of cycles:

> means "perferred to"
= means "equally prefereable"
< means "less preferable than

Figure 5. ORCON Methodology

5. Materials and Methods

In previous researches [9-12], we had good results when we compared the results generated by the Zapros III-*i* method concerning the SBSE methods. In these works, it was reported that the VDA method is no better than the SBSE methods, but the gain of this research is the opening of this new front in the VDA field, leaving this field as an alternative nonexistent to select and prioritize software requirements.

However, just prioritizing software requirements, in the case of that 20-unit work, can be a tiresome task because of the number of possibilities within ORCON. In general, we know that in a software project, not all requirements are implemented by the fact that the available resource value is almost always less than the sum of the implementation costs of all the requirements. Thus, it is interesting if we can select before prioritizing deliveries, those requirements that are most important to key customers, or another item at the discretion of the DM. So, if we classify these requirements into two classes in advance, "implementable" and "non-implementable," we will gain time in the prioritization process because fewer requirements will come for this step. This classification is oriented in criteria established by the DM and will work in this process as an agent that filters and passes to the next step only the requirements with higher chances of being implemented.

In Figure 6, we present the workflow used in this paper. We first use an instance generator for terms of the simulations that will be worked out since this work uses empirical data that simulate the situations faced by the DM's in the process of choosing requirements taking into account their characteristics, objectives, and constraints. In session 6, we show the details of this stage of the work. After completing this first step and having the data already generated, we submit them to the classification and ordering process using the SBSE, NSGA-II and MOCell methods. Both metaheuristics receive the same data, but as expected, generating a set of distinct but approximate results. These data are used to compare the results obtained by the VDA, SAC and ORCON methods.

In session 7, we describe how the methodologies and configurations adopted by the NSGA-II and MOCell metaheuristics work. In the following session, the method of adapting the quantitative problems solved by SBSE to be solved by qualitative methodologies is described. After this adaptation, the data are submitted to the VDA methods. Initially, we used the SAC method to classify the requirements and later, we used the ORCON method to prioritize them. This is described in section 8.

After we had all the results, we presented this to the DM who had the opportunity to evaluate the results generated. We present a comparison of results as well as assessments by DM's in session 9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

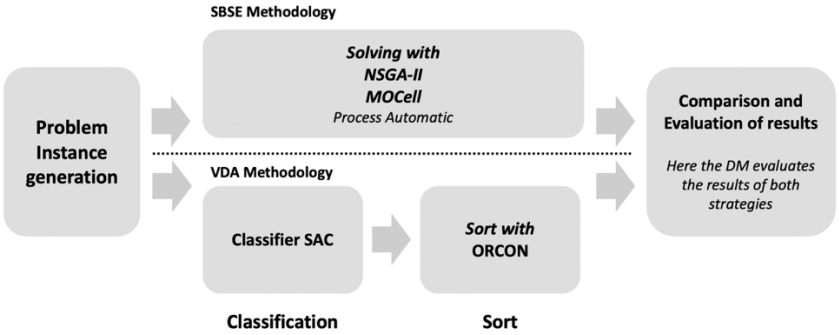


Figure 6. Workflow

6. Problem Generation

As seen, the data used in this research are empirical, yet they portray the difficulties faced by companies that develop software. So, to represent it well, we adopt a mathematical model that generates very close empirical data of the context of these companies. These data were created by applying the proposed mathematical formulation, by which we call the instance generator. This generator produces random data, within a scale, representing four software projects. In order to compare the results of this research with previous ones [9-11], we try to maintain the similar mathematical formulation as follows:

$$Maxf_{VALUE}(y) = \sum_{i=1}^{N} S_i \cdot y_i, \quad (1)$$

$$Minf_{VOLATILITY}(x^{Pos}) = \sum_{i=1}^{N} (B_i \cdot x^{Pos}{}_i) \cdot y_i, \quad (2)$$

Subject to:

$$x^{Pos}{}_i < x^{Pos}{}_j \quad (3)$$

$$\sum_{i=1}^{N} cost_i \cdot y_i \leq R \quad (4)$$

The equations proposed as the objectives for the problem are presented in the following functions:

Function 1 – In this equation, the objective is to maximize customer satisfaction by increasing business value (S_i), where $S_i = \sum_{m=1}^{W} w_m \cdot Value\ (m, i)$ represents the value for the business of a particular requirement r_i. As the client w_m attaches value to the requirement r_i with its preferences $(Value\ (m, i))$, this function is incremented as the choices are made. That way, as your most important customers point to your most important requirements, this function gains value.

Function 2 – Presents the degree of stability (B) for a group of project requirements (i), implementing those more stable requirements. This function works by measuring the product between the stability of the requirement and the location where it was allocated $x^{Pos}{}_i$. This function has a low value if it is prioritized on more stable requirements. In this way, the function tends to execute the most stable requirements first.

The restrictions of this project were formulated as follows:

Function 3 - It presents the formulation as to the precedences between the requirements and their technical constraints, where r_i precedes by r_j. In this case, r_i must be developed before r_j $(x^{Pos}{}_i < x^{Pos}{}_j)$.

Function 4 - Shows the constraint on costs for implementing a set of requirements against the available budget A. In this case, y_i indicates where a requirement r_i can be developed. Generally, the sum of costs to implement all requirements is greater than the sum of the resources available.

 doi:10.20944/preprints202004.0306.v1

Therefore, some requirements cannot be implemented. Thus, in order to take advantage of available resources, it is convenient to implement those essential requirements first for both the development company and the client.

6.1 Configuring Generated Instances

Each software project has a considerable number of requirements to implement. In the field of SBSE, we can find satisfactory solutions for a search in large sets of requirements (2000 requirements) [32]. In VDA, the number of criteria is restricted and small because, among other things, the fact that the human response to extensive questionnaires is taken into consideration [9-10]. Thus, in order to maintain equality in problem-solving, we adopted the number of 20 requirements for both SBSE and VDA in each of the four scenarios studied in this research. We have adopted in this work that the characteristics of requirements (SBSE) are called criteria in VDA. These criteria will be known later in Table 3.

To force the algorithms to seek reasonable solutions, we set the value of resources available for the project to be less than the sum of costs required to execute all the requirements. Therefore, we maintain a variable value of available resources between 70% and 80% for each of the four project scenarios.

Clients have values of importance (weight, significance) different for the company. Customers have personal preferences and assign scores for each of the 20 requirements. In this case, we seek to prioritize the most valuable requirements for the most significant customers for the company, thereby increasing the satisfaction of this group of customers. A more significant number of customers, therefore more restrictions and objectives, significantly increase the difficulty of finding a viable solution to the problem. In a previous work [12], we considered five clients per project; in this work, we will consider seven clients per project. Therefore, we have an increase in complexity.

Another factor that we consider here is technical precedence, as this is also a real trait of software development problems. The technical precedence ratio was adjusted between 10% and 20% of the total number of requirements.

These project factors are presented in figure 1.

Table 1. Settings adopted in this work.

File	Quantity requirements	Number of clients involved in the project	Available resources	Technical precedence level
File 1	20	7	70%	10%
File 2	20	7	80%	10%
File 3	20	7	70%	20%
File 4	20	7	80%	20%

Our research followed the flow shown in figure 6 using the four files adjusted according to the table above and the mathematical formulation presented at the beginning of this session. Our goal is to find satisfactory solutions to each of the problems using VDA methods through a classifier (SAC) and a sorting method (ORCON). At the same time, SBSE methods will receive copies of the same files available for VDA methods. As stated earlier in this paper, we expect the results of the VDA methods to approach those of the SBSE methods, but not to exceed them. Even so, this is already an essential indicator that this research is heading in the right direction since we are solving quantitative problems with qualitative methods. We hope that further research can further improve these results using different methodologies.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

7. Applying the SBSE methodology

As in previous works, to base the results generated by VDA, we use the NSGA-II and MOCell metaheuristics as a way to have a baseline, since we expect the results in VDA not to exceed these metaheuristics, but give us a level of approximation between these results.

Table 2 shows how the NSGA-II and MOCell metaheuristics were adjusted in this work.

Table 2. Settings adopted in this work.

Metaheuristic	Adjustment
NSGA-II	• Population: 500; • Maximum ratings: 2,000,000; • Crossing: 0.9; • Mutation: 1.0; • Use the binary method turner.
MOCell	• Population: 512; • Size file: 512; • Evaluations: 102,400; • Return: 20; • Crossing: 0.9 (rate); • Mutation: 1.0 (rate); • Use the binary method turner.

We use the jMetal framework [33] for the application of the NSGA-II and MOCell metaheuristics. This tool contains other embedded metaheuristics and can be checked in [33]. In this way, the data extracted from the instance generator was executed in this framework following the proposed configurations and the mathematical formulation. Each round of solutions was executed ten times in each of the four files (table 1), and an average solution from that set was extracted.

Thus, we obtained a set of Cartesian points. At each generated point, we have an ordered sequence of requirements. These points vary from one extreme (since we are dealing with a multiobjective problem) that portrays the level of customer satisfaction by the essential requirements to another that represents the degree of stability (volatility) of the requirements. In session nine, we will present the results for the four files graphically.

8. Applying the VDA methodology

We did not find in the literature other lines of research in VDA that deal with the classification and prioritization of software requirements. Therefore, this task in this field is relatively new. The methodology used in this work has already been tested in other scenarios in previous studies [9-12] and has shown to be efficient to solve qualitatively (VDA) quantitative problems (SBSE) with satisfactory results. For this, it was necessary to create an adaptation methodology for this purpose. In the next sub-sessions, we will see the configurations of this methodology.

8.1 Adaptation to the VDA methodology

In this work, we consider empirical data and generated with a technical basis. As it is a quantitative problem, the data produced by an instance generator built for this purpose (subsection 6.1) are in numerical formats to represent a characteristic of the requirement (for example, stability, cost, the importance of stakeholders, among others.) Where scales are used with number intervals. Thus, for the particular importance of the stakeholder, we have a variation from 0 to 10, where 0 represents that stakeholder of lesser importance to the company to that of greater importance (10). To help the decision-maker in the data entry process, we have created the table below. In this table,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

we can have the transformation of quantitative data into qualitative data in a fair way, so that the SAC and ORCON methods can interpret them. This guarantees an isonomic comparison.

In this way, we consider naming the characteristics of the requirements as "criterion". Therefore, requirement *r* has its stability as a criterion. Each criterion of this requirement has its values, and here we call this "criteria values". Table 3 shows the table that was considered to make this adaptation.

Table 3. VDA method adaptation table.

Criterion	Values
1 Stability	1.1 There is a low probability of change
	1.2 Relative probability of change
	1.3 High probability of change
2 Cost	2.1 Low-cost Impact
	2.2 Relative cost
	2.3 High-cost impact
3 Stakeholder	3.1 Notorious stakeholder for the company
	3.2 Stakeholder indifferent to the company
	3.3 Stakeholders have low significance to the company
4 Value added to the requirement by the customer	4.1 Stakeholder adds high value for a requirement
	4.2 Stakeholder adds low value for a requirement

Therefore, if a requirement r has a value of 3 for the stability, according to the table above, it becomes a criterion value for stability "High probability of change". After this step, we proceed to the implementation of the SAC method.

8.2 Application of the SAC method

As stated in the previous section, our methodology classifies requirements into two groups: "implementable" and "non-implementable". As in previous work [12], we invited 12 experts with project management experience to respond to questionnaires regarding the classification of requirements since the SAC method does not have a classification platform. These managers had at least 12 months of experience in the area. The method of our research was clarified to the DM's, and they accepted to participate, giving their valuable contribution to this work.

Constraining the characteristics of the requirements, the project's constraints and objectives, and the professional experience, the DM's answered questions about the elicitation of preferences. After collecting the questionnaires, we were able to compile and extract two sets of requirements. In one-off cases, where data inconsistency occurred, we called the DM to answer one or two more questions to resolve this. We talked about inconsistencies in VDA in sub-session 4.2 when dealing with ORCON.

Furthermore, to follow, only the requirements considered as implementable will be considered for the next step. We will apply the ORCON methodology to sort these requirements into software releases.

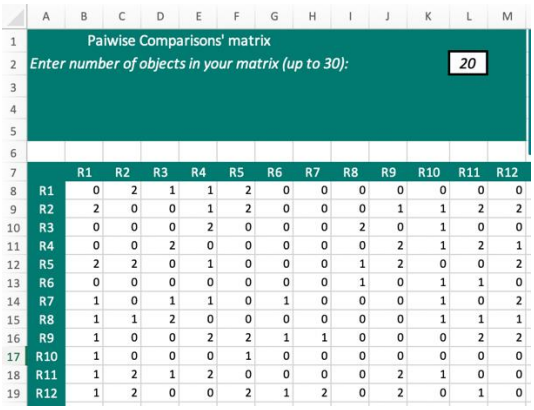
8.3 Application of the ORCON methodology

Having here a set of requirements considered implementable, we will considerably reduce the number of options to DM in this step. We apply the ORCON methodology through its tool.

The same experts who participated in the previous stage performed this stage. They could rely on the participation of one of the authors of this project to insert and process the data in the ORCON tool, as explained above. They were able to enter their preferences, and the tool itself dealt with the inconsistencies, which are natural in VDA. The DM's were once again aware of the objectives and constraints of each of the four scenarios/projects.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

In figure 7, we have a print that shows the use of ORCON.



Paiwise Comparisons' matrix

Enter number of objects in your matrix (up to 30): 20

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
R1	0	2	1	1	2	0	0	0	0	0	0	0
R2	2	0	0	1	2	0	0	0	1	1	2	2
R3	0	0	0	2	0	0	0	2	0	1	0	0
R4	0	0	2	0	0	0	0	0	2	1	2	1
R5	2	2	0	1	0	0	0	1	2	0	0	2
R6	0	0	0	0	0	0	0	1	0	1	1	0
R7	1	0	1	1	0	1	0	0	0	1	0	2
R8	1	1	2	0	0	0	0	0	0	1	1	1
R9	1	0	0	2	2	1	1	0	0	0	2	2
R10	1	0	0	0	1	0	0	0	0	0	0	0
R11	1	2	1	2	0	0	0	0	2	1	0	0
R12	1	2	0	0	2	1	2	0	2	0	1	0

Figure 7. ORCON tool

The subjectivity arising from the DM experience can lead to a positive difference during the analysis and choice of order of implementation for a particular requirement. This is an essential factor that incorporates the characteristics of VDA methods. We can also consider that this methodology gives us the possibility to solve problems of inconsistencies that are natural to happen to the human.

The ORCON tool is very objective, which made it possible to solve a problem in 15 minutes with the support of one of the authors of this research, as stated. This does not let the process of choice be tiring.

Finally, the DM's had the opportunity to get to know and comment on their satisfaction with the solution generated in this research. In this way, the results generated by this work as well as a consolidated DM's assessment will be checked in the next session.

9. Results and Discussion

SBSE, by its very nature, presents results groups containing thousands of solutions to the same problem. Therefore, in the same result, we can have different sequences of ordering requirements. In VDA, it is generated in a single result that was extracted from the application processes of SAC and ORCON.

In Table 4, we show the sequence of requirements that were generated for file 4. VDA methods generated this only result. We realize that requirement 12, for example, will be the first to be implemented, and requirement 20 will be the last one. We can see that in table 4 there are 15 requirements and not the 20 initially inserted. This is because the constraints of the problem have cut the 05 requirements that are not here. This is because these requirements were not classified in the selection process or, if they were classified, they were in a lower position and, therefore, the cut made by the lack of resources disqualified them (table 1).

The requirements have already been allocated in each release according to the resources available for each one. In this case, the group requirements of release one will be implemented first because they are the less volatile (more stable), lower cost, and more essential requirements for the company's most important customers, for example.

The technical precedencies between the requirements were respected in this project, is one of the constraints of the problem.

Table 4. File 4 - Results

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

Priority for implementation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Requirement	12	03	04	06	07	11	19	14	16	13	17	10	05	15	20
Release number	1	1	1	1	1	1	1	1	2	2	2	3	4	5	5

In order to have equal and fair results between the methods (quantitative and qualitative), we have been cautious in the application of the methodology, since it is a very complicated process since it involves the search for the solution of the same problem in two very different forms. However, this guarantee results comparable to each other in addition to the gains it provides.

For better presentation, the results obtained from VDA and SBSE were organized in graphs. This gives us a clear idea of comparing results. In all the graphs we can see a real Pareto front [34] created between the NSGA-II and MOCell methods and a single black dot that represents the results generated by the VDA methods.

In Figures 8, 9, 10 and 11, these results are presented. Each graphic represents a file, being respectively the files 1, 2, 3 and 4.

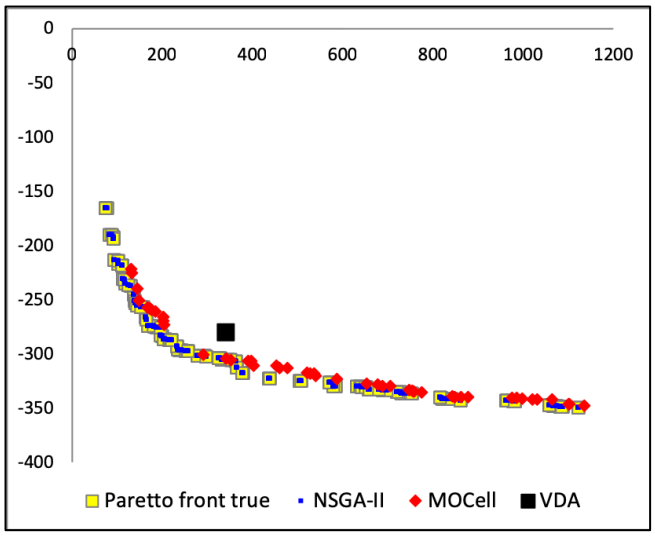


Figure 8. Results for File 1

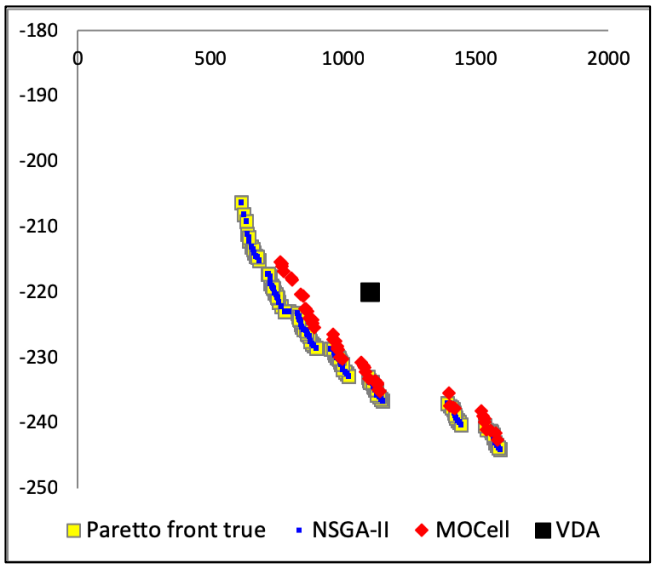


Figure 9. Results for File 2

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

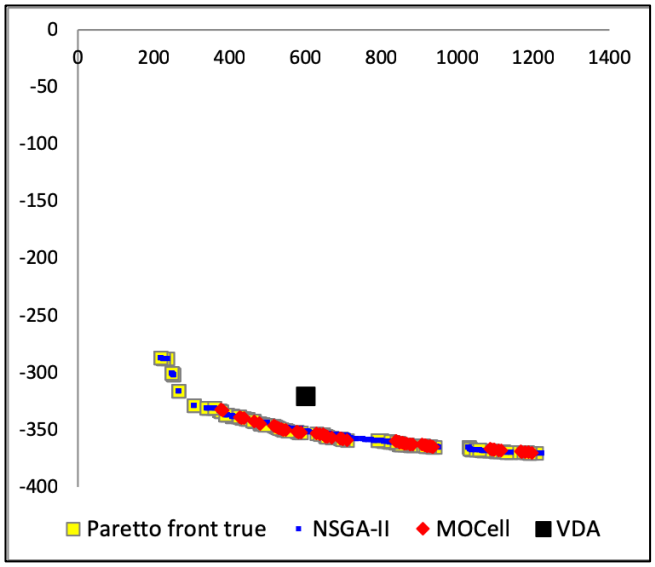


Figure 10. Results for File 3

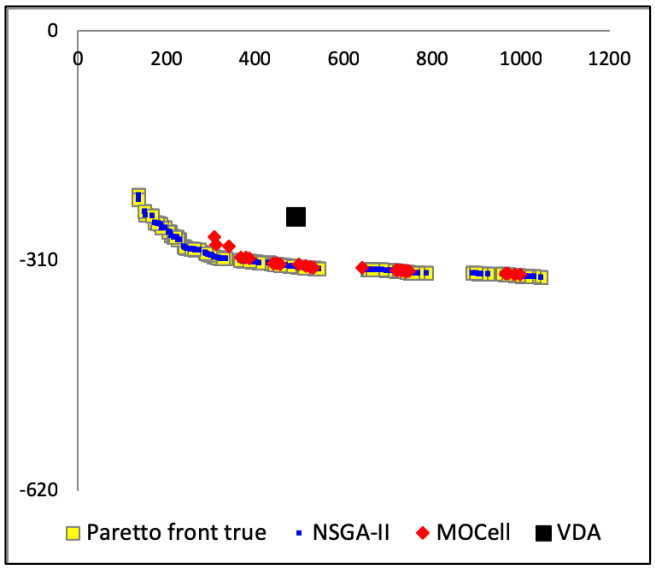


Figure 11. Results for File 4

We can see the formations of the data sets created by the metaheuristics and the point created by the VDA methods. This unique VDA solution represents the DM's desire since it was, he who made his choices, taking into account his technical expertise as well. The other points represent the solutions of NSGA-II and MOCell, according to the legend, besides the actual Pareto front that we form. Figures 8 through 11 represent the results for each of the files. The settings of these files are set out in table 1, shown above.

We can see that the results of the VDA methods were always close to those of the SBSE methods (NSGA-II and MOCell). Therefore, we understand that these are good results because SBSE already has in its portfolio dozens of methods that can generate better results than these, but VDA does not. Both areas gain from this research because VDA starts to have an open path to research that involves selection and prioritization of requirements and SBSE opens another field that is the Verbal Analysis of Decision since we note that it is possible to even in a complicated way, interdisciplinarity between these areas. In general, we consider the results of this research to be good.

Four files were generated so that VDA methods could be challenged. We make higher and lower budget configurations, higher and lower technical precedence. Even so, the results from all the files were similarly close to the actual Pareto front. We might consider it difficult for our research if these

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

results were divergent or were far removed from the Pareto front or even if they far outweighed the results of metaheuristics. In the trials we had, we did not realize this behavior anomaly it.
In order to be able to see the gains of this research mathematically, we try to calculate the distance between the actual Pareto front and the black point generated by the VDA methods. For this, we use a metric known as Generational Distance (GD) [35].

9.1 Generational Distance (GD) metric

The strategy of measuring the distances between a Pareto front PF_{known} and the Pareto Real front PF_{true} [34] is a solution when we need to know this distance mathematically. One of the methods that do this task is called Generational Distance (GD) [35]. Since the VDA solution is a single point, we calculate the distance between this point PF_{known} and the front of Real Pareto PF_{true}.

$$d_i = \min_j(||f_{true}^j - f_{known}^i||)_2$$

To find this distance, d_i is multiplied by e_i^{GER}. When applied $(d_i - \underline{f_j})/(\overline{f_j} - \underline{f_j})$, we can normalize this. The metric is defined as follows:

$$NDWD_{GER} = \sum_{i=1}^{N} e_i^{GER} \left(\frac{d_i - \underline{f_j}}{\overline{f_j} - \underline{f_j}}\right)$$

As defined in [34], closer proximity between the two fronts being evaluated is indicated by a low-value result. Therefore, when a point is contained in the front of Pareto real, we have a value of 0 as a result. We can see the results of the GD metric for the four graphs shown previously, as described in table 5.

Table 5. Results analyzed by GD

File	Result for GD	Result for GD in [12]	The efficiency of the work approach
File 1	0.1552	0.1543	▼ 0.58%
File 2	0.3601	0.3585	▼ 0.44%
File 3	0.3202	0.3287	▲ 2.65%
File 4	0.4241	0.4283	▲ 0.99%

In Table 5, we can see that file 1 had the lowest value for GD; that is, we obtained more satisfactory results. File 4, on the other hand, had the worst results for the observed solutions. The difference between the four files is the degree of difficulty in reaching a good solution. Table 1 shows this level of difficulty in detail.

We can notice something precious for this work in relation to the previous work [12], when we check the results for files 1 and 2 (which have fewer prerequisites, therefore being a simple problem), in relation to the files 3 and 4 (which have more prerequisites, therefore more complex) from both surveys. We noticed that this work has a significant performance gain for bigger and more complex problems (files 3 and 4).

Considering that metaheuristics already have a long history and tradition regarding the solution of this type of problem, we understand that our solutions were reasonable because they were in a region close to the Pareto front area, even though this is not yet a key area for VDA.

After presenting the results, that is, the sequence of requirements requested from the DMs after the consolidation of their responses, they were able to assess their level of satisfaction with these

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2020 doi:10.20944/preprints202004.0306.v1

results. They answered a questionnaire with a score between 0 and 100, where 0 represents low satisfaction and 100 high satisfaction. In figure 12, we present these consolidated results. We realized that the DMs were satisfied with the results achieved by the qualitative methods (VDA).

As an average for all the solutions of all DM's, the value of 83.5 points was reached. Satisfactory notes to this approach demonstrate that the methodology employed here has excellent potential to address the selection and prioritization of software requirements.

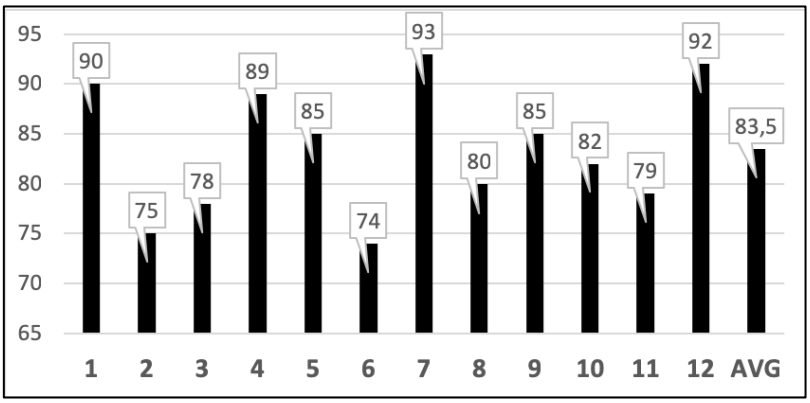


Figure 12. DM's score for the solutions generated by the VDA methodology

10. Conclusions

In this work, we can demonstrate the ability of Verbal Decision Analysis and its methods to solve problems related to the selection and prioritization of software requirements. We note that this task is almost always difficult for the decision-maker and that wrong choice in this processing context can impact the whole development project.

Thus, to demonstrate the accuracy of this research, we seek to bring our empirical development scenarios to those found in real development projects, as well as the technical capabilities of the VDA methods, which are still limited in this field. It is essential to put strict restrictions on forcing methods to seek efficiency. The importance of the requirement for a customer and its importance to the company is another bias we consider because in some cases, a customer wants to implement a requirement that he can afford in that period.

In previous research [9-11] good evaluations were achieved using only the ordering of requirements, but we understand that a classification preceding this helped in the preference elicitation process because the DM was able to do it more lightly and less costly. We did this in [12], and we saw that if we changed the methods, we could find even better results, like the one we observed here. Therefore, the use of the SAC method as a classifier was relevant here as well, because those requirements that were far from the objectives and constraints have already been disregarded.

We observed that the method adopted in this work to select or prioritize software requirements using SAC and ORCON obtained better results for more complex problems in relation to the ORCLASS and Zapros III-*i* methods. Thus, it is worth mentioning that the idea of creating a structure capable of identifying the complexity of a problem and automatically adopting the most appropriate methods to solve it is entirely feasible. This work shows us that this is possible.

Another decisive factor in this work is that we were able to listen to the DM's opinion during the process precisely because he was the protagonist in the choices of the best alternatives for him considering, in addition to everything, his professional experience, which gives value to his results.

Search-Based Software Engineer has a recognized role in the area of optimization, using metaheuristics of the most varied to solve several types of problems such as verification, selection, allocation, prioritization and order of requirements, among others. We note that optimization reduces time in the search process for optimal solutions [36,37]. This work demonstrates that there may be other alternatives, albeit modest ones, to solve these types of problems. In this case, both areas gain from this research because there are innumerable possibilities for projects between them.

The feasibility of this work is positive when we use Verbal Decision Analysis as a tool to support the decision making of a software development project in the task of classifying software requirements. The SBSE field has been carrying out this task for some time, but we have the opportunity to see that another area, in this case, the VDA, can do the same task satisfactorily. This research allowed us, more and more, to evolve more in the search for the validation of this new methodology and also in the feasibility of building a framework that decides and applies the best VDA methods for specific problems.

In future work, one can increase the degree of difficulty (Table 1) for VDA methods. One can try to find results with a more significant number of requirements. A framework developed using the methodology proposed here would be of great importance as it would leave the process even faster.

Author Contributions: P. A. M. B. conceptualization, formal analysis, writing—original draft preparation, resources; P. R. P. conceptualization, methodology, validation, supervision, writing—review and editing; F. R. V. S. conceptualization, data curation, project administration and M. S. F. investigation, software. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Postgraduate and Research sector of the Federal Institute of Ceará – Brazil (PRPI-IFCE).

Acknowledgements: Paulo Alberto Melo is grateful for all the support received by the IFCE (Federal Institute of Ceará). Plácido Rogério Pinheiro is grateful for the support received for the development of this project by the National Council for Scientific and Technological Development (CNPq).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Colanzi, T. E., Vergilio, S. R., Assunção, W. K. G., Pozo, A. Search Based Software Engineering: Review and analysis of the field in Brazil. *In Journal of Systems and Software* **2013**, *Vol. 86*, Issue 4, pp. 970-984, ISSN 0164-1212. DOI 10.1016/j.jss.2012.07.041.
2. Ruhe, G., Saliu, M. O. The Art and Science of Software Release Planning. *IEEE Software*, **2005**, *vol. 22*, pp. 47- 53. DOI 10.1109/MS.2005.164.
3. Sommerville, I. *Software Engineering*, 7 ed.; Pearson Addison-Wesley: São Paulo, 2004.
4. Sagrado, J. D., Aguila, I. M. D. Ant Colony Optimization for Requirement selection in Incremental Software development. In Proceedings of the 1st International Symposium on Search-Based Software Engineering (SSBSE '09), IEEE Computer Society, Windsor, England, 2009; pp. 67-76.
5. Bagnall, A. J., Rayward-Smith, V. J., Whittley, I. M. The next release problem. *Information and Software Technology*, **2001**, *vol. 43*, pp. 883-890.
6. Paixão, M. H. E., Souza, J. T. A robust optimization approach to the next release problem in the presence of uncertainties. *The Journal of Systems and Software*, **2015**, *vol. 103*, pp. 281-295. DOI http://dx.doi.org/10.1016/j.jss.2014.09.039.
7. Ferreira, T. N., Araújo, A. A. P., Basilio Neto, A. D., Souza, J. T. Incorporating User Preferences in Ant Colony Optimization for the Next Release Problem. *Applied Soft Computing (Print)* **2016**, *vol. 49*, Pages 1283-1296. DOI http://dx.doi.org/10.1016/j.asoc.2016.06.027.
8. Brasil, M. M. A., Silva, T. G. N., Freitas, F. G., Souza, J. T., Cortés, M. I., A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements. *Lecture Notes in Business Information Processing* **2012**, *v. 102*, pp. 300-314. DOI http://dx.doi.org/10.1007/978-3-642-29958-2_20.
9. Barbosa, P. A. M., Pinheiro, P. R., Silveira, F. R. V., Filho, M. S. Applying Verbal Analysis of Decision to prioritize software requirement considering the stability of the requirement. In 6th Computer Science Online Conference, Advances in Intelligent Systems and Computing, Czech Republic, April 2017; ISBN 978-3-319-57141-6, pp. 26–29. DOI: doi.org/10.1007/978-3-319-57141-6_45.
10. Barbosa, P. A. M., Pinheiro, P. R., Silveira, F. R. V., Filho, M. S. Selection and prioritization of software requirements using the Verbal Decision Analysis paradigm. In The 29th International Conference on Software Engineering and Knowledge Engineering - SEKE, Pittsburgh USA, 2017. DOI 10.18293/SEKE2017-150.

11. Barbosa, P. A. M., Pinheiro, P. R., Silveira, F. R. V. Towards the Verbal Decision Analysis Paradigm for Implementable Prioritization of Software Requirements. *Algorithms (MDPI)* **2018,** *vol 11,* pp.176. DOI: 10.3390/a11110176.
12. Barbosa, P. A. M., Pinheiro, P. R., Silveira, F. R. V., Filho, M. S. Selection and Prioritization of Software Requirements Applying Verbal Decision Analysis. *Complexity Journal – Hindawi*, **2019**, *vol. 2019*. DOI https://doi.org/10.1155/2019/2306213.
13. Karlsson, J., Ryan, K. Supporting the selection of Software Requirements. International Workshop On Software Specification And Design (IWSSD '96), 1996; 8th. Proceedings, pp. 146-149.
14. Cordeiro, A. G., Freitas, A. L. P. Priorização de requisitos e avaliação da qualidade de software segundo a percepção dos usuários. *CI. Inf.* **2011**, *v. 40 n. 2*, pp.160-179.
15. Nurmuliani, N., Zowghi, D., Fowell, S., Analysis of requirements volatility during the software development life cycle. In the Australian Software Engineering Conference (ASWEC'04), Australia, 2004.
16. Curis, B., Krasner, H., Iscoe, N., A Field Study of the Software Design Process for Large Systems. *Communications of the ACM* **1988**, *vol. 31*, pp. 1268-1287.
17. Greer, D., Ruhe, G., 2004. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, **2004**, pp. 243-253.
18. Colanzi, T. E., Vergilio, S., Pozo, A. T. R., Assuncão, W. K. G. Search Based Software Engineering: A Review from the Brazilian Symposium on Software Engineering. In XXV Simpósio Brasileiro de Engenharia de Software (SBES-CBSoft 2011), Brasil, 2011; pp. 49-54.
19. Harman, M., Jones, B. F., Search-based software engineering. *Information and Software Technology*, **2001**, pp. 833-839.
20. Nebro, A. J., Durillo, J. J., Machín, M., Coello, C. A. C., Dorronsoro, B. A Study of the Combination of Variation Operators in the NSGA-II Algorithm. In Conference of the Spanish Association for Artificial Intelligence, Spain, September 2013; pp. 269–278.
21. Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., Alba, E. MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. *International Journal Intell. Systems*, **2009**, *vol. 24*, pp. 726–746. DOI 10.1002/int.20358.
22. Tamanini, I., Pinheiro, P. R. Reducing Incomparability in Multicriteria Decision Analysis: An Extension of The ZAPROS Methods. *Pesquisa Operacional (Print)* **2011**, *vol. 31 n. 2*, pp. 251-270. DOI: 10.1590/S0101-74382011000200004.
23. Tamanini, I., Pinheiro, P. R. Challenging the Incomparability Problem: An Approach Methodology Based on ZAPROS. *Communications in Computer and Information Science (Print)* **2008**, *vol. 14*, pp. 338-347. DOI 10.1007/978-3-540-87477-5_37.
24. Tamanini, I., Pinheiro, P. R., Machado, T. C. S. Hybrid Approaches of Verbal Decision Analysis in the Selection of Project Management Approaches. *Procedia Computer Science*, **2015**, *vol. 55*, pp. 1183-1192. DOI http://dx.doi.org/10.1016/j.procs.2015.07.093.
25. Moshkovich, H., Mechitov, A., Ordinal Inconsistencies in pairwise comparisons: problems and solutions. In 23rd International Conference on Multiple Criteria Decision Making, Hamburg, Germany, 2015; pp. 63. DOI 10.1504/IJMDM.2017.082511.
26. Larichev, O., Moshkovich, H. M. Verbal decision analysis for unstructured problems. *Boston: Kluwer Academic Publishers*, **1997**.
27. Tamanini, I. Hybrid Approaches to Verbal Decision Analysis Methods. Doctor Thesis, Graduate Program in Applied Informatics, University of Fortaleza, 2014.
28. Larichev, O., Kortnev, A., Kochin, D. Decision Support System for Classification of a Finite Set of Multicriteria Alternatives. *Decision Support Systems* **2002**, *v. 33 n. 1*, pp. 13–21, 2002. DOI: 10.1016/S0167-9236(01)00132-4.
29. Moshkovich, H. M., Mechitov, A. Verbal Decision Analysis: Foundations and Trends. *Advances in Decision Sciences* **2013**, *v. 2013*, 9 pages, DOI: 10.1155/2013/697072.
30. Larichev, O. I., Moshkovich, H. M. ZAPROS-LM - a method and system for ordering multiattribute alternatives. *European Journal of Operational Research*, **1995**, vol. 82, pp. 503-521.
31. Gass, S.I., 1998. Tournaments, transitivity and pairwise comparison matrices. *Journal of Op. Res. Society*, **1998**, *vol. 49*, pp. 616-624.

32. Barbosa, P. A. M. An Optimal-Based to the Prioritization of Software Requirements, Considering the Stability of the Requirement. Master Thesis, Academic Master in Computer Science - Ceará State University, Brazil, 2013.
33. Durilo, J. J., Nebro, A. J., 2006. JMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. *Tech-Report ITI*, **2006**, vol. *2006-10*.
34. Coello, C. A., Lamont, G. B., Veldhuizen, D. A. V. Evolutionary Algorithms for Solving Multi-Objective Problem. In Kluwer Academics Publishers, Boston, USA, 2007.
35. Veldhuizen, D.A.V., Lamont, G. B. Evolutionary Computation and Convergence to a Pareto Front. In Late Breaking Papers at the Genetic Programming 1998 Conference, Stanford, USA, 22-25 July 1998; Stanford University Bookstore, 1998; 221-228.
36. Filho, M. S., Pinheiro, P. R., Albuquerque, A. Task Assignment to Distributed Teams aided by a Hybrid Methodology of Verbal Decision Analysis. *IET Software,* **2017**, *vol. 11*, pp. 245–255. http://dx.doi.org/10.1049/iet-sen.2016.0306.
37. Filho, M. S., Pinheiro, P. R., Albuquerque, A., Rodrigues, J. J. P. C. Task Allocation in Distributed Software Development: A Systematic Literature Review. *Complexity*, **2018**, *vol. 2018*, 1-13.