Understanding Clusters of Optimal Solutions in Multi-Objective Decision Problems

Varsha Veerappa
Department of Computer Science
University College London
London, UK
v.veerappa@cs.ucl.ac.uk

Emmanuel Letier
Department of Computer Science
University College London
London, UK
e.letier@cs.ucl.ac.uk

Abstract- **Multi-objective decisions problems are ubiquitous in requirements engineering. A common approach to solve them is to apply search-based techniques to generate a set of non-dominated solutions, formally known as the Pareto front, that characterizes all solutions for which no other solution performs better on all objectives simultaneously. Analysing the shape of the Pareto front helps decision makers understand the solution space and possible tradeoffs among the conflicting objectives. Interpreting the optimal solutions, however, remains a significant challenge. It is in particular difficult to identify whether solutions that have similar levels of goals attainment correspond to minor variants within a same design or to very different designs involving completely different sets of decisions. Our goal is to help decision makers identify groups of strongly related solutions in a Pareto front so that they can understand more easily the range of design choices, identify areas where strongly different solutions achieve similar levels of objectives, and decide first between major groups of solutions before deciding for a particular variant within the chosen group. The benefits of the approach are illustrated on a small example and validated on a larger independently-produced example representative of industrial problems.**

Keywords- Cost-value based requirements selection; multi-objective decision making, hierarchical clustering, search-based software engineering.

I. INTRODUCTION

Requirements engineering problems are inherently multi-objective: they must take into consideration a multitude of stakeholders' goals that are generally not directly comparable one to another. In such context, there is generally not one single solution that performs better than all others for all objectives simultaneously. Tradeoffs must therefore be made between the different objectives. Multi-objective decision techniques aim to help decision makers make such tradeoffs in an informed way.

One of the simplest and most common multi-objective problems in requirements engineering is that of selecting requirements based on their cost and value [1]. Given a set of requirements with estimated cost of development and estimated value to the stakeholders, the problem consists in selecting a set of requirements to be implemented so as to maximize value and minimize cost. When performed during system evolution, the problem is known as the Next-Release-Planning problem [2]. Another important example –found in NASA's Defect Detection Prevention (DDP) framework– is the problem of selecting a set of mitigation actions to reduce the project risks so as to maximize the level of goal attainment while minimizing cost [3]. For large number of requirements or large number of mitigations actions, these problems are computationally expensive to solve –for N elements to select from, there are up to 2^N possible solutions to explore. These problems are therefore generally solved by applying search-based multi-objective optimization algorithms [2][4][5][6][7] that aim to generate a set of non-dominated optimal solutions, known as the Pareto front, that corresponds to solutions for which there are no other solutions with highest value for a lower cost.

The solutions generated by search-based optimization algorithms help decision makers to explore tradeoffs between the objectives by exposing how much one goal can be improved by compromising on some other goals – e.g. how much value can be gained by increasing the cost. For industrial problems, these algorithms generate hundreds of solutions, which make the tasks of understanding them and selecting one among them difficult and time consuming. One interesting characteristic of such problems is that a set of solutions with similar cost and value could either be composed entirely of solutions that are minor variants of one another (i.e. they agree on all major decisions and only differ on smaller, less important ones) or it could be composed of solutions that include major design alternatives (i.e. they select significantly different sets of requirements). Currently, such information – which is important to make informed design decisions – is not exposed to decision makers and is almost impossible to find manually in large solutions set.

Our objective is to help decision makers identify groups of strongly related solutions in a set of optimal solutions so that they can make better informed decisions when selecting a particular solution in that set. Our approach consists in applying a clustering technique to form such groups based on how close they are in terms of design decisions. The paper contributions are (1) the design of a clustering approach using an adequate notion of solution "closeness" to form useful clusters in requirements selection problems; (2) the selection of simple clusters visualizations to fill in specific information needs in such decision processes; (3) the development of a tool that supports these techniques in Matlab; and (4) indication of the benefits of these techniques on an independently-generated solution set representative of typical industrial problems.

II. Related Work

In previous work, we have applied clustering techniques to group stakeholders based on the values they assign to requirements [8]. Our purpose there was to make multi-objective decision techniques and fairness analysis more tractable when requirements values are elicited from very large numbers of stakeholders in web-based elicitation tools [9] [10]. In that work, clustering is applied to the *inputs* of multi-objective optimization techniques to make them tractable; in this paper, it is applied to their *outputs* to provide additional information about the generated solutions to decision makers.

Our objective is also different from the objectives of analysing the sensitivity of optimal solutions with respect to variations in the model parameters – e.g. the cost and value of individual requirements [4] – or analysing the robustness of the solutions with respect to the non-determinism of the genetic algorithm search [11]. Sensitivity and robustness analysis are concerned with understanding how the set of optimal solutions could change due to uncertainties in the model parameters or caused by the genetic algorithm; we are concerned with helping decision makers understand the set of generated solutions. Our objective is therefore orthogonal and complementary to sensitivity and robustness analysis.

The idea of clustering solutions in a Pareto optimal set is not new (e.g. [12][13][14]). The approach taken by previous techniques is to cluster solutions according to how close they are in term of objective attainment (which is useful to help understanding solutions when the number of objective is larger than 3); our approach in contrast is to cluster solutions according to how close they are in terms of design decisions. This latter approach has also been proposed to help understanding optimal solutions in industrial design problems (such as optimizing the dimensions of a combustion engine's exhaust pipe) [6]. The design decisions in such problems consist in selecting optimal values for a small number of continuous variables. In contrast, the design decisions for requirements selection problems consists in making decisions for a large number of Boolean variables (indicating whether a requirement is selected or not). The difference is significant as it requires an entirely different specification of distance functions, different clustering approaches, and different cluster visualizations.

An alternative to the quantitative approach to solving multi-objective decision problems used in this paper is to apply qualitative conflict analysis techniques such as Win-Win [15], the NFR framework [16], or the soft system methodology [17]. Such approaches help clarifying the different influence on the decision problem, but provide only very limited support for choosing among alternatives [18]. Qualitative techniques may therefore be useful upstream of quantitative ones to inform and guide the quantitative model elaboration or in replacement of these when no accurate data can be obtained.

III. Background

A. *Cost-value based requirements selection*

Our techniques can be applied to any quantitative multi-objective *selection* problem, i.e. problems that consist in selecting items in a set like selecting a set of requirements to be implemented in the next release of a system [2] [7] or selecting a set of actions to mitigate risks impacting a set of goals [3]. To make the exposition simple, we will focus on the simple problem of selecting a set of requirements so as to maximize value while minimizing costs [9]. This problem is defined as follows.

Let R = $\{r_1, r_2, \ldots, r_N\}$ be a set of N requirements. Each requirement has a cost c_i that denotes how much it costs to develop it and a value v_i that denotes its value to the project's stakeholders. Cost and value are represented as two vectors: $c= [c_1, c_2, \ldots, c_n]$ and $v = [v_1, v_2, \ldots, v_n]$. A selection of some subset of requirements in R is represented by a bit vector $x = [x_1, x_2, \ldots, x_n]$ where x_i is equal to 1 if requirement r_i is selected, and equal to 0 otherwise. The total value and total cost of a set of selected requirements are defined as follows:

$$Value(x) = \textstyle\sum_{i=1..n} v_i * x_i$$

$$Cost(x) = \textstyle\sum_{i=1..n} c_i * x_i$$

A requirements selection *x* is Pareto-optimal if there is no requirements selection *y* that has both a higher value and lower cost, i.e. *Value(y)> Value(x)* and *Cost(y) < Cost(x)*. The Pareto front of the problem is the set of all Pareto-optimal solutions. When N is large finding the exact Pareto front is computationally expensive. Search-based optimization algorithms are then used for generate good approximations for this set.

Cost-value based requirements selection is a common industrial problem [1] [9] supported by commercial tools (e.g. [28] [29]). As an illustration, consider a small requirements selection problem (Problem 1) with 8 requirements whose cost and values are given in Table I. The requirements R1 to R8 in this case may for example correspond to potential features for a mobile phone device such as a touchscreen feature, a USB interface, the ability to open Excel files, etc. Applying a state-of-the-art optimization technique [9] generates a set of 27 solutions. The solutions of cost of up to 9 are shown in Table II. Fig. 1 plots all 27 solutions in terms of their cost and value. For the larger number of requirements the generated solutions can be much larger and their cost and value on the Pareto front much denser. For example, Fig. 2 shows the Pareto front for a typical industrial problem (Problem 2) with 20 requirements for which 104 solutions have been generated (the set of optimal solutions come from [5] based on original data from [19]).

Once a Pareto front has been generated, decision makers have to select one solution that they find preferable in the set. A visual inspection of the Pareto front can provide useful information to decision makers about the possible tradeoffs between cost and value; regions where the slope of the curve is steep indicate regions where a small increase in cost yield high return in value; regions where the slope is more flat indicate regions where increase in cost does not yield much additional value. This may help identifying a region on the Pareto set in which to select a solution, but it does not help decision makers select a particular solution in that region.

TABLE I. REQUIREMENTS COST AND VALUE FOR PROBLEM 1

	R1	R2	R3	R4	R5	R6	R7	R8
Value	5	3	3	2	2	1	1	1
Cost	10	5	6	5	6	3	2	3

TABLE II. SOLUTIONS OF COST UP TO 13 FOR PROBLEM 1

Sol.	R1	R2	R3	R4	R5	R6	R7	R8	Value	Cost
S1	0	0	0	0	0	0	1	0	1	2
S2	0	1	0	0	0	0	0	0	3	5
S3	0	1	0	0	0	0	1	0	4	7
S4	0	1	0	1	0	0	0	0	5	10
S5	0	1	0	0	0	1	1	0	5	10
S6	1	0	0	0	0	0	0	0	5	10
S7	0	1	0	0	0	0	1	1	5	10
S8	0	1	1	0	0	0	0	0	6	11
S9	0	1	1	0	0	0	1	0	7	13

IV. FINDING CLUSTERS OF OPTIMAL SOLUTIONS

A. *Motivation*

While the shape of the Pareto front gives information about possible tradeoffs among the conflicting objectives, it gives no information about the composition of the individual solutions in the generated set. Such information could be identified by looking at the details of each individual solution, but this approach is tedious and no support is provided to help decision makers identify relations between the different solutions. Identifying such relations may be important for decision making. An important example is the case where the Pareto front contains solutions that are close to each other in terms of cost and value but very different in terms of selected requirements. In Table II, for example, there are 4 solutions with cost 10 and value 5: S4 consists in selecting R2 and R4; S5 in selecting R2, R6, and R7; S6 in selecting the single requirement R1; and S7 in selecting R2, R7, and R8. S6 is radically different from the others because it consists of a single requirement R1, which is not present in any of the other solution around that cost. Solutions S5 and S7, however, are similar in their selection of requirements because they both include R2 and R7, and they only differ on whether to include R6 or R8. Furthermore, solution S3 which is of slightly lower cost can also be viewed as similar to S5 and S7 because it includes the same two requirements R2 and R7 and differs from them only by not including any of R6 or R8. We wish to highlight these similarities and differences to decision makers by presenting groups of related solutions. Solutions S3, S5, and S7 would for example form one group, S6 another, and S4 yet another.

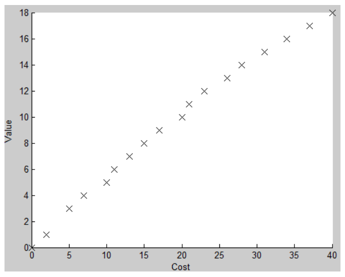


Figure 1. Pareto Front for Problem 1

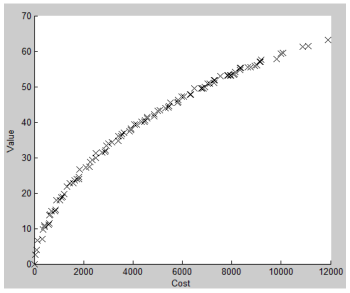


Figure 2. Pareto Front for Problem 2

Identifying such groups of solutions that share common design decisions is useful for the following reasons:

- It can help decision makers understand large solutions sets: instead of having to inspect a large number of individual solutions, they can look at a much smaller number of groups of related solutions, and focus their attention on the important characteristics of the group rather than the particularities of their individual solutions.
- This would allow decisions to be made incrementally; instead of having to select one solution in a large set of individual solutions, decision makers can first decide for a group of solutions before selecting one solution within the group.
- Such groupings would also reveal areas on the generated Pareto front where significantly different requirements selections have similar levels of objective attainment. This may be important for reasoning about system extension and contraction [21]. In our running example, assuming S6 is selected, if the project runs late, it will not be possible to deliver fewer requirements, whereas if S5 or S7 had been selected, it would be possible to drop R6, R7 or R8 and still deliver value.

Achieving these benefits depends on being able to form adequate groups within the solutions set, which in turn depends on having an adequate definition of what it means for two solutions to be closely related.

B. *Similarity Measures*

Achieving the benefits outlined in the previous section depends on being able to form meaningful groups within the solutions set using a suitable definition of what it means for two solutions to be similar.

Different distance measures can be used to compare bit vectors. A well-know measure is the Hamming distance that counts the number of bits that are different between two vectors. For example, the Hamming distance between S4 and S5 is 3 because they disagree on 3 requirements selection decisions. Such a measure, however, is not ideal in our context because it gives the same importance to requirements selection and rejection. This has undesirable consequences when assessing how close two solutions are to each other. Imagine that we have large number of requirements to choose from and two solutions that each includes 2 requirements, one of which is common to both solutions and one that differs. The Hamming distance between these solutions is 2. If we add the same third requirement to both solutions (so that they both include 2 requirements in common and differ on the third), their Hamming distance is still 2 despite the fact that they now have one more requirement in common. If we keep adding common requirements to both solutions so that they each include the same 99 requirements, for example, and differ on their selection of a hundredth, their Hamming distance will still be 2 despite the fact that they are now very similar in terms of the development activities they would require and only differ on a small fraction of what would need to be done.

Another measure more suitable to our needs is the Jaccard distance [22]. This distance measures the overlap in the number of selected requirements between two solutions. It is defined in our framework as follows:

$$\mathrm{Dist}(x,y) = 1 - \frac{\sum_{i=1..n} x_i \wedge y_i}{\sum_{i=1..n} x_i \vee y_i}$$

This distance can best be understood in set-theoretic terms: if x and y denote sets of selected requirements, their Jaccard distance is given by 1 minus the ratio of the number of requirements they have in common over the number of all requirements involved in either of the solutions:

$$\mathrm{Dist}(x,y) = 1 - \frac{\#(x \cap y)}{\#(x \cup y)}$$

The Jaccard distance between two solutions is always a number between 0 and 1. If we consider our previous example of two solutions that each have two requirements, one in common with the other and one different, their Jaccard distance is 1-1/2 = 0.5; if we consider two solutions that each have 100 requirements of which 99 are common and one differs, their Jaccard distance is 1 – 99/101 = 0.02.

The Jaccard distance has the inconvenience, however, that it gives the same importance to all requirements when comparing solutions. In practice, this is generally not the case; some requirements will have much more impact on the system design than others. For example, a requirement for a mobile phone device to include a touch screen has much more impact on the system design than a requirement for the phone to have an alarm. When assessing how close solutions are to each others, two solutions that include both the touch screen requirement and differ only on their inclusion or not of the alarm should be seen as closer to each other than two solutions that both include the alarm but differ on their inclusion of a touch screen. To allow for the importance of requirements to be taken into account, we will use a weighted version of the Jaccard distance:

$$WeightedDist(x,y,w) = 1 - \frac{\sum_{1=1..n} w_i * (x_i \wedge y_i)}{\sum_{1=1..n} w_i * (x_i \vee y_i)}$$

In this definition, w = *[w1, w2, …, wn]* is a weight vector where *wi* represents the weight of requirements r_i in the design decision. Any weight vector could be chosen by the decision makers to reflect the importance they want to give to requirements when assessing the distance between two solutions. A sensible approach could be to give high weights to requirements that have a high impact on the system architecture and low weight to those that have low impact on the architecture. Another approach that does not require additional input from decision makers is to use the cost vector as a measure of the importance of a requirement on the system design. The clustering algorithm will therefore tend to group together solutions that have a lot of development effort (or cost) in common. This reflects our perspective that the decision makers are the product developers. If the decision making process is viewed from the perspective of a product user, a good choice for the weight vector would be to use the requirements values (i.e. $w=v$). We have developed tool support that allows alternative weight vectors to be used when clustering solutions. In the rest of the paper, we will illustrate our techniques using a cost-weighted approach only.

C. *Generating Clusters*

Clustering is the partitioning of a set of elements into disjoint groups (called clusters) so that each element within a group will tend to be close to each other, and elements in different groups will tend to be far from each other. Exploring all possible clustering arrangements before selecting some "best" one is impractical due to the exponential grow in the number of all partitioning of a set. All practical clustering algorithms are therefore heuristic methods to generate a "good" clustering efficiently without guarantee that the one it finds is necessarily the best.

In this paper, we will use a standard agglomerative hierarchical clustering technique [23]. We have chosen a hierarchical clustering technique (over a partitioning one, such as k-means) because it allows us to easily adapt the level of granularity of the clusters, and it does not require the number of clusters to be specified in advance. Hierarchical clustering techniques consist in incrementally building a tree structure representing clusters at different levels of granularity, where the root node corresponds to a single cluster that includes all elements, the sub-nodes of a node represent how the parent cluster is split into sub-clusters, and the leaf nodes are clusters with a single element. Agglomerative hierarchical clustering techniques,

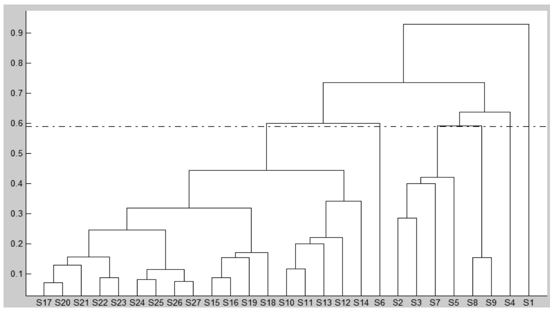


Figure 3. Dendrogram for the Solutions in Table II

as opposed to divisive ones, build this tree structure bottom-up.

The output of a hierarchical clustering algorithm is a *dendrogram* which is a representation of the tree of hierarchical clusters together with information about the distance between the clusters. In addition to specifying the distance measure to be used for comparing individual solutions, such technique needs a specification of the inter-cluster distance to be used when constructing the dendrogram. Alternative measures for such distance include the minimal, maximal, or average distance between all pair of elements in each cluster. These inter-cluster distances are called single, complete, and average linkage, respectively. We will use both the average and complete linkage as they are known to be more robust and reliable than single linkage in generating good quality clusters [23].

In our approach, we generate two dendrograms, one using complete linkage and one using average linkage, and automatically select the best one using a standard heuristic measure for comparing the quality of a dendrogram known as the cophenetic correlation [23]. A benefit of generating two dendrograms, in addition to being able to select the best of the two, is that it also allows us to assess the robustness of the generated clusters with respect to the linkage method used to generate them.

Fig. 3 shows the dendrogram generated using the cost-weighted Jaccard distance from the solutions to problem 1 shown in Table II. In this case, the best dendrogram according to their cophenetic correlation is the one using average linkage.

Visual inspection of a dendrogram can give indications about the spread of solutions in the solution set. The height of a node represents the distance between its two sub-clusters. Therefore, nodes that are low on the dendrogram form clusters whose elements are closer to each other than nodes that are placed higher up. For example, the cluster {S8, S9} forms a cluster whose elements are closer to each other than the cluster {S2, S3, S7, S5}. We can also see that splitting this latter cluster according to the partitioning shown on the dendrogram will generate a cluster {S2, S3, S7} that is not much tighter than its parent. This suggest that a good clustering for solutions S2, S3, S7, S5, S8, and S9 is to form two groups, one with the first four solutions and the second with the last two. The visual inspection of a dendrogram can also reveal long branches leading to isolated solutions, such as the ones leading to S1, S4, and S6 in Fig. 3, which may indicate the presence of "outliers". An outlier in our context is a solution that is radically different from other solutions in the set. S1, for example, which is the lowest cost and value solution in the set, includes a single low cost requirement and is significantly different from all others. This method also helps us identifying S6 and S4 as solutions significantly different from others.

Once the dendrogram has been generated, we need to decide a cut-off value for the inter-distance cluster distance at which to generate the actual clusters. For example, with a cut-off value of 0.59, shown with the dotted line on Fig. 3, we generate 6 clusters formed by the set of elements under the six branches cut by that line. Our technique chooses a default cut-off value using a standard heuristics, known as Mojena's rule [24], that estimates the best cut-off value from the distribution of the branch height on the dendrogram. We have chosen this method over alternative techniques that consists in analyzing how the clustering quality (measured by indices such as its silhouette or C-index) changes for different cut-off value because it performs much faster than these alternatives. Our tool also allows decision makers to change the default cut-off value, for example based on their visual inspection of the dendrogram or through assessing the clusters cohesion and robustness as described in the following section.

We have checked the performance of our system on large solutions sets obtained from randomly generated cost and value vectors of up to 100 requirements. On a basic laptop, generating the clusters for such models takes around 5 seconds, compared to around 10 minutes for generating the set of optimal solutions using Matlab.

D. *Assessing Clusters Cohesion and Robustness*

An optional step in our clustering procedure allows decision makers to check the cohesion and robustness of the clusters that have been generated. This can be used to gain confidence in the quality of the clusters and, if necessary, adapt the cut-off value on the dendorgram to generate better clusters. Assessing the clusters cohesion refers to assessing how close the solutions within a cluster are to each other and how far apart the different clusters are to each other. The objective is to minimize the intra-cluster distance (the mean distance between all pairs of elements within a cluster) and maximize their inter-cluster distance. Assessing the clusters robustness refers to checking how sensitive the generated clusters are with respect to the choice of clustering algorithm and parameters.

We measure clusters cohesion by computing their C-index, a standard internal quality measure adequate for analyzing cluster of bit-vectors [25]. The C-index of a cluster within a data set is given by

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}$$

where S is the sum of the distances between all pairs of solutions in the cluster, and if n is the number of such pairs, S_{min} is the sum of the n smallest distances of all pairs within

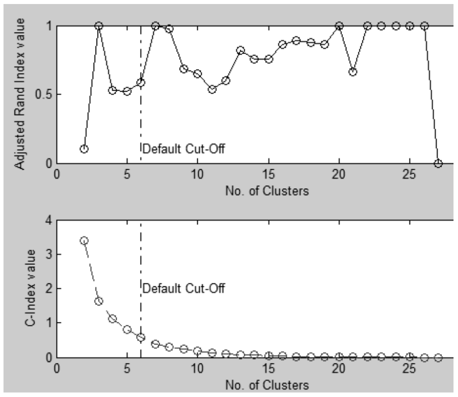


Figure 4. Rand index and C-index for example

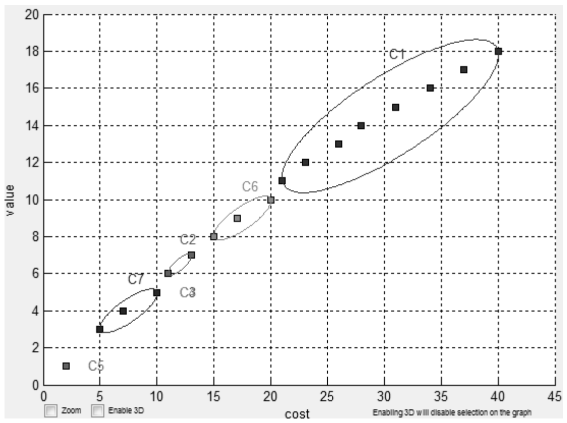


Figure 5. Cluster Visualization on the Pareto Front for Problem 1

the full data set, S_{max} is the sum of the n largest distance of all pairs within the full data set. S_{min} and S_{max} correspond to the smallest possible and largest possible sum of distances between the clusters points for all possible clusters of size n within the data set. A good cluster is one where S is close to $S_{min,}$ and therefore C close to 0. The C-index of a clustering is the average of the C-index over all the clusters.

To estimate the clusters' robustness, we compare the clusters generated by the two different dendrograms (the one using complete linkage and the other using average linkage) by measuring the Adjusted Rand Index (ARI) [26] between them. This index can be thought as measuring the percentage of solutions that are placed in the same cluster by the two methods. An ARI of 1 means that the two clustering are exactly the same.

These two measures can be computed for different cut-off values on the dendrogram in order to analyse how they vary with the number of clusters. Fig. 4, for example, shows how the ARI and C-index vary with the number of clusters in our running example. The default cut-off (given by the Mojena's value) is shown with a vertical dashed line. The figure shows that increasing slightly the number of clusters from 6 to 7 or 8 would generate a more robust clustering and a better C-index. Generating more than 8 clusters would not much improve C-index and the clusters will be less robust. For this example, we have therefore decided to generate 7 clusters. This corresponds to a cut-off value of 0.4444 (which is determined automatically by our tool from the required number of clusters).

E. Understanding Clusters

Once a cut-off value has been set, visualizing the generated clusters can be used to gain insights about the set of optimal solutions. We have developed a set of views intended to help decision makers identifying useful information about them.

1) Clusters' distribution on the pareto front

The first view consists in visualizing the clusters on the Pareto front as shown in Fig. 4. Each cluster is delimited by an ellipse of a different color whose major diameter joins the lowest and highest cost solutions in the cluster and the minor diameter is proportional to the value range of the cluster. Each solution point belonging to a cluster is also colored in its cluster's color. In Fig. 5, some solutions points, such as the 4 solutions of cost 10 and value 5, are superimposed on the Pareto Front. This may also happen for clusters. For example, clusters C1 and C3, two singleton clusters composed of solutions S6 and S4, respectively, are superimposed. A 3D visualization of this graph (not shown here) allows us to separate these superimposed clusters.

Ellipses that overlap on the Pareto front (i.e. that have non disjoint cost and value ranges) indicate an area where there may be solutions with very different requirements selections that achieve similar levels of objective attainment. The overlapping between C7, C1, and C3 is an example of this. Not all overlapping ellipses, however, will correspond to strongly different solutions. If the inter-cluster distance between two overlapping clusters is small, it is likely that the boundary between them is not so clear-cut and that solutions that belong to their intersection on the Pareto front are in fact close to each other in terms of selected requirements. Inspecting the requirements distribution for each cluster (as supported by the views presented in the next section) allows decision makers to check whether this is the case or not.

Areas with adjacent but non-overlapping clusters may indicate points on the Pareto front where there are significant differences between solutions below and above a certain cost point. Such areas exist in Fig. 5 notably between C7 and C2, C2 and C6, and C6 and C1. Again inspecting the requirements distribution within adjacent cluster will help understanding the relation between them.

Clusters with a single solution are also worth inspecting because they may denote solutions that are very different from all others around them on the Pareto curve. The clusters containing the single solutions S1, S4, and S6 are example of this.

On the figure, it is possible to zoom on an area of a Pareto front to distinguish clusters more clearly when there are many clusters and many solutions in a small region.

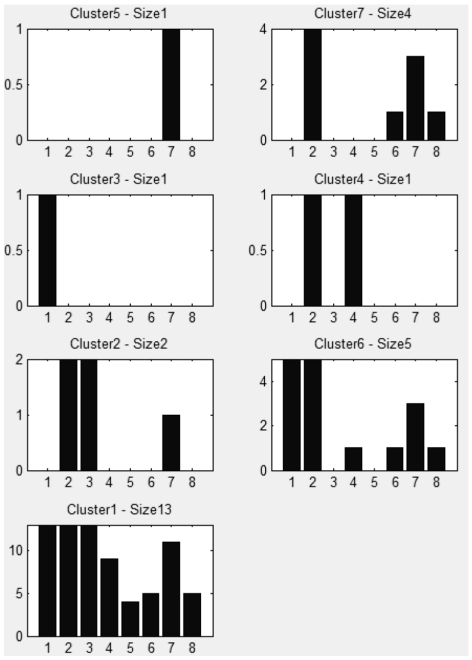
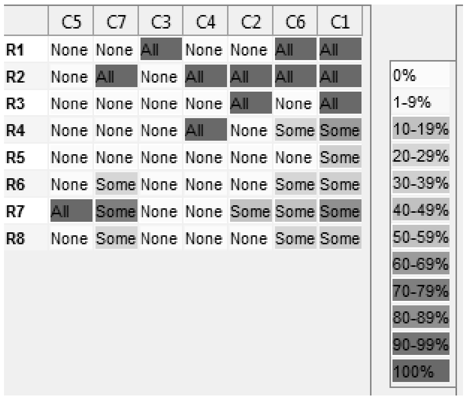


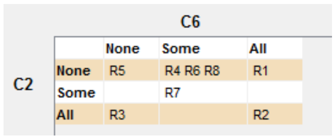
Figure 6. Requirements distribution per cluster for Problem 1



	C5	C7	C3	C4	C2	C6	C1
R1	None	None	All	None	None	All	All
R2	None	All	None	All	All	All	All
R3	None	None	None	None	All	None	All
R4	None	None	None	All	None	Some	Some
R5	None	None	None	None	None	None	Some
R6	None	Some	None	None	None	Some	Some
R7	All	Some	None	None	Some	Some	Some
R8	None	Some	None	None	None	Some	Some

0%
1-9%
10-19%
20-29%
30-39%
40-49%
50-59%
60-69%
70-79%
80-89%
90-99%
100%

Figure 7. Requirements distribution per cluster in table format



C6

C2	None	Some	All
None	R5	R4 R6 R8	R1
Some		R7	
All	R3		R2

Figure 8. Pair-wise comparison between C2 and C6 in Problem 1

Alternatively, one can ask for clusters to be re-regenerated by taking into accounts only the solutions that are within some cost range. The regenerated clusters will in general be different from the first set of clusters. This can be used notably to focus on a region of the Pareto front that includes overlapping clusters to check whether solutions that were in different clusters in the first partitioning remain in different clusters in the second clustering. If this is the case, it would increase the chance of finding strongly different solutions within that region.

Details about each cluster's composition are also given by extending the solution table (such as Table II) with a column indicating the solutions' cluster and the rows are colored with the solutions' cluster's color. Such tables can be ordered by clusters to inspect details of individual clusters. They can also be ordered by cost or by value which, thanks to the cluster colors, makes it easy to see cluster's overlap in the table.

2) *Requirements distribution per clusters*

The second set of views aims to help decision makers understand the compositions of each clusters and the relations between different clusters as cost increases.

Fig. 6 shows the requirements distribution view for each cluster in our running example. Such view shows for each cluster a bar chart that gives for each requirement (labeled on the x-axis) the percentage of solutions in the cluster in which it is selected. For example, all solutions in cluster 7 include R2, 75% of them include R7, 25% include R6, and 25% include R8. The clusters are organized by increasing order of average cost and annotated with their size. This view can help visualizing whether overlapping or adjacent cluster are strongly different or not. For example, this view shows that clusters C7, C3, and C4 that overlap on the Pareto curve have very different compositions. There are also significant differences between the adjacent clusters C2 and C6: all solutions in lower cost cluster C2 include R3, whereas none of the solutions in higher cost cluster C6 does.

A related view presents the same information as the bar charts but in a tabular form as shown in Fig. 7. In this view, the clusters are again ordered by increasing order of cost, and the distribution of each requirement within each cluster is now indicated by a color scheme (the darker the cell, the more the requirement is present in solutions within the cluster). The labels 'None', 'Some', and 'All' are also used to indicate whether the requirement is present in none, some, or all solutions within the cluster. Such view allows one to easily identify how the clusters' compositions evolve with cost and to identify which requirements tend to be present in lower cost, middle cost and higher cost clusters.

3) *Pairwise comparison of cluster's compositions*

We also found it useful to perform pairwise clusters comparisons. The purpose of this view is to make it easy to identify what is common and what is different between solutions found in both clusters. For example, Fig. 8 shows the pairwise comparison between the adjacent clusters C2 and C6. This view helps highlighting which requirements are present in both solutions (R2), absent in both (R5), and which are present in all solutions of one and absent in all solutions or the other (R3 and R1).

This view helps decision makers verifying that the separation between adjacent clusters is meaningful, and it helps them understand the key differences between the two clusters (in this example, the highest cost solutions in C6

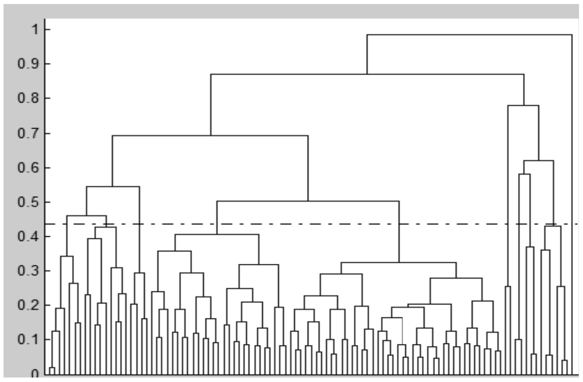


Figure 9. Dendrogram for solutions in Problem 2

include R1 that is absent from solutions in C2 but exclude R3 that is included in C2).

V. CASE STUDY

In the previous section, we have presented our approach and illustrated its relevance to support decision making on a small, artificial example. In this section, we wish to show the insights that can be gained by applying our technique to a larger solution set typical of industrial size problems.

A. *Generating the clusters*

The initial data set contain requirements whose values vary between 1 and 5 (originally from an industrial project in [19]) and whose costs estimations (randomly generated in [5]) range from 36 to 1057. The set of solutions generated for this problem in [5] contains 104 solutions shown in Fig. 2. From these solutions, our technique generates the dendogram shown in Fig. 9 and computes a default cut-off value of 0.435 which generates 10 clusters. Checking the ARI and C-index in Fig. 10 shows that this number of clusters is reasonable for this solution set; increasing the number of clusters would only yield small improvement in the C-index.

B. *Analysing the Clusters*

1) *Insights from cluster's cost-value distribution*

The visualization of the generated clusters is shown in Fig. 11. We observe that the clusters in the lower cost region are small and tend to have many overlaps. As cost increases, the clusters' sizes increase and the amount of overlap decreases. This phenomenon, which could also be observed in Fig. 5 for problem 1, is a consequence of our use of a cost-weighted Jaccard distance as clustering criteria. With such distance function, solutions consisting of fewer requirements will tend to have more differences between them than solutions composed of a larger numbers of requirements.

We also observe in Fig. 11 potentially interesting overlaps in the cost region between 2000 and 3000 and in the cost region around 6000. We will explore solutions in this second region in more details.

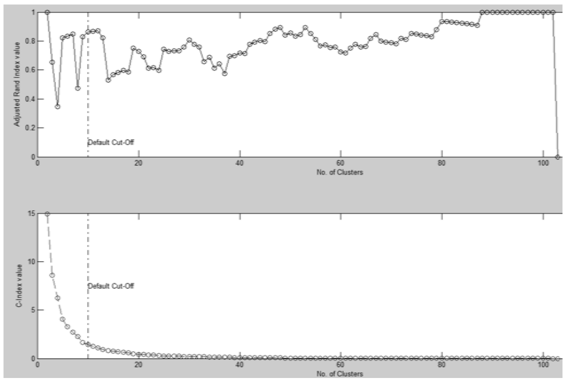


Figure 10. RAI and C-index for clusters in Problem 2

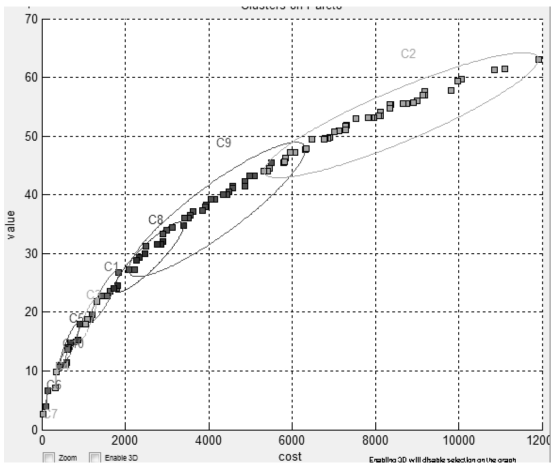


Figure 11. Clusters for Problem 2

2) *Insights from the requirements distribution evolution*

The requirements distribution bar chart in Fig. 12 shows that the clusters have different profiles and tend to incrementally include more requirements. Fig. 13 allows us to see more easily where each requirement starts appearing in optimal solutions. We see for example that R15 is included in all optimal solutions from C6, and R5 is included in all optimal solutions from C5 (remember the clusters in this view are ordered by cost). We also see that R7 and R20 are hardly part of any solution.

3) *Checking clusters overlap*

The clusters on the Pareto front show an overlap between clusters C9 and C2 around cost 6000. This overlap may indicate the presence of optimal solutions with significantly different designs in that region. To check whether this is indeed the case, we can regenerate clusters in the cost range 4000-8000 to check whether the new clustering will again partition the solutions at roughly the same place. If there are

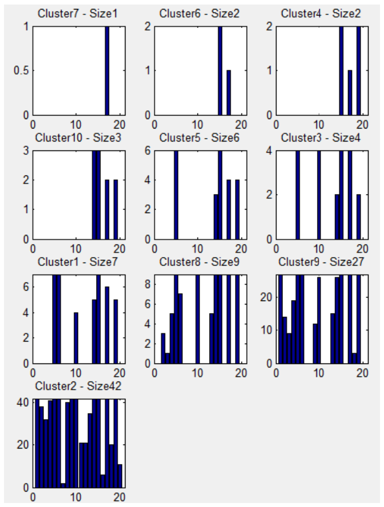
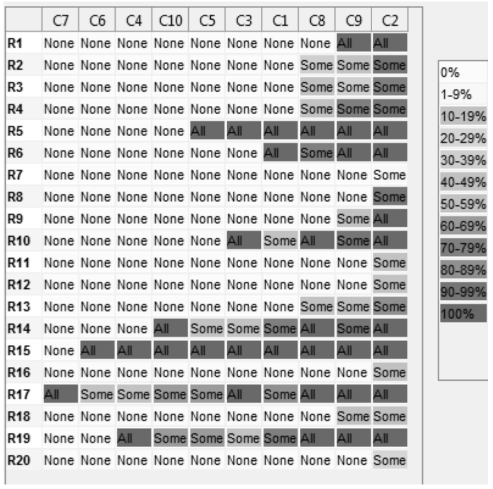


Figure 12. Requirements Distribution per Cluster for Problem 2



	C7	C6	C4	C10	C5	C3	C1	C8	C9	C2
R1	None	None	None	None	None	None	None	None	All	All
R2	None	None	None	None	None	None	None	Some	Some	Some
R3	None	None	None	None	None	None	None	Some	Some	Some
R4	None	None	None	None	None	None	None	Some	Some	Some
R5	None	None	None	None	All	All	All	All	All	All
R6	None	None	None	None	None	None	All	Some	All	All
R7	None	None	None	None	None	None	None	None	None	Some
R8	None	None	None	None	None	None	None	None	None	Some
R9	None	None	None	None	None	None	None	None	Some	All
R10	None	None	None	None	None	All	Some	All	Some	All
R11	None	None	None	None	None	None	None	None	None	Some
R12	None	None	None	None	None	None	None	None	None	Some
R13	None	None	None	None	None	None	None	Some	Some	Some
R14	None	None	None	All	Some	Some	Some	All	Some	All
R15	None	All	All	All	All	All	All	All	All	All
R16	None	None	None	None	None	None	None	None	None	Some
R17	All	Some	Some	Some	Some	All	Some	All	All	All
R18	None	None	None	None	None	None	None	None	Some	Some
R19	None	None	All	Some	Some	Some	Some	All	All	All
R20	None	None	None	None	None	None	None	None	None	Some

0%
1-9%
10-19%
20-29%
30-39%
40-49%
50-59%
60-69%
70-79%
80-89%
90-99%
100%

Figure 13. Table of Requirements Distribution for Problem 2

again different clusters at cost 6000 this would reinforce the possibility of finding significantly different design alternatives in that region; if not, this means that the overlap between C9 and C2 may have been artificial.

Regenerating clusters in that region generates 8 clusters shown in Fig. 14. The figure shows 3 overlapping clusters at cost 6000: C6, C1, and C2. These clusters are composed of 8, 5, and 4 solutions, respectively. Their pairwise comparison tables are shown in Fig. 15. The most interesting cells in pairwise comparison tables are always the All-None ones showing requirements selected in all solutions of one cluster and in no solution of the other. The tables in Fig. 15 show that the main difference between C1 and C6 is the selection of R18 in C1's solutions, the main difference between C2 and C6 is the selection of R11 in all of C2's solutions; and the main difference between C2 and C1 is whether to select R11 or R18. The tables also show

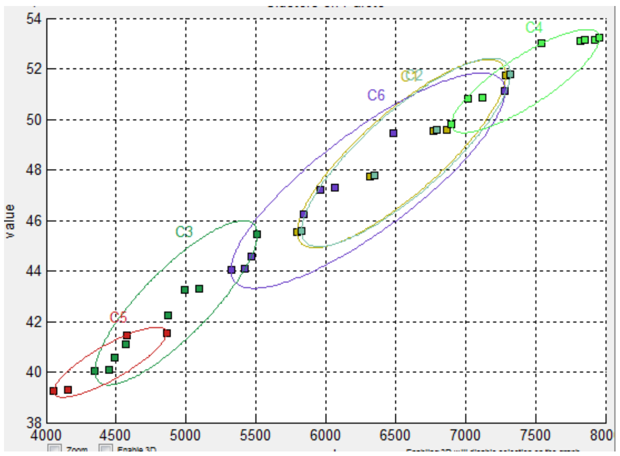
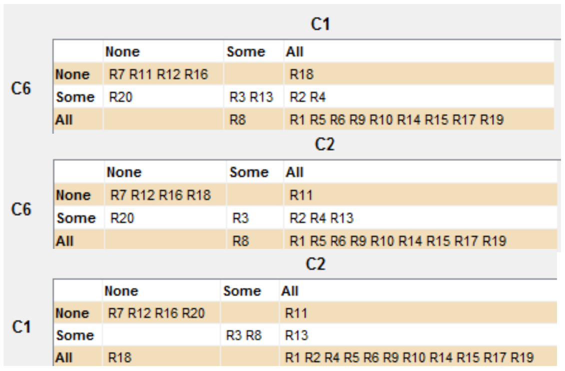


Figure 14. Clusters for Cost Range 4000-6000



C1

C6	None	Some	All
None	R7 R11 R12 R16		R18
Some	R20	R3 R13	R2 R4
All		R8	R1 R5 R6 R9 R10 R14 R15 R17 R19

C2

C6	None	Some	All
None	R7 R12 R16 R18		R11
Some	R20	R3	R2 R4 R13
All		R8	R1 R5 R6 R9 R10 R14 R15 R17 R19

C2

C1	None	Some	All
None	R7 R12 R16 R20		R11
Some		R3 R8	R13
All	R18		R1 R2 R4 R5 R6 R9 R10 R14 R15 R17 R19

Figure 15. Pairwise Comparison between C1, C2 and C6

the long list of requirements that are common between these clusters. If decision makers were to select a solution within that cost range, they would be able to first decide at the cluster level between C1, C2, and C6 before choosing one of the variants in the selected cluster.

VI. CONCLUSION

Many decision problems in requirements engineering, such as the cost-value based requirements selection problem and NASA's DDP risk mitigation selection problem, rely on quantitative multi-objective decision techniques and search-based algorithms to generate sets of optimal solutions. These sets are usually analysed in the objective space (by visualising the Pareto front curve) to inform possible trade-offs between conflicting objectives. Little work has been done so far to support decision makers in understanding variations between solutions in the design space (i.e. how they vary in terms of selected requirements).

We have argued that identifying groups of strongly related solutions may improve the quality and ease of the decision making process by (1) helping decision makers in understanding how groups of solutions are spread on the Pareto front, (2) helping them in identifying areas where strongly divergent solutions achieve similar objectives (which can notably be important when planning for potential extension and contraction of a solution), and (3) allowing them to make decisions incrementally by first selecting among groups of solutions before selecting one of the variants within the chosen group.

We have proposed a hierarchical clustering technique relying on a cost-weighted distance function as an appropriate technique to group solutions for requirements selection problems. We have then proposed a series of visualizations to support decision makers achieving the three goals mentioned in the previous paragraph.

Further validation in the field would be needed to test to what extent our approach helps reducing the cognitive loads of decision makers and helps them identifying useful information about the solution sets that they could not identify otherwise. Through the examples presented in this paper – one small and illustrative; the other large, independently-produced, and representative of typical industrial problems – we hope to have convinced the readers that our approach is likely to achieve these objectives.

REFERENCES

[1] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE software*, vol. 14, 1997, pp. 67–74.

[2] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis, "Search based approaches to component selection and prioritization for the next release problem," *Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on*, 2006, pp. 176–185.

[3] M. Feather and T. Menzies, "Converging on the optimal attainment of requirements," *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, 2002, pp. 263-270.

[4] M. Harman, J. Krinke, J. Ren, and S. Yoo, "Search based data sensitivity analysis applied to requirement engineering," *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, New York, NY, USA: ACM, 2009, pp. 1681–1688.

[5] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, and Y. Zhang, ""Fairness Analysis" in Requirements Assignments," *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, IEEE Computer Society, 2008, pp. 115-124.

[6] T. Aittokoski, S. Ayramo, and K. Miettinen, "Clustering aided approach for decision making in computationally expensive multiobjective optimization," *Optimization Methods & Software*, vol. 24, Apr. 2009, pp. 157–174.

[7] Y. Zhang, A. Finkelstein, and M. Harman, "Search based requirements optimisation: Existing work and challenges," *Requirements Engineering: Foundation for Software Quality*, 2008, pp. 88–94.

[8] V. Veerappa and E. Letier, "Clustering Stakeholders for Requirements Decision Making," *Requirements Engineering: Foundation for Software Quality*, 2011.

[9] Y. Zhang, "Multi-Objective Search-based Requirements Selection and Optimisation," Department of Computer Science, King's College London, 2010.

[10] S.L. Lim, "Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation.," School of Computer Science and Engineering, University of New South Wales, 2010.

[11] G. Gay, T. Menzies, O. Jalali, G. Mundy, B. Gilkerson, M. Feather, and J. Kiper, "Finding robust solutions in requirements models," *Automated Software Engineering*, vol. 17, Mar. 2010, pp. 87–116.

[12] J. Morse, "Reducing the size of the nondominated set: Pruning by clustering," *Computers & Operations Research*, vol. 7, 1980, pp. 55-66.

[13] M.A. Rosenman and J.S. Gero, "Reducing the pareto optimal set in multicriteria optimization.," *Engineering Optimization*, vol. 8, 1985, p. 189.

[14] C.A. Mattson, A.A. Mullur, and A. Messac, "Smart Pareto filter: obtaining a minimal representation of multiobjective design space," *Engineering Optimization*, vol. 36, 2004, p. 721.

[15] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin spiral model: a case study," *Computer*, vol. 31, 1998, pp. 33-44.

[16] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Springer, 1999.

[17] P. Checkland and J. Poulter, *Learning for Action: A Short Definitive Account of Soft Systems Methodology, and Its Use Practitioners, Teachers and Students*, John Wiley & Sons, 2006.

[18] E. Letier and A.V. Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," Newport Beach, CA, USA: ACM, 2004, pp. 53-62.

[19] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 46, 2004, pp. 243--253.

[20] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, and Y. Zhang, "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making," *Requirements Engineering*, vol. 14, 2009, pp. 231-245.

[21] D. Parnas, "Designing Software for Ease of Extension and Contraction," *IEEE Transactions on Software Engineering*, vol. SE-5, 1979, pp. 128-138.

[22] P. Jaccard, "Nouvelles recherches sur la distribution florale.," *Bulletin de la Société Vaudoise des Sciences Naturelles.*, 1908, pp. 223-270.

[23] B. Everitt, L. SABINE, L. MORVEN, and M. Leese, *Cluster Analysis*, Hodder Arnold, 2001.

[24] R. Mojena, "Hierarchical Grouping Methods and Stopping Rules: An Evaluation," *Computer Journal*, vol. 20, 1977, pp. 353-363.

[25] L.J. Hubert and J.R. Levin, "A general statistical framework for assessing categorical clustering in free recall," *Psychological Bulletin*, vol. 83, Nov. 1976, pp. 1072-1080.

[26] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, 1985, pp. 193-218.

[27] M. Harman, "The Current State and Future of Search Based Software Engineering," *2007 Future of Software Engineering*, Washington, DC, USA: IEEE Computer Society, 2007, pp. 342–357.

[28] IBM Focal Point, http://www.ibm.com/software/awdtools/focalpoint/ (Last accessed June 2011)

[29] Release Planner, http://www.releaseplanner.com/ (Last accessed June 2011)