Requirements Eng (2007) 12:127–143
DOI 10.1007/s00766-007-0041-5

ORIGINAL RESEARCH

MiHOS: an approach to support handling the mismatches between system requirements and COTS products

Abdallah Mohamed · Guenther Ruhe · Armin Eberlein

Received: 28 September 2006 / Accepted: 2 January 2007 / Published online: 27 February 2007

Abstract In the process of selecting Commercial Off-The-Shelf (COTS) products, it is inevitable to encounter mismatches between system requirements and COTS products. These mismatches occur as a result of an excess or shortage of the COTS attributes. This paper proposes a decision support approach, called MiHOS (Mismatch Handling for COTS Selection), that aims at addressing COTS mismatches during and after the selection process. MiHOS can be integrated with existing COTS selection methods at two points: (1) *For evaluating COTS candidates*: MiHOS estimates the anticipated fitness of the candidates if their mismatches are resolved. This helps to base our COTS selection decisions on the fitness that the candidates will eventually have if selected. (2) *Mismatch resolution after selecting a COTS product*: MiHOS suggests alternative plans for resolving the most appropriate mismatches using suitable actions, such that the most important risk, technical, and resource constraints are met. A case-study is used to illustrate MiHOS and to discuss its added value.

Keywords COTS versus requirements mismatch · Decision support · COTS selection

A. Mohamed (✉) · G. Ruhe
University of Calgary, 2500 University Drive, NW, Calgary, AB T2N1N4, Canada
e-mail: asamoham@ucalgary.ca

G. Ruhe
e-mail: ruhe@ucalgary.ca

A. Eberlein
American University of Sharjah, Sharjah, P.O. Box 26666, UAE
e-mail: eberlein@ucalgary.ca

1 Introduction

1.1 Background

In the last decade, Commercial Off-The-Shelf (COTS) software products have received a lot of industry attention due to their potential for time and effort saving when building software systems [1]. COTS refer to software products that are developed by a third party for the general public for the purpose of integration into a larger system [2]. Open source software (OSS) has also been recently considered as COTS products due to the fact that both OSS and COTS are treated the same way in practical applications [3]. In this paper we use the term COTS to refer to both OSS and COTS.

Several challenges are encountered when using COTS products such as the lack of control over COTS development and how it meets our specific requirements [4, 5], the incompleteness and uncertainty of information, and the large number of evaluation criteria and constraints [6]. In order to cope with such challenges, suitable methods and techniques should be used to evaluate existing COTS products, and then select the most appropriate COTS. Many proposals were made in literature to model the COTS selection process, (e.g., [7–10]). However, most of the existing COTS selection methods fail to deal properly with the situation in which many mismatches are encountered between the stakeholders' requirements and the COTS candidates[1]. This problem, which is the focus of this paper, is described in more details next.

[1] Some efforts were made to support the alignment of COTS functionality and customer requirements in [11, 12] which focus on COTS products named ERP (Enterprise Resource Planning) systems.

1.2 Problem description and motivation

With the increasing proliferation and diversity of COTS products, searching for COTS products based on some requirements would result in identifying many candidates that satisfy the requirements with different degrees; yet, probably none of these candidates would perfectly match the requirements [13]. These mismatches, which appear between COTS attributes[2] and the requirements, occur as a result of an excess or shortage of COTS attributes. Such mismatches are inevitable because COTS products are made for broad use while stakeholders' requirements are specific to their context [5, 14].

The so called buy-and-adapt approach [15] is usually employed when encountering many COTS mismatches. In buy-and-adapt approach, the COTS product with the highest fitness with the stakeholders' requirements is selected. After selecting this COTS, as many mismatches as possible are resolved in order to improve its fitness with the requirements.

We argue here that in such cases and during the COT selection process, the COTS candidates should be evaluated, (i.e., compared) based on their anticipated fitness if their mismatches are resolved. However, there are usually limited resources, (e.g., *effort* and *budget*) for resolving these mismatches. This means we typically can only resolve a subset of all COTS mismatches. Resolving different subsets of mismatches will have different impact on the overall COTS fitness. The question arises: what is the "right" subset of mismatches that should be resolved under the given resource constraints? This right subset has to be roughly identified when comparing different COTS candidates in order to estimate their anticipated fitness, and thus select the best-fit COTS product in a more efficient way.

In addition, it is not enough to only identify the "right" mismatches, but also to choose the "right" resolution actions. Typically, alternative resolution actions can be used to resolve each mismatch. Different resolution actions require different amounts of resources, (i.e., *effort* and *budget*), and impose different *risks* on the system under development. Therefore, the "right" subset of resolution actions should be chosen such that the "right" mismatches are resolved, with the least risk, and within the given resource constraints.

This problem gets more complex as the number of mismatches and their alternative resolution actions increase. Consequently, this increases the difficulty of making appropriate decisions for: (i) selecting the best-fit COTS product, and (ii) resolving the "right" mismatches of the selected product using the "right" resolution actions. Inefficient decisions not only affect the functionality and quality of the system, but might also result in serious consequences, e.g., losing market share.

This paper tries to address the COTS mismatch problem by providing support to human decision makers during COTS selection. The selection is done based on pro-active analysis of the impact of *mismatch handling*. We define COTS mismatch handling as:

> The process in which mismatches and their resolution actions are analyzed; and a selected set of mismatches is resolved using certain resolution actions.

For that, the paper proposes an approach called MiHOS (Mismatch-Handling for COTS Selection). MiHOS can be integrated into existing COTS selection methods to support handling COTS mismatches during and after the COTS selection process. MiHOS has two major uses:

(1) MiHOS is used when comparing the COTS candidates, (i.e., during COTS evaluation) when making decisions to select a COTS. The objective of MiHOS here is to estimate the anticipated fitness of the COTS candidates if the 'right' mismatches for each candidate are resolved. This use of MiHOS is referred to as *MiHOS.Evaluation*.
(2) MiHOS is used after selecting a COTS product when resolving its mismatches. MiHOS is used to support planning the resolution of the 'right' mismatches using the 'right' set of resolution actions (for the selected COTS). This use is referred to as *MiHOS.Planning*.

MiHOS follows an iterative and evolutionary decision support framework called EVOLVE* [16]. Instead of providing only one solution to decision makers, MiHOS suggests a portfolio of qualified solutions. The decision makers are then invited to explore and analyze these solutions, and then either accept one solution, or refine the problem model and regenerate further refined solutions. This kind of *interactive decision support* allows decision makers to have more participation in and control over the decision making process. It is worth mentioning that MiHOS is not intended to address mismatches related to quality requirements. This is, however, discussed in more details in Sect. 6.

1.3 Paper structure

This paper consists of seven sections. In Sect. 2, an overview of the COTS selection and mismatch

[2] The term "COTS attributes" is used to refer to COTS functionalities and qualifications. These attributes are defined based on COTS vendors' documentations and based on our understanding for the domain.

handling problems is given. Sect. 3 proposes a formal model of the problem: COTS mismatch handling. In Sect. 4, the proposed approach is described. A case study illustrates the approach in Sect. 5. Next, Sect. 6 describes the limitations of the proposed approach. Finally, we provide conclusions and suggestions for future research in Sect. 7.

2 COTS selection and mismatch handling in a nutshell

2.1 Goal-driven COTS selection

Several efforts have been made during the last decade to model the COTS selection process, e.g., OTSO [7], PORE [17], CAP [10], and IQMC [18]. There is no commonly accepted method for COTS selection [6]. Nevertheless, all COTS selection methods agree on some key steps that can be iterative and overlapping; these key steps formulate a General COTS Selection (GCS) process which is described as follows (Fig. 1):

Step 1: Define evaluation criteria based on system requirements.
Step 2: Search for COTS products.
Step 3: Filter search results based on a set of 'must-have' requirements in order to define a short list of COTS candidates to be evaluated in more details.
Step 4: Evaluate COTS candidates, (i.e., in the short list)
Step 5: Analyze the evaluation data, (i.e., the output of Step 4) and select the COTS product that has the best fitness with the requirements.

Note that after Step 5, an additional step is usually performed to resolve some of the mismatches within the selected COTS product, i.e., to customize the COTS product as needed.

COTS selection is often done based on a simple weighing and aggregation method. Each criterion is given a weight to represent its importance, and COTS candidates are evaluated (scored) against the criteria. Scores are then multiplied by the weights, and the weighted scores are summed.

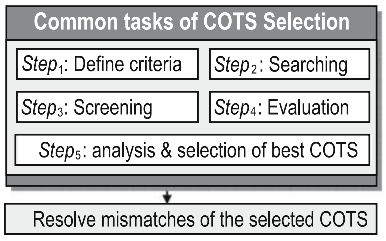

Common tasks of COTS Selection
Step₁: Define criteria
Step₂: Searching
Step₃: Screening
Step₄: Evaluation
Step₅: analysis & selection of best COTS
Resolve mismatches of the selected COTS

Fig. 1 The general COTS selection (GCS) process

We suggest the use of a goal-driven approach to define hierarchical evaluation criteria. A goal is [19] "*an objective that the system under consideration should achieve. They may be formulated at different levels of abstraction, ranging from high-level, strategic concerns to low-level, technical concerns*." Based on this, we define two types of goals: strategic goals and technical goals. Strategic goals refer to high level requirements that cannot be directly measured in a COTS candidate, i.e., by measuring the performance of one functional attribute of the COTS. For example, a strategic goal might be "to have secure communications". Each strategic goal is decomposed into more refined subgoals [20, 21]. The decomposition process continues until defining goals that are at the same level of granularity of COTS features. We refer to the goals at this level as technical goals which represent measurable criteria; for example, "to support SSL protocol". More examples of strategic goals decomposed into technical goals will be given later in Sect. 5.1.

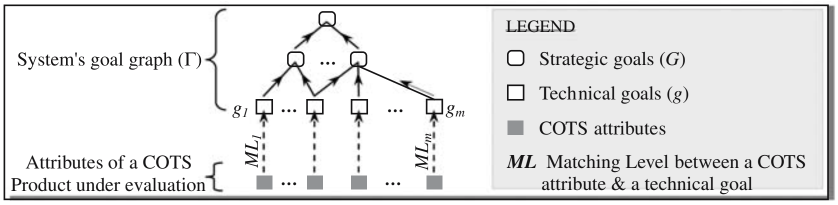
The decomposition process should be done in parallel with COTS exploration and evaluation. The reason for this is that an analyst should be aware of existing COTS capabilities and constraints so as to define a goal graph that addresses stakeholders' strategic needs at the high-levels, and in the same time maps to COTS features at the technical-goal levels, i.e., could be used to evaluate COTS products based on one-to-one comparison between technical-goals and COTS features.

Formally, the decomposition results in a hierarchical goal graph $\Gamma = (V, A)$, where V is divided into two types of nodes: strategic-goals (G) and technical-goals (g), and A is a set of arcs connecting these nodes (Fig. 2). An arc connecting two goals indicates that the child goal at the arc's tail is contributing to achieving the parent goal at the arc's head. The number of hierarchical levels depends on the system, but should be kept small to reduce complexity, (e.g., 3–4 levels).

Using the hierarchical criteria definition, the weights are defined on each arc in the goal graph Γ to represent the importance of child-goals to achieving their parent-goals. The relative weight (as described in Sect. 3.1.4) can be then estimated for each technical-goal.

The same weighing and aggregation method described in the GCS process applies here, but with small differences: (i) the relative-weights of technical-goals are used instead of the simple weights in the GCS process, and (ii) the COTS products are evaluated (*scored*) against the technical-goals. Each score represents a matching level (ML) between a technical-goal and a COTS attribute (Fig. 2). The value of ML $\in [0,1]$

Fig. 2 Hierarchical definition for COTS evaluation criteria


System's goal graph (Γ)
g_1
g_m
ML_1
ML_m
Attributes of a COTS Product under evaluation
LEGEND
Strategic goals (G)
Technical goals (g)
COTS attributes
ML Matching Level between a COTS attribute & a technical goal

indicates different levels of satisfaction of the technical-goals by the corresponding COTS attributes; 1 indicates full satisfaction and 0 no satisfaction. When a mismatch occurs, the ML value is less than 1. The value of ML is estimated using several methods and on different scales, but eventually any estimation has to be normalized to the range [0,1][3].

It is worth mentioning that there exist some goal-driven approaches and tools for COTS selection, e.g., DesCOTS [22] which is a tool that relies on IQMC [18]. DesCOTS uses the i* framework [23] to model goals based on the ISO9126 [24] quality model. The user is then asked to define the selection requirements based on the developed goal graph, and the weighted score of these requirements is used to support the selection. MiHOS on the other hand assumes the use of the approach described in [20, 21] to define strategic and technical goals. COTS products are evaluated against technical goals, and the scores of selected strategic goals are estimated using weighted score method described in the previous paragraph. The benefit of using such approach is two-folds:

(1) It allows decision makers at Step 5 of the GCS process to analyze the scores of different strategic-goals instead of having only total scores of COTS candidates. Total scores tend to mask individual criteria that may present weaknesses or strengths; i.e., a high score in one criterion hides a low score in another.
(2) It allows more flexibility for the interactive decision support. Decision makers can easily explore different scenarios with certain strategic-goals, e.g., "*During COTS selection, what is the best COTS product if all mismatches related to the "security strategic goal" are resolved?* " This is illustrated in more details in our case-study in Sect. 5.

2.2 COTS mismatch handling

Several efforts for analyzing the COTS mismatch problem have been reported. For example, Alves et al. [20] studied conflicts that arise from different types of mismatches. Mohamed et al. [25] introduced a classification of possible mismatch types, (i.e., problem space) as well as a classification of possible resolution actions, (i.e., solution space).

In short, a mismatch occurs whenever a COTS attribute does not exactly match the corresponding technical goal. Typically, such mismatches can be classified into one of five types: (i) "*ZeroMatch*" which indicates that a COTS product fails to satisfy a technical goal, (ii) "*PartialMatch*" which indicates that a COTS partially satisfies a technical goal, (iii) "*Surplus*" which indicates that a COTS has extra functionality that is not required, (iv) "*Overmatch*" which indicates that a COTS functionality exhibits more capability than required, and finally (v) "*Equivalence*" which indicates that a COTS functionality contributes to achieving the strategic goal behind a required technical goal, but does not match the technical goal itself . A detailed description of all mismatch types is outside the scope of this paper and can be found in [25].

Practically, COTS mismatches can be resolved using different ways:

(i) *Adapting our requirements*. That is, to revise our requirements and thus make a compromise that reduces the mismatch [26]
(ii) *Tailoring or extending the COTS product*. This technique is used to fix mismatches related to COTS functionality. As stated by Vigder et al. [15], this can be realized by:

- *Add-ons*: to acquire additional add-ons that add functionality to the COTS product.
- *Scripting*: to write custom code using a scripting language supported by the COTS product. Examples of such scripting languages include JavaScript, VisualBasic, Perl.
- *API*: to develop a controlling program that calls COTS functions using its API interface.
- *Modifying source*: to modify the COTS source code if available. This is a risky action because of several maintenance issues [1, 2, 27]

In order to decide which mismatches are to be resolved using which actions, several factors should be considered: (a) the impact of different mismatches on

[3] More details on ML estimation are documented in a technical report that is available upon request.

the COTS-fitness, and (b) the effort, cost, and risk of alternative resolution actions for each mismatch. In addition, the application of the selected set of resolution actions should not exceed available resources.

MiHOS.Evaluation and MiHOS.Planning take into account the above factors when being integrated into the GCS process. Before describing MiHOS approach, a formal model of the problem (which is the core of MiHOS) is described in the next section.

3 Mismatch handling: a formal model

The core of our approach, MiHOS, is a formal model of the mismatch handling problem. We give a description of this formal model in this section. Next, MiHOS is described in details in Sect. 4.

3.1 Problem facets

3.1.1 Decision variables

Suppose we have a COTS product that has a set of mismatches $M = \{m_i: i = 1 \text{ to } I\}$. For each mismatch m_i, assume we identified a set A_i of J possible resolution actions, $(\forall m_i)$: $A_i = \{a_{ij}: j = 1 \text{to } J\}$. The goal is to select one resolution action a_{ij} (at most) for each mismatch m_i. This is described by the decision variables, $(\forall m_i)$: $\{x_{ij}: j = 1 \text{to } J\}$, where $x_{ij} = 1$ if the resolution action a_{ij} is chosen to resolve the mismatch m_i, and $x_{ij} = 0$ if a_{ij} is not chosen.

For the set of resolution-actions A_i applicable to a mismatch m_i, we define each resolution action $a_{ij} \in A_i$ to be sufficient to completely resolve m_i. Therefore, we are constrained by choosing only one resolution action to resolve each m_i, This constraint is represented by, $(\forall m_i)$: $\sum_j x_{ij} \le 1$. On the other hand, a mismatch m_i is tolerated if no resolution action is chosen, i.e., $(\forall_{j=1 \text{ to } J})\ x_{ij} = 0$.

3.1.2 Mismatch amount

Based on the discussion of the ML in Sect. 2, whenever $\text{ML}_i < 1$ then we have a mismatch m_i. We estimate the mismatch amount for each m_i as

$$(\forall m_i) : Amount_i = 1 - ML_i \tag{1}$$

3.1.3 Technical risk

We assume applying each resolution action a_{ij} imposes a technical risk equal to r_{ij}. This technical risk can be refined into *failure risk* and *instability risk*. The *failure risk* measures the risk that the developer might fail to resolve a mismatch. For example, using "scripting" to resolve a mismatch might fail if developers do not have enough expertise with the scripting language to be used. The *instability risk* refers to the probability that a resolution action would cause instability of the existing system, i.e., the system to which the COTS is integrated. For example, using a beta version of an "add-on" to resolve a mismatch might cause instability of the final system. Both *failure risk* and *instability risk* are estimated by experts on a nine-point scale, see Table 1.

Assuming equal distance between the above nine-point scale, the technical risk of a resolution action can be estimated from the formula,

$$r_{ij} = \alpha \cdot failure\_risk_{ij} + \beta \cdot instability\_risk_{ij} \tag{2}$$

where $\alpha + \beta = 1$; α and β indicate the relative importance of the two types of risks, and are defined by experts. In critical systems, the risk of causing system instability might be more critical than failing to resolve mismatches. In this case, β should be set to a value higher than α. In our work, we see both risks of equal importance, and thus we use $\alpha = \beta = 0.5$.

3.1.4 Relative importance of technical-goals

Consider a mismatch m_i exists between a COTS attribute and a technical-goal g_i. This technical-goal is at the leaf level of a hierarchical graph of Y higher-level strategic-goals $G_{i,y}$:$y = 1$ to Y. Now, assume g_i is assigned a weight of importance υ_i;and $G_{i,y}$ is assigned a weight of importance equals to ω_{iy}. Then for a uniquely defined path $\rho[g_i, G_{i,Y}]$ between the nodes $g_i$and $G_{i,Y}$, the relative importance Ω_i of the node g_i with respect to $G_{i,Y}$ is obtained by multiplying all weights of the arcs along ρ, (see Fig. 3):

$$\boldsymbol{\Omega}_i = \boldsymbol{\upsilon}_i \cdot \prod_{y=1\ldots(Y-1)} \omega_{iy}. \tag{3}$$

Table 1 Meaning of values of resolution actions' *failure risk* and *instability risk*

Value	Meaning
Failure risk	
1	*Extremely low*
3	*Very low*
5	*High*
7	*Very high*
9	*Extremely high*
Instability risk	
1	*Extremely low*
3	*Very low*
5	*High*
7	*Very high*
9	*Extremely high*

Even numbers can be interpreted as a refinement of values above and below

In case there is more than one path from a technical-goal to the highest-level strategic-goal, then the path resulting in the maximum relative importance is considered.

3.1.5 Stakeholders

In many cases, several stakeholders are involved in the estimation process of "*the goals' relative importance*" and "*the technical risk of resolution actions*". Assume we have a set S of K stakeholders abbreviated by $S = \{s_k\colon\ k = 1 \text{to } K\}$. The degree of importance of each stakeholder s_k is denoted by the relative importance $\lambda_k \in\{1,3,5,7,9\}$; this indicates very low, low, medium, high, and very high importance, respectively. Even numbers, (e.g., λ_k = 2, 4, 6, 8) indicate a refinement of values above and below. We use a nine-point ordinal scale to allow sufficient differentiation between stakeholders' degrees of importance.

For aggregating different stakeholders' estimations, we use a weighted average function. For each stakeholder s_k:

(i) If s_k has estimated a relative importance Ω_{ik} for a goal g_i, then the resulting relative importance is

$$\boldsymbol{\Omega}_i = \sum_k \lambda_k \cdot \boldsymbol{\Omega}_{ik} \tag{4}$$

(ii) If s_k has estimated a technical risk r_{ijk} for a resolution action a_{ij}, then the resultant technical risk is determined by

$$\mathbf{r}_{ij} = \sum_k \lambda_k \cdot \mathbf{r}_{ijk} \tag{5}$$

3.1.6 Constraints

3.1.6.1 Resource constraints Different resources are required for applying resolution actions. Each resource has a maximum capacity that should not be exceeded. For this paper, we assume two types of resources related to effort and to cost. The principal steps of the MiHOS method will remain the same if further types of resources would be added. For the two resource types: budget and effort, we introduce Cap_1 for the budget available and Cap_2 for the total effort available. We assume that every mismatch resolution action a_{ij} uses an amount $Usage_{ijt}$ of resource t. Thus, the total amount of resources consumed by the selected set of resolution actions (represented by decision variables x_{ij}) must be less than the total capacity of these resources,

$$\sum_{ij} x_{ij} \cdot Usage_{ijt} \le \text{Cap}_t, \quad t = 1, 2 \tag{6}$$

3.1.6.2 Pre-assignments The user has an option to pre-assign a certain mismatch to be either "resolved" or "tolerated" before running the model. If m_i is pre-assigned to be resolved, then the model must select exactly one resolution action to apply. This means, the sum of decision variables related to all resolution actions A_i applicable to m_i must equal to 1, (i.e., $\sum_j x_{ij}$ = 1). On the other hand, if a mismatch m_i is to be tolerated, then no resolution actions should be suggested for it, (i.e., $\sum_j x_{ij}$ = 0).

3.2 Objective function

The objective function brings together the problem facets described so far in Sect. 4. Typically, planning for mismatch handling aims at: (a) maximizing the resultant COTS fitness, and (b) using resolution actions with minimum risk. Point (a) is influenced by two aspects: goals relative importance Ω_i and mismatches amount $Amount_i$, while point (b) is influenced by only one aspect: technical risk r_{ij} of selected resolution actions. The proposed objective function brings these aspects together in a balanced way. The objective is to maximize function *F(x)* subject to the satisfaction of the above constraints. *F(x)* is given as:

$$F(x) = \sum_i [Amount_i \cdot \boldsymbol{\Omega}_i \cdot \sum_j (x_{ij}\, \Delta r_{ij})] \tag{7}$$

where $\Delta r_{ij} = 10 - r_{ij}$ indicates how safe an action a_{ij} is. Similar to r_{ij}, Δr_{ij} is also based on an ordinal scale, but is handled as ratio scale. We use Δr_{ij} instead of r_{ij} because maximizing *F(x)* should yield the minimum risk, (i.e., the maximum safety) of selected actions.

4 MiHOS: an approach to handle COTS mismatches

MiHOS is a three-phase approach that allows decision makers to interactively participate in and have more

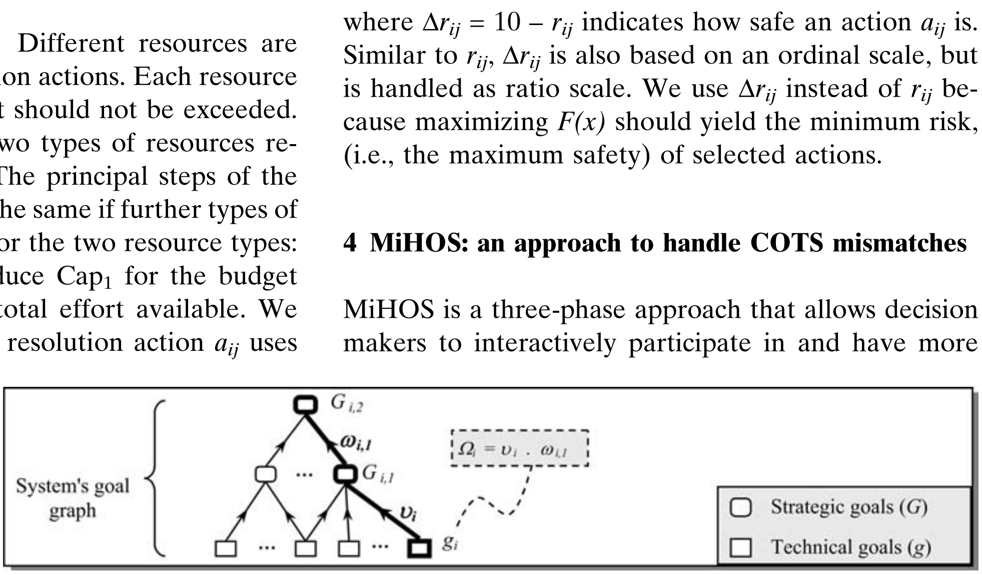


Fig. 3 Relative importance of technical-goals

control over the decision making process. MiHOS adopts the concept of providing a portfolio of *qualified solutions* with maximum *diversification;* this concept was originally presented in [16]. In the following, we explain this concept and then describe the main phases of MiHOS.

4.1 Qualified solutions

The formal model described in Sect. 3 constitutes an optimization problem. A qualified solution to this problem is any solution that is of a high enough level of quality to solve the right mismatches using the right resolution actions. To identify these qualified solutions, we initially rely on optimization algorithms.

The formalized problem in Sect. 3 can be solved by the optimization package LINDO [28]. Assume that solving such problem would yield an optimum solution X_0 with the maximum possible objective function value equal to $F(X_0)$. Based on this, we define a qualified solution as any solution X^* that possesses two characteristics:

(1) X^* lies in the feasible space which is delimited by the problem constraints described in Sect. 3.1.6.
(2) $F(X^*) \geq \alpha \cdot F(X_0)$, where α is a predefined quality level, and $\alpha \in (0,1]$.

In our approach, we choose the quality level α = 0.95 in order to address the uncertainty involved when defining the values of problem parameters, e.g., 'mismatch amount' and 'technical risk'. Due to inherent uncertainty, it would make little sense to require higher accuracy.

4.2 Qualified solutions with maximum diversification

Diversification of qualified solution is a technique that provides more efficient support to decision makers. It provides a portfolio of qualified solutions that are structurally diversified. The human experts can then review different solution alternatives which have a guaranteed level of quality. It is assumed that following such an approach would be more appropriate for supporting human decisions than providing only one mathematically optimal solution that, however, might not match the actual problem needs [16]. Providing a portfolio of qualified and diversified solutions would give the decision makers more participation in and control over the decision making process.

The necessity of diversification stems from the fact that if the solutions are not significantly different, they are not really alternatives. Thus, to make best use of the diversification notion, we need to provide a set of qualified solutions that have the maximum diversification. The strategy used in MiHOS to get this set is described in "Appendix 1".

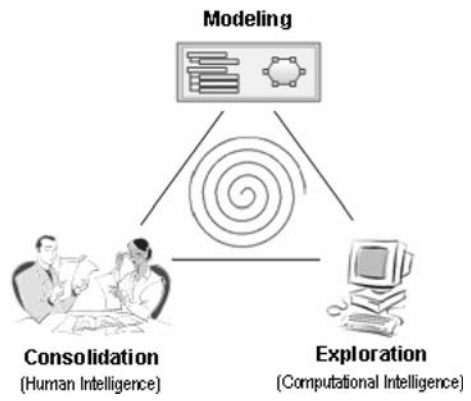


Fig. 4 Hybrid intelligence to support planning for mismatch handling

4.3 A hybrid approach for mismatch-handling during COTS selection

MiHOS follows a framework called EVOLVE* [16] that provides decision support using hybrid intelligence that brings computational and human intelligence together. The computational part of MiHOS is used to solve the formal-model of the problem and to provide a set of qualified solutions with maximum diversification to human decision makers. Then, the decision makers, (i.e., human intelligence) are invited to analyze these solutions and either accept one of them, or refine the formal model based on their knowledge that evolves through successive iterations. Such iterative and evolutionary approach allows addressing the wicked and uncertain character of the problem.

Overall, MiHOS includes several tasks performed during three phases: (1) Modeling, (2) Exploration, and (3) Consolidation (Fig. 4). Before describing these three phases, we want to give the big picture of how MiHOS is integrated into the COTS selection process. MiHOS has two main uses: *MiHOS.Evaluation* and *MiHOS.Planning* (refer to Sect. 1.2). As seen in Fig. 5, the output of MiHOS is a set of alternative resolution plans for handling COTS mismatches. These plans are used at two points during COTS selection: (1) when selecting the best COTS product from the most promising candidate, (i.e., *MiHOS.Evaluation*), and (2) after selecting a COTS product, (i.e., *MiHOS.Planning*).

4.3.1 Modeling

The modeling phase aims at describing the settings of the mismatch problem to be suitable for the format given in Sect. 4. This includes three main tasks:

(1) *Identify mismatches and resolution actions*. For each COTS candidate, identify:

a. The set of mismatches m_i and the applicable resolution actions A_i.
b. The technical risk r_{ij} associated with each resolution action a_{ij}.
c. The resources required for each resolution action a_{ij}. In our approach, we consider two types of resources: ‘‘*Effort*$_{ij}$’’ which refers to the person-hours needed to apply a_{ij}, and ‘‘*Cost*$_{ij}$’’ which refers to any financial expenses required for applying a_{ij}. Estimating the effort and cost using a combination of expert judgment with formal models, (e.g., [29]) has proven promising [30, 31].

(2) *Identify relative goal importance*. Estimate the relative importance Ω_i of each technical-goal g_i that has a mismatch m_i with a COTS attribute.
(3) *Identify constraints*. Identify the problem constraints (see Sect. 3.1.6). This includes:

a. Defining any ‘pre-assignments’ for resolving or tolerating the mismatches.
b. Estimating available resource constraints, i.e., *Effort* and *Budget*. Here, MiHOS allows the project manager to choose one of two options:
   - *Fixed-resources option*: the project manager defines ‘one value’ for each resource type, e.g., effort = 60 person-hours. This option allows the project manager to explore the output of MiHOS only at the defined value.
   - *Ranged-resources option*: the project manager defines a ‘value range’ for a resource, e.g., effort = 40–100 person-hours. This option allows the project manager to explore the output of MiHOS at different resources values. This option helps to make more informed decisions as it gives decision makers information about different possible scenarios at different resource values within a defined range. This is illustrated in more details in Sect. 5.

4.3.2 Exploration

In this phase, the computational part of MiHOS explores the solution space. A set of qualified alternative plans is generated based on our formal model given in Sect. 3. This formal model constitutes a linear integer programming problem [32] as all the model’s objectives and constraints are linear functions, and the decision variables are integers.

(1) *For MiHOS.Evaluation*: MiHOS’s exploration phase is used to estimate the average anticipated fitness of COTS candidates under evaluation based on the five suggested plans. The average anticipated fitness is calculated as follows:

$$\mathit{Avg.\,Anticipated\ Fitness} = \frac{\sum_{n=1..N} \text{Anticipated fitness if } \mathit{Plan}_n \text{ is applied}}{N} \qquad (8)$$

where N = total number of suggested plans. In case of choosing ‘*Fixed constraints*’ option in the *Modeling* phase, the anticipated fitness is calculated only once for the given resource constraints. In case of using ‘*ranged-resources*’ option, the anticipated fitness is calculated at several successive values in the defined resource range.

(2) *For MiHOS.Planning*: after the decision maker selects a COTS product, the exploration phase helps planning the resolution of the COTS mismatches by suggesting several alternative plans. These plans are given to decision makers to analyze them during the Consolidation phase of MiHOS.

4.3.3 Consolidation

In this phase decision makers review the output of the exploration phase. They might then refine the problem settings as necessary and go to the next iteration.

(1) *For MiHOS.Evaluation*: after evaluating COTS products, MiHOS’s consolidation phase allows decision makers to perform *what-if* analysis to examine the impact of ‘changing the problem settings’ on the ‘anticipated fitness’ of different COTS candidates. For example, decision makers might want to compare COTS candidates if all mismatches related to security are resolved. This is done using the ‘‘pre-assignment’’ feature of our model. Decision makers can then analyze COTS

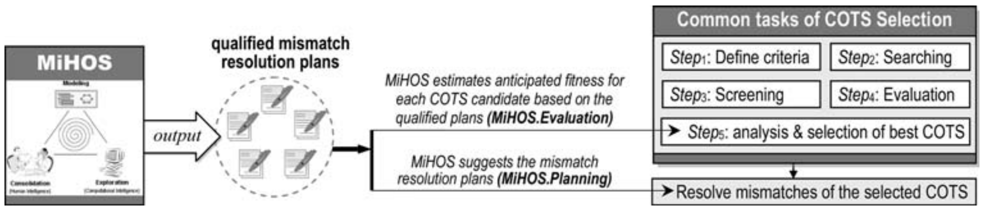

MiHOS
Modeling
Consolidation
Exploration
output
qualified mismatch resolution plans
MiHOS estimates anticipated fitness for each COTS candidate based on the qualified plans (MiHOS.Evaluation)
MiHOS suggests the mismatch resolution plans (MiHOS.Planning)
Common tasks of COTS Selection
Step1: Define criteria
Step2: Searching
Step3: Screening
Step4: Evaluation
Step5: analysis & selection of best COTS
Resolve mismatches of the selected COTS

Fig. 5 Integrating MiHOS to COTS selection process

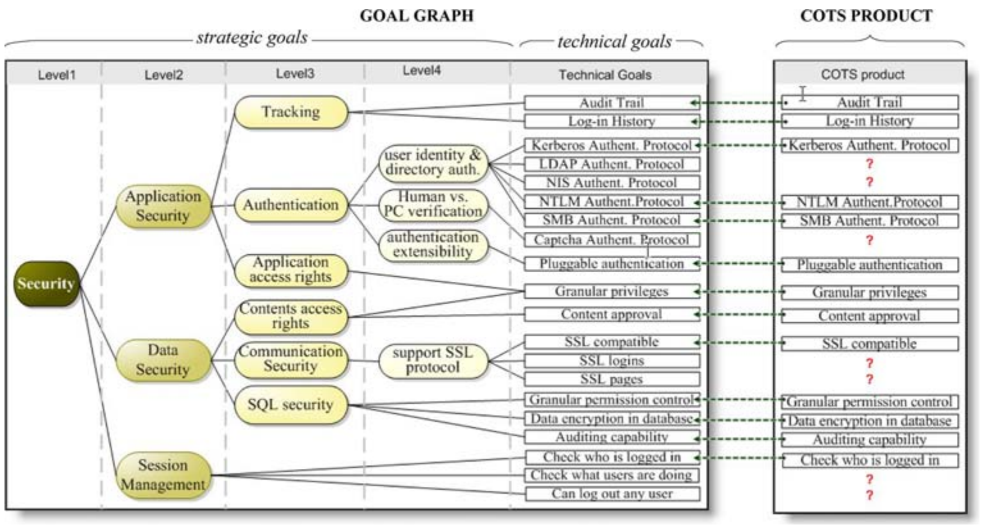


Fig. 6 Evaluating a COTS product against technical goals

candidates based on their 'anticipated security scores' as well as their 'anticipated overall scores'.

(2) *For MiHOS.Planning*: when trying to solve mismatches of the selected COTS product, MiHOS's consolidation phase enables decision makers to review and analyze the alternative plans suggested in the exploration phase. The decision makers then have the choice either to accept one of the plans, or refine the problem settings and go to the next iteration. During each iteration, decision makers' knowledge evolves and more refinement of the underlying model occurs. The iterations continue until a desirable solution alternative is found.

5 Case study

5.1 Settings

We use a case study from the e-services domain: the acquisition of a content management system (CMS) for creating an e-business solution. CMS are COTS products which can be used to facilitate creating online news portals using managed contents, (i.e., text, images, videos, etc).

Initially, a set of 275 goals were defined to represent system requirements at different levels of abstraction. From these, 153 goals were defined at the technical-goals level, and 122 were strategic goals. While the strategic goals were defined based on stakeholders' high-level requirements, (i.e., those which cannot be directly measured in COTS products), the technical goals were defined so as to map COTS features as means to operationalize the strategic goals (refer to Sect. 2.1). Figure 6 shows a part of the goal graph and how it is used in COTS evaluation. On the left is the strategic goals and how they are decomposed to more refined goals. the figure shows four levels of strategic goals, and one level of technical goals. The right part of the figure shows the features of a COTS product under evaluation, and how they are compared with the technical goals. Some mismatches are shown by the symbol "?".[4]

The COTS market was searched, and a subset of 30 "must-have" technical-goals was used to define a shortlist of five COTS candidates using progressive filtering [17]. All candidates had different mismatches with different amounts. As an example, we show in "Appendix 2" (Table 4) the mismatches of COTS4.

Next, the resource constraints were estimated:

- For the *Budged* constraint, the '*Fixed resource option*' was chosen, and the *available_budget* was set to $2,000.
- For the *Effort* constraint, the '*Ranged-resource option*' was chosen, and the *available_effort* ranges was initially set to a range from 10 to 100 person-hours.

5.2 Results for MiHOS.Evaluation: applying MiHOS after COTS evaluation

MiHOS was applied to estimate the average anticipated fitness for COTS candidates if their mismatches are resolved[5]. The average is based on the five qualified plans suggested by MiHOS. The anticipated fitness is used to compare different COTS candidates. In this case study, MiHOS's output was analyzed through three successive phases as follows:

[4] Full details of the case study are available upon request.

[5] In most cases, the differences between the anticipated fitness in these plans did not exceed ±1%.

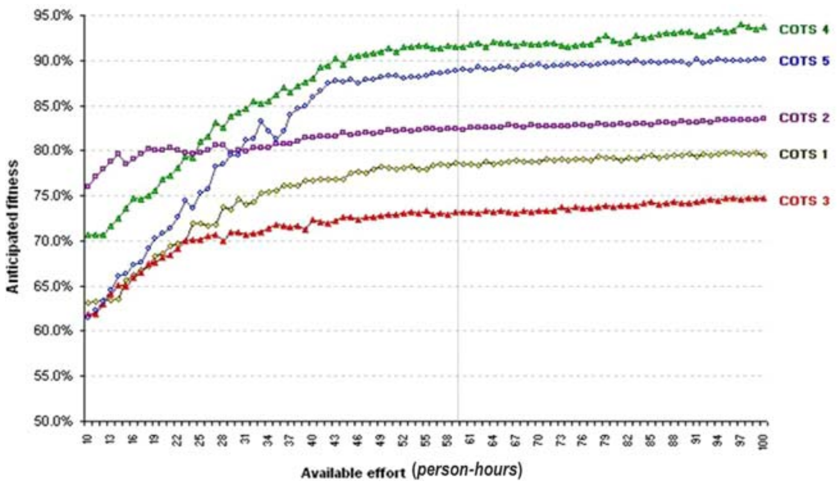


Fig. 7 Anticipated fitness value of COTS candidates

- Phase 1: Given the resource constraint defined in Sect. 5.1, i.e., *Available_Budget=\$2,000*, and *Available_Effort = 10–100 person-hours*, the anticipated fitness of the COTS candidates was estimated and the output was depicted as seen in Fig. 7; where the *y*-axis represents the anticipated fitness, and the *x*-axis represents the change in the *available_effort* value.

Figure 7 provided a better insight on the resolvability of COTS candidates' mismatches under different resource values. Having this figure and based on the current project settings, the *available_effort* was set by decision makers to 60 person-hours.

- Phase 2: Based on Phase 1, the resource constraints were set in this phase to *Available_Budget=\$2,000*, and *Available_Effort=60 person-hour*. Table 2 summarizes the output of MiHOS. The *anticipated_fitness* under the given resource constraints is listed in Table 2, column 2. Based on this information, (i.e., in column 2), it was found that both $COTS_4$ and $COTS_5$ have the highest anticipated-fitness. Yet, it was hard to choose between $COTS_4$ and $COTS_5$ as we could not rely on the accuracy of such a small difference between their anticipated-fitness values. Therefore, it was decided to examine the scenario if the 'security' of COTS candidates is maximized, i.e., if all mismatches related to the '*security*' goal of COTS candidates are resolved (as described in Sect. 4.3.3).
- Phase 3: After applying the scenario: maximize the '*security*' of COTS candidates, it was found that $COTS_4$ is the best choice. This is because $COTS_4$ showed the best anticipated overall fitness (Table 2, column 3) with the maximum anticipated security score (Table 2, column 4). Consequently, we decided to select $COTS_4$ as the best COTS product.

Note that Table 2 also compares COTS selection with and without the use of MiHOS. If MiHOS was not used, then COTS candidates would have been compared based on their current fitness. In such case, $COTS_2$ might have been selected due to its high fitness value (column 5 in Table 2). This shows the added valued of using MiHOS at this stage of COTS selection, i.e., when comparing different COTS candidates.

5.3 Results for MiHOS.Planning: applying MiHOS after COTS selection

After selecting $COTS_4$, we wanted to choose the best plan to solve its mismatches. The same values used before for *MiHOS.Evaluation*, i.e., the mismatch amount, effort to solve mismatches, etc. for $COTS_4$ are reused here. The plans suggested by MiHOS for resolving $COTS_4$'s mismatches are analyzed more extensively to choose the best one.

In more details, $COTS_4$ had an overall of 64 mismatches between its attributes and the original

Table 2 Analysing COTS candidates before and after applying MiHOS

	After resolving the 'right-mismatches'			Fitness before using MiHOS[a] (%)
	Anticipated fitness[a] (%)	Anticipated fitness with max security[a] (%)	Security before → after[a]	
$COTS_1$	78	68	84 → 100%	63
$COTS_2$	82	81	86 → 92 %	75
$COTS_3$	73	70	70 → 77.9 %	61
$COTS_4$	92	91	72 → 100%	68
$COTS_5$	89	69	58 → 95%	53

[a] The 'fitness-values' were obtained using the weighing-and-aggregation method described in Sect. 2

technical-goals (see “Appendix 2”). The set of 95% qualified solutions were explored and the most diversified five solutions were identified. Table 3 shows these solutions.

In Table 3, the rows show alternative plans, while the columns refer to different mismatches. The cells containing ‘x’ suggest tolerating relevant mismatches, while other cells suggest solving relevant mismatches m_i using actions a_{ij}. The last four columns show the evaluation of the suggested plans. As can be seen, these five suggested plans have a consensus on the majority of resolution-actions. For example, for mismatch m_1, all suggested plans, (i.e., X_0 –X_4) suggest resolution action $a_{1,1}$. Therefore, we felt more confident to use those actions. One the other hand, we focused our analysis on the differences between the suggested plans. All alternative plans were seen reasonable and eventually alternative X_0 was adopted as it required the least resources.

6 Limitations of MiHOS

As is the case with any approach, the quality of MiHOS’s results relies on the quality of the input data. As mentioned in [33], every evaluation is subject to uncertainty, e.g., how meaningful are the results obtained if the effort estimation is inaccurate? and if the weights are imprecise? This uncertainty creates risk to the validity of the results. To mitigate the risk, MiHOS suggests some techniques to improve estimations. More precisely, estimation by analogy as described in [29] is suggested to improve effort estimation. Similarly, the analytical hierarchy process (AHP) [34] is suggested to determine weights of criteria. In addition to that, MiHOS tries to reduce the uncertainty using two fundamental techniques:

(i) Hybrid problem solving making synergy between computational and human expert intelligence (as suggested in [14] to provide decision support for software release planning).
(ii) The use of a portfolio of qualified solutions with maximum diversification. These solutions are selected from the set of solutions exceeding 95% of maximum objective function value allows for 5% tolerance in accepting output solutions. With such tolerance, a plan suggested by MiHOS is likely to remain within the acceptable range, (i.e., the top 5%) even when having some inaccuracies in input values. A similar approach that uses comparable techniques to cope with the input uncertainties has proven promising in the area of software release planning [16, 35].

On the other hand, we argue that the extra effort needed for applying MiHOS during and after the selection process is more helpful and less risky than acting reactively if problems occur. We suggest using MiHOS in big projects in which inefficient decisions would have critical consequences, e.g., related to finances or human safety. We also recommend applying MiHOS during the selection process only after having a small set of the most promising COTS candidates so as to reduce the effort required for defining the problem settings. However, once the problem settings are defined, it is very easy and almost effortless to re-run MiHOS several times so as to explore different scenarios of the problem. For example, in our case study the scenario “*evaluate COTS candidates if their security is maximized*” was done in almost no time by few clicks on the computer keyboard, and yet yielded important information that significantly affected our decision.

In addition, handling mismatches related to quality requirements (QR) might be difficult. This problem arises from the difficulty of estimating the cost and effort for achieving a specific target level by solving such mismatches. In our case study, there was an exceptional case in which MiHOS was able to deal with the “security” quality attribute. This reason is that it was possible to map “security” to a determinate set of functional requirements. However, applying MiHOS to other QRs, (e.g., reliability) would be difficult.

7 Summary and outlook

We proposed a decision-support approach, called MiHOS, that can be integrated with any existing COTS selection method at two points: (1) when evaluating most promising COTS candidates; the role of MiHOS here is to estimate the anticipated fitness if the right mismatches are resolved for each candidate; and (2) after selecting a COTS product in order to help planning the resolution of the right mismatches using the right resolution-actions. MiHOS takes in consideration factors such as the impact of a mismatch on COTS fitness, and the cost, effort and risk of resolution actions. Our case study showed the significance of using MiHOS in qualifying the decisions related to the COTS selection process.

Although MiHOS tries to reduce uncertainty by various means as described in [6], there will still exist some uncertainty that could cause risks for the validity of the output. For this reason, another component was developed to be used along with MiHOS; namely MiHOS-SA which stands for MiHOS-Sensitivity

Table 3 Suggested resolution plans for COTS4 (mismatches *pre-assigned* as to be tolerated are not shown)

Mismatches	Alternative sets of resolution actions				
	X_0	X_1	X_2	X_3	X_4
m (1)	a (1,1)	a (1,1)	a (1,1)	a (1,1)	a (1,1)
m (2)	a (2,1)	a (2,1)	a (2,1)	a (2,1)	a (2,1)
m (3)	a (3,1)	a (3,1)	a (3,1)	a (3,1)	a (3,1)
m (4)	a (4,1)	a (4,1)	a (4,1)	a (4,1)	a (4,1)
m (5)	a (5,1)	a (5,1)	a (5,1)	a (5,1)	a (5,1)
m (6)	a (6,1)	a (6,1)	a (6,1)	a (6,1)	a (6,1)
m (7)	a (7,1)	a (7,1)	a (7,1)	a (7,1)	a (7,1)
m (8)	a (8,1)	a (8,1)	a (8,1)	a (8,1)	a (8,1)
m (9)	a (9,2)	a (9,2)	a (9,1)	a (9,1)	a (9,2)
m (10)	X	X	X	X	X
m (11)	X	X	X	X	X
m (12)	a (12,1)	a (12,1)	a (12,1)	a (12,1)	a (12,1)
m (13)	a (13,1)	a (13,1)	a (13,1)	a (13,1)	a (13,1)
m (15)	a (15,1)	a (15,1)	a (15,1)	a (15,1)	a (15,1)
m (16)	X	X	X	X	a (16,1)
m (17)	a (17,1)	X	a (17,1)	a (17,1)	X
m (18)	X	a (18,1)	X	a (18,1)	X
m (19)	X	X	a (19,1)	a (19,1)	a (19,1)
m (20)	a (20,1)	X	a (20,1)	X	X
m (23)	a (23,1)	a (23,1)	a (23,1)	a (23,1)	a (23,1)
m (24)	a (24,1)	a (24,1)	a (24,1)	X	a (24,1)
m (25)	X	X	a (25,1)	a (25,1)	a (25,1)
m (26)	a (26,1)	a (26,1)	a (26,1)	a (26,1)	a (26,1)
m (27)	a (27,1)	a (27,1)	a (27,1)	a (27,1)	a (27,1)
m (30)	a (30,1)	X	X	a (30,1)	X
m (31)	X	X	X	X	X
m (32)	X	X	X	X	X
m (41)	a (41,1)	a (41,1)	a (41,1)	a (41,1)	a (41,1)
m (44)	a (44,1)	a (44,1)	a (44,1)	a (44,1)	a (44,1)
m (45)	a (45,1)	a (45,1)	a (45,1)	a (45,1)	a (45,1)
m (46)	X	a (46,1)	a (46,1)	a (46,1)	X
m (47)	X	a (47,1)	a (47,1)	X	X
m (48)	a (48,1)	X	X	a (48,1)	X
m (49)	X	X	X	X	a (49,1)
m (50)	a (50,1)	a (50,1)	a (50,1)	a (50,1)	a (50,1)
m (51)	a (51,1)	a (51,1)	a (51,1)	a (51,1)	a (51,1)
m (52)	a (52,1)	a (52,1)	a (52,1)	a (52,1)	a (52,1)
m (53)	a (53,1)	a (53,1)	a (53,1)	a (53,1)	a (53,1)
m (54)	a (54,1)	a (54,1)	a (54,1)	a (54,1)	a (54,1)
m (55)	a (55,1)	a (55,1)	a (55,1)	a (55,1)	a (55,1)
m (56)	X	a (56,1)	X	a (56,1)	X
m (57)	X	X	X	X	X
m (58)	a (58,1)	a (58,1)	X	a (58,1)	X
m (59)	a (59,1)	a (59,1)	a (59,1)	a (59,1)	a (59,1)
m (60)	a (60,1)	X	a (60,1)	X	X
m (61)	a (61,1)	a (61,1)	a (61,1)	a (61,1)	a (61,1)
m (62)	a (62,1)	a (62,1)	a (62,1)	a (62,1)	a (62,1)
m (63)	X	a (63,1)	a (63,1)	X	X
m (64)	a (64,1)	a (64,1)	a (64,1)	a (64,1)	a (64,1)
Objective Function Value	1,535	1,520	1,514	1,502	1,496
Effort (person-hours)	59	60	60	60	60
Cost($)	1,180	1,200	1,200	1,200	1,200
Resultant COTS Fitness (%)	91	92	91	92	90

Analysis. MiHOS-SA aims at checking the robustness of MiHOS's results against input errors. MiHOS-SA asks analysts to identify input parameters to be analyzed as well as the error range at the inputs for the analysis. MiHOS then simulates the error and shows the corresponding potential results. This allows decision makers to have better insight about the validity risks of the output, and hence make more appropriate

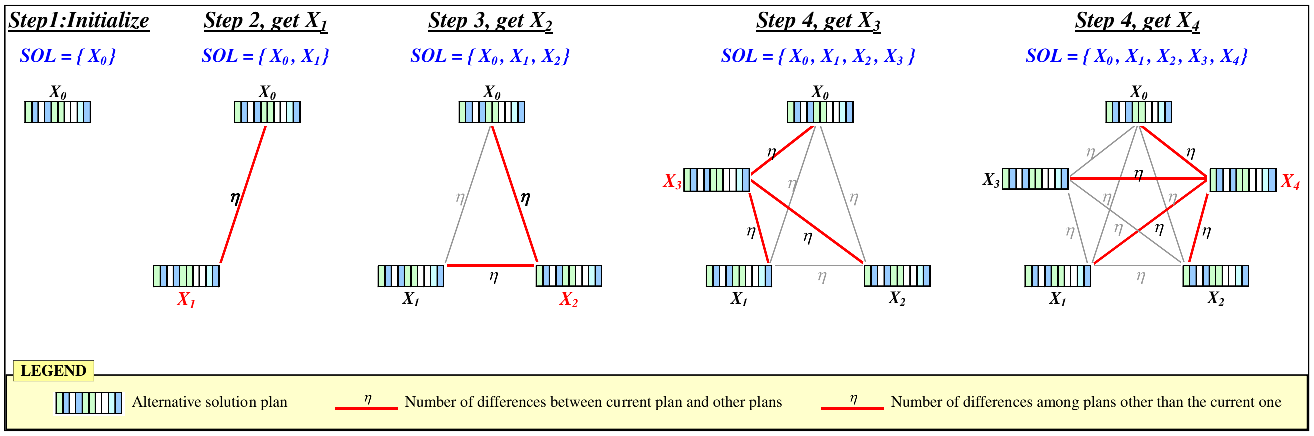

Step1:Initialize
SOL = { X0}
Step 2, get X1
SOL = { X0, X1}
Step 3, get X2
SOL = { X0, X1, X2}
Step 4, get X3
SOL = { X0, X1, X2, X3 }
Step 4, get X4
SOL = { X0, X1, X2, X3, X4}
η
LEGEND
Alternative solution plan
Number of differences between current plan and other plans
Number of differences among plans other than the current one

Fig. 8 Procedure of finding a subset of qualified solutions with maximum diversification

decisions. It is worth mentioning that MiHOS-SA was applied to the case-study used in this paper and the results showed that MiHOS's output was relatively robust in this case-study. A detailed discussion about MiHOS-SA is beyond the scope of this paper.[6]

Despite the limitations described in Sect. 6, MiHOS can be extended to address more complex problems. For example, instead of considering the overall-effort for applying resolution-actions, the formal model can be easily modified to address more refined types of effort; e.g., development, testing, and documentation efforts. This differentiation between effort types is necessary when having different teams in the project with different effort constraints.

Also, instead of assuming that each mismatch is resolved by only one resolution-action, the formal model can be modified to consider several resolution actions that together can solve one or more mismatches. This can be done by adding technological constraints that define the dependencies between resolution-actions.

We are also working on exploring the use of *'linear-combination'* for formulating the objective function, and comparing the results to the current objective function. Linear combination might be useful to conduct more complex *'what-if'* analysis. This increases the customizability of the objective function towards experts' preferences.

To better judge the usefulness and applicability of MiHOS, a more rigorous evaluation is needed. We plan to conduct further case studies in different domains in order to evaluate its performance more deeply and to allow better insight about its performance under different circumstances.

[6] MiHOS-SA is described in a technical report that is available on request.

Acknowledgments We appreciate the support of the Natural Sciences and Engineering Council of Canada (NSERC) and of the Alberta Informatics Circle of Research Excellence (iCORE) to conduct this research.

Appendix 1: Diversification strategy used in MiHOS

Searching for a subset having the maximum diversification among a set of solutions is a challenging problem [16]. To illustrate this, suppose we have a COTS product with 70 mismatches m_i each of which can be solved by using one of two resolution actions $a_{i,1}$and $a_{i,2}$. The solution space for such problem would contain 3^{70} possible solutions; this is because each mismatch m_i has three handling actions: solve using $a_{i,1}$, solve using $a_{i,2}$, and tolerate m_i. If only 0.0001% of these solutions are qualified, then the number of qualified solutions for this problem equals $0.0001\% \times 3^{70} \approx 2.58 \times 10^{26}$. Now, it is required to search this subset for the most diversified solutions. Assuming that we need a subset of five diversified solutions, we would have $2.58 \times 10^{26}!/[(2.58 \times 10^{26}–5)! \times 5!] \approx 9.53 \times 10^{129}$ alternative subsets.

A strategy that simplifies the problem by searching only a subset of the feasible space for the most diversified solutions was introduced in [16]. However, as stated by the authors, this strategy does not guarantee the identification of the solutions with the maximum diversification among the whole feasible space.

In our paper, we propose another strategy. Suppose that we want to obtain a subset *SOL* that contains five solutions having the maximum diversification among the set of all qualified solutions. And suppose that if we have two solution plans $X_i$and $X_j \in SOL$, then we can measure the level of diversification by comparing the structure of $X_i$and X_j and simply counting the number of differences $\aleph_{i,j}$ between them. Based on this, we can identify the subset SOL by applying the following procedure (see Fig. 8):

1. Initialize the procedure:

1.1. Formulate the original problem P_0 as described in Sect. 3 and get the optimum solution X_0 by solving P_0.
1.2. Set $SOL = \{X_0\}$.
1.3. Set $\eta = 1$ (where η is required degree of diversification. η represented by the number of differences that are counted between the different solution plans).

2. Get a qualified solution X_1 which has η differences with X_0. This is done as follows:

2.1. Formulate a new problem P_1 which is the same as P_0 in addition to a new constraint $\aleph_{1,0} = \eta$ (see Fig. 8).
2.2. Get the solution X_1 by solving P_1.
2.3. Add X_1 to the solution subset SOL; i.e., change $SOL = \{X_0, X_1\}$.

3. Repeat Step 2 to get the solution X_2, and add X_2 to SOL subset. Note that the new formulated problem P_2 has the same formulation as P_0 but has additionally two more constraints, $\aleph_{2,0} = \eta$ and $\aleph_{2,1} = \eta$.
4. Similarly to Step 3, repeat Step 2 to get $X_3$and X_4, while adding more constraints that guarantees a number of differences equal to η among all solutions (see Fig. 8).
5. If the optimization engine, (i.e., LINDO [28]) has successfully found all diversified solutions X_0–X_4, then increment η by one.
6. Repeat Steps 2–5 (while η is being incremented) until the optimization engine fails to find a feasible solution to at least one of the new formulated problems (this often happens when the optimization engine fails to solve P_4).
7. The required subset SOL would then include the diversified solutions identified in the most recent successful run of the optimization engine.

The above procedure also cannot guarantee identifying the set of qualified solutions with maximum diversification. This is because it assumes equal distances between the diversified solutions, while in reality, the distances between the diversified solutions might vary. In addition, adding constraints such as $\aleph_{1,0} = \eta$ makes the problem non-linear which makes it harder to solve.

Appendix 2: Mismatches between COTS4 and technical goals

Table 4 A list of COTS4 mismatches

Technical goals[a]		Mismatches, mi			First resolution action, a(i,1)					Second resolution action, a(i,2)				
Name	Relative importance (1–10)	ID	Amount[b] (0–1)	Pre-assignment	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)
Kerberos Auth. protocol	8.00	m_1	1.00	Resolve	a(1,1)	SCRP[c]	10	200	5	a(1,2)	–	–	–	–
LDAP Auth. protocol	8.00	m_2	1.00	Resolve	a(2,1)	Add-on	1	20	1	a(2,2)	SCRP	10	200	5
NIS Auth. protocol	8.00	m_3	1.00	Resolve	a(3,1)	SCRP	10	200	5	a(3,2)	-	-	-	-
NTLM Auth. protocol	8.00	m_4	1.00	Resolve	a(4,1)	Add-on	1	20	1	a(4,2)	SCRP	10	200	5
SMB Auth. protocol	8.00	m_5	1.00	Resolve	a(5,1)	Add-on	2	40	1	a(5,2)	–	–	–	–
Captcha Auth. protocol	3.20	m_6	1.00	Resolve	a(6,1)	Add-on	1	20	1	a(6,2)	SCRP	20	400	5
Pluggable authentication	8.00	m_7	1.00	Resolve	a(7,1)	Add-on	1	20	1	a(7,2)	–	–	–	–
Support SSL protocol	4.80	m_8	0.20	Resolve	a(8,1)	Add-on(SSL pages)	1	20	1	a(8,2)	SCRP	20	400	3
Trash	8.00	m_9	1.00	–	a(9,1)	Add-on	2	40	5	a(9,2)	SCRP	4	80	1
Zip archives	2.00	m_{10}	1.00	–	a(10,1)	SCRP	15	300	3	a(10,2)	–	–	–	–
WebDAV compliant	4.00	m_{11}	1.00	–	a(11,1)	SCRP	12	240	5	a(11,2)	–	–	–	–
Content staging support	4.00	m_{12}	1.00	–	a(12,1)	Add-on	1	20	1	a(12,2)	SCRP	20	400	1
Auto content staging support	4.00	m_{13}	1.00	–	a(13,1)	Add-on	1	20	1	a(13,2)	SCRP	10	200	1
Manual content staging	2.40	m_{14}	1.00	Tolerate	a(14,1)	–	–	–	–	a(14,2)		–	–	–
Visitor tracking	8.00	m_{15}	1.00	–	a(15,1)	Add-on	1	20	1	a(15,2)	SCRP	10	200	1
Webpage tracking	4.80	m_{16}	1.00	–	a(16,1)	Add-on (alpha)	2	40	9	a(16,2)	SCRP	6	120	1
Control which banner to show	1.00	m_{17}	1.00	–	a(17,1)	Add-on	1	20	1	a(17,2)	SCRP	10	200	1

Table 4 continued

Technical goals[a]		Mismatches, mi			First resolution action, a(i,1)					Second resolution action, a(i,2)				
Name	Relative importance (1–10)	ID	Amount[b] (0–1)	Pre-assignment	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)
Control when to show banners	0.50	m_{18}	1.00	–	a(18,1)	Add-on	1	20	1	a(18,2)	SCRP	10	200	1
Control banners' time duration	0.50	m_{19}	1.00	–	a(19,1)	Add-on	1	20	1	a(19,2)	SCRP	10	200	1
Contact management	1.00	m_{20}	1.00	–	a(20,1)	Add-on	1	20	1	a(20,2)	SCRP	30	600	1
Size of third-party plug-ins dev.	6.00	m_{21}	0.25	Tolerate	a(21,1)	–	–	–	–	a(21,2)	–	–	–	–
API extensiveness	6.00	m_{22}	0.25	Tolerate	a(22,1)	–	–	–	–	a(22,2)	–	–	–	–
Extensible user profiles	3.60	m_{23}	1.00	–	a(23,1)	Add-on (beta)	1	20	3	a(23,2)	SCRP	20	400	3
Prototyping	1.73	m_{24}	1.00	–	a(24,1)	Add-on	1	20	1	a(24,2)	SCRP	30	600	3
Dashboard	0.48	m_{25}	1.00	–	a(25,1)	Add-on	1	20	1	a(25,2)	SCRP	60	1,200	3
Accessibility/WAI	4.80	m_{26}	1.00	–	a(26,1)	Add-on (beta)	1	20	3	a(26,2)	SCRP	40	800	1
Drag and drop	4.00	m_{27}	1.00	–	a(27,1)	Add-on	1	20	1	a(27,2)	SCRP	30	600	1
UI_p look and feel	6.00	m_{28}	0.25	Tolerate	a(28,1)	–	–	–	–	a(28,2)	–	–	–	–
UI_a look and feel	6.00	m_{29}	0.25	Tolerate	a(29,1)	–	–	–	–	a(29,2)	–	–	–	–
Static content export	0.80	m_{30}	1.00	–	a(30,1)	Add-on	1	20	1	a(30,2)	SCRP	60	1,200	1
Load balancing	1.60	m_{31}	1.00	–	a(31,1)	SCRP	40	800	3	a(31,2)	–	–	–	–
Database replication	1.20	m_{32}	1.00	–	a(32,1)	SCRP	30	600	3	a(32,2)	–	–	–	–
Support by experts to instal	1.60	m_{33}	0.25	Tolerate	a(33,1)	–	–	–	–	a(33,2)	–	–	–	–
Professional services to install	1.60	m_{34}	0.25	Tolerate	a(34,1)	–	–	–	–	a(34,2)	–	–	–	–
Public forum for installation	8.00	m_{35}	0.13	Tolerate	a(35,1)	–	–	–	–	a(35,2)	–	–	–	–
Support by experts for usage	1.60	m_{36}	0.25	Tolerate	a(36,1)	–	–	–	–	a(36,2)	–	–	–	–
Professional services for usage	1.60	m_{37}	0.25	Tolerate	a(37,1)	–	–	–	–	a(37,2)	–	–	–	–
Training for usage	2.40	m_{38}	0.25	Tolerate	a(38,1)	–	–	–	–	a(38,2)	–	–	–	–
Mailing list for usage	4.00	m_{39}	0.25	Tolerate	a(39,1)	–	–	–	–	a(39,2)	–	–	–	–
Public forum for usage	8.00	m_{40}	0.13	Tolerate	a(40,1)	–	–	–	–	a(40,2)	–	–	–	–
API docs for extensions (extns.)	10.00	m_{41}	0.10	Resolve	a(41,1)	get detailed docs	10	200	5	a(41,2)	–	–	–	–
Developers forum for extns.	10.00	m_{42}	0.13	Tolerate	a(42,1)	–	–	–	–	a(42,2)	–	–	–	–
Third-party developers for extns.	10.00	m_{43}	0.25	Tolerate	a(43,1)	–	–	–	–	a(43,2)	–	–	–	–
Code skeleton for extns.	10.00	m_{44}	1.00	–	a(44,1)	Add-on	1	20	1	a(44,2)	SCRP	200	4,000	5
Test framework for extns.	8.00	m_{45}	1.00	–	a(45,1)	Add-on (beta)	1	20	3	a(45,2)	SCRP	120	2,400	5
Blog	0.80	m_{46}	1.00	–	a(46,1)	Add-on	1	20	1	a(46,2)	SCRP	30	600	3
Chat	0.80	m_{47}	1.00	–	a(47,1)	Add-on	1	20	1	a(47,2)	SCRP	30	600	5
Classified	0.80	m_{48}	1.00	–	a(48,1)	Add-on	1	20	1	a(48,2)	SCRP	20	400	1
Database reports	0.80	m_{49}	1.00	–	a(49,1)	Add-on (alpha)	2	40	7	a(49,2)	SCRP	6	120	1
Forum	8.00	m_{50}	1.00	–	a(50,1)	Add-on	1	20	1	a(50,2)	SCRP	150	3,000	7
Calendar	8.00	m_{51}	1.00	–	a(51,1)	Add-on	1	20	1	a(51,2)	SCRP	40	800	3
FAQ management	6.40	m_{52}	1.00	–	a(52,1)	Add-on	1	20	1	a(52,2)	SCRP	20	400	1
Graph and charts	6.40	m_{53}	1.00	–	a(53,1)	Add-on (beta)	1	20	5	a(53,2)	SCRP	120	2,400	1
Image gallery	8.00	m_{54}	1.00	–	a(54,1)	Add-on	1	20	1	a(54,2)	SCRP	50	1,000	1
Guest book	4.80	m_{55}	1.00	–	a(55,1)	Add-on	1	20	1	a(55,2)	SCRP	10	200	1
Bug report	0.80	m_{56}	1.00	–	a(56,1)	Add-on (beta)	1	20	3	a(56,2)	SCRP	10	200	3
HTTP proxy	0.80	m_{57}	1.00	–	a(57,1)	SCRP	15	300	3	a(57,2)	–	–	–	–
Job posting	1.60	m_{58}	1.00	–	a(58,1)	Add-on	1	20	1	a(58,2)	SCRP	15	300	1
Polls	6.40	m_{59}	1.00	–	a(59,1)	Add-on	1	20	1	a(59,2)	SCRP	30	600	1
Project tracking	1.60	m_{60}	1.00	–	a(60,1)	Add-on (beta)	1	20	5	a(60,2)	SCRP	80	1,600	3

Table 4 continued

Technical goals[a]		Mismatches, mi			First resolution action, a(i,1)					Second resolution action, a(i,2)				
Name	Relative importance (1–10)	ID	Amount[b] (0–1)	Pre-assignment	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)	ID	Type	Effort (prsn-hrs)	Cost ($)	Risk (1–9)
Search engine	8.00	m_{61}	1.00	–	a(61,1)	Add-on	1	20	1	a(61,2)	SCRP	60	1,200	3
Surveys	8.00	m_{62}	1.00	–	a(62,1)	Add-on	1	20	1	a(62,2)	SCRP	60	1,200	1
Tests/quizzes	0.80	m_{63}	1.00	–	a(63,1)	Add-on	1	20	1	a(63,2)	SCRP	30	600	1
Weather info	4.80	m_{64}	1.00	–	a(64,1)	Add-on	1	20	1	a(64,2)	SCRP	10	200	3

[a] The full description of the goals and the complete goal graph are available upon request

[b] When the mismatch amount equals 1, then this means that the technical-goal is fully unsatisfied. But, when the mismatch amount is less than 1, then this means that the technical-goal is partially satisfied

[c] SCRP stands for "scripting"

References

1. Morisio M, Seaman CB, Parra AT, Basilli VR, Kraft SE, Condon SE (2000) Investigating and improving a COTS-based software development process. In: ICSE'00, Limmerick, Ireland, pp 32–41
2. CeBASE: center for empirically based software engineering, at http://www.cebase.org
3. Torchiano M, Morisio M (2004) Overlooked aspects of COTS-based development. IEEE Softw 21(2):88–93
4. Boehm B, Abts C (1999) COTS integration: plug and pray? Computer 32(1):135–138
5. Alves C (2003) COTS-based requirements engineering. In: Component-based software quality—methods and techniques, vol 2693. Springer, Heidelberg, pp 21–39
6. Ruhe G (2003) Intelligent support for selection of COTS products, Web, Web-services, and database systems, lecture notes in computer science, vol 2593. Springer, Heidelberg, pp 34–45
7. Kontio J (1995) OTSO: a systematic process for reusable software component selection. University of Maryland, Maryland, CS-TR-3478, Dec 1995
8. Lozano-Tello A, Gomez-Pérez A (2002) BAREMO: how to choose the appropriate software component using the analytic hierarchy process. In: Proceedings of the 14th international conference on software engineering and knowledge engineering, ACM, Ischia, pp 781–788
9. Ncube C, Maiden N (1999) Guiding parallel requirements acquisition and COTS software selection. In: IEEE international symposium on requirements engineering (RE'99), University of Limerick, Ireland, pp 133–140
10. Ochs M, Pfahl D, Chrobok-Diening G, Nothhelfer-Kolb B (2000) A COTS acquisition process: definition and application experience. In: ESCOM-SCOPE 2000, Munich, Germany
11. Rolland C, Prakash N (2001) Matching ERP system functionality to customer requirements. In: The fifth IEEE international symposium on requirements engineering, IEEE Computer Society, Toronto
12. Zoukar I, Salinesi C (2004) Matching ERP functionalities with the logistic requirements of French railways: a similarity approach. In: The sixth international conference on enterprise information systems (ICEIS'04), Universidade Portucalense, Port0-Portugal, pp 444–450
13. Carney D (1998) COTS evaluation in the real world. SEI interactive, Carnegie Mellon University, December
14. Carney D, Hissam SA, Plakosh D (2000) Complex COTS-based software systems: practical steps for their maintenance. J Softw Maint 12(6):357–376
15. Vigder MR, Gentleman WM, Dean J (1996) COTS software integration: state of the art. National Research Council Canada (NRC), 39198, January 1996
16. Ruhe G, Ngo-The A (2004) Hybrid intelligence in software release planning. Int J Hybrid Intell Syst 1(2):99–110
17. Maiden NA, Ncube C (1998) Acquiring COTS software selection requirements. IEEE Softw 15(2):46–56
18. Franch X, Carvallo JP (2003) Using quality models in software package selection. IEEE Softw 20(1):34–41
19. Lamsweerde AV (2001) Goal-oriented requirements engineering: a guided tour. In: Fifth IEEE international symposium on requirements engineering, pp 249–263
20. Alves C, Finkelstein A (2003) Investigating conflicts in COTS decision-making. Int J Softw Eng Knowl Eng 13(5):473–493

21. Alves C, Franch X, Carvallo JP, Finkelstein A (2005) Using goals and quality models to support the matching analysis during COTS selection. In: The fourth international conference on COTS based software systems (ICCBSS'05), Bilbao, Spain, pp 146–156
22. Grau G, Carvallo JP, Franch X, Quer C (2004) DesCOTS: a software system for selecting COTS components. In: The 30th EUROMICRO conference, Rennes, France, pp 118–126
23. I*-homepage: http://www.cs.toronto.edu/km/istar
24. ISO/IEC1926–1, Software engineering—product quality model—Part 1: quality model, International Organization for Standardization, Geneve, Switzerland
25. Mohamed A, Ruhe G, Eberlein A (2006) A basis for mismatch handling during OTS acquisition. SEDS Lab, UofC, Calgary, TR, 056/2006, July 2006, http://www.enel.ucalgary.ca/~asamoham/SEDS/TR056-2006-Short.pdf
26. Carney D (1999) Requirements and COTS-based systems: a thorny question indeed. In: SEI interactive, Carnegie Mellon University, June 1999
27. Brownsword L, Carney D, Oberndorf T (1998) The opportunities and complexities of applying commercial off-the-shelf components. CrossTalk 11(4):25–30
28. LINDO_Systems-homepage: http://www.lindo.com
29. Li J, Ruhe G, Al-Emran A, Richter M (2006) A flexible method for effort estimation by analogy. Empir Softw Eng. (doi: 10.1007/s10664-006-7552-4)
30. Humphrey WS (1989) Managing the software process, 1st edn. Addison-Wesley, Reading
31. Ruhe G, Saliu MO (2005) The art and science of software release planning. IEEE Softw 22(6):47–53
32. Wolsey LA, Nemhauser GL (1998) Integer and combinatorial optimization. Wiley, New York
33. Comella-Dorda S, Dean JC, Lewis G, Morris E, Oberndorf P, Harper E, A (2004) Process for COTS software product evaluation. Carnegie Mellon University, Software Engineering Institute, CMU/SEI-2003-TR-017, July 2004, http://www.springerlink.com/index/MAFK2LV46922UW45
34. Saaty TL (1990) The analytic hierarchy process. McGraw-Hill, New York
35. Du G, McElroy J, Ruhe G (2006) Ad hoc versus systematic planning of software releases—a three-staged experiment. In: Seventh international conference on product focused software process improvement (PROFES06), Amsterdam