A Semi-Automated Tool for Requirements Trade-off Analysis

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. In designing most systems, requirements analysts face many competing requirements, such as performance, usability, costs, and so forth. Ideally, analysts would like to quantitatively measure consequences of solutions on requirements and risks, and extract stakeholders' preferences in terms of numerical weights. However, during the early stages of requirements and system design, it is hard to quantitatively measure all factors on a similar scale and quantify stakeholders' preferences. This contribution proposes a semi-automated decision aid tool which allows the use of available but potentially incomplete quantitative and qualitative requirements and risk measures. It removed the need to elicit importance weights of requirements. Instead, stakeholders are asked how much they would relax the demand on one objective to better achieve another. The proposed tool extends the Even Swap method with formally defined rules for suggesting the next swap to decision stakeholders.

Key words: Requirements trade-offs, qualitative decision analysis, preferences, quantitative data.

1 Introduction

Requirements analysts need to make key decisions early in the project, such as which architectural or design solution to employ [1]. Each alternative solution satisfies different functional and non-functional requirements to varying extents. Selecting a solution among multiple alternatives involves making trade-offs among requirements, with respect to stakeholders' preferences and consequences of alternatives on the requirements. Requirements analysts and project leaders also require objective risk measures to select good-enough countermeasures. In practice, however, quantifying risk factors, estimating probability and damage of risks, and quantifying the mitigating impacts of controls is challenging and error-prone [2].

Related Work: Faced with the typical absence of reliable quantitative data, some Requirements Engineering (RE) techniques, such as i^* [3] and Tropos [4] treat quality goals as soft goals. Goal model evaluation techniques such as [5–7],

enable reasoning about the partial satisfaction of soft goals by propagating qualitative labels such as partially satisfied (✓̇), sufficiently satisfied (✓), partially denied (✗̇), and fully denied (✗). In some other RE approaches, requirements and alternatives are quantified by using ordinal measures or a probabilistic layer for reasoning about partial goal satisfaction [8–11].

Some decision analysis methods evaluate consequences of alternative solutions in terms of precise and meaningful quantitative measures [1, 8, 12]. Some Multi-Criteria Decision Analysis (MCDA) methods such as Analytical Hierarchy Process (AHP) [13] and Even Swaps [14], circumvent the need to measure requirements and consequences of solutions.

Problems: The main problems when making trade-offs among requirements to decide over alternative design solutions are:

1. Manual Prioritization: extracting stakeholders' preferences over multiple criteria in terms of numerical importance weights is error-prone and labor-intensive.
2. Incomparable Scales: aggregating requirements measures in different scales is usually error-prone or not possible.
3. Extensive Data Collection: eliciting required information to make an objective decision usually involves an extensive data collection from stakeholders.
4. Lack of Quantitative Risk Factors: quantitatively measuring the probability and damage of all risks is challenging, if possible at all.
5. Scalability: the decision problem may become complicated and impossible to be analyzed manually due to several requirements and/or alternatives.

Contributions: This paper describes a decision aid tool that addresses above problems. The tool adopts the Even Swaps multi-criteria decision analysis approach [14] to make trade-offs among requirements. The Even Swaps is a recently introduced decision analysis method in management science that consists of a chain of trading one decision criterion for another. These trades are called swapping. Swaps are even, which means stakeholders are asked to hypothetically improve one criterion, and in return, reduce another one proportionally (evenly). The main advantage of this method when dealing with software requirements is that it does not require extracting numerical importance weights and satisfaction level of all requirements on the same scale. Requirements can be evaluated in a mixture of scales and by different measurement methods.

Although the Even Swaps method solves the problem 1, 2, and 3 (mentioned above), it can fail in practice due to scalability issues: when several software requirements and alternative solutions need to be considered, decision stakeholders may not be able to determine the best swap among numerous possibilities [15]. The main contribution of this paper is introducing an algorithm (and tool) that:

- Semi-automates the Even Swaps process, in the sense that the process is still interactive with stakeholders while being performed and controlled by an automated algorithm.
- The tool suggests which requirements to select for the next swap.

2 Motivating Example

We illustrate the use of the tool by analyzing a prototypical Digital Rights Managements (DRM) player [16]. The player gets an encrypted media and given a valid user key, decrypts the media, and decodes the digital content to an analogue audio. The user needs to purchase an activation code for the player, and the player only works if the activation code is valid at the time of using the player.

The DRM player contains two hard-coded credentials: *Valid activation code* and *Player key*. A software cracker can use the DRM player without buying activation code, either through static code analysis to extract the valid code or by tampering the binary code to bypass the license checks. The main protection strategies against *Tampering the binary code* attack are [16]:

1. *Obfuscation:* By obfuscating a program, the code is transformed into a form that is more difficult for an adversary to understand or change than the original code. Obfuscation adds an overhead to the code which causes performance drop downs.
2. *Tamper Proofing:* Tamper proofing algorithms detect that the program has been modified. Once tampering has been detected, a tamper response is executed which usually causes the program to fail.
3. *Distribution with Physical Token:* Physical tokens are hardware-based protections that try to provide a safe environment for data, code and execution. By employing a physical token, the user needs to show possession of a token to use the software.

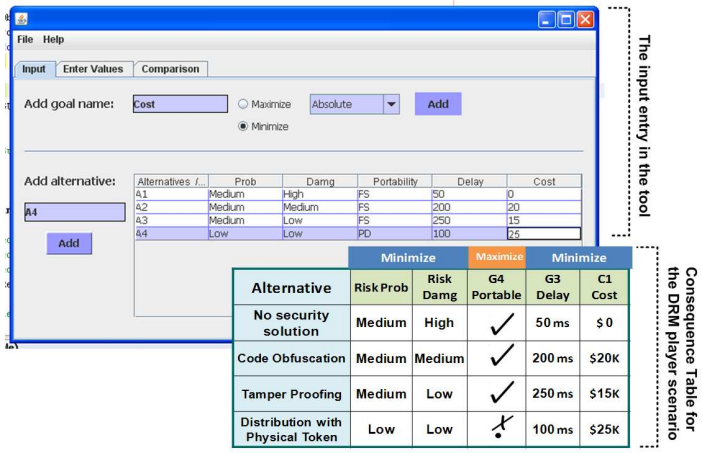

Fig. 1. Consequences of alternative security solutions on the DRM player requirements and the risk of tampering

These alternative security solutions have side-effects on other goals such as portability, delay, and cost. The corresponding consequence table in Figure 1 contains heterogeneous data, i.e., different goals are evaluated in different scales and by different techniques. Some of the criteria are measurable variables that need to be minimized or maximized. For example, stakeholders are able to estimate *Delay* (G_3) in milliseconds based on the properties and specification of

alternatives. However, enough information is not available to quantitatively measure the risk of tampering attack, so consequences of alternatives on the damage and probability of this risk is evaluated in the ordinal scale of Low, Medium Low, Medium, Medium High, and High.

3 Basics of the Even Swaps Method

In an even swap, the decision analyst, collaborating with the stakeholders, hypothetically changes the consequence of an alternative on one requirement, and compensates this change with a preferentially equal change in the satisfaction level of another requirement. Swaps aim to either make criteria irrelevant, in the sense that both alternatives have equal consequences on the criteria, or create a dominant alternative. Alternative A dominates alternative B, if A is better than (or equal to) B on every criteria [15]. Irrelevant goals and dominated alternatives can both be eliminated, and the process continues until only the most preferred alternative remains [15].

Notation Remark. A swap between two goals g_x and g_y that changes the satisfaction value of g_x from x to x' and compensates this change by modifying the satisfaction level of g_y from y to y' is written as:

$$(g_x : x \rightarrow x' \Longleftrightarrow g_y : y \rightarrow y')$$

Case Study: The DRM Player Decision Scenario. We illustrate the Even Swaps method by analyzing and comparing the first two alternatives security solutions for the DRM player (Figure 1):

1. Compare *No security solution* (A_1) and *Code obfuscation* (A_2)
2. Ask stakeholders: If the $RiskDamg$ could be reduced from High to Medium, how much *Delay* they would tolerate instead of 50 ms delay?
   ($RiskDamg : High \rightarrow Medium \Longleftrightarrow Delay : 50ms \rightarrow ?$)
   Stakeholders agree with increasing the Delay to 500 ms.
3. Consequences of A_1 are revised based on the above swap. The revised alternative is called A_1', which is a virtual alternative and subsidence of A_1:
   Consequences of $A_1' = \{Medium, Medium,\ \checkmark,\ 500ms,\ \$0\}$
   Consequences of $A_2 = \{Medium, Medium,\ \checkmark,\ 200ms,\ \$20K\}$
4. $RiskProb$, $RiskDamg$, and Portability are irrelevant decision criteria and can be removed:
   Consequences of $A_1' = \{500ms,\ \$0\}$
   Consequences of $A_2 = \{200ms,\ \$20K\}$
   on $\{Delay, Cost\}$
5. Ask stakeholders: If the Delay could be reduced from 500 ms to 200 ms, what Cost they would pay?
   ($Delay : 250ms \rightarrow 200ms \Longleftrightarrow Cost : \$0 \rightarrow ?$)
   Stakeholders agree to pay \$10K for A_1.
6. Consequences of A_1' are revised based on the above swap.
   Consequences of $A_1' = \{200ms,\ \$10K\}$
   Consequences of $A_2 = \{200ms,\ \$20K\}$
   on $\{Delay, Cost\}$

7. Delay is irrelevant and is removed.
8. With respect to price, A_1' dominates A_2.
9. A_2 is removed from the problem and the process continues by applying the Swap Method to A_1 and A_3.

4 The Automated Even Swaps Tool

This paper introduces a semi-automated Even Swaps tool, that given a consequence table, suggests a chain of swaps to determine the overall best alternative. The algorithm consists of several Even Swaps cycles. In the beginning of each cycle, the algorithm selects a pair of alternatives for the next Even Swaps process. The algorithm suggests a chain of swaps to stakeholders intending to find the preferred alternative in the pair. The dominant alternative is kept in the list of solutions, but the dominated alternative is removed. The algorithm then selects another pair of alternatives to compare and a new cycle starts. These cycles continue until one alternative remains, which is the best solution overall.

4.1 Automatically Suggesting Swaps

The decision aid algorithm has two main goals: 1) minimizing the number of swapping steps required to make one of the alternatives dominant, and 2) generating swaps that are easy to make for stakeholders. Toward these goals, we develop a set of rules in the following sections for suggesting the next swap to stakeholders.

Goal one: Minimizing the number of swapping steps: The tool minimizes the number of swapping steps in 2 ways: 1) suggests swaps that help make one of the alternatives dominant, and 2) reuses previously made swaps to avoid asking repetitive swap queries from stakeholders.

Rule 1: Make swaps that help toward creating a dominated alternative. Swaps make one of the decision criteria irrelevant, which helps remove one goal from the decision problem in each step. Removing criteria one by one is a time-consuming approach to apply the Even Swaps process. The tool suggests swaps that help toward making one of the alternatives dominated. That means if an alternative such as A is dominant for n goals like g_1, g_2, ... g_n, and B is a better solution only for one goal, like g, then we need to remove g by swapping it with one of those n goals. By removing g, solution A might still be dominant with respect to all goals (g_1, g_2, ... g_n). However, swapping any two goals from g_1, g_2, ... g_n will not change the situation between A and B and neither of them becomes dominated with respect to all relevant and remaining goals.

In the Even Swaps process, between the pair of A_1 and A_2 in the DRM player scenario, with respect to $RiskDamg$, A_2 is a better solution, and with respect to $Delay$ and $Cost$, A_1 is a better solution. Note that $RiskPrb$ and $Portability$ are irrelevant. To reduce the number of swaps needed in the next step, $RiskDamg$ must be swapped with either $Delay$ or $Cost$, aiming to remove $RiskDamg$, so A_1 would dominate A_2 with respect to all relevant goals.

Rule 2: Pick the most reusable swaps. When stakeholders make a swap, their input can be reused for another alternative, without further consultation with human stakeholders. This reduces the number of swap queries from stakeholders. A swap from previous steps can be reused in the current step if the goals and their values are identical with the previous swap. Assume stakeholders have made a swap as $(g_x : x \rightarrow x' \Leftrightarrow g_y : y \rightarrow y')$ in a previous cycle. Now to decide between two alternatives such as A and B in a different cycle, this swap is reusable iff:

- Consequences of A on g_x and g_y are equal to x and y
- Consequences of B on g_x is x'

Under these conditions, alternative A can be replaced with A' where consequences of A' on g_x and g_y are revised to be x' and y'. Thus g_x becomes an irrelevant goal and can be removed.

Goal Two: Suggesting Easy Swaps: In addition to considering swaps reusability, the algorithm suggests swaps that decision stakeholders would be willing to make. Hammond et al. [14] suggest making the easiest swaps first, e.g., money is an easy goal to swap. What would make a swap easy for stakeholders? For example, stakeholders may easily agree to improve on a goal that is not sufficiently satisfied and compensate it with decreasing the satisfaction level of a requirement that is highly satisfied, to reach a balance among goals.

In addition, if consequences of two alternatives on the first goal of the swap are close, with an insignificant change, such a goal can become irrelevant. In this way, the goals that do not differentiate alternatives are eliminated from the problem earlier. The tool swaps the first goal with another goal for which consequences of alternatives are highly differentiable, because it would be more probable that the revised virtual alternative after applying the swap is still better than the other alternative.

Rule 3, make the easiest swap: When comparing two alternatives A and B, two goals such as g_1 and g_2 are swapped where the satisfaction level of A on g_1 is minimum compared to any other goal, and the satisfaction level of g_2 is the highest level compared to other goals. We define a distance factor between alternatives on every goal, which aggregates these desired properties into a value. The distance factor on the goal g_x is $\triangle(A, B, g_x)$ and calculated as:

$$\triangle(A, B, g_x) = \frac{|a_x - b_x| + a_x}{max_{g_x}}$$

where a_x and b_x are the consequences of A and B on g_x. max_{g_x} is the maximum satisfaction level of g_x in the consequence table, and by dividing $|a_x - b_x| + a_x$ to the maximum value, distance factors of different alternatives are normalized to values in the scale of 0 to 2.

For example, the max of G_4 in Figure 1 is ✓ (fully satisfied). For the goals that need to minimized, the lower their value is, the higher the satisfaction level would be. Thus, if the consequence of A on g_x is a_x, then a_x is replaced with $max_{g_x} - a_x$. The tool revises the satisfaction level of goals in the consequence table in Figure 1 are as:

	RiskProb	*RiskDamg*	G_4	G_3	c_1
A_1	Medium Low	0	✓	200 ms	$25 K
A_2	Medium Low	Medium Low	✓	50 ms	$5 K
A_3	Medium Low	Medium High	✓	0 ms	$10 K
A_4	Medium High	Medium High	✗	150 ms	$0 K
max_g	Medium High	Medium High	✓	200 ms	$25 K

These modifications to the consequence table are only used for calculating the distance factor. Figure 2 shows the distance factors of A_1 and A_2 on the goals. (For calculations, the interval scale of Low to High is mapped to the interval values of 1 to 5, and ✓, ✓, ✗, and ✗ are mapped to 3, 2, ,1, 0.)

Alternative	Risk Prob	Risk Damg	G4 Portable	G3 Delay	C1 Cost
No security solution	2	0	3	200 ms	$25k
Code Obfuscation	2	2	3	50 ms	$5K
Distance factors of A1 and A2	$\frac{\lvert 2-2\rvert +2}{4}$ =0.5	$\frac{\lvert 0-2\rvert +0}{4}$ =0.5	$\frac{\lvert 3-3\rvert +3}{3}$ =1	$\frac{\lvert 200-50\rvert +200}{200}$ =1.75	$\frac{\lvert 25-5\rvert +25}{25}$ =1.8

Fig. 2. Distance factors of alternatives A_1 and A_2 on DRM player goals

By applying rule 1, we concluded that *RiskDamg* must be swapped with either *Delay* or *Cost*. Based on rule 3, the lowest distance factor (*RiskDamg*) must be swapped with the highest distance factor (*Cost*).

Rule 4, Swap goals with tangible scales: Tangible goals that are measured in absolute values, such as costs or delays, are easier to trade [14]. The final rule for suggesting the next swap is selecting goals that are measured in more granular and tangible scales, because dealing with tangible factors is easier for stakeholders. For example, costs in terms of money is more tangible than the risk level expressed as Medium, Low, High. Therefore, the tool prefers goals that are measured in absolute values to goals measured by percentages, percentages are preferred to ordinal values, and ordinal values are preferred to qualitative labels.

4.2 The Automated Swaps Suggestion Tool

The automated Even Swaps tool takes a set of goals and a consequence table, and in each round of the Even Swaps method, identifies a list of potential goals to be swapped next. The list is first generated by applying rule 1. Then the list is trimmed by only keeping the most reusable swaps (rule 2). To find the best swap, the tool applies rule 3, and if still there are more than one possible swap for the next step, rule 4 is applied. A demo of the proposed tool in this paper is available at [17]. The core features of the tool and a graphical user interface is developed (in Java), and further improvements and tests are undergoing.

Given m goals, in each round, at most $m-1$ swaps are made, and for n alternatives, at most $n-1$ rounds of Even Swaps are needed. Thus, in the worst case scenario, with $m-1 \times n-1$ swaps the best solution is identified. By reusing swaps and applying rule 1, we aim to minimize this number.

5 Conclusions and Limitations

In this work, we adopt and enhance the Even Swaps [14] method for analyzing trade-offs among requirements when multiple alternative design solutions satisfy different requirements to some extent. The main contribution of this tool is applying a set of rules for suggesting next swaps to the decision stakeholders. The algorithm and prototype tool are able to handle different types of input data: absolute and ordinal values in different scales.

A threat to practicality of the tool is the diversity of evaluation scales in the consequence table. Stakeholders may not be able to swap a goal measured in absolute values with a goal that is evaluated by qualitative labels such as ✓ and ✗.

References

1. 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*, vol. 25, pp. 49–56, 2008.
2. *Security Risk Management Guide*, Microsoft Corporation: Microsoft Solutions for Security and Compliance and Microsoft Security Center of Excellence, 2006.
3. E. Yu, "Modeling Strategic Relationships for Process Reengineering," Ph.D. dissertation, University of Toronto, 1995.
4. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal reasoning techniques for goal models," *Journal of Data Semantics*, vol. 1, pp. 1–20, 2003.
5. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Kluwer Academic, 1999.
6. J. Horkoff and E. Yu, "A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models," in *CAiSE Forum*, 2009.
7. P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the tropos methodology," *Eng. Appl. Artif. Intell.*, vol. 18, no. 2, pp. 159–171, 2005.
8. W. Ma, L. Liu, H. Xie, H. Zhang, and J. Yin, "Preference model driven services selection," in *Proc. of CAiSE'09*, 2009, pp. 216–230.
9. P. Giorgini, G. Manson, and H. Mouratidis, "On security requirements analysis for multi-agent systems." in *SELMAS'03*, 2003.
10. Y. Asnar and P. Giorgini, "Modelling risk and identifying countermeasure in organizations," 2006, pp. 55–66.
11. E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *SIGSOFT '04/FSE-12*, 2004, pp. 53–62.
12. H. P. In, D. Olson, and T. Rodgers, "Multi-criteria preference analysis for systematic requirements negotiation," in *Proc. of the COMPSAC'02*, ser. COMPSAC '02, 2002, pp. 887–892.
13. T. Saaty, *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. New york: McGraw-Hill, 1980.
14. J. S. Hammond, R. L. Keeney, and H. Raiffa, *Smart choices : a practical guide to making better life decisions*. Broadway Books, 2002.
15. J. Mustajoki and R. P. Hämäläinen, "Smart-swaps - a decision support system for multicriteria decision analysis with the even swaps method," *Decis. Support Syst.*, vol. 44, no. 1, pp. 313–325, 2007.
16. C. Collberg and J. Nagra, *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Professional, 2009.
17. (2011) Decision analysis tool demo, release 1, dcs, univ. of toronto, available at http://www.cs.toronto.edu/~gelahi/Release1/Release1.html.