Article

Towards the Verbal Decision Analysis Paradigm for Prioritization of Software Requirements Implementable

Paulo Alberto Melo Barbosa [1,*], Plácido Rogério Pinheiro [1] and Francisca Raquel de Vasconcelos Silveira [2]

[1] Graduate Program in Applied Informatics, University of Fortaleza, Fortaleza 60811-905, Brazil; albertobmap@hotmail.com (P.A.M.B); placidrp@gmail.com (P.R.P)
[2] IT Department, Federal Institute of Ceará, Tianguá 62.320-000, Brazil; raquel.vsilveira@hotmail.com
* Correspondence: albertobmap@hotmail.com; Tel.: +55-88-99653-9467

Abstract: The activity of prioritizing software requirements should be done as efficiently as possible. Selecting the most stable requirements for the most important customers for the development company can be a positive factor when we consider that the available resource does not always encompass the implementation of all requirements. Quantitative methods for reaching software prioritization in releases are many in the field of Search-Based Software Engineering **(SBSE)**. However, we show that it is possible to use qualitative Verbal Decision Analysis (VDA) methods to solve this same type of problem. Moreover, we will use the ZAPROS III-*i* methods to prioritize requirements considering the opinion of the decision-maker, who will participate in this process. Finally, the results obtained in the VDA structured methods were quite satisfactory when compared to the methods using SBSE. A comparison of results between quantitative and qualitative methods will be made and discussed later.

Keywords: verbal decision analysis; multi-objective optimization; software release planning; ZAPROS III-*i*

1. Introduction

Among the actions of a software manager, we find the decision-making process. It is the role of the decision maker (DM) to recognize, evaluate and select the best alternatives taking into account technical and human factors (experience, perceptions). The preferences for the alternatives and criteria available should lead the decision to an expected and satisfactory goal. Software Engineering has a branch of activity linked to the manufacture of tools and methods that facilitate the decision-making activity of the software manager. One of these areas that produce automatic, semi-automatic, interactive, and others. Methods are known by Search-Based Software Engineering (SBSE), the name given to a body of work in which Search-Based Optimisation is applied to Software Engineering. This approach to Software Engineering has proved to be very successful and generic. It has been a subfield of software engineering for ten years [1]. Moreover, SBSE seeks to reformulate Software Engineering problems as 'search problems' [1,2]. Moreover, this is not to be confused with textual or hypertextual searching. Instead, for Search-Based Software Engineering, a search problem is one in which optimal or near-optimal solutions are sought in a search space of candidate solutions, guided by a fitness function that distinguishes between better and worse solutions [3].

The techniques found in Search Engineered Software Engineering can solve many optimization problems related to the area of software engineering as well as help in support of the decision-maker. Algorithms called metaheuristics can be a solution to find satisfactory solutions in a set of data. Metaheuristics can easily incorporate new constraints and explore regions of a set in an attempt to overcome local optimality. Although they can not guarantee global optimality, they can identify numerous points of great locations.

On the other hand, problems of software engineering often involve conflicting constraints, ambiguous and imprecise information within a broad set of choices or decisions. Solving these problems is a complex task considering that there is no optimal solution [4]. If we take into account a set of requirements for composing multiple releases, the sequences of input requirements for this problem may be too numerous. Besides, the solutions' compositions must comply with constraints such as customer satisfaction, time, cost, among others. A critical aspect to the success of a software project based on iterative and incremental lifecycle is the planning of which requirements are going to be delivered in each release of the software [5].

The releases model, derived from incremental software development, allows customers to receive portions of the software in advance [6]. A problem faced by companies developing and maintaining large and complex software systems developed for large and diverse customers is to determine what requirements will be implemented in the next software release [7]. The more complex the software, the higher the time to arrive at a satisfactory result concerning the planning of Releases [8].

It is essential that the task of selecting and prioritizing requirements to take effect in the most efficient way possible. Requirements changes are often the primary factor in increasing time and cost in software development projects. Therefore, selecting and prioritizing requirements taking into account their degree of stability can increase the effectiveness of the entire software development process. Volatile requirements are considered as a factor that can cause significant difficulties during software development. Furthermore, this is because these requirements may change throughout the project implementation. In this way, changes in features are expected, which can cause problems for the software development company [9].

As seen, software requirements planning problems are already solved by the SBSE with extensive coverage in the literature. Deciding what requirements will be implemented first is a decision problem for the software manager who can use metaheuristics, and their quantitative structures, to solve it. However, the task of prioritizing also has a subjective bias related to the decision-makers experience. In this scenario, Verbal Decision Analysis (VDA) may appear as an alternate option to work around these issues. It is emphasized that Verbal Decision Analysis is a subjective method widely used to solve qualitative problems inherent to personal options.

The requirements planning problems are without their multiobjective majority, and therefore, the solution is almost always composed of a family of solutions located in front of Pareto and must be considered equivalent [10]. The evaluated multiobjective approach consists of treating the human's preferences as another object to be maximized, as well as maximizing the overall client satisfaction and minimizing the project risk [11]. Multiobjective optimization addresses optimization problems that have multiple objective functions to be simultaneously maximized or minimized. The methodology VDA is structured on the assurance that most decision-making problems can be qualitatively described. The Verbal Decision Analysis supports the decision making the process by the verbal representation of problems. Although the decision-maker ability to choose is very dependent on the occasion and the interest's stakeholders, the methods of decision-making support are universal [12]. There are in the literature several methods structured in VDA that help the decision-maker to choose from within multicriteria set alternatives that best meet their personal preferences [12].

Therefore, in the context of planning software releases, the objective of this work is to compare, taking into account requirements ordering in solved software releases, the results obtained by the metaheuristics (with SBSE quantitative methodology) with the results obtained by methods of Verbal Decision Analysis (with qualitative methodologies). Finally, we will make a study of this comparison and propose a new methodology to solve requirements prioritization problems in software releases using VDA methods.

This new methodology gives the decision-maker of a software development project an alternative to a software release planning problem, indicating the best order of implementation of software requirements taking into account technical aspects (implementation cost, technical precedence between requirements) and human aspects.

2. Verbal Decision Analysis

Verbal Decision Analysis (VDA) proposes a systematic analysis and support of decision-based on verbal factors, as opposed to the quantitative methods generally used, as it uses a method of qualitative analysis of the attributes. Therefore, no numerical conversions are performed. VDA comprises a set of several methods for classifying and ordering alternatives, which consider multiple criteria in solving problems [13]. Figure 1 shows the VDA methods for classification and ordering.

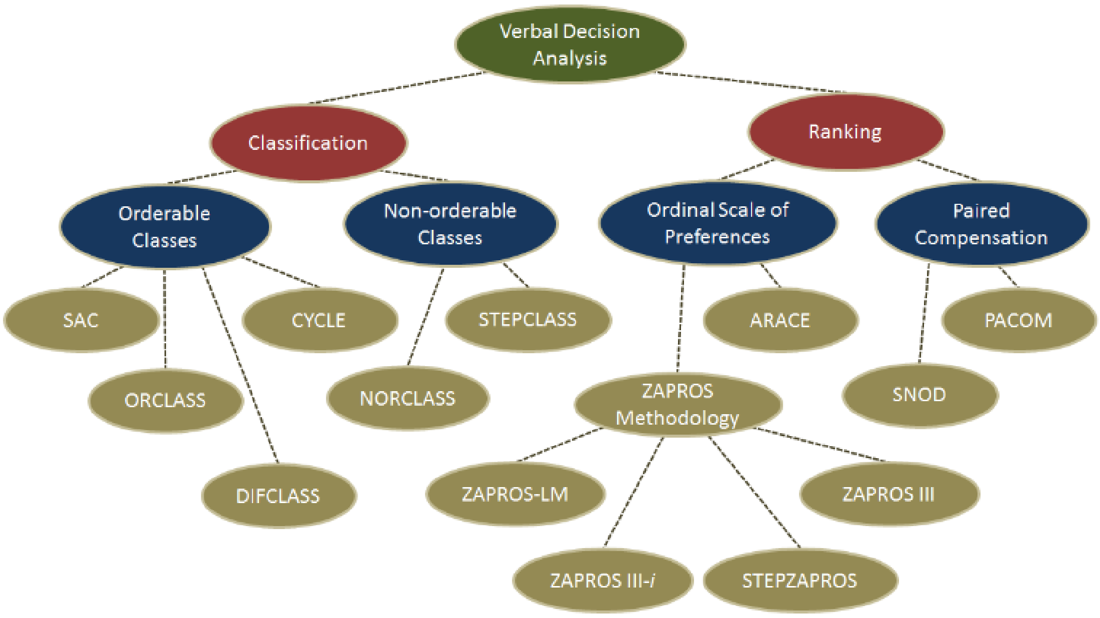

Figure 1. VDA Methods for Classification and Ordering [18].

Furthermore, Verbal Decision Analysis has excellent applicability in problems that present a considerable number of alternatives and a relatively small set of criteria and their values. The methods that make up the VDA framework have many features and benefits [14], among which we highlight the following:

(i) Its purpose is to describe the problems, VDA methods use language that is natural for the decision-maker;

(ii) methods use the verbal information to induce preferences, which allows them to implement psychologically valid measures from the decision-maker viewpoint;

(iii) methods include steps to process inconsistent entries in the decision-maker preferences, such as consistency checks and criteria independence;

(iv) methods use transparent procedures from the decision-maker viewpoint;

(v) they allow us to review the preferences that were given and how they generated the result, providing explanations about the results generated.

The application of these methods, in particular, the ZAPROS method, to a given problem presents a significant amount of solution possibilities. However, this is due to the numerous combinations of criteria values to generate situations to be analyzed, which, at the end of the process, refer to the decision rule. This high number of combinations can leave the stages of preference elicitation and comparison of alternatives so sophisticated that it would be impossible to perform them manually.

The estimation of the number of unparalleled alternatives (and consequently of the decision power of the method) can be done by calculating the general number of alternative pairs $Q = 0.5n^N(n^N - 1)$, where N is the number of criteria, n is the number of criteria values) and the subset that will be related by Pareto dominance (*D*). From the difference between *Q* and *D*, we have the set of alternatives that depends directly on the Scale of Preferences obtained by the answers of the decision-maker, this is the set that is more likely to contain opposite pairs of alternatives. After that, we will have the decision power index of a method by means of the calculation: $P = 1 - {}^{S}/_{B}$, where *B* is the difference between *Q* and *D*, *S* is the number of alternatives that cannot be compared based on the preference scale of the decision-maker, or which represent incomparable alternatives [15].

3. The ZAPROS III-*i* Method

The project manager has among his activities the role of making decisions. To come up with an alternative, he has a set of his choice. Each alternative has its own or similar criteria. The use of methodologies that support the decision-maker can minimize possible negative impacts caused by wrong decisions [16]. The VDA walks precisely in this direction to present to the alternative decision-maker in friendly language and as human as possible. There are many methods in VDA that work with this, among them the ZAPROS III method [16] that make the process of eliciting preferences less inconsistent with previous methods [16]. It is structured in the elicitation of preferences of values that represent the distances between the evaluations of two criteria, denominated Quality Variations (QV). Besides, it uses the Formal Quality Index (FIQ) to order the established alternatives to minimize the number of pairs of alternatives to be compared to obtain the result of the problem [17]. Some alternatives may be unmatched, and this leads to unsatisfactory results in decision-making models. Thus, the ZAPROS III-*i* method originated, very similar to ZAPROS III, but presents modifications mainly in the process of comparing alternatives to improve the decision method [17]. In this way, the use of the ZAPROS III-*i* methodology as a means to solve problems of ordering software requirements can be promising, since this method takes into account, in addition to the factors described in the previous item, the opinion of the project manager.

ZAPROS III-*i* consists of a VDA method that aims at sorting alternatives in scenarios involving a reduced set of criteria and values and a large number of alternatives. The method relies on obtaining preferences around values that represent the distances between two criteria judgments. A preference scale can be structured, allowing the comparison of alternatives [18].

As explained in [18], the ZAPROS III-*i* method is structured in three stages: Problem Formulation, Elicitation of Preferences and Comparison of Alternatives. In the first step, we obtain the criteria and their values relevant to the decision-making process. In the second step, we generate the preference scale based on the preference of the decision-maker. The process occurs in two stages: (i) elicitation of preferences for quality variation of the same criterion, and (ii) preference elicitation between pairs of criteria. In the last step, the method performs the comparison between the alternatives based on the preferences of the decision-maker. For details on the procedure, see [11].

The method follows the same formal statement of the problem proposed in [15,16]:

Given:

1. K = 1, 2,..., N, representing a set of N criteria;

2. n_q represents the number of possible values on the scale of q-th criterion, (q ∈ K); for ill-structured problems, as in this case, usually $n_q \leq 4$;

3. $X_q = \{x_{iq}\}$ represents a set of values to the q-th criterion, and this set is the scale of this criterion; $|X_q| = n_q$ (q ∈ K), where the values of the scale are ranked from best to worst, and this order does not depend on the values of other scales;

4. $Y = X_1 * X_2 * \ldots * X_n$ represents a set of vectors y_i (every possible alternative: hypothetical alternatives + real alternatives) in such a way that: $y_i = (y_{i1}; y_{i2}; \ldots; y_{iQ})$, and $y_i \in Y$, $y_{iq} \in X_q$ and $Q = |Y|$, such that $|Y| = \prod_{q=1}^{Q} n_q$

5. $A = \{a_i\} \in Y$, i =1, 2, ..., t such that the set of t vectors represents the description of the real alternatives.

Required: The ranks of multi-criteria alternatives based on the decision-maker preferences.

The flowchart with steps to apply the ZAPROS III-*i* method procedure to rank order a set of alternatives was presented in [18] and is shown in Figure 2. In the first stage, Problem Formulation,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2018 doi:10.20944/preprints201807.0045.v1

the relevant criteria, and their values are obtained through the decision-making process. In the second stage, Elicitation of Preferences, the preference scale is generated based on the decision-maker preference. As mentioned, this stage occurs in two steps: (i) elicitation of preferences for quality variation of the same criterion, and (ii) elicitation of preferences between pairs of criteria. In the last stage, Comparison of Alternatives, the alternatives are compared based on the decision-maker preferences.

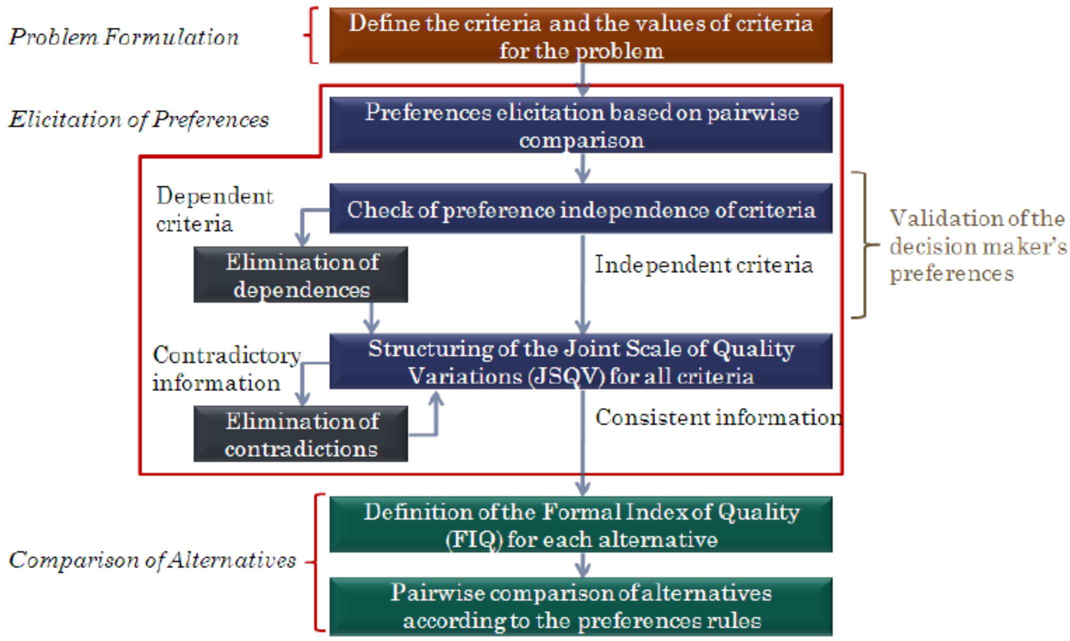

Figure 2. Procedure to apply ZAPROS III-*i* Method [18].

In the elicitation of preferences stage, decision-maker responses allow ranking of all quality variations (QV) from the scales of two criteria. This ranking is called the Joint Scale of Quality Variation (JSQV) for two criteria. All criteria are submitted to the same process. In the end, the scale of preferences for quality variations (JSQV) for all criteria is constructed [18].

To facilitate the decision-making process and to carry it out consistently, a tool called ARANAÚ was developed [19]. The tool was first developed to support the ZAPROS III method. In this work, we use an updated version of the ZAPROS III-*i* method.

4. Prioritize Software Requirements

Bagnall [7] deals with the determination of the requirements that must be executed for the next release of the software. The author predicts that customers have different levels of importance to the company and point out the requirements that have prerequisites and that must be performed in a previous or parallel release that is being implemented. The algorithms applied in this strategy show the obtaining of quick solutions to small problems.

Greer [20] state that defining which release to deliver the requirement is a decision that depends on several variables that are complexly related. They deal with different stakeholder perspectives and release planning, including effort restraint.

In allocating requirements, it is important to note that we must take into account the resources that will implement those requirements.

It is difficult to meet all the requirements identified for a system, mainly due to time and budget constraints. Requirements are usually developed in stages and prioritization helps to define which ones should be implemented first [21].

According to Karlsson [22], the requirements must be allocated in different versions of the software and, for Berander [23], the "correct" selection of the requirements that will be part of each

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2018 doi:10.20944/preprints201807.0045.v1

version is the primary step towards the success of a project or product. Therefore, it is necessary to distinguish those that will have the most significant impact on user satisfaction.

In addition to the factors already seen, we can find other aspects, such as the volatility that impacts the prioritization of requirements. Considerable effort is required to select and prioritize volatile requirements. This type of requirement is generally considered an undesirable problem. Previous studies have already identified that their characteristics may produce adverse impacts on software development processes [24]. For example, a study by Curtis [25] indicates that the volatile requirements correspond to a significant portion of the problems faced by software development companies.

Nurmuliani [24] conducted a real study in a software development company to identify the causes of volatility in requirements and the impact of this on company projects. In descending order, the author considered that the most significant changes in requirements are due to: a) inclusion of new requirements in the system, b) exclusion of requirements and c) modification of the characteristics of the requirements.

Bagnall [7] proposed a work called *the next release problem*, where the author, pioneer in this field of research, presents the problem as a search of which characteristics should be chosen concerning the variables, dependencies between requirements and priority of requirements. Next, Figure 3 shows that the requirements r(*n*), where *n* represents the requirement identification, are associated with clients (*n*), where *n* represents the customer's identification.

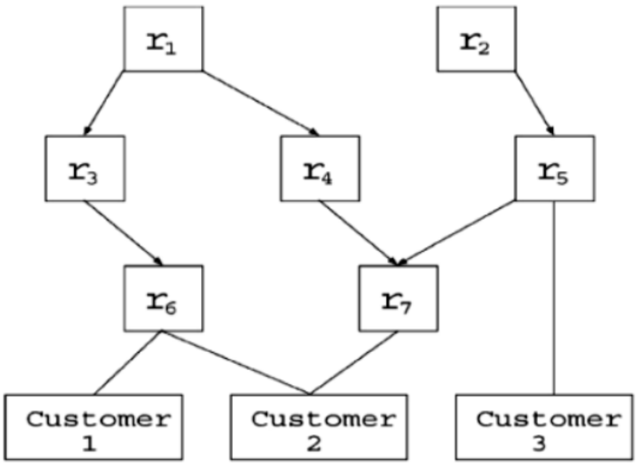

Figure 3. Representation of the requirements associated with customers [7].

The problems faced by the Search-Based Software Engineering (SBSE) are usually solved through metaheuristics. According to Becceneri [26], metaheuristic is a general algorithmic tool, which can be applied to different optimization problems, with relatively small modifications, to make them adaptable to a specific problem. Thus, we can consider metaheuristics as heuristic procedures that have generic strategies for escaping from good locations. Metaheuristics can easily incorporate new constraints and explore regions of a set in an attempt to overcome local optimality. Although they cannot guarantee global optimality, they can identify numerous points of great locations. Our work proposes to prioritize software requirements in the order in which they will be implemented using a VDA method. On the other hand, the characteristics of a family of methods [14], here we will use the method ZAPROS III-*i* [13].

The results will be compared with those obtained when using the metaheuristics (quantitative methods) Mocell [27], NSGA-II [28] and SPEA2 [29]. The choice of multicriteria resolution methods among those available owes the characteristics of the problem in question. To help validate the resulting information, let's insert a random search algorithm, which does not offer any specific search methodology.

 doi:10.20944/preprints201807.0045.v1

The methodology adopted in this work is represented in Figure 4. We will emphasize in detail the methodology used by VDA to solve the proposed problem. Later we will comment on the results of the solutions obtained by the metaheuristics of SBSE.

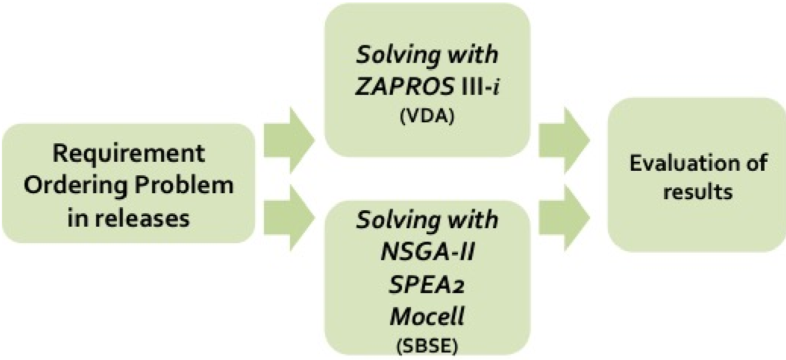

Figure 4. The methodology adopted in this work

5. Problem Generation

In this work, we are dealing with empirical problems of prioritization of requirements. Therefore, we seek to get as close to the scenario faced by companies that develop software. The mathematical formulation for the elaboration of the strategy to be studied was elaborated as follows.

$$Max\ f_{VALUE}(y) = \sum_{i=1}^{N} score_i . y_i \ , \tag{1}$$

$$Min\ f_{VOLATILITY}(x^{Pos}) = \sum_{i=1}^{N} (stability_i . x^{Pos}{}_i) . y_i \ , \tag{2}$$

Subject to:

$$\sum_{i=1}^{N} cost_i . y_i \leq resourceProject \tag{3}$$

$$x^{Pos}{}_i < x^{Pos}{}_j, \quad \text{se } Dr_i, r_j = T1\ (Technical\ Precedence: r_i\ precedes\ technically\ r_j) \tag{4}$$

The variable $x^{Pos}{}_i$ points out the position of the r_i, being able to assume a $\{0, 1, 2, \ldots N\}$, in the order of implementation established by the prioritization, for $i = 1, 2, \ldots N$.

The variable y_i indicates whether the requirement r_i will be implemented $(y_i = 1)$ or not $(y_i = 0)$, para $i = 1, 2, \ldots N$.

Function I - Demonstrates the degree of satisfaction of the stakeholders in the implementation of a set of requirements, where the $score_i = \sum_{m=1}^{M} w_m . Value\ (m, i)$ expresses the business value to the requirement r_i. In this way, and considering the importance of the client, the function adds more value as more requirements are selected.

Function II - Represents the degree of stability of the project requirements, through the advanced implementation of the requirements considered more stable. This function calculates the product between the stability of the requirement and the position to which it was allocated. Thus, a smaller value of the function indicates that the requirements with greater stability were prioritized.

Finally, the strategy constraints are presented in III and IV, where III is the cost constraint of implementing the requirements to the available budget and IV represents the constraint of precedence between the requirements. If a requirement r_i precedes a requirement r_j, then r_i must be implemented before r_j $(x^{Pos}{}_i < x^{Pos}{}_j)$.

We consider that each problem generated has 20 software requirements. Besides, we consider as seven the number of customers who are interested in the project. Among these seven clients, we have each one of them has an importance for the technology company software developer. Some clients

 doi:10.20944/preprints201807.0045.v1

(managers, CEO) may be more important than others. This is taken into account. We also consider that each of these clients may have a preference for a set of requirements that, for example, are inherent to their professional activities in the client company. To generate a more realistic problem make the process challenging, we consider that the amount of resources available is between 70% and 80% of the total value needed to implement the 20 requirements, where each of these 20 has an estimated individual value. Logically the sum of these values corresponds to the estimated overall project execution value. With a tight budget, need to implement those more stable requirements. Furthermore, this is a small guarantee that the more stable, the less prone to change it will be throughout the implementation process, thus ensuring more efficient resource spending. Therefore, implementing stable requirements first appears as an advantage to the IT company, and we consider this stability as a sorting criterion. As is known, requirements also have technical precedence between them. This characteristic was considered in this work. The representation of the simulations generated for these situations is shown in table 1.

File description	No. of Requirements	No. of Clients	Percentage of technical precedence between the requirement	Budget available for the project
A.20.7.10.70	20	7	10 %	70 %
A.20.7.10.80	20	7	10 %	80 %
A.20.7.20.70	20	7	20 %	70 %
A.20.7.20.80	20	7	20 %	80 %

Table 1. Representation of the variants for the problems generated

6. Use of the methodology VDA

Whereas the purpose of applying the decision-making procedure, a shadowing tool was used by ARANAÚ. This tool gives graphical support to the use of the ZAPROS III-*i* methodology throughout the completion of the project data required by this method to work.

To arrive at a useful classification using ARANAÚ, we follow some steps. These are a) Identification of the Alternatives, b) Definition of the Criteria and the Criteria Values, c) The ARANAÚ tool Application.

6.1. Alternatives

Initially, we considered for the set of alternatives, the 20 requirements of the software project generated by Table 1. Note that this table generated four variants. We will use them all in essays separately.

6.2. Definition of the Criteria and the Criteria Values

Since the generation of alternatives occurred in a quantitative format, we have numerical values ranging from a minimum to a maximum. For example, for the cost of a requirement, we have values between 10 and 20, where 10 represent the minimum cost added to a requirement and 20 the maximum value. Thus, it is necessary to convert these numerical values to a format that goes from the term 'low cost' to 'high cost,' as shown in Table 2. In this way, we can define and evaluate the criteria to be used in ARANAÚ.

Criteria	Criteria values

1 Cost	1.1 Requirement has low cost 1.2 Cost of the requirement is reasonable 1.3 Cost of the requirement is high
2 Stability	2.1 The requirement will hardly change 2.2 The requirement may change 2.3 The requirement will change
3 Stakeholders	3.1 The stakeholder is significant 3.2 The stakeholder has partial and isolated importance 3.3 The stakeholder is of little importance
4 Customer requirement value	4.1 The requirement is of great value to the customer 4.2 The requirement is of low importance to the customer

Table 2. Criteria and values of criteria adopted

6.3. The ARANAÚ tool Application

With all the values of the files, represented in Table 1, defined and converted to the criteria presented in Table 2, we can make use of the ARANAÚ tool. Each professional was invited to answer the questions related to the prioritization of requirements taking into account their skill and experience in this type of choice. As this is a multicriteria problem, some conflict issues have been presented to these professionals to indicate the most feasible solution. The questions were elaborated by the tool itself taking into account the data-informed about the project and requirements. Figure 5 shows the format of the ARANAÚ application.

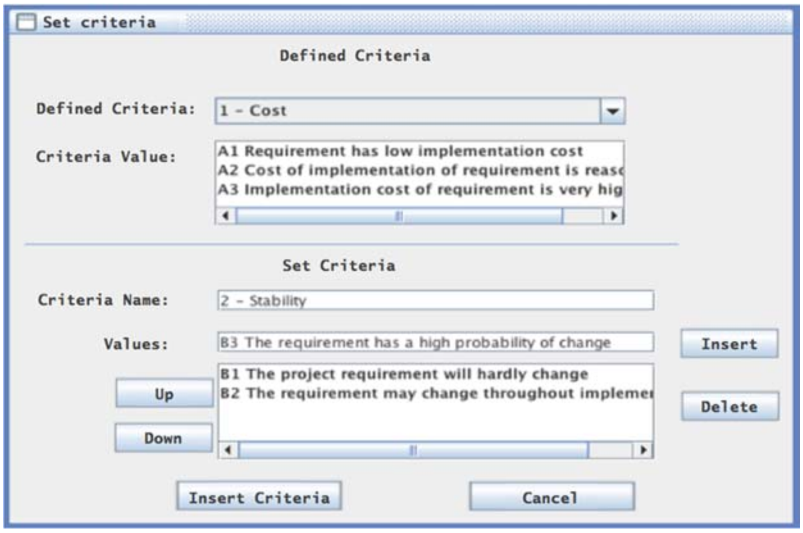

Figure 5. ARANAÚ Tool

Here the decision-maker decides what his preferences are when asked by the tool that presents solutions available in a context. As mentioned, the decision-maker is now dealing with a set of criteria and alternatives in natural language and no longer with a set of often indecipherable numbers.

7. Use of the methodology SBSE

The metaheuristics NSGA-II, Mocell, and SPEA-2, respectively presented previously, were applied to find solutions to the problem. Besides these, in this work, we also used, as a reference, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2018 doi:10.20944/preprints201807.0045.v1

algorithms of the random search. Also, this is due to the fact of the possibility of comparison and legitimation between the results obtained by these algorithms and the metaheuristics.

According to Harman [30], in the SBSE a metaheuristic must surpass a random algorithm so that it can be considered adequate.

The parameters of the algorithms were conceived along the tests of the approaches in the search for the best solution to the problem. For this, the parameters described in Table 3 were defined, as follows.

Algorithm	Parameters used
NSGA-II SPEA-2	• Initial population size: 250 individuals; • Maximum number of evaluations: 100,000 (resulting in 400 generations); • The Probability of crossing: 0.9 (TwoPointsCrossover operator); • The probability of mutation: 1.0 (SwapMutation operator); • Selection using the binary turner method.
MOCell	• Initial population size: 256 individuals; • External file size: 256; • Maximum number of evaluations: 102,400 (resulting in 400 generations); • Feedback mechanism: 20; • Crossing rate: 0.9 (TwoPointsCrossover operator); • Mutation rate: 1.0 (SwapMutation operator); • Selection using the binary turner method.
Random Search	• The maximum number of evaluations: 100,000.

Table 3. Definition of metaheuristic parameters

The results for SBSE metaheuristics will be discussed in the next session.

8. Results and Discussion

The ARANAÚ tool resulted in a set of requirements ordered according to the order of implementation, respecting the criteria for each requirement and the choices made by the decision-maker. Table 4 shows the ranking of requirements generated by the ARANAÚ tool, where requirement 8, for example, will be the first to be implemented and requirement nine will be the 20th if there is a resource available for such implementation.

Ranking	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Requirement	8	10	19	20	7	12	15	18	5	14	2	17	3	4	11	16	1	13	6	9

Table 4. Ranking generated for the problem file A.20.7.10.80

Figure 6 shows a graph of dominance among the alternatives of the set of prioritized requirements.

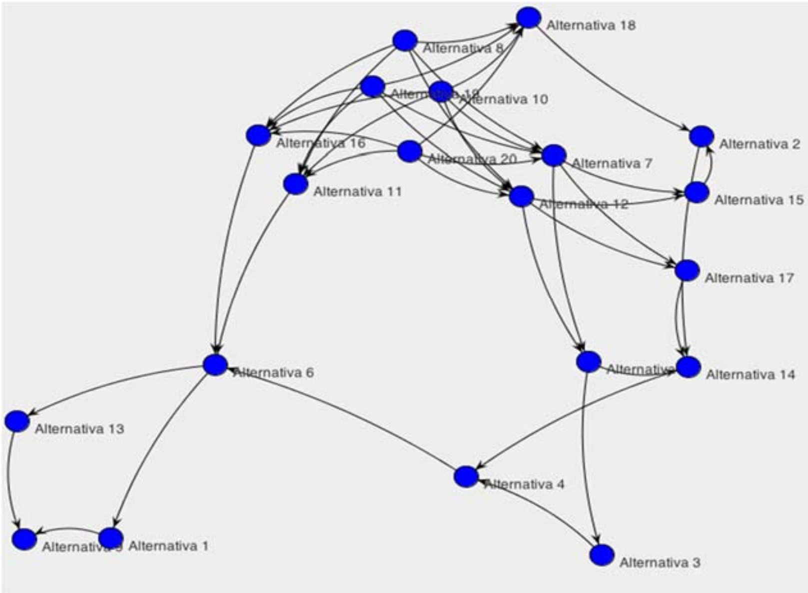

Figure 6. Graph of dominance for the file A.20.7.10.80

The results obtained from the four problems executed by the methods were extracted, tabulated. Finding the set of non-dominated solutions is one of the premises of multiobjective optimization. This set can be called the front of Pareto.

With this set, the decision-maker to choose which of the solutions best meets their needs in the context of the project.

We can see in Figures 7 and 8, the results for two of these problems. The figures show the executions of NSGA-II, SPEA2, Mocell, ZAPROS III-i and Random search algorithm.

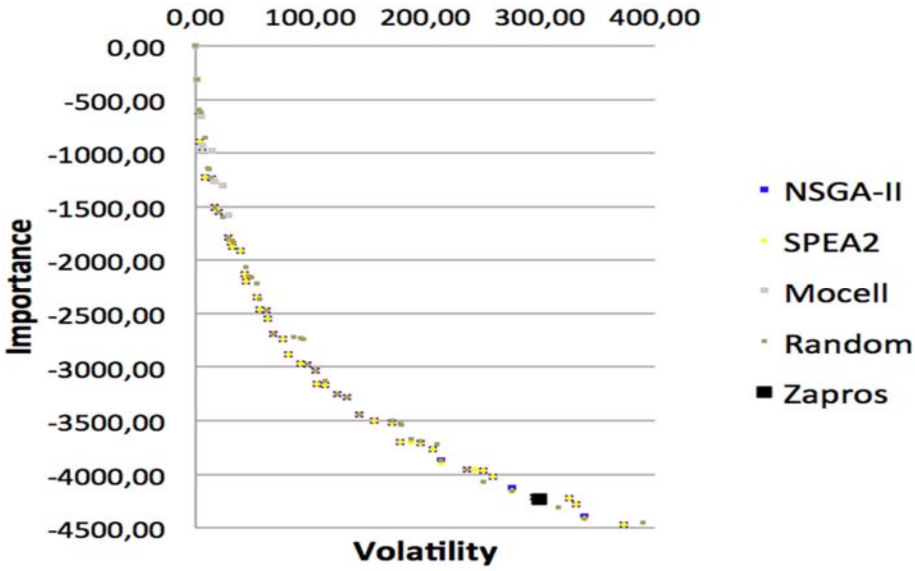

Figure 7. The graph for file A.20.7.10.80

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2018 doi:10.20944/preprints201807.0045.v1

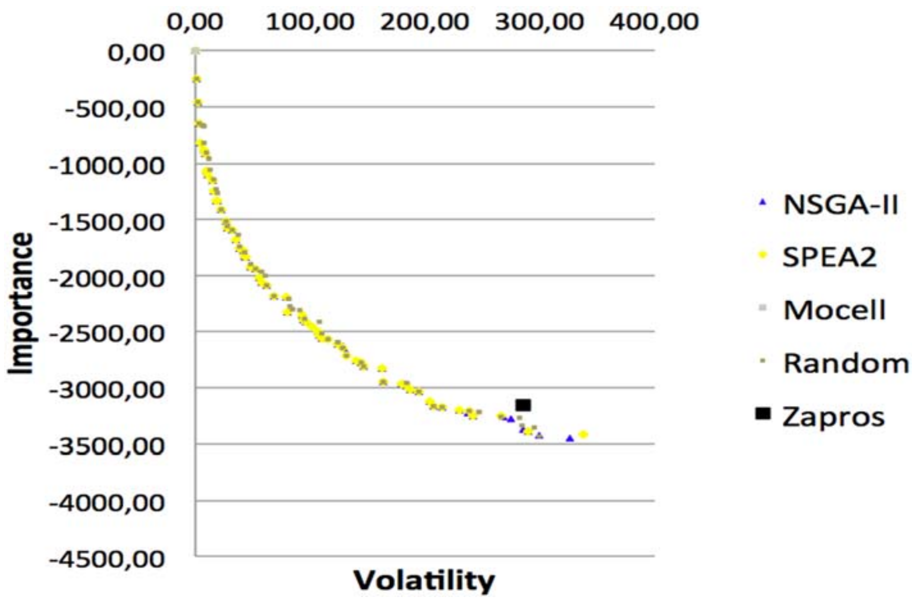

Figure 8. The graph for file A.20.7.20.80

Due to the high disparity and the difference of scales between the results obtained concerning the Pareto front, a normalization of the values of the solutions was applied to match the results.

Also, we can observe in the figures above five fronts of Pareto superimposed and differentiated by colors according to the legend. The graphs take into account the requirements importance criteria and the stability of the requirement. As already stated, the more stable requirements are implemented first, the better for the developer company, which will make the most of the available resource. However, an unstable (more volatile) requirement may be of great importance to a valued customer to the developer company. That way we have a different situation. Figures 7 and 8 represent a problem with twenty requirements, seven customers interested in this requirement and only 80% resource available to implement the software project. Figure 7 presents 10% of technical precedence between requirements while figure 8 presents 20%.

The results obtained by metaheuristics are already expected by the long way that these methods run throughout the literature. Many are the methods to solve this same type of problem, and the solutions are always similar. Now, we see that the result obtained ZAPROS III-i is represented by a single black dot in the graph. This point represents the only solution available to the decision-maker through the ARANAÚ tool. The other algorithms do not provide a single solution, but a set of them.

When we question the effectiveness of the results of metaheuristics being superior to those pointed out by the decision-maker, it is evident what the work of [31] says when it can prove in experiments of this class, this fact. However, although metaheuristics may present better results, they do not represent the expressed will of the decision-maker as a whole. The black dot shown in Figures 7 and 8 shows that this is the best choice for him, or the best solution to the problem, given his particularities and experience.

At the end of the execution and obtained the solution of ordered requirements, the decision-maker was asked to provide a note regarding its evaluation for the solution generated by ARANAÚ, where 0 is the worst score and 100 the best score. On average we have 87 points as an overall rating. However, this is entirely satisfactory when we consider that research in this field is taking its first steps. Satisfactory evaluation of humans the solution generated by ZAPROS III-i is of great importance for this research. We can verify the scores corresponding to each decision-maker and the average obtained in Figure 9.

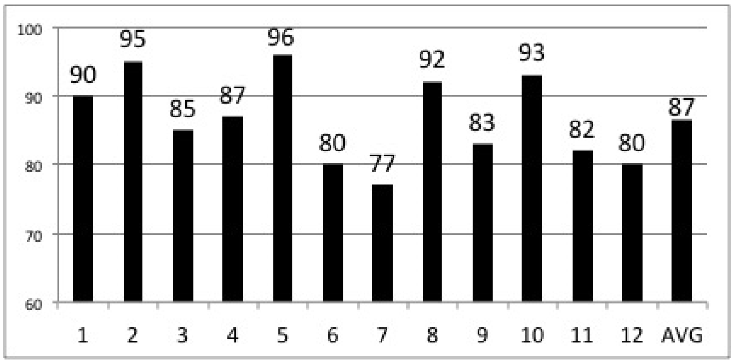

Figure 9. The score of each decision-maker for the solution generated by ZAPROS III-*i*

Moreover, this can significantly increase the time of enterprise production if we consider that the project manager will not need to look at the other options to decide on the best one. Relying on the criteria informed by himself and his professional experience, the solution presented by ZAPROS III-*i* is the one that best applies to the business.

9. Conclusions

In the process of incremental software development, we have that software release planning is one of the most complex activities [32]. This article demonstrates a new methodology for solving software release planning problems. This problem was developed for multiobjective search, where the objectives are to maximize the satisfaction of stakeholders and to minimize problems of reimplementation of volatile requirements, increasing the efficiency of the resource expense since the more stable requirements will be implemented first. By using this methodology, project managers have a range of options to increasingly customize the solutions generated to slow down the wrong decision-making.

In Barbosa [33, 34] corroborates the results of this research. When we observed the number of requirements that were tested in both approaches, the author used ten requirements and five clients interested in the project. The results obtained were very similar to those seen in this study, which used 20 software requirements and seven interested customers. Also, this demonstrates that while increasing the number of requirements, and consequently the difficulty of finding a satisfactory solution, the efficiency of this method remains.

Automated methods generate excellent solutions[35], but it is essential for the software release planning process to take into account the opinion of the decision maker and thus provide a solution in which it participates and contributes to the process of comparing alternatives.

Besides, the significant contribution we can make is that qualitative methods structured in Verbal Decision Analysis can generate solutions to solve Software Requirements Allocation Problems already faced by SBSE quantitative solutions. This fact opens precedents for further research in this field, hitherto little discussed in the literature.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author Plácido Pinheiro is thankful to the National Council of Technological and Scientific Development (CNPq) for the support received for this project. The author Paulo Alberto Melo is thankful to the Federal Institute of Ceará e a pro-rector of research and post-graduation (PRPI) for the support received for this project.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2018 doi:10.20944/preprints201807.0045.v1

References

1. M. Harman and B. F. Jones, Search-based software engineering, "Information and Software Technology," 43(14), pp. 833–839, 2001.
2. Mark Harman, Afshin Mansouri, and Yuanyuan Zhang, "Search-based software engineering: A comprehensive analysis and review of trends techniques and applications," Technical Report TR-09-03, Department of Computer Science, King's College London, April 2009. http://discovery.ucl.ac.uk/id/eprint/170689, 2009.
3. M. Harman, P. McMinn, J. T. de Souza and S. Yoo, "Search-Based Software Engineering: Techniques, Taxonomy, Tutorial. In Empirical Software Engineering and Verification," Springer Berlin Heidelberg, pp. 1-59, 2012.
4. S. Vergilio, T. E. Colanzi, A. T. R. Pozo, W. K. G. Assuncao, "Search-Based Software Engineering: A Review from the Brazilian Symposium on Software Engineering," CbSoft 2011, XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011), 2011. 49-54p.
5. F. Colares, J. T. Souza, R. A. Carmo, C. I. P. S. Padua, G. R. Mateus, "A New Approach to the Software Release Planning. In: XXII Simpósio Brasileiro de Engenharia de Software," Fortaleza. Anais do XXII Simpósio Brasileiro de Engenharia de Software. Los Alamitos, CA, USA: IEEE Computer Society, 2009. p. 207-215. DOI http://dx.doi.org/10.1109/SBES.2009.23, 2009
6. G. Ruhe, M. O. Saliu, "The Art and Science of Software Release Planning." IEEE Software, Nov./Dec. pp. 47-53, 2005.
7. A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," Information and Software Technology, vol. 43, 15, pp. 883-890, 2001.
8. J. Karlsson, C. Wohlin, B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements," Information and Software Technology, pp. 939-947, 1998.
9. B. Curtis, H. Krasner, N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, vol. 31, pp. 1268-1287, 1988.
10. C. M. Fonseca, P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization," Fifth International Conference on Genetic Algorithms, San Mateo, USA, pp. 416-423, 1993.
11. R. Saraiva, A. A. Araujo, A. D. B. Neto, I. Y. M. Bruno, J. T. Souza, "Incorporating decision-maker preferences in a multi-objective approach for the software release planning," JOURNAL OF THE BRAZILIAN COMPUTER SOCIETY (ONLINE), v. 23, pp. 1-11, DOI http://dx.doi.org/10.1186/s13173-017-0060-0, 2017
12. I. Tamanini, P. R. Pinheiro, T. C. S Machado, "Project management aided by verbal decision analysis approach: a case study for the selection of the best SCRUM practices." International Transations in Oper. Res., v. 22, DOI 10.1111/itor.12078, pp. 287-312, 2015.
13. I. Tamanini, P. R. Pinheiro, "Reducing Incomparability in Multicriteria Decision Analysis: An Extension of The ZAPROS Methods, Pesquisa Operacional (Print)," v. 31, n. 2, DOI: 10.1590/S0101-74382011000200004, p. 251-270, 2011.
14. I. Tamanini, P. R. Pinheiro, T. C. S. Machado, et al., "Hybrid Approaches of Verbal Decision Analysis in the Selection of Project Management Approaches," Procedia Computer Science, v. 55, DOI http://dx.doi.org/10.1016/j.procs.2015.07.093, pp. 1183-1192, 2015.
15. O. Larichev, H. M. Moshkovich, "Verbal decision analysis for unstructured problems," Boston: Kluwer Academic Publishers, 1997.
16. J. Figueira, S. Greco, M. Ehrgott, "Multiple Criteria Decision Analysis: State of the Art Surveys," Boston, Dordrecht, London: Springer Verlag, 2005.
17. M. Doumpos, C. Zopounidis, "Multicriteria Decision Aid Classification Methods," Boston, Dordrecht, London: Springer Verlag, DOI: 10.1007/b101986, 2002.
18. I. Tamanini, P. R. Pinheiro, "Challenging the Incomparability Problem: An Approach Methodology Based on ZAPROS. Communications in Computer and Information Science (Print)," v. 14, DOI 10.1007/978-3-540-87477-5_37, p. 338-347, 2008.
19. I. Tamanini, P. R. Pinheiro, A. L. Carvalho, "Aranau Software: A New Tool of the Verbal Decision Analysis," Technical Report, University of Fortaleza, 2007.
20. D. Greer, G. Ruhe, "Software release planning: an evolutionary and iterative approach," Information and Software Technology, pp. 243-253, 2004.

21. J. H. Allen, S. J. Barnun, R. J. Ellison, W. G. Mcgraw, N. R. Mead, "Software security engineering: a guide for project managers," Upper Saddle River, NJ Addison-Wesley, pp. 368, 2008.
22. J. Karlsson, C. Wohlin, B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements," Information and Software Technology, pp. 939-947, 1998.
23. P. Berander, "Prioritization of Stakeholder Needs in Software Engineering Understanding and Evaluation," Thesis (Licentiate of Technology in Software Engineering) - Department of Systems and Software Engineering, Blekinge Institute of Technology, Sweden, pp. 172, 2004.
24. N. Nurmuliani, D. Zowghi, S. Fowell, "Analysis of requirements volatility during software development life cycle," Australian Software Engineering Conference (ASWEC'04), 2004.
25. B. Curtis, H. Krasner, N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, vol. 31, pp. 1268-1287, 1988.
26. J. C. Becceneri, "Metaheurísticas e otimização," Computers and Operations Research, 2007.
27. J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, E. Alba, "jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics," TECH-REPORT ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, Campus de Teatinos, Universidad of Málaga, Malaga, 2006.
28. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," Evolutionary Computation IEEE Transactions on 6, https://doi.org/10.1109/4235.996017, pp. 182-197, 2002.
29. E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," IEEE Trans on Evol. Computation 3, pp. 257-271, 1999.
30. M. HARMAN and B. F. JONES, "Search-based software engineering," Information and Software Technology, pp. 833-839, 2001.
31. J. T. Souza, C. L. Maia, F. G. Freitas, D. P. Coutinho, "The Human Competitiveness of Search-Based Software Engineering. In: II International Symposium on Search-Based Software Engineering," Benevento, Proceedings of the II International Symposium on Search-Based Software Engineering, Los Alamitos, IEEE Computer Society, v. 1. pp. 34-43, DOI http://dx.doi.org/10.1109/SSBSE.2010.25, 2010.
32. M. S. Filho, P. R. Pinheiro, A. B. Albuquerque, "Applying Verbal Decision Analysis to Task Allocation in Distributed Development of Software," SEKE 2016, DOI 10.18293/SEKE2016-181, 2016.
33. P. A. M. Barbosa, P. R. Pinheiro, F. R. V. Silveira, M. S. Filho, "Applying Verbal Analysis of Decision to prioritize software requirement considering the stability of the requirement," 6th Computer Science Online Conference 2017(CSOC) Advances in Intelligent Systems and Computing, Vol. 575. ISBN: 978-3-319-57141-6. DOI 10.1007/978-3-319-57141-6_45, 2017.
34. P. A. M. Barbosa, P. R. Pinheiro, F. R. V. Silveira, M. S. Filho, "Selection and prioritization of software requirements using the Verbal Decision Analysis paradigm," The 29th International Conference on Software Engineering and Knowledge Engineering, DOI 10.18293/SEKE2017-150, 2017.
35. Marum Simão Filho, Placido R. **Pinheiro, Adriano** Albuquerque. Task Assignment to Distributed Teams aided by a Hybrid Methodology of Verbal Decision Analysis. IET Software, v. 1, p. 1-23, 2017.