Requirements Trade-offs Analysis by Pairwise Comparison of Alternatives and Automated Even Swaps

Golnaz Elahi[1] and Eric Yu[2]

[1] Department of Computer Science, University of Toronto, Canada, M5S 1A4
gelahi@cs.toronto.edu
[2] Faculty of Information, University of Toronto, Canada, M5S 3G6
yu@ischool.utoronto.ca

Abstract. Information systems analysts may need to make trade-offs among Non-Functional Requirements (NFRs) in order to decide on alternative design solutions or technologies to pursue. Quantitative cost-benefit analysis is not often feasible, because accurate quantitative values are often hard to obtain and biased, and decisions made using these numbers are thus unreliable. We propose a decision analysis method that assists making trade-off in the absence of numerical data. In this method, stakeholders compare the consequences of alternatives on decision criteria. The method uses a heuristic algorithm to examine all possible satisfaction levels of goals with respect to the relative rankings of alternatives. We adopt the Even Swaps decision analysis method [9] to determine the best solution. We enhance the Even Swaps method with rules for automatically suggesting swaps to decision analysts. The algorithms are implemented in a prototype tool and evaluated in an industrial case study.

1 Introduction

Information systems developers and managers need to make some key decisions such as which technologies, architectural pattern, design solutions, or products to use [6]. Each option results in different satisfaction levels of Non-Functional Requirements (NFRs). Such decisions involve making critical trade-offs among NFRs, because usually the satisfaction of one requirement, by adoption of a solution, can aid or detract from the satisfaction of other requirements and obstruct some functionality [23]. In the ideal case, if the (financial and non-financial) costs, benefits, risks, and utility of each alternative solution were known and one could accurately estimate how well each solution satisfies the requirements, alternative solutions could be objectively compared.

Quantitative cost-benefit analysis enables applying mathematical operations on numerical trade-off factors to select the optimum solution. These mathematical operations are seen as important tools to support objective decision analysis; therefore, many of the requirements decision methods [6, 12, 17, 14] rely extensively on the availability, accuracy, and meaningfulness of numerical estimations of risks, costs, benefits, and the satisfaction level of requirements. A fundamental assumption made by these methods is that either software analysts and stakeholders are able to measure how well NFRs are satisfied or they have the cognitive abilities and empirical knowledge required to estimate the requirements satisfaction level for alternative solutions.

In practice, however, formal and rigorous methods to effectively estimate the satisfaction level of various NFRs do not exist, because while some NFRs can be refined into measurable variables, many NFRs have a fuzzy or soft nature, which makes them hard, if not impossible, to measure. Stakeholders are usually biased about some solutions, and quantitative values elicited from stakeholders are often subjective and unreliable for decision making. Furthermore, time and budget limitations preclude elaborate methods for obtaining quantitative data at early stages of the development. Hence, faced with the typical absence of reliable quantitative data, objective trade-off decision analysis methods using qualitative values are needed.

We propose a trade-off analysis method to enable a systematic decision analysis in the absence of numerical data. In this method, stakeholders are asked to compare consequences of alternative solutions instead of estimating satisfaction levels of NFRs quantitatively. Unlike the way that AHP [22] works, in this method, comparisons of alternatives are not transformed to a numerical representation of utility or preferences. Instead, it considers all valid satisfaction levels that the goals could possibly have, with respect to the relative rankings of the alternatives. We do not know which of these possible satisfaction levels of goals is the actual consequences of the alternative, so the method determines the optimum alternative for each of these possible goal satisfaction levels (by using a MCDA method called Even Swaps [9]). If an alternative is the best solution for all or a majority of these possibilities, it is probably the overall optimum solution. If this condition does not hold, the final decision is made by consulting with domain experts to elicit the missing essential data.

Even Swaps is a recently introduced decision analysis method that makes trade-offs by asking decision stakeholders to give up on one NFR for better satisfaction level of another NFR. The main advantage of this method is avoiding challenges of estimating or measuring the satisfaction level of all NFRs. By using the Even Swaps method, the preferences of NFRs are implicitly extracted with respect to the level of sacrifice and compensation that stakeholders make. In the MCDA community, a tool and a set of guidelines for suggesting swaps to decision stakeholders are developed [18, 19]. Mustajoki and Hamalainen [19] develop a tool called Smart Swaps, which suggests next swaps to the user based on preference programming. In this paper, we propose a set of factors (and rules), such as considering the reusability of swaps and cognitively-easy to make, in order to suggest swaps to decision analysts. In addition, our tool is tailored for analyzing enumerations of the goals satisfaction levels.

2 Motivating Scenario

We motivate our work with a scenario at the Ministry of Transportation, Ontario (MTO), Canada. MTO managers need to decide whether to keep an old existing traffic management system (which we refer to as A_1) or deploy a new web-service based Intelligent Transportation System (ITS) (called ONE-ITS and referred to as A_2). In the traffic monitoring systems, operators control and manage the traffic by constantly monitoring traffic videos. The centers cannot share these video feeds or grant access to other departments to modify the camera settings. The ONE-ITS provides and distributes the data necessary to carry out traffic management operations and amalgamates all of the information sources into one platform.

Figure 1 shows a goal model for the Variable Message Sign (VMS) management sub-system. The VMS sub-system needs to `Update the messages easily`, which requires `Easy to learn VMS management` and `Simple VMS manipulation toolkit`. The alternative systems (A_1, the new ONE-ITS system and A_2, the current system) contribute to the requirements of the VMS system differently. For example, The ONE-ITS system `Display the VMS devices on an electronic map`, which “helps” satisfy `Easy to learn VMS management` and `Simple VMS manipulation toolkit`. However, ONE-ITS system `Enables operations over a web portal` which “hurts” the `Secure modification of messages`.

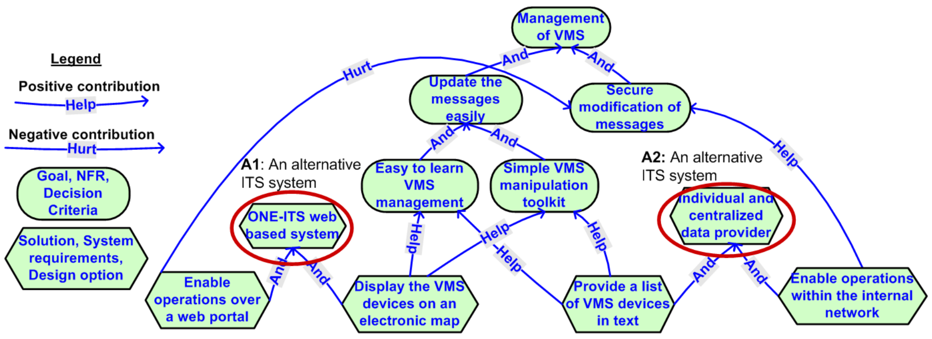

Legend
Positive contribution
Help
Negative contribution
Hurt
Goal, NFR, Decision Criteria
Solution, System requirements, Design option
Management of VMS
Update the messages easily
Secure modification of messages
Easy to learn VMS management
Simple VMS manipulation toolkit
A1: An alternative ITS system
A2: An alternative ITS system
ONE-ITS web based system
Individual and centralized data provider
Enable operations over a web portal
Display the VMS devices on an electronic map
Provide a list of VMS devices in text
Enable operations within the internal network
And
Help
Hurt

Fig. 1. The goal model of VMS sub-system. The modeling notation is described in [5].

The new system facilitates sharing and distributing traffic data, but MTO managers are concerned about unknown security threats against the web-service access to the traffic data. MTO managers have to deal with other trade-offs, such as usability, performance, implementation and maintenance costs. Quantitatively measuring the security level, usability, and maintainability of the existing system is challenging (if not impossible); measuring those qualities for the ONE-ITS system, which only exists as a system requirements specification, is not feasible. In order to make a final decision, the MTO managers were to rely on estimations of those measures for the ONE-ITS; In the rest of this paper, we will use our method to propose a solution to the MTO manager.

3 Background and Current Challenges

Multi-Criteria Decision Analysis (MCDA) is an umbrella term that groups a collection of formal approaches that take into account multiple criteria and help decision makers explore decisions when intuitive gut-feeling decision making is not satisfactory [3]. For example, in the Multi Attribute Utility Theory (MAUT) (Keeney and Raiffa [15]) the criteria preference and values are conjointly used to calculate the total utility value for each alternative. Preference (utility) elicitation requires an appropriate sequence of queries and interactions with the decisions stakeholders to obtain enough information about individual preferences. AHP [22] is a theory of relative measurement of intangible criteria to derive a numerical scale of priorities (preferences, importance weights) from pairwise comparisons of elements. Yen and Tiao [24] also use pairwise comparisons of criteria, but drive the priorities by Marginal Rate of Substitution [15]. The Even Swaps [9] method avoids eliciting explicit numerical priorities, and yet incorporates the stakeholders' preferences in determining the optimum solution by querying value trade-offs between stakeholders' goals.

Requirements trade-off analysis is the systematic examination of advantages and disadvantages of requirements as well as the design choices for a system to achieve the right balance among several competing goals [1]. Architecture Trade-off Analysis Method (ATAM) [2] is used to evaluate whether an architecture decision satisfies particular quality goals. However, ATAM is a labor-intensive analysis method which relies on the subjective opinion of the domain experts.

Feather et al. [6] propose a quantitative model for strategic decision analysis and trade-off analysis considering quality requirements, by the "coarse quantification" of relevant risk factors and their interactions. In [24], fuzzy logic values are used to represent and reason about the satisfaction degree of imprecise (quality) requirements. In [13], attribute values, such as contribution values and preference matrices, are added to goal graphs to choose and adopt a goal from the alternatives and to recognize the conflicts among the goals. Letier and Lamsweerde [16] argue that due to the lack of accuracy and measurability of goal formulations and lack of impact propagation rules through goal models, domain-specific quality variables are needed to reason on partial goal satisfaction. In [16], goals are specified in a precise probabilistic way, and the impacts of alternative decisions on the degree of goal satisfaction are analyzed.

These requirements trade-off analysis methods [6, 11, 24, 13, 16] rely on availability, accuracy, and meaningfulness of solutions' utility, or availability of precise probabilities and quantitative measures for goal-related quality variables [16]. Measuring the quality of each alternative solution, when the solution is still in form a requirements specification and not a running system, is not possible. Thus, in early RE phases, estimations are made; however, these quantitative estimates elicited from stakeholders about their preferences and solutions are imprecise, uncertain, or ill-defined, and there is the risk of relying on the results of the decision analysis despite the high levels of estimation involved [7, 20].

For these reasons, various goal-oriented early RE approaches [4, 10] focus on qualitative goal satisfaction evaluation and assessment. These techniques typically rely on structural refinement of AND-OR goal graphs to make fine-grained distinctions among alternatives, while making a minimal differentiation among degrees of goal satisfaction (and contributions of alternatives). Goal model evaluation techniques such as the ones in [4, 10, 8] enable analyzing requirements satisfaction in the absence of numerical data by using qualitative goal evaluation labels such as pratially and sufficiently satisfied (denied). However, most often, the final results of goal model evaluation with different alternatives are indifferent, and indicate that top quality goals are partially satisfied (denied), which does not sufficiently differentiate the alternative solutions.

4 Overview

This section overviews the basics of the proposed decision analysis method: we explain why comparing the consequences of alternatives on the decision criteria either provides sufficient information or narrows down the required information for making a decision over alternative solutions. This section also discusses some preliminaries and definitions, and reviews the basics of the Even Swap decision analysis method.

4.1 Making Choices by Pairwise Comparison of Alternatives

"When we deal with intangible factors, which by definition have no scales of measurement, we can compare them in pairs. Making comparisons is a talent we all have." Tomas L. Saaty [21]

Comparing intangible factors has been an alternative approach to measurements or estimations in Multi-Criteria Decision Analysis (MCDA) methods such as Analytic Hierarchy Process (AHP) [22]. The foundation of methods such as AHP is based on the intuitiveness of comparing instead of direct estimation or measurement. In the proposed method in this paper, decision stakeholders are asked to compare consequences of a pair of alternatives on the decision criteria. To illustrate how relative rankings of alternatives on decision criteria helps select a solution, let us assume two alternatives, A and B and two hypothetical decision criteria: price and quality. If price of A is better than B, we write $A > B$ on price. In general, the possible relative rankings of A and B are limited to:

1- One of A or B has a better price and quality (e.g. $A > B$ on price and quality)

2- One of A or B has a better price and the other one has a better quality (e.g. $A > B$ on price and $B > A$ on quality)

Deciding over A and B in the first situation is straightforward. In the second situation, a trade-off between price and quality needs to be made. For example, when $A > B$ on price and $B > A$ on quality, if price is more important than quality for the stakeholders, then A should be selected. However, when the number of criteria grows, making such decisions is not as straightforward as this example. Besides, the relative ranking relations of alternatives can become complicated when stakeholders are able to describe the magnitude of difference that consequences of alternatives have. For example, suppose the price of A is "highly" better than B while the quality of B is only "slightly" better than A.

In this condition, probably A is preferred to B even if quality is the priority of stakeholders, because the quality of A and B are not much different, while the price of A is highly better.

Empirical psychological evidence, on which AHP is also based, shows that humans can easily distinguish high/medium/low, and then subdivide again into high/medium/low within each interval, which in total results in 9 different levels for distinguishing the decision elements. If comparisons of alternatives are described using such 9 categories, analyzing the rankings of alternatives in conjunction with priorities over criteria becomes even more complex. To provide a systematic way to deal with these complications, we propose a heuristic search-based algorithm which enumerates and explores the possible satisfaction level of criteria with respect to the relative rankings of alternatives. In the following section, we establish a set of definitions and notations to explain the method.

4.2 Preliminaries

The Scale for Comparing. Nine levels for distinguishing the decision criteria is cognitively the maximum level of comparisons that humans can easily compare. For efficiency purposes, we use a less granular scale for comparing the alternatives consequences: $0, Low, Medium\ Low$, $Medium$, $Medium\ High$, $High$. The difference between two successive intervals is equal to Low. The maximum difference of the relative orderings of two alternatives is $High$. $High$ $(-High)$ represents the maximum satisfied (denied) level of satisfaction. We will use the abbreviations $0, L, ML, M, MH$, and H respectively.
Comparing Alternatives. Stakeholders are asked to specify which alternative in the pair better satisfies (or denies) each criterion or stakeholders goals. Decision stakeholders are also asked to estimate the difference in the consequences of alternatives toward the satisfaction of each goal, using the scale introduced above. For example, if the price of alternative A is highly better than B's, we write $\frac{A}{B}$ =High and if $B > A$ on quality, and the difference of A and B's quality is Medium, we write $\frac{B}{A}$ =Medium.
The Concept of Placement Case: To deal with complex relative ranking relations and the lack of numerical data about how well the decision criteria and stakeholders' goals are satisfied by each alternative solution, we enumerate possible satisfaction level of goals according to the rankings. To illustrate how the possible satisfaction levels are enumerated, consider the example of alternatives A and B with respect to price and quality. Assume $\frac{A}{B} = Medium\ High$ on price and $\frac{B}{A} = High$ on quality. Thus, price of A and B is either

1) $Sat(price, A)$[3]= H and $Sat(price, B)$ = L Or
2) $Sat(price, A)$ = MH and $Sat(price, B) = 0$

$\frac{A}{B} =$ MH on price, which means $Sat(price, A) - Sat(price, B) =$ MH; thus, if $Sat(price, A)$= H, then $Sat(price, B)$ =H - MH = L. On the other hand, since $\frac{B}{A} = High$ on quality, $Sat(quality, B)$ =H and $Sat(quality, A)$ =0, and there is no other possible satisfaction level for quality.

[3] $Sat(g, A)$ denotes the satisfaction level of goal g by alternative A.

We call one possible satisfaction level of goals for a pair of alternatives a "placement case". It is called a placement case, because it is one possible case of placing the satisfaction level of goals for the alternatives on the scale of satisfaction levels.

For example, two possible placement cases of A and B on the criteria are P_1 and P_2. Each placement case is the satisfaction levels of price and quality by A and B:

- $P_1(A) = \{H, 0\}$ and $P_1(B) = \{L, H\}$ on $\{price, quality\}$
- $P_2(A) = \{MH, 0\}$ and $P_2(B) = \{0, H\}$ on $\{price, quality\}$

Using Placement Cases in Decision Analysis. Each placement case is a guess about the absolute consequences of alternatives, derived from relative rankings of alternatives and the magnitude of their difference. We do not know which of the enumerated placement cases is the actual consequences of alternatives. However, finding the actual consequences of alternatives among all the possibilities may not affect the final decision, if an alternative is the preferred one in all of the placement cases. For example, in the above example of A and B, A is the better solution for both P_1 and P_2, so A is the better solution anyways. Even if a solution is preferred for a majority of the placement cases (not all), it is probably the best alternative. We will discuss the detailed algorithm and steps of the method for analyzing the enumerations and deciding over the alternatives in Section 5. The bottom line in this process is identifying the preferred alternative for each placement case. For this purpose, we adopt the Even Swaps [9] decision analysis method. The basics of the Even Swaps method are explained in what follows.

4.3 Basics of the Even Swaps Method

In an even swap, the decision analyst, collaborating with the stakeholders, changes the consequences of an alternative on one goal, and compensates this change with a preferentially equal change in the satisfaction level of another goal. For instance, in P_1, consequences of A on price and quality are $\{H, 0\}$, and consequences of $B = \{L, H\}$. Imagine the stakeholders' priority is the price. In an even swap, stakeholders agree that if the quality of A is improved from 0 to H, this improvement would be evened out by paying a higher price, so the price satisfaction level of B is decreased from H to MH. Note that such a swap shows how much stakeholders are willing to sacrifice the good price for quality, and the alternatives are not actually improved. In this example, the stakeholders have not agreed to sacrifice the good price for quality; even though the quality of A is dramatically increased, the price is only changed from H to MH.

The swap creates a new virtual alternative B' with revised consequences. The virtual alternative is as preferred as the initial one, and it can be used as a surrogate. The irrelevant goals, i.e., goals on which the consequences of alternatives are indifferent, are removed from the decision process. For example, after the swap, A and B' are indifferent with respect to quality (the quality level of both is High), so the quality criterion can be removed from the process of deciding between A and B in P_1. The underlying purpose of the swaps is to either make goals irrelevant, in the sense that both alternatives have equal consequences on this goal, or create a dominant alternative, in the sense that the other alternative is at least as good as this alternative on every attribute. A swap is defined as follows:

Definition 1. Assume two alternatives A_i and A_j and two goals g_x and g_y, where $Sat(g_x, A_i) = i_x$, $Sat(g_y, A_i) = i_y$, $Sat(g_x, A_j) = j_x$, and $Sat(g_y, A_j) = j_y$. (For brevity, we will use $Sat(g, A)$ to denote the satisfaction level of goal g by alternative A). A swap changes i_x to i'_x ($i_x \rightarrow i'_x$) and compensates this change by modifying i_y to i'_y ($i_y \rightarrow i'_y$). We write the swap as:

$$(g_x, g_y, i_x \rightarrow i'_x \Longleftrightarrow i_y \rightarrow i'_y)$$

5 The Heuristic Method For Requirements Trade-Offs Analysis

In this section, we present the heuristic decision analysis method that enumerates all placement cases based on the comparisons of alternatives, and determines the overall best solution. The method consists of 4 main steps in a loop: 1) comparing a pair of alternatives 2) enumerating the placement cases 3) deciding on the preferred solution for each placement case 4) determining the overall best alternative in the pair.

In the first step of a cycle, the algorithm takes a pair of alternatives, and given the stakeholders' goals and criteria of comparison, the analyst interacts with stakeholders, asking them to compare consequences of alternatives on each goal. In the second step, the placement cases for the pair of alternatives are enumerated according to their relative ranking. In the third step, the algorithm applies the Even Swaps method to decide which alternative in the pair is a better solution for each placement case. This solution is called the dominant one. In the fourth step, the overall best alternative is decided according to the results of the analysis for the placement cases. The overall better alternative is kept in the list of alternative solutions, and the dominated solution is dropped from the list of alternatives. The dominant solution is compared with the next alternative solution in each cycle of the algorithm. The cycles continue until only one alternative remains, which is proposed as the best available design option. In the following sections, we describe the steps of the algorithm in more details and illustrate them by analyzing the scenario at MTO.

5.1 Step 1: Comparing the Alternatives Consequences

A solution either has a positive or negative impact on the satisfaction of a goal. When eliciting and comparing the consequences of two alternatives such as A_i and A_j on a goal g, three main pieces of information are collected: the contribution type: negative or positive, which alternative better satisfies g, and the value of $\frac{A_i}{A_j}$ (or$\frac{A_j}{A_i}$).

For example, Figure 2 shows the comparisons of two hypothetical alternative solutions, A_1 and A_2, on three example goals: Usability (g_1), Performance (g_2), and Maintainability (g_3). The model shows that both alternatives have a negative impact on Performance, but A_1 has a *Medium High* stronger impact on this goal than A_2. This comparison does not indicate whether A_1's Performance is

$-High$ or $-Low$, but we know that whatever negative value that A_1 contribute to Performance, A_2 contributes $Medium\ High$ levels less than that.

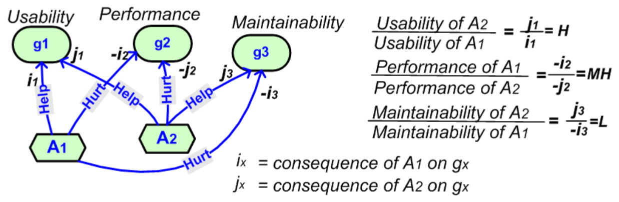

Usability
Performance
Maintainability
g1
g2
g3
A1
A2
Help
Hurt
Help
Hurt
Help
Hurt
i1
j1
-i2
-j2
j3
-i3
Usability of A2 / Usability of A1 = j1 / i1 = H
Performance of A1 / Performance of A2 = -i2 / -j2 = MH
Maintainability of A2 / Maintainability of A1 = j3 / -i3 = L
ix = consequence of A1 on gx
jx = consequence of A2 on gx

Fig. 2. Comparisons of two alternative systems at MTO.

5.2 Step 2: Enumerating the Placement Cases

In the second step, the placement cases are enumerated by generating possible satisfaction levels of every goal and combining them to unique placement cases. Goals satisfaction levels depend on the type of alternatives' contributions on the goals (positive or negative) and the difference between the consequences of alternatives. Consider a pair of alternatives A_i and A_j that contribute to a set of goals $G = \{g_1, g_2, ...g_m\}$, a placement case P, where $P(A_i) = \{i_1, i_2, ...i_m\}$ and $P(A_j) = \{j_1, j_2, ...j_m\}$. Possible satisfaction level of a goal like g_x by A_i and A_j is enumerated as follows: (for $1 \leq x \leq m$, $d_x = |i_x - j_x|$):

If $Sat(g_x, A_j) > Sat(g_x, A_i)$ then:

1- If A_j and A_i's contributions are both positive:
$d_x \leq j_x \leq H$ and $i_x = H - j_x$

2- If A_j and A_i's contributions are both negative:
$-H \leq j_x \leq -d_x$ and $i_x = -H - j_x$

3- If A_j's contribution on g_x is positive and A_i's contributions is negative:
$0 \leq j_x \leq d_x$ and $i_x = j_x - d_x$

If $Sat(g_x, A_i) > Sat(g_x, A_j)$, in the above statements, i_x shall be switched with j_x. Note that although the scale of $-High$ to $High$ has 11 different intervals, there are at most 5 different satisfaction levels for any given goal: in conditions 1 and 2, where $\frac{A_j}{A_i} = Low$, g_x can have 5 different satisfaction levels at the positive (negative) side of the scale. In the third condition, where $\frac{A_j}{A_i} = High$, g_x can have 5 possible satisfaction levels, in which, one alternative is placed at the negative side of the scale and the other is at the positive side.

For example, Figure 3 shows that, on the scale of $-High$ to $High$, there are only two possible different sets of values that can be assigned to A_1 and A_2's Performance. Either the A_1's Performance is $-High$ (and the A_2's Performance is $-Low$), or A_1's Performance is $-Medium\ High$ (and A_2's Performance is 0).

At the same time, there are two possible satisfaction levels for Maintainability of these two alternatives. A_1 has a negative impact on Maintainability and A_2 has a positive impact on Maintainability, while the difference of these two contributions is Low. Therefore, A_1's Maintainability is either $-Low$ or 0 and accordingly, A_2' Maintainability is 0 or Low. Since Usability of A_1 and A_2 has

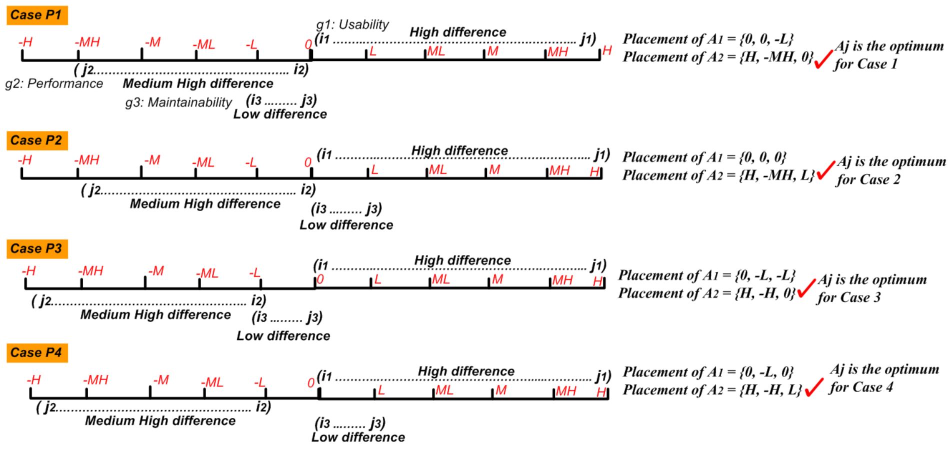


Fig. 3. Possible placement cases for A_1 and A_2 in the MTO scenario

a $High$ difference, there is only possible Usability level for the alternatives: A_1's Usability is 0 and A_2's Usability is $High$.

By combining 2 possible satisfaction levels for Performance and Maintainability and the only possible level of Usability, A_1 and A_2 could result in 4 different goals satisfaction statuses. The right-hand side of Figure 3 shows these 4 possible placement cases, P_1, P_2, P_3, and P_4 for A_1 and A_2. Each case includes one possible way to place the contributions of A_1 and A_2 on the scale, according to the difference between the consequences of alternatives on the the goals.

5.3 Step 3: Determining the Dominant Alternative for Placement Cases

Once all possible placement cases are generated, the algorithm finds the dominant alternative for each placement case. An absolute dominant alternative better satisfies all goals. More formally, absolute dominance is defined as:

Definition 2. An alternative A_i is called absolute dominant for a placement case P, if $\forall (i_x \in P(A_i) \wedge j_x \in P(A_2))$, $i_x \geq j_x$. In that case, A_j is dominated by A_i, and we write $A_i > A_j$.

In none of placement cases in Figure 3, neither of A_1 and A_2 are absolute dominant. This is a typical situation because each alternative better satisfies some of the goals. We use the Even Swaps [9] method to determine the dominant alternative, i.e., the overall better alternative, when neither of them is the absolute dominant. This method has been discussed in Section 2.3. For example, in the placement case 1 in Figure 3, we ask stakeholders "if Performance of A_1 is decreased from 0 to -MH", how much improvment they expect on Usability. This swap is written as $(g_2, g_1, 0 \rightarrow -MH \Leftrightarrow 0 \rightarrow ?)$. Assume stakeholders state that they expect that Usability of A_1 to be increased from the level of 0 to "M". Consequences of A_1 (now A_1') are modified to $\{M, -MH, -L\}$, and compared to the consequences of A_2 on the goals ($\{H, -MH, 0\}$), A_2 is obviously a better choice.

When the problem scales and several NFRs and alternative solutions need to be considered, determining the best swap among numerous possibilities is hard for human decision makers [19]. In the proposed method, the Even Swaps decision method is invoked for every placement case; thus expert and non-expert users may need to make numerous swaps in a tedious and long process. In Section 4, we propose an algorithm for finding the right swaps in each step of the Even Swap process. The algorithm intends to minimize the number of swap queries, reuse previous swaps, and at the same time, finds swap queries that are easy to answer for stakeholders.

5.4 Step 4: Determining the Overall Better Alternative in the Pair

Once all possible placement cases are generated, the method determines the dominant alternative for each placement case by using the Even Swaps method [9]. Then the heuristic method decides which alternative is overall a better solution out of the pair of alternatives. If an alternative is decided as the dominant one all of the possible placement cases, the algorithm suggests that solution as the definite optimum solution in the pair of alternatives.

In a pair of alternatives, such as A_i and A_j, typically, A_i is dominant for a number of cases, let us assume w_1 number of placement cases and A_j is dominant for w_2 placement cases. If w_2 is sufficiently small, those w_2 cases are probably exceptional placement cases for which A_i is not dominant. For example, assume A_i is better than A_j except for few cases where $Sat(g, A_i) <$Medium on a goal like g. Such cases are exceptional patterns of placement cases.

The algorithm provides the pattern of the exceptional cases to domain experts for a final evaluation and judgment. If the domain expert judges that A_i's contributions does match those w_2 exceptional cases, A_i is the overall dominant solution in the pair. For example, consider the patterns of placement cases where $Sat(g, A_i) <$Medium. If the domain expert believes that $Sat(g, A_i)$ is Medium or higher than Medium (not matched with the exceptional cases), then A_i is definitely the better solution in the pair of A_i and A_j. On the other hand, if the domain expert believes that the absolute value of $Sat(g, A_i)$ is actually lower than Medium, all w_1 cases, where $Sat(g, A_i) >$Medium, are invalid placement cases that do not match the reality. The actual consequence of A_i and A_j is one of the placement cases among those w_2 cases, and A_j is the better solution.

If domain experts are not able to evaluate the patterns, the proposed method cannot determine the overall best alternative in the pair due to the lack of information. In this situation, any numerical method would not be able to determine the best solution either. When neither w_1 nor w_2 are small enough and for nearly half of the placement cases, A_i is dominant and for the other half of placement cases A_j is dominant, one possible conclusion is that the usefulness of the alternatives are too similar that the proposed algorithm cannot differentiate the alternatives.

6 Automatically Suggesting Swaps

The maximum number of possible satisfaction levels for each goal is 5; thus, having m goals as the decision criteria, the Even Swaps process may be invoked

5^m times. Since the swapping process involves user queries, 5^m number of Even Swaps user queries quickly become a labor-intensive and tedious process. The number of swap enquiries can be reduced by reusing swaps from the previous placement cases for other cases. In addition, if purposefully suggested, swaps can help "create" an absolute dominant alternative with the fewest enquiries. We develop a set of rules for automatically suggesting useful swaps; for example, suggesting swaps that lead to an absolute dominance situation, finding the most reusable swaps, swapping the minimally satisfied goals with the maximum satisfied goals, and reusing the existing determined dominance situations.

6.1 Creating a Dominance Situation

In the placement case P_1 in Figure 3, where $P_1(A_1) = \{0, 0, -L\}$ and $P_1(A_2) = \{H, -MH, 0\}$, by swapping g_2 and g_1 and removing g_2, A_2 may become the absolute dominant alternative. On the other hand, swapping g_1 and g_3 is not useful for creating a dominance situation, because before swapping them, at least one swap is needed to reach a dominance situation for P_1, and by swapping g_1 and g_3 and removing one of them from the list of goals, we still need to make another swap (with g_2) to reach an absolute dominance. In order to reach an absolute dominance we need to swap two goals that are in trade-offs, where one alternative is dominant for satisfying the first goals and the other alternative is stronger for satisfying the second goal. A swap that resolves such a trade-off situation helps create an absolute dominant alternative.

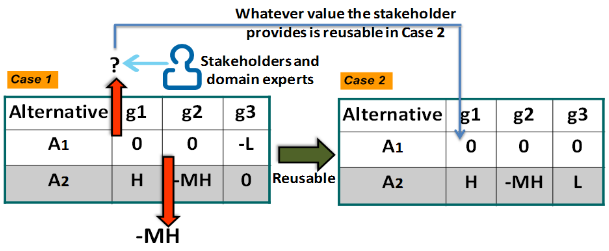
Rule 1, create a dominance situation: Given a set of goals $G = \{g_1, g_2, ...g_m\}$, if consequences of $A_i = \{i_1, i_2, ...i_m\}$ and consequences of $A_j = \{j_1, j_2, ...j_m\}$, and $w_1 =$ number of goals where $i_x > j_x$ and $w_2 =$ number of goals were $j_x > i_x$ (for $1 \leq x \leq m$), then $w_1 \times w_2$ swaps exist that could potentially reduce the number of steps of the Even Swaps process to make an alternative dominant.

6.2 Suggesting the Most Reusable Swap

When a swap query is asked from stakeholders, it can be reused for another identical case, without further consultation with human stakeholders (if the satisfaction level of goals are the same). For example, the swap $(g_2, g_1, 0 \rightarrow -MH \Leftrightarrow 0 \rightarrow ?)$ (from the placement case P_1 in Figure 3) is reusable in the placement case P_2 because $Sat(g_1, A_1)$, $Sat(g_2, A_1)$, and $Sat(g_1, A_2)$ are equal in both placement cases (illustrated in Figure 4). On the other hand, swapping g_3 and g_2 in P_1 as $(g_3, g_2, -L \rightarrow 0 \Leftrightarrow 0 \rightarrow ?)$ is not reusable in P_2, because $Sat(g_3, A_1)$ in P_1 and P_2 are not equal $(-L \neq 0)$.

Rule 2, pick the most reusable swap: Given a set of tuples of goals, as swap candidates, this rule states that the most reusable tuple should be selected for the next swap. The notion of reusable swap is formally defined as follows:

Definition 4. A swap like $(g_x, g_y, i_x \rightarrow i'_x \Leftrightarrow i_y \rightarrow i'_y)$ is reusable in a placement case P, where $Sat(g_x, A_i) = I_x$, $Sat(g_y, A_i) = I_y$, $Sat(g_x, A_j) = J_x$, and $Sat(g_y, A_j) = J_y$, if $i_x = I_x$, $i'_x = J_x$, and $i_y = I_y$.


Whatever value the stakeholder provides is reusable in Case 2
?
Stakeholders and domain experts
Case 1
Case 2
Reusable
-MH

Alternative	g1	g2	g3
A1	0	0	-L
A2	H	-MH	0

Alternative	g1	g2	g3
A1	0	0	0
A2	H	-MH	L

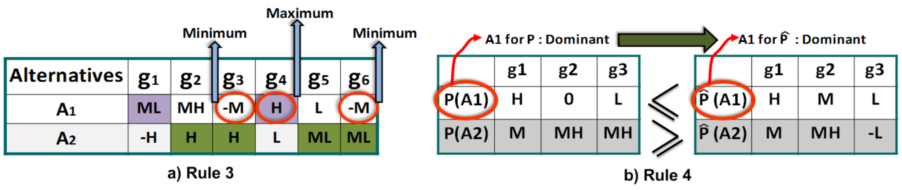
Fig. 4. An example of reusable swap

6.3 Suggesting Easy Swaps

Hammond et al. [9] suggest making the easiest swaps first, e.g., money and costs are easy goals. What would make a swap easy for the stakeholders? For example, stakeholders may easily agree to increase the satisfaction level of a goal that is not sufficiently satisfied and compensate it with decreasing the satisfaction level of a goal that is highly satisfied, intending to reach a balance among software requirements.

Rule 3, swap goals with minimum satisfaction level and goals with the maximum satisfaction level: This rule states that among the candidate tuples like (g_x, g_y), two goals such as g_i and g_j should be swapped where g_i has the minimum satisfaction level among all g_x, and g_j has the maximum satisfaction level among all g_y.

For example, in Figure 5 (a), (g_x, g_1) and (g_x, g_4), $x = 2, 3, 5, 6$, are candidate tuples to create an absolute dominance situation. Among those 8 tuples, rule 3 suggests selecting tuples like (g_x, g_4), because between g_1 and g_4, A_1 consequence on g_4 is the maximum. Among g_2, g_3, g_5, and g_6, the minimum satisfaction level provided by A_1 is on g_3 and g_6. Thus, by applying rule 3, two tuples, (g_3, g_4) and (g_6, g_4), are selected the next swap.



Alternatives	g1	g2	g3	g4	g5	g6
A1	ML	MH	-M	H	L	-M
A2	-H	H	H	L	ML	ML

	g1	g2	g3
P(A1)	H	0	L
P(A2)	M	MH	MH

	g1	g2	g3
P̂ (A1)	H	M	L
P̂ (A2)	M	MH	-L

Fig. 5. Examples of the swap suggestion rules

6.4 Reusing Dominance Situations

Once an alternative is decided as the dominant one, this knowledge might be reusable for deciding between other pairs of alternatives without the need to the Even Swaps process. For example, in the placement case P in Figure 5 (b) $(P(A_1) = \{H, 0, L\}$, $P(A_2) = \{M, MH, MH\})$, let us assume A_1 is decided as the dominant alternative. We can conclude that A_1 is the dominant alternative in the placement case $\widehat{P}$ as well, because since A_1 in P' is definitely stronger than A_1 in P, and A_2 in $\widehat{P}$ is definitely weaker than A_2 in P.

Rule 4, reuse the dominance situation: Assume two placement cases like P and $\widehat{P}$ such as $P(A_i) = \{i_1, i_2, ..., i_m\}$ an $P(A_j) = \{j_1, j_2, ...j_m\}$, $\widehat{P}(A_i) = \{\hat{i_1}, \hat{i_2}, ..\hat{i_m}\}$ and $\widehat{P}(A_j) = \{\hat{j_1}, \hat{j_2}, ...\hat{j_m}\}$. If A_i is preferred to A_j for the case P, then A_i is preferred to A_j for the case $\widehat{P}$ too, iff:
- $\forall i_x \in P(A_i)$ and $\hat{i_x} \in \widehat{P}(A_i)$ $\quad \hat{i_x} \geq i_x$, and
- $\forall j_x \in P(A_j)$ and $\hat{j_x} \in \widehat{P}(A_j)$ $\quad \hat{j_x} \leq j_x$

6.5 The Automatic Even Swaps Suggestion Algorithm

Given a set of goals, $G = \{g_1, g_2, ...g_m\}$ and two alternatives A_i and A_j, the Even Swaps method for determining the optimum solution between A_i and A_j consists of 6 main steps. In the algorithm, $Sat(g_x, A_i) = i_x$ and $Sat(g_x, A_j) = j_x$, the algorithm stores the swaps in a Knowledge Base (KB), and tuples of candidate goals for the next swap are stored in temporary array lists called L, L', L''.

```
While NOT($A_i$ is dominant OR $A_j$ is dominant)
    Step 1: Remove irrelevant goalss
        For all $g_x$ in $G$:
            If $i_x = j_x$ Then remove $g_x, i_x, j_x$ from $G, P(A_i), P(A_j)$
    Step 2: Reuse swaps
        For all $swap$ In $Swaps\ KB$
            If $swap$ is reusable to $P$ Then
                Apply $swap$ to $P$
                Repeat Step 1:
                    Remove irrelevant goalss after swapping
    Step 3: Apply Rule 1, create a dominance situation
        For $x = 1$ To $m$
            If $i_x > i_y$ AND $j_x > j_y$ Then $T$.add($(g_x, g_y)$)
    Step 4: Apply rule 2, find the most reusable swap
        For all $(g_x, g_y)$ in $T$
            $T'$.add(the most reusable $(g_x, g_y)$)
        If NOT exist a reusable $(g_x, g_y)$ in $T$ Then $T' = T$
    Step 5: Apply rule 3, find min and max satisfaction levels
        For all $(g_x, g_y)$ in $T'$
            If $i_x$ is Min AND $i_y$ is Max Then
                $T''$.add($(g_x, g_y)$)
    Step 5: Ask the swap from stakeholders
        $(g_x, g_y)$ = random tuple in $T''$
        in swap $(g_x, g_y, i_x \to i'_x \Leftrightarrow i_y \to i'_y)$ Ask $i'_y$ value
        in $P(A_i)$ $i_y = i'_y$, $i_x = i'_x$
        $SwapsKB$.add$(g_x, g_y, i_x \to i'_x \Leftrightarrow i_y \to i'_y)$
End While
If $A_i$ is dominant OR $A_j$ is dominant
    Step 6: Apply rule 4, reuse the alternative dominance
        For all $Placement\ Cases\ P_x$
            If $A_i, A_j$ dominance in $P$ is reusable in $P_x$ Then
                Apply the dominance to $P_x$
```

To illustrate the algorithm, let us trace the steps the analysis of P_1 and P_2 in Figure 3. In the first cycle of the algorithm, there is no irrelevant goal in $P_1(A_1)$ and $P_1(A_2)$. The swaps KB is empty, so the algorithm needs to suggest a swap to the stakeholders. In step 3, (g_2, g_1) and (g_2, g_3), two candidate goal tuples for creating a dominance are generated. In step 4, the algorithm determines that (g_2, g_1) is the most reusable swap, and in step 5, the swap $(g_2, g_1, 0 \to -MH \Leftrightarrow 0 \to ?)$ is asked from the stakeholders. The swap is stored in the swaps KB, and as a result of increasing g_1 to H and reducing g_2 to $-MH$, both g_2 and

g_1 become irrelevant goals. By removing g_2 and g_1, the final decision for P_1 is made: $A_2 > A_1$. However, the decision about the dominance of A_2 is not reusable for the other three placement cases. When analyzing the placement case P_2, the swap previously made for P_1 can be reused without the need for enquirying the stakeholders, and so A_2 is recognized as the dominant solution for P_2 as well. The chain of swap suggestions and decisions contrinues untill all placement cases are examined. In all four placement cases, $A_2 > A_1$, thus, overall, A_2 is the better solution.

6.6 Discussion

Given m goals and n alternative solutions, $m \times n$ knowledge-intensive and cognitively hard queries need to be asked from stakeholders to elicit the absolute satisfaction level of each goal by every alternative. By applying the proposed method in this work, decision stakeholders need to answer $m \times (n - 1)$ comparison queries which require less cognitive abilities. The number of placement cases enumerated based on these comparison grows in the order of 5^m. When the problem scales, the algorithm may ask numerous swap queries and may return several exceptional patterns of satisfaction level to human experts for the final judgment. However, in reality, stakeholders are only enquiried about the few exceptional patterns of placement cases (not all 5^m of them). The number of swap queries from stakeholdersare are dramatically reduced by reusing the previously asked swaps and reusing the dominance situations (rules 2 and 4).

A major limitation of this method is the possibility of having an alternative that is dominant for half of the placement cases; in such a condition, stakeholders are required to examine numerous exceptional cases (the other half of the placement cases), which might be impossible or more time-consuming than finding the absolute satisfaction level of all goals. Another threat to validity of the method is using the interval scale of $-High$ to $High$ for comparing the alternatives and expecting that stakeholders are able to differentiate the strength of contributions in this interval scale. The way that stakeholders understand the intervals of comparisons may differ from the way in fact we use them in the algorithm.

Although the suggested heuristic algorithm may not always provide a definitive answer, it enables objective trade-off decision making even though detailed numerical data is not available. The heuristic algorithm may ultimately rely on the human judgment, for which experts are asked to judge whether the contribution values match the exceptional patterns. Nevertheless, the proposed method reduces the required input data about the absolute satisfaction levels of goals.

7 Case Study and Comparison with AHP

We applied the proposed method to decide about the trade-offs of switching to the ONE-ITS system from the existing traffic monitoring system at MTO. The MTO expert who collaborated in this study described the goals of his department for employing traffic monitoring systems. In a separate interview session that took one hour long, the MTO expert compared the alternative systems with respect to a number of goals. Figure 6 shows how the alternative solutions affect the goals of the VMS application.

7.1 Applying the Heuristic Method to the ONE-ITS Decision Case

The heuristic method generated 540 different placement cases for the comparisons give in Figure 6. The automated Even Swaps were used to decide on the dominant alternative system for each of the 540 cases. The algorithm asked 8 swap queries from the requirements analyst[4]. For example, the automated swap suggestion algorithm decided that swapping G_3 and G_6 is reusable and useful and accordingly stakeholders were enquired the swap: $(G_3, G_6, MH \rightarrow 0 \Leftrightarrow L \rightarrow x)$. The user believes that G_6 is much more important than G_3, hence, reducing G_3 by 4 levels was swapped with increasing G_6 by 3 levels, from L to MH $(x = MH)$. When the user was enquired about another swap such as $(G_1, G_4, 0 \rightarrow H \Leftrightarrow H \rightarrow x)$, since the user believes that G_1 and G_4 are the most and the least preferred goals, increasing G_1 by 5 levels was swapped with reducing G_4 by 10 levels from $High$ to $-High$ $(x = -H)$.

The algorithm determines *ONE-ITS* as the dominant solution for all 540 possible placement cases, and therefore, without further human judgement, the ONT-ITS (A_1) is suggested to the MTO expert as the optimum solution. The MTO expert stated that ONE-ITS is the better solution in their opinion as well, which provides a support for the correctness of the decision suggested by our method.

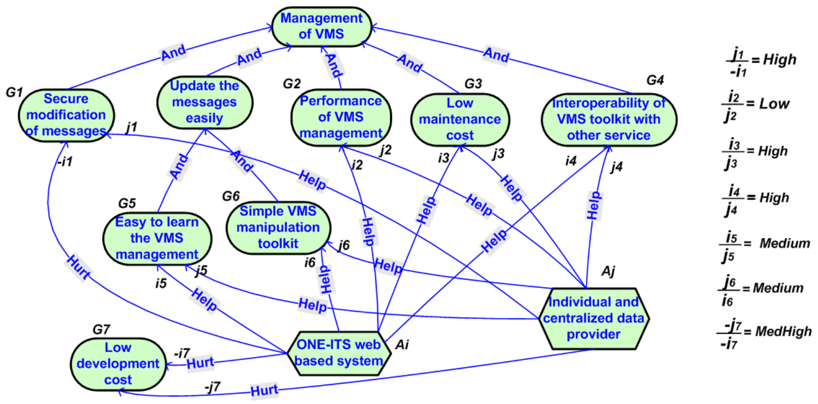


Fig. 6. Comparison of alternative solutions for the VMS sub-system

7.2 Applying AHP to the ONE-ITS Decision Case

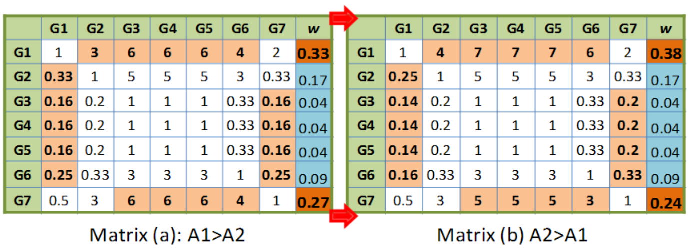
AHP is the most similar existing work to our contribution, hence, we analyze the MTO scenario using AHP. In AHP, the scale of priorities is derived from pair-wise

[4] Due to project constraints, the MTO expert did make the swaps, and a requirements analyst in the project ranked the preferences and made the swaps.

comparison of goals preferences. Paired comparisons are made with judgments using numerical values taken from the AHP absolute fundamental scale of 1-9, where 1 indicates that two elements are of equal value and 9 indicates favoring one element over another with the highest order of affirmation.

In order to make the proposed method comparable with the AHP, we elicited the user's preferences from swaps. The goals preference rankings are as $G_1 > G_7 > G_2 > G_6 > G_5$ and $G_5 = G_4 = G_3$. The pair-wise comparison of preferences over these 7 goals in Figure 6 was done based on the assumptions made about the ranking of their preferences. Figure 7 gives two different paired comparisons of goals made by the same user. The last column in the matrices shows the final importance weight calculated by the Eigen value of the matrices. By using the importance weights of goals in the matrix (a) in Figure 7 and calculting the final utility of A_1 and A_2, A_1 (the ONE-ITS) is proposed as the overall optimum solution.

The pairwise comparisons of goals preferences are slightly modified in Figure 7 (b). By using the importance weights of goals in matrix (b), A_2 (the existing system) is determined as the overall optimum solution. This contradiction (also in contrast with the results of our method) shows that the final decision suggested by the AHP is highly sensitive to the ordinal pairwise comparisons. Besides, by specifying that the preference of G_1 to G_3 is 7 in the AHP ordinal scale (Matrix (b)), the final importance weight for G_1 is calculated 7 times greater than G_3's weight. This means a qualitative description of the comparison in an ordinal scale is transformed to its proportional numerical representation, while it may not reflect the actual intents behind those qualitative comparisons made by the human user; for instance, G_1's importance weight may not necessarily be 7 times greater than G_3's, and by converting the human users description of the comparisons to AHP ordinal scale the comparisons are exaggerated.



	G1	G2	G3	G4	G5	G6	G7	w
G1	1	3	6	6	6	4	2	0.33
G2	0.33	1	5	5	5	3	0.33	0.17
G3	0.16	0.2	1	1	1	0.33	0.16	0.04
G4	0.16	0.2	1	1	1	0.33	0.16	0.04
G5	0.16	0.2	1	1	1	0.33	0.16	0.04
G6	0.25	0.33	3	3	3	1	0.25	0.09
G7	0.5	3	6	6	6	4	1	0.27

Matrix (a): A1>A2

	G1	G2	G3	G4	G5	G6	G7	w
G1	1	4	7	7	7	6	2	0.38
G2	0.25	1	5	5	5	3	0.33	0.17
G3	0.14	0.2	1	1	1	0.33	0.2	0.04
G4	0.14	0.2	1	1	1	0.33	0.2	0.04
G5	0.14	0.2	1	1	1	0.33	0.2	0.04
G6	0.16	0.33	3	3	3	1	0.33	0.09
G7	0.5	3	5	5	5	3	1	0.24

Matrix (b) A2>A1

Fig. 7. Pair-wise comparison of goals preferences using AHP, changes to the ordinal comparisons of AHP are highlighted.

8 Conclusions and Future Work

This work proposes a notation for expressing requirements trade-offs by a simple goal model. We proposed a heuristic decision making algorithm which uses comparison of alternatives' contributions as the basis of trade-off analysis. The stakeholders preferences are incorporated into the decision analysis by using the Even Swaps method enhanced with automated swap suggestions. Although the suggested heuristic algorithm may not always find the optimum solution, it enables objective trade-off decision making even though detailed numerical data are not available. The heuristic algorithm may ultimately rely on human judgment, for which experts are asked to judge whether the contributions strengths match the exceptional patterns. The number of possible placement cases can be reduced by adding numerical measures of contributions to the model, whenever such data is available, specially for factors such as cost and time. This may reduce the number of cases (patterns) that we need to provide to stakeholders and experts for the final judgment.

In future work, we try to reduce the number of placement cases by adding thresholds for critical requirements, so stakeholders would be able to express a threshold of acceptance for critical goals. Further case studies and application of the method in action research and experiments is needed for evaluating the utility and usability of the method and the tool. In an ongoing study, in order to further compare the proposed method with AHP, we are applying the our method to the case studies presented in the existing contributions that use AHP for requirements trade-off analysis [12, 17].

Acknowledgment

Financial support from the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged. The authors thank Roger Browne at Ministry of Transportation, Ontario (MTO) for valuable inputs into the case study. Authors thank Jocelyn Simmonds for her insightful comments on this work.

References

1. I. F. Alexander. Initial industrial experience of misuse cases in trade-off analysis. In *In Proc. of RE'02*, pages 61–70. IEEE Computer Society, 2002.
2. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice.* Second Edition, Addison Wesley, 2003.
3. V. Belton and T. J. Stewart. *Multiple Criteria Decision Analysis: An Integrated Approach.* Springer, 2001.
4. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering.* Kluwer Academic, 1999.
5. G. Elahi and E. Yu. Requirements trade-offs analysis in the absence of quantitative measures: A heuristic method. *SAC'11*, 2011.
6. M. S. Feather, S. L. Cornford, K. A. Hicks, J. D. Kiper, and T. Menzies. A broad, quantitative model for making early requirements decisions. *IEEE Software*, 25:49–56, 2008.
7. J. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys.* Springer Verlag, 2005.
8. P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Eng. Appl. Artif. Intell.*, 18(2):159–171, 2005.
9. J. S. Hammond, R. L. Keeney, and H. Raiffa. *Smart choices : a practical guide to making better life decisions.* Broadway Books, 2002.

10. J. Horkoff and E. Yu. A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In *CAiSE Forum*. CEUR Workshop Proceedings, 2009.
11. S. H. Houmb, J. Jrjens, G. Georg, and R. France. An integrated security verification and security solution trade-off analysis. *In Integrating Security and Software Engineering*, 2006.
12. H. P. In, D. Olson, and T. Rodgers. Multi-criteria preference analysis for systematic requirements negotiation. In *COMPSAC '02*, pages 887–892, 2002.
13. H. Kaiya, H. Horai, and M. Saeki. AGORA: Attributed goal-oriented requirements analysis method. *In RE'02*, 0:13, 2002.
14. J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *IEEE Softw.*, 14(5):67–74, 1997.
15. R. L. Keeney and H. Raiffa. *Decisions with multiple objectives : preferences and value tradeoffs*. Wiley, 1976.
16. E. Letier and A. van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *SIGSOFT '04/FSE-12*, pages 53–62, 2004.
17. W. Ma, L. Liu, H. Xie, H. Zhang, and J. Yin. Preference model driven services selection. In *Proc. of CAiSE'09*, pages 216–230, 2009.
18. J. Mustajoki and R. P. Hämäläinen. A preference programming approach to make the even swaps method even easier. *Decision Analysis*, 2(2):110–123, 2005.
19. J. Mustajoki and R. P. Hämäläinen. Smart-swaps - a decision support system for multicriteria decision analysis with the even swaps method. *Decis. Support Syst.*, 44(1):313–325, 2007.
20. J. Noppen, P. van den Broek, and M. Aksit. Dealing with imprecise quality factors in software design. *SIGSOFT Softw. Eng. Notes*, 30(4):1–6, 2005.
21. T. Saaty. The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 345–408. Springer Verlag, 2005.
22. T. L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. Mcgraw-Hill, 1980.
23. A. van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Trans. Softw. Eng.*, 26(10):978–1005, 2000.
24. J. Yen and W. A. Tiao. A systematic tradeoff analysis for conflicting imprecise requirements. In *RE '97*, page 87, 1997.