A Clustering Based Technique for Large Scale Prioritization during Requirements Elicitation

Philip Achimugu, Ali Selamat, and Roliana Ibrahim

UTM-IRDA Digital Media Centre, K-Economy Research Alliance &
Faculty of Computing, Universiti Teknologi Malaysia,
Johor Baharu, 81310, Johor, Malaysia
check4philo@gmail.com, {aselamat,roliana}@utm.my

Abstract. We consider the prioritization problem in cases where the number of requirements to prioritize is large using a clustering technique. Clustering is a method used to find classes of data elements with respect to their attributes. K-Means, one of the most popular clustering algorithms, was adopted in this research. To utilize k-means algorithm for solving requirements prioritization problems, weights of attributes of requirement sets from relevant project stakeholders are required as input parameters. This paper showed that, the output of running k-means algorithm on requirement sets varies depending on the weights provided by relevant stakeholders. The proposed approach was validated using a requirement dataset known as RALIC. The results suggested that, a synthetic method with scrambled centroids is effective for prioritizing requirements using k-means clustering.

Keywords: Software, requirements, weights, prioritization, clustering.

1 Introduction

During software development process, there are more prospective requirements specified for implementation with limited time and resources. Therefore, a meticulously selected set of requirements must be considered for implementation with respect to available resources [1]. The process of selecting preferential requirements for implementation is referred to as requirements prioritization. This process aims at determining an ordered relation on specified sets of requirements [2].

There are so many advantages of prioritizing requirements before implementation. First, prioritization aids the implementation of a software system with preferential requirements of stakeholders [3]. Also, the challenges associated with software development such as limited resources, inadequate budget, insufficient skilled programmers among others makes requirements prioritization really important. It can help in planning software releases since not all the elicited requirements can be implemented in a single release due to the challenges associated with software development [4]. Consequently, determining which, among pool of requirements to be implemented first and the order of implementation is a critical success factor in software development.

T. Herawan et al. (eds.), *Recent Advances on Soft Computing and Data Mining*
SCDM 2014, Advances in Intelligent Systems and Computing 287,
DOI: 10.1007/978-3-319-07692-8_59,

To prioritize requirements, stakeholders will have to compare them in order to determine their relative value through preference weights [5]. These comparisons grow with increase in the number of requirements [6]. State-of-the-art prioritization techniques such as AHP and CBRanks seem to demonstrate high capabilities [7]. These techniques have performed well in terms of ease of use and accuracy but, still lacking in scalability and rank reversals respectively. Rank reversals refer to the ability to update or reflect rank status anytime an attribute is added or deleted from a set. In this paper, an enhanced approach for software requirements prioritization is proposed based on the limitations of existing approaches.

2 Related Work

Different prioritization techniques have been proposed in the literature. According to the research documented in [8], existing prioritization techniques are classified under two main categories, which include: techniques that are applied to small number of requirements (small-scale) and techniques that applied to larger number of requirements (medium-scale or large-scale). Examples of small-scale techniques include round-the-group prioritization, multi-voting system, pair-wise analysis, weighted criteria analysis, and the quality function deployment approach. However, techniques for prioritizing larger number of requirements include: MoSCoW, binary priority list, planning game, case based rank and the wiegers's matrix approaches.

A further classification of existing prioritization techniques was provided by [9]. They similarly divided existing techniques into two main categories: (1) techniques which enable values or weights to be assigned by project stakeholders against each requirement to determine their relative importance and (2) methods that include negotiation approaches in which requirements priorities result from an agreement among subjective evaluation by different stakeholders. Examples of techniques that apply to the first category are analytical hierarchy process (AHP), cumulative voting, numerical assignment, planning game and wieger's method. An example of the second category would be the win-win approach and the multi criteria preference analysis requirement negotiation (MPARN).

The most cherished and reliable prioritization technique as reported in the literature is the AHP technique; although it also suffers scalability problems with increase in the number of requirements. An in-depth analysis and descriptions of existing prioritization techniques with their limitations can be found in [10]. Nonetheless, obvious limitations that cut across existing techniques ranges from rank reversals to scalability, inaccurate rank results, increased computational complexities and unavailability of efficient support tools among others. However, this research seeks to address most of these limitations with the aid of clustering algorithms.

3 The Proposed Approach

Clustering is an optimization problem where the aim is to partition a given set of data objects into a certain number of clusters in order to determine the relative closeness between those objects [11]. In this paper, we concentrated on the development of a large-scale prioritization approach using k-means, where the numbers of requirement

sets (*R*), constructed clusters (*k*), and attributes (*A*) are relatively huge. K-means utilizes a two-phased iterative algorithm to reduce the sum of point-to-centroid distances, summed over all *k* clusters described as follows: The first phase implores the "batch" updates to re-assign points to their nearest cluster centroid, which initiates the re-calculation of cluster centroids. The second phase uses the "online" updates to re-assign points so as to reduce the sum of distances which causes the re-computation of cluster centroids after each reassignment. In this research, the former was adopted because; the clusters are updated based on minimum distance rule. That is, for each entity *i* in the data table, its distances to all centroids are calculated and the entity is assigned to the nearest centroid. This process continues until all the clusters remain unchanged. Before loading the datasets for the algorithm to run, there is need to pre-process or standardized them.

K-Means is an unsupervised clustering method that is applicable to a dataset represented by set of *N* to *Ith* entity with set of *M* to *Vth* feature. Therefore, the entity-to-feature matrix *Y* will be given by (*yiv*), where *yiv* is the feature value v∈ V at entity i∈ I. This process generate a partition $S= \{S_1, S_2,\ldots, S_K\}$ of *I* in *K* non-overlapping classes *Sk*, referred to as clusters. Each of these cluster have specific centroids denoted as *ck* = (*ckv*) with an *M*-dimensional vector in the feature space (k=1, 2,…K). Centroids form set $C = \{c_1, c_2,\ldots, c_K\}$.The criterion, minimized by this method, is the within-cluster summary distance to the centroids. A partition clustering can be characterized by (1) the number of clusters, (2) the cluster centroids, and (3) the cluster contents. Thus, we used criteria based on comparing either of these characteristics in the generated data with those in the resulting clustering while; the centroids are calculated by finding the average of the entries within clusters.

During requirements prioritization, the project stakeholders converge to assign weights to requirements. Before weights assignment takes place, the elicited requirements are described to the relevant stakeholders in order to understand each requirement and the implication of weighting one requirement over the other. Therefore, the main aim of this research is to propose a technique of prioritizing requirements based on the preference weights provided by the stakeholders. The metric distance function was utilized in approximating the distances between each requirement weight. These requirements can thus be considered as points in a *K* dimensional Euclidean space. The aim of the clustering in this research work is to minimize the intra-cluster diversity (distortion) when ranking or prioritizing large requirements.

The case presented in this paper has to do with the calculation of relative importance of requirement sets across relevant stakeholders based on the preferential weights of attributes contained in each set. These weights are partitioned into clusters with the help of centroids to determine the final clusters of requirement sets based on the Euclidean space of each attribute weight. The cluster centroids are responsible for attracting requirements to their respective clusters based on a defined criterion. Prioritization can therefore be achieved by finding the average weights across attributes in all the clusters. For instance, if we have requirement sets as $R = \{r_1, r_2,\ldots, r_k | i = 1,\ldots, N\}$ of dimensional attributes *A*, defined by $(a_1, a_2,\ldots, a_K)$ over 5 stakeholders. Prioritization will mean computing all the relative weights of attributes provided by stakeholders based on a weighting scale over each requirement set. These requirement sets are partitioned into various clusters given

as $K = \{k_1, k_2, \ldots, k_M\}$. Each cluster will contain the relative weights of all the stakeholders for a particular requirement set. The algorithm is described below:

1. *Initialize* m_i, $i = 1,\ldots, n$, *for example, to k random* x^t
2. *Repeat*
3. *For all* x^t *in X*
   i. $k_i^t \leftarrow 1$ *if* $\| x^t - m_i \| = min_j \| x^t - m_j \|$
   ii. $k_i^t \leftarrow 0$ *otherwise*
4. *For all* m_i, $i = 1,\ldots,n$
   i. $m_i \leftarrow$ *sum over t* $(k_i^t x^t)$ */ sum over t* (k_i^t)
5. *Until* m_i *converge*

Algorithm 1. Computation of relative weights

The vector *m* contains attribute weights with mean under each cluster, while *X* stands for the centroids and *k* represent the estimated cluster labels. The algorithm executes as follows:

1. It will select a pattern in which to initialize m_i to form clusters, and do it.
2. For each attribute in a requirement set, the algorithm captures the weights provided by the stakeholders for that set and assigns it to a new cluster (represented by m_i).
3. For each m_i, a new centroid is calculated by finding the average of the weights and the cluster is re-calculated to reflect the mean relative weights of the set.
4. Steps 2-3 are repeated until m_i converges.

Therefore, each cluster k_i (i=1,…, n) has requirements classified by the centroid. Rank reversals can be addressed by calculating a new centroid and mean each time an attribute is added or deleted from the list. The mean of a given requirement set is:

$$\overline{Z}^i = \left(\sum_{r \in k_i} R\right) \Big/ |n_i| \tag{1}$$

Assuming, the requirements are points of a Euclidean space, the normalization σ of weights in a cluster is defined as:

$$\sigma = \frac{1}{n_i} \sum_{i=1}^{N} \overline{Z}^i \tag{2}$$

However, Equation (3) is used to compute the distance or disagreement measures between requirement sets. This is achieved by computing the mean distance of attributes in each requirement set with respect to their cluster centroids.

$$d = \sum_{k=1}^{K} \left(a_k^{(i)} - a_k^{(j)}\right) \tag{3}$$

Equation 4, which is the square root of the variance, is used to prioritize requirements.

$$P = \sqrt{\sum_{k=1}^{K} \left(a_k^{(i)} - a_k^{(j)}\right)} \tag{4}$$

We ran the straight K-Means algorithm for different weights, *W* of attributes *A* in a range from START value (typically 1, in our experiments) to END value (typically,

10) with respect to the number of stakeholders *S* (Algorithm 2). The average weights of each cluster is obtained and normalized. Given a cluster *K*, the smallest *W(S, A)* is subtracted from the largest and the square root of the difference is obtained to reflect the overall relative weights of each requirement set (Equations 3 and 4).

K-Means Results Generation

1. *For K=The number of clusters START: END*
2. *For diff_init=1: number of different K-means initializations*
3. *randomly select K entities as initial centroids and normalize*
4. *run Straight K-Means algorithm*
5. *calculate the WK, the value of W(S, A)*
6. *for each K , take the average W among different clusters*
7. *compute the disagreement values and find its square root*
8. *end diff_init*
9. *end K*

Algorithm 2. Requirement prioritization process

A solution to a clustering problem can be depicted by a partitioning table and cluster centroids. These two techniques are intertwined; that is, if one is given, the optimal choice of the second one can be uniquely generated. However, this is executed based on two optimal conditions:

a. Nearest neighbour condition: The attributes for a given set of cluster centroids can be optimally classified by assigning it to a cluster with the closest centroid.
b. Centroid condition: The disagreement of the optimal cluster representative given in a partition is minimized with the help of the centroid of the cluster members.

Clustering problems can be addressed by using either the centroid based (CB) technique or partition based (PB) technique. However, in this research, the CB technique was adopted. In centroid-based technique, the centroid *X* for a given set of requirement is determined by summing all the attributes in the cluster, divided by their numbers. Each cluster is visited at least once to avoid erroneous results. The weights in each cluster is computed in a greedy way is to ensure efficient processing of clusters with large numbers of attributes and to minimize inter-cluster discrepancies. Each cluster in the solutions is assigned a number and cluster size, indicating the number of attributes that belongs to it.

The proposed technique for requirements prioritization was enhanced by applying a few steps of the k-means algorithm for each new solution. This operation first generates a rough estimate of the solution which is then refined by the k-means algorithm. This modification allows faster convergence of the solution.

4 Experimental Setup

The experiments described in this research investigated the possibility of computing preference weights of requirements across all stakeholders in a real-world software project using k-means algorithm. As mentioned previously, the metrics evaluated in this experiment are (1) the number of generated clusters, (2) the cluster centroids, and

(3) the cluster contents. The RALIC datasets was used for validating the proposed approach. The PointP, RateP and RankP aspect of the requirement datasets were used, which consist of about 262 weighted attributes spread across 10 requirement sets from 76 stakeholders. RALIC stands for replacement access, library and ID card [12]. It was a large-scale software project initiated to replace the existing access control system at University College London. The datasets are available at: http://www.cs.ucl.ac.uk/staff/S.Lim/phd/dataset.html. Attributes were ranked based on 5-point scale; ranging from 5 (highest) to 1 (lowest). As a way of pre-processing the datasets, attributes with missing weights were given a rating of zero [13].

For the experiment, a Gaussian Generator was developed, which computes the mean and standard deviation of given requirement sets. It uses the Box-Muller transform to generate relative values of each cluster based on the inputted stakeholder's weights. The experiment was initiated by specifying a minimum and maximum number of clusters, and a minimum and maximum size for attributes. It then generates a random number of attributes with random mean and variance between the inputted parameters. Finally, it combines all the attributes into one and computes the overall score of attributes across the number of clusters k. The algorithms defined earlier attempt to use these combined weights of attributes in each cluster to rank each requirement set. For the k-means algorithm to run, we filled in the variables/observations table which has to do with the three aspect of RALIC dataset that was utilized (PointP, RateP and RankP), followed by the specification of clustering criterion (Determinant W) as well as the number of classes. The initial partition was randomly executed and ran 50 times. The iteration completed 500 cycles and the convergence rate was at 0.00001.

5 Experimental Results

The results displayed in Table 1 shows the summary statistics of 50 experimental runs. In 10 requirement sets, the total number of attributes was 262 and the size of each cluster varied from 1 to 50 while, the mean and standard deviation of each cluster spanned from 1-30 and 15-30, respectively.

Table 1. Summary statistics

Variables	Obs.	Obs. with missing data	Obs. with missing data	Min	Max	Mean	Std. deviation
Rate P	262	0	262	0.000	262	5.123	15.864
Point P	262	0	262	2.083	262	28.793	24.676
Rank P	262	0	262	0.000	262	1.289	16.047

*Obs. = Objects

Also, Figure 1 shows the results of running the clustering algorithm on the data set when trying to find 10 clusters. It displays the generated 10 clusters which represent 10 sets of requirements with various numbers of weighted attributes and the within-class variance. Figure 2 shows the statistics summary of the experimental iteration. The error function value was within 3.5.

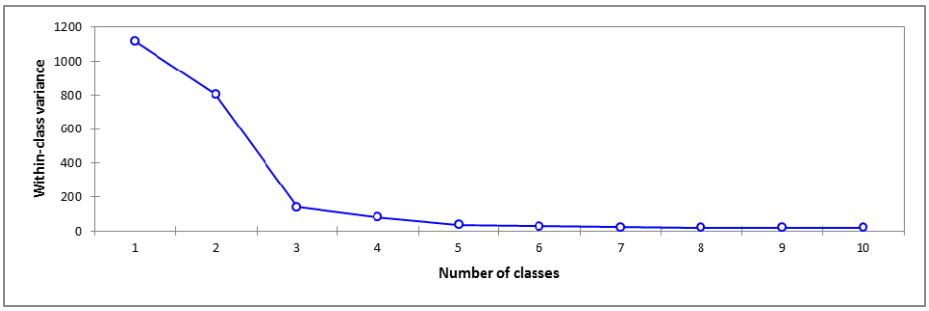

Fig. 1. Evolution of variances within classes

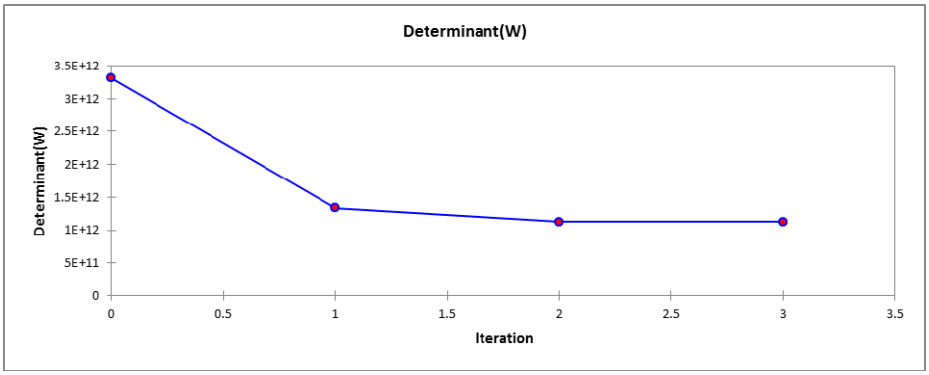

Fig. 2. Statistics for each iteration

Analysis of multiple runs of this experiment showed exciting results as well. Using 500 trials, it was discovered that, the algorithm guessed or classified requirement sets correctly. This is reflected in table 2, where the centroids for each variable were computed based on the stakeholder's weights. The sum of weights and variance for each requirement set was also calculated. The former aided in the prioritization of requirement sets, while the latter shows the variances existing between each requirement set.

Table 2. Class centroids

Class	Rate P	Point P	Rank P	Sum of weights	Within-class variance
1	4.604	17.347	0.276	53.00	7.302
2	4.230	7.6520	0.277	61.00	8.283
3	4.258	52.831	0.346	31.00	37.89
4	3.714	85.639	0.270	14.00	172.8
5	4.370	24.396	0.368	27.00	2.393
6	4.172	39.844	0.302	29.00	12.69
7	1.276	19.435	0.290	12.00	3.607
8	4.167	30.188	0.302	30.00	1.992
9	4.410	27.635	0.437	8.000	1.190
10	262.0	262.00	262.0	1.000	0.000

Table 3. Contribution (Analysis of variance)

Observation	DF(Model)	Mean squares (model)	DF (Error)	Mean square error	F	Pr > F
Rate P	1	733.847	264	249.847	2.937	0.088
Point P	1	82946.75	264	297.017	279.266	<0.0001
Rank P	1	774.132	264	255.557	3.029	0.083

Further analysis was performed using a two-way analysis of variance (ANOVA). On the overall dataset, we found significant correlations between the ranked requirements. The results of ANOVA shown in Table 3 produced significant effect on the Rate P and Rank P with minimized disagreement rates (p-value = 0.088 and 0.083 respectively). Also, the results of the ranked requirements are shown on the profile plot depicted in Figure 3. Our experiments generated 10 Gaussian clusters datasets as presented in Figure 1 and Table 2 respectively. Table 2 reflect the visual representations of the results, where the computed centroids were used in determining the relative ranks of generated clusters. The cluster shape, spread and spatial sizes are labelled according to variables specified during the experiment. Therefore, from Figure 3, it can be observed that Requirement set 4 was most ranked, followed by 3, 6, 9, 7, 1 in that order.

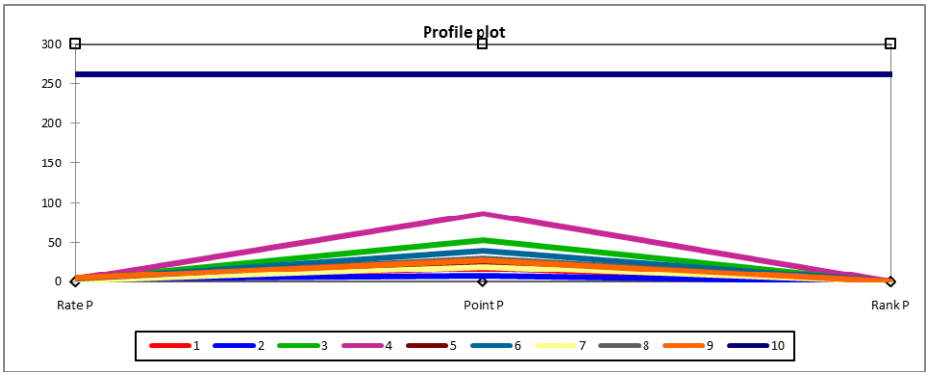

Fig. 3. Results by classes

6 Discussion

The aim of this research was to develop an enhanced prioritization technique based on the limitations of existing ones. It was eventually discovered that, existing techniques actually suffer from scalability problems, rank reversals, large disparity or disagreement rates between ranked weights as well as unreliable results. These were addressed at one point or the other during the course of undertaking this research. The method utilized in this research consisted of clustering algorithm with specific focus on k-means algorithm. Various algorithms and models were formulated in order to enhance the viability of the proposed technique. The evaluation of the proposed

approach was executed with relevant datasets. The performance of the proposed technique was evaluated using ANOVA. The results showed high correlation between the mean weights which finally yielded the prioritized results. On the overall, the proposed technique performed better with respect to the evaluation criteria described in Section 4. It was also able to classify ranked requirements with the calculation of maximum, minimum and mean scores. This will help software engineers determined the most valued and least valued requirements which will aid in the planning for software releases in order to avoid breach of contracts, trusts or agreements. Based on the presented results, it will be appropriate to consider this research as an improvement in the field of computational intelligence.

7 Conclusion and Future Work

In conclusion, prioritizing requirements is an important aspect of software development process. In this paper, a clustering based technique has been proposed for prioritizing large number of requirements. This technique can help software engineers make qualitative decisions which include: (1) Requirement elicitation (2) setting criteria that constitute each requirement (3) envisioning the expected result or output (4) determining the weights of each criterion and (5) prioritizing the requirements. Most importantly, the ability to ensure objective selection or grading process will help in the quest to develop acceptable and robust software products. In our approach, the basic elements consist of criteria which define a specific requirement, ranked with weights which are combinations of numeric values. However, the benchmark of rank accuracy between the proposed and existing techniques is worth exploration. Also, in the future, we hope to develop a parallel hybridization of clustering and evolutionary algorithms to solve requirement prioritization problem.

Acknowledgement. This work is supported by the Research Management Centre (RMC) at the Universiti Teknologi Malaysia under Research University Grant (Q.J130000.2510.03H02), the Ministry of Science, Technology & Innovations Malaysia under Science Fund (R.J130000.7909.4S062) and the Ministry of Higher Education (MOHE) Under Exploratory Research Grant Scheme (R.J130000. 7828.4L051).

References

1. Perini, A., Susi, A., Avesani, P.: A machine learning approach to software requirements prioritization. IEEE Transactions on Software Engineering 39(4), 445–461 (2013)
2. Tonella, P., Susi, A., Palma, F.: Interactive requirements prioritization using a genetic algorithm. Information and Software Technology 55(1), 173–187 (2013)
3. Ahl, V.: An experimental comparison of five prioritization methods. Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden (2005)

4. Berander, P., Andrews, A.: Requirements prioritization. In: Engineering and Managing Software Requirements, pp. 69–94. Springer, Heidelberg (2005)
5. Kobayashi, A., Maekawa, M.: Need-based requirements change management. In: Proceedings of the Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS 2001, pp. 171–178. IEEE (2001)
6. Kassel, N.W., Malloy, B.A.: An approach to automate requirements elicitation and specification. In: International Conference on Software Engineering and Applications (2003)
7. Perini, A., Ricca, F., Susi, A.: Tool-supported requirements prioritization: Comparing the AHP and CBRank methods. Information and Software Technology 51(6), 1021–1032 (2009)
8. Racheva, Z., Daneva, M., Herrmann, A., Wieringa, R.J.: A conceptual model and process for client-driven agile requirements prioritization. In: 2010 Fourth International Conference on Research Challenges in Information Science (RCIS), pp. 287–298. IEEE (2010)
9. Berander, P., Khan, K.A., Lehtola, L.: Towards a research framework on requirements prioritization. SERPS 6, 18–19 (2006)
10. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N.R.: A systematic literature review of software requirements prioritization research. Information and Software Technology (2014)
11. Kaur, J., Gupta, S., Kundra, S.: A kmeans clustering based approach for evaluation of success of software reuse. In: Proceedings of International Conference on Intelligent Computational Systems, ICICS 2011 (2011)
12. Lim, S.L., Finkelstein, A.: StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Transactions on Software Engineering 38(3), 707–735 (2012)
13. Lim, S.L., Harman, M., Susi, A.: Using Genetic Algorithms to Search for Key Stakeholders in Large-Scale Software Projects. In: Aligning Enterprise, System, and Software Architectures, pp. 118–134 (2013)