

# Sensitivity analysis in the process of COTS mismatch-handling

Abdallah Mohamed · Guenther Ruhe ·  
Armin Eberlein

Received: 4 September 2007 / Accepted: 28 December 2007 / Published online: 29 January 2008  
© Springer-Verlag London Limited 2008

**Abstract** During the selection of commercial off-the-shelf (COTS) products, mismatches encountered between stakeholders' requirements and features offered by COTS products are inevitable. These mismatches occur as a result of an excess or shortage of functionality offered by the COTS. A decision support approach, called mismatch handling for COTS selection (MiHOS), was proposed earlier to help address mismatches while considering limited resources. In MiHOS, several input parameters need to be estimated such as the level of mismatches and the resource consumptions and constraints. These estimates are subject to uncertainty and therefore limit the applicability of the results. In this paper, we propose sensitivity analysis for MiHOS (MiHOS-SA), an approach that aims at helping decision makers gain insights into the impact of input uncertainties on the validity of MiHOS' results. MiHOS-SA draws on existing sensitivity analysis techniques to address the problem. A case study from the e-services domain was conducted to illustrate MiHOS-SA and discuss its added value.

**Keywords** COTS vs. requirements mismatch · COTS selection · MiHOS · Sensitivity analysis · Case study

## 1 Introduction

Developing software systems based on commercial off the shelf (COTS) products requires the evaluation of several COTS candidates in order to select the one that best fits our requirements [1–3]. During this process, it is inevitable to encounter mismatches between these requirements and COTS features [4, 5]. This is because COTS products are developed for broad use while stakeholders' requirements are specific to their project [6, 7]. Consequently, developers first select the COTS that fits best, and then try to resolve as many mismatches as possible by tailoring the selected COTS [3].

In order to help decision makers analyze these mismatches, an approach called mismatch handling for COTS selection (MiHOS) was previously proposed [8] MiHOS provides the following decision support related to the COTS-selection process:

1. *Support during COTS selection:* Since the selected COTS is eventually tailored after the selection to resolve its mismatches, COTS candidates should be compared during the selection process based on the *anticipated fitness* if their mismatches are resolved. However, due to limited resources, only a subset of mismatches is eventually resolved. Here, MiHOS can be used to do proactive analysis and identify the right subset of mismatches, and therefore help estimate the anticipated fitness of COTS candidates. The anticipated fitness is the first output of MiHOS, which is used during COTS selection.

---

A. Mohamed  
Shoubra Faculty of Engineering, Banha University,  
108 Shoubra St, Cairo, Egypt  
e-mail: asamoham@ucalgary.ca

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

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

2. *Support after COTS selection*: It is also important to apply the right resolution actions in order to resolve the right subset of mismatches. Alternative resolution actions can be used to resolve each mismatch. These actions require different amounts of resources, and impose different risks on the system. MiHOS can be used after the selection process to generate *mismatch-resolution plans*. These plans suggest the use of appropriate resolution actions to resolve the right mismatches with the least risk, and within the given resource constraints. These plans are the second output of MiHOS, which is used after selection.

In order to obtain the results discussed above, decision makers have to identify a variety of inputs for MiHOS, e.g., “the amount of mismatches”, “the resources required to apply each resolution action”, and “the resource constraints”. The problem is that the process of estimating these inputs is subject to uncertainty. The uncertainty might emerge from a variety of sources such as inaccurate information and linguistic imprecision. This problem is inherent to all software engineering processes and products [9]. Failing to deal properly with uncertainty might reduce the applicability of the results.

Therefore, decision makers need a technique to examine the impact of the uncertainty on the outputs obtained from MiHOS. A valuable technique to achieve this objective is sensitivity analysis (SA). SA involves determining the extent to which the outputs change as the inputs vary. Using SA helps make more informed decisions as it allows considering uncertainty when making decisions.

A question arises, which SA techniques should be used with MiHOS? Several techniques can be used to employ SA, e.g., variance-based methods and Monte Carlo analysis [10]. Which technique is appropriate depends on the structure and purpose of the model under analysis. In this paper, we propose sensitivity analysis for MiHOS (MiHOS-SA), a method designed on top of MiHOS. Its purpose is to help humans make more effective decisions by allowing them to analyze the impact of input uncertainties on the two main outputs of MiHOS. Further, in this paper we apply MiHOS-SA to a case study from the e-services domain in order to illustrate MiHOS-SA and discuss its added value.

This paper is structured as follows: Sect. 2 discusses the background necessary to understand the proposed work. Section 3 describes the proposed MiHOS-SA method. Section 4 presents the results obtained from the case study. The limitations of MiHOS-SA are discussed in Sect. 5. Finally, conclusions are given in Sect. 6.

## 2 Background

This section is divided into two subsections, which discuss the background necessary to better understand MiHOS-SA: (1) an overview of sensitivity analysis techniques, and (2) a short description of MiHOS approach.

### 2.1 Sensitivity analysis

Sensitivity analysis (SA) is a common technique used in different contexts of decision-making. The basic idea of SA is to vary the inputs to simulate the uncertainty, run the model, and analyze the impact on the outputs. If small changes in the inputs cause significant changes in the outputs, this means low robustness of the outputs [10]. Two approaches may be considered when performing SA [10]:

- *One-at-a-time approach*, in which a single input-parameter is selected for the analysis. This parameter’s value is changed either *systematically* or *randomly* in successive model runs—all other parameters are fixed during the analysis. “Systematically” means the value is gradually incremented (or decremented) by a predefined amount before each model run. “Randomly”, on the other hand, indicates that a random amount is added or subtracted to the original value before each model run.
- *All-together approach*, in which several input parameters are changed during the analysis. These parameters are simultaneously changed either systematically or randomly before each model run.

Sensitivity analysis is a fundamental technique to evaluate the robustness of solutions in the context of models and problems stated in various contexts and disciplines [11]. Software engineering literature includes many applications of SA in different contexts such as software architecture [12] and software quality and cost control [13]. More examples from software engineering area as well as other disciplines can be found in [14].

### 2.2 MiHOS in a nutshell

MiHOS [8] is a *COTS selection* method that focuses on handling COTS mismatches. A typical COTS selection process usually defines a set of evaluation criteria based on stakeholders’ requirements, and then uses the criteria to evaluate existing COTS and to select the one that fits best. The COTS fitness is calculated based on a weighing and aggregation method. The key idea is to give a weight to



$$(\forall m_i) : A_i = \{a_{i,1}, a_{i,2}, \dots, a_{i,J}\}$$

For instance, if the mismatch  $m_i$  indicates a required functionality that a COTS does not support, then  $A_i$  will include possible actions to ensure that the COTS will support this functionality and thus resolve  $m_i$ ; A classification of possible resolution actions can be found in [3]. In MiHOS, the goal is to select (at most) one resolution action  $a_{i,j}$  for each mismatch  $m_i$ . This is described by the set of decision variables

$$X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,J}\}$$

where  $x_{i,j} = 1$  if the resolution action  $a_{i,j}$  is chosen to resolve the mismatch  $m_i$ , and  $x_{i,j} = 0$  otherwise. This means if a mismatch  $m_i$  is to be resolved using the action  $a_{i,1}$ , then  $X_i = \{1, 0, 0, \dots, 0\}$ . If  $a_{i,J}$  is to be used, then  $X_i = \{0, 0, 0, \dots, 1\}$ . All the other mismatch resolution actions are defined correspondingly.

- **Resources constraints:** Applying resolution actions requires different amount of resources. Each resource has a maximum capacity that should not be exceeded. For this paper, we assume two types of resources related to cost and to effort: *available\_budget* and *available\_effort*. The method remains applicable also for the more general case of more constraints.
- **Resource consumptions:** MiHOS assumes each resolution action  $a_{i,j}$  requires an effort equal to  $effort_{i,j}$  and costs an amount of  $cost_{i,j}$  (see Table 1). The total effort and budget consumed by the set of selected resolution actions (represented by decision variables  $x_{i,j}$ ) must be less than the total amounts of resources available:

$$\begin{aligned} \sum_{i,j} x_{i,j} \cdot effort_{i,j} &\leq available\_effort \\ \sum_{i,j} x_{i,j} \cdot cost_{i,j} &\leq available\_budget \end{aligned} \quad (1)$$

- **Technical risk:** MiHOS assumes that applying each resolution action  $a_{i,j}$  imposes a technical risk  $r_{i,j}$  on the target system. The technical risk is estimated based on: (1) Development risk: the risk that the developers might fail to apply the action  $a_{i,j}$ , and (2) Instability risk: the risk that an action  $a_{i,j}$  would cause instability of the target system. The developers may estimate the technical risk  $r_{i,j}$  on a 9-point ordinal scale,  $r_{i,j} \in \{1, 3, 5, 7, 9\}$ ; this indicates very low, low, average, high, and very high risk. Even numbers can be interpreted as intermediate values.

Based on the above formal description, a mismatch-handling process should aim at:

- Maximizing COTS fitness.
- Minimizing risk for resolution of mismatches.
- Fulfillment of resource constraints.

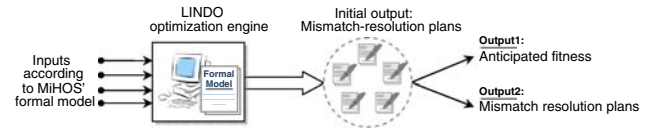


Fig. 2 MiHOS process (overview)

Point (a) is influenced by the requirements priorities  $\Omega_i$  and the mismatch amounts  $Amount_i$ , point (b) is influenced by the technical risk  $r_{i,j}$ , and point (c) is relevant to the resource consumptions and constraints. MiHOS defines an objective function that brings these aspects together in a balanced way. The objective is to maximize function  $F(x)$  subject to the satisfaction of the resource constraints:

$$F(x) = \sum_{i=1}^{\mu} \left( Amount_i \cdot \Omega_i \cdot \sum_{j=1}^J (x_{i,j} \cdot \Delta r_{i,j}) \right) \quad (2)$$

where  $\Delta r_{i,j} = 10 - r_{i,j}$  indicates how safe an action  $a_{i,j}$  is. MiHOS uses  $\Delta r_{i,j}$  instead of  $r_{i,j}$  because maximizing  $F(x)$  should yield the minimum risk (i.e., the maximum safety) of selected actions.

Based on the above formal model, MiHOS' two types of results are generated as follows (see Fig. 2):

(1) The analyst first defines the input parameters according to the above formal model, e.g., determines the amount of mismatches for each COTS candidate, and the resource constraints and consumptions.

(2) This formal model constitutes a integer linear programming problem [16] as all the constraints are linear functions and the decision variables are integers. Integer linear programming is known to belong into the class of NP-complete problems. Applying an optimization package called LINDO [17], MiHOS generates a set of five optimum or near optimum solutions for each COTS under evaluation. The advantage of having five instead of just one solution is that it allows accommodate additional concerns (as discussed later). In dependence of the size of the problem and the response time request, the solution process might be terminated also when the solution is sufficiently good, but not necessarily optimal. Each solution constitutes a mismatch-resolution plan that uses appropriate actions to resolve a subset of mismatches of one COTS.

To explain the plans' structure, consider a COTS having a set of mismatches  $\{m_1, m_2, \dots, m_{\mu}\}$ . A mismatch-resolution plan is represented by the set  $\{y_1, y_2, \dots, y_{\mu}\}$ , where for each mismatch  $m_i$ , a suggestion  $y_i$  may take one of the following values:

- $y_i = \text{"Do not resolve } m_i\text{"}$ . Here, all decision variables  $x_{i,j}$  are set to zero; i.e., no resolution action is used.
- $y_i = \text{"Resolve } m_i \text{ using resolution action } a_{i,1}\text{"}$ . Here, only  $x_{i,1} = 1$  and  $x_{i,j} = 0$  otherwise; i.e.,  $X_i = \{1, 0, 0, \dots, 0\}$ .

- $y_i = \text{“Resolve } m_i \text{ using resolution action } a_{i,2}\text{”}$ . Here, only  $x_{i,2} = 1$  and  $x_{i,j} = 0$  otherwise; i.e.,  $X_i = \{0, 1, 0, \dots, 0\}$ .
- ...
- $y_i = \text{“Resolve } m_i \text{ using resolution action } a_{i,J}\text{”}$ . Here, only  $x_{i,J} = 1$  and  $x_{i,j} = 0$  otherwise; i.e.,  $X_i = \{0, 0, 0, \dots, 1\}$ .

(3) Based on the five mismatch-resolution plans  $Y_n = \{y_1, y_2, \dots, y_\mu\}$ ,  $n = 1, \dots, 5$ , two main outputs of MiHOS are generated for each COTS candidate  $x$  as follows:

*Output1:* The anticipated fitness  $AAF(x)$  of a COTS candidate  $x$  is estimated in three steps:

1. Assume that the first mismatch-resolution plan is applied, and the mismatches are resolved as it suggests.
2. Having the assumption in (1), recalculate the fitness of COTS candidates using the weighing and aggregation method of which the key idea was described at the beginning of this section.
3. Repeat (1) and (2) for the remaining four mismatch-resolution plans

The overall anticipated fitness (i.e., Output1) is the average of the five fitness values obtained in the above three steps:

$$\text{Anticipated fitness} = \frac{\sum_{n=1, \dots, 5} \text{Anticipated fitness assuming plan } Y_n \text{ is applied}}{5} \tag{3}$$

where  $Y_n$  is a mismatch-resolution plan generated by MiHOS,  $Y_n = \{y_1, y_2, \dots, y_\mu\}$ .

*Output2:* The second output of MiHOS are the same mismatch-resolution plans originally generated with the aid of LINDO. These plans are given to the decision maker so that s/he selects the one that best suit his/her interests. The motivation of offering five plans to the decision maker instead of only one is that it allows address implicit and additional concerns not formally described in the original problem statement. In fact, the quality of the five solutions is rather the same, but they may differ in terms of their ability to accommodate additional concerns. This principle of diversification as a means to handle uncertainty was applied in [18] for the case of software release planning.

### 3 Sensitivity analysis for MiHOS

In this section, we describe the structure of the proposed MiHOS-SA approach, and show how it can be used to analyze robustness of main results of MiHOS.

### 3.1 Overview of MiHOS-SA

MiHOS-SA consists of several activities that are organized in three main phases: initialization, computation, and analysis. The main flow of the process is described in Fig. 3.

1. *Initialization:* This phase includes two activities performed by analysts:
  - (a) *Selecting the input parameters to be analyzed:* The selected input parameters should be the ones most critical in terms of their degree of uncertainty. This is determined based on the analysts’ judgement. However, the user still has the choice of applying global SA, and thus analysing all of the parameters at once.
  - (b) *Determining the sampling range  $\mathfrak{R}$ :* MiHOS-SA uses a sampling approach, where uncertainty is represented by a range  $\mathfrak{R}$ . This range is defined by lower and upper bounds that reflect the uncertainty interval for the input-parameter’s value.
2. *Computation:* This phase includes two activities performed by a computer:
  - (a) *Sampling the input parameter:* This activity aims at generating a sample from  $\mathfrak{R}$  for every input parameter selected in phase 1. The samples are generated randomly or systematically to simulate an input uncertainty.

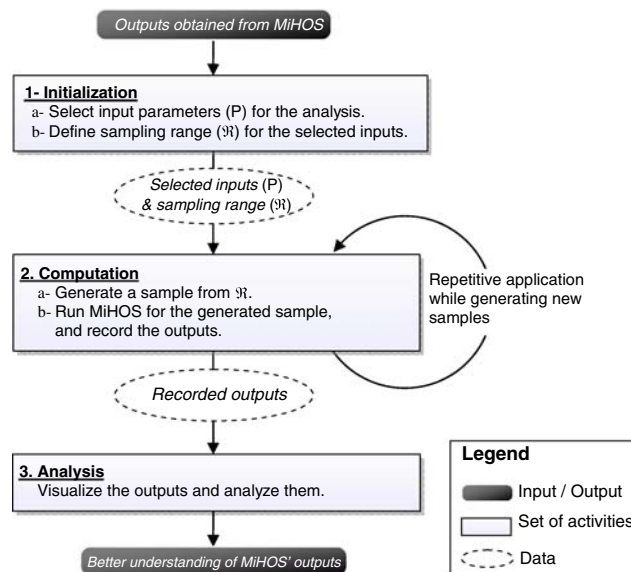


Fig. 3 The MiHOS-SA approach

- (b) *Running MiHOS*: This activity calculates and records the outputs corresponding to the generated sample.

The above two activities are repeatedly applied for a series of generated problem instances. The number of runs is context-specific and determined by the analyst in accordance to the time available and the requests for reliability of results.

3. *Analysis*: This phase aims at analyzing and interpreting the results obtained in Phase 2. MiHOS-SA suggests analyzing by visualizing them, and then allowing decision makers to analyze them in order to help make decisions.

### 3.2 MiHOS-SA in detail

In the following, we describe the different elements of MiHOS-SA in detail.

#### 3.2.1 Input parameters ( $P$ )

The input parameters  $P$  include all parameters defined as input for MiHOS. These parameters can be grouped into two main categories: ResourceCapacities and MismatchParameters with

*ResourceCapacities*

$$= \{available\_effort, available\_budget\}.$$

*MismatchParameters* =  $Amount^* \cup r^* \cup \Omega$

$$^* \cup effort^* \cup cost^*.$$

We use the star (\*) to indicate a set of input parameters. For example, consider  $COTS_1$  in Table 1. The set  $Amount^*$  includes  $\{Amount_i : i = 1 \text{ to } \mu\}$  and indicates the amounts of the mismatches  $\{m_i : i = 1 \text{ to } \mu\}$ . Also, the set  $effort^*$  includes all effort values estimated to apply the individual resolution actions  $a_{i,j}$ ; i.e.,  $effort^* = \{effort_{i,j} : i = 1 \text{ to } \mu, j = 1 \text{ to } J\}$ .

#### 3.2.2 Sampling range ( $\mathfrak{R}$ )

The sampling range  $\mathfrak{R}$  is defined for each input parameter being analyzed.  $\mathfrak{R}$  is defined differently for ResourceCapacities and MismatchParameters.

- (1) *Sampling range for ResourceCapacities*:

We use *systematic sampling* to generate samples for input parameters  $\rho \in \text{ResourceCapacities}$  as follows: the

sampling range  $\mathfrak{R}$  is first determined by defining the lower and upper bounds of  $\rho$ . Then, during the sensitivity analysis the value of  $\rho$  is gradually incremented from the lower bound to the upper one. For each single increment, MiHOS is run once and the output is recorded. The increment amount is defined based on the required granularity of the results. For example, assuming the sampling range  $\mathfrak{R}$  for the *available\_effort* is [900, 1,100] and the step size is defined to be 40, then the output is computed at *available\_effort* = 900, 940, ..., 1,060, 1,100.

- (2) *Sampling range for MismatchParameters*:

We use *random sampling* with uniform distribution [10] to generate samples for the MismatchParameters. This is performed as follows: Consider a parameter set  $\rho^* = \{\rho_1, \rho_2, \dots, \rho_\mu\}$ , where  $\rho^* \subset \text{MismatchParameters}$ , i.e.,  $\rho^*$  could refer to *Amount^\**, *effort^\**, etc. For each input parameter  $\rho_i \in \rho^*$ , a sampling range  $\mathfrak{R}_i$  is first defined in terms of a factor  $\xi$ , where:<sup>3</sup>

$$\mathfrak{R}_i = [(1 - \xi) \cdot \rho_i, (1 + \xi) \cdot \rho_i], \quad 0 < \xi < 1$$

The result is a set of sampling ranges  $\mathfrak{R}^* = \{\mathfrak{R}_1, \dots, \mathfrak{R}_\mu\}$  defined for the parameter set  $\rho^*$ . For instance, for the parameter set *Amount^\** with  $\xi = 0.1$ , the elements of  $\mathfrak{R}^*$  are:

$$\begin{aligned} \mathfrak{R}_1 &= [0.9 \cdot Amount_1, 1.1 \cdot Amount_1] && \text{defined for } Amount_1 \\ \mathfrak{R}_2 &= [0.9 \cdot Amount_2, 1.1 \cdot Amount_2] && \text{defined for } Amount_2 \\ &\dots && \dots \\ \mathfrak{R}_\mu &= [0.9 \cdot Amount_\mu, 1.1 \cdot Amount_\mu] && \text{defined for } Amount_\mu \end{aligned}$$

After defining  $\mathfrak{R}^*$ , samples are randomly generated for each input parameter  $\rho_\mu \in \rho^*$ . For example, for the *Amount^\** in the above example, a sample is randomly generated for each  $Amount_i$  from within the its corresponding sampling range  $\mathfrak{R}_i$ . For each random sample, MiHOS is run and the output is recorded. This process is executed as many times as possible to address the different scenarios of random values. The number of executions depends on the time limits for the analysis. In our case study, we ran the model 80 times, and found that they are sufficient to obtain relatively good results.

Why do we use random sampling for MismatchParameters instead of systematic sampling? For a set of mismatch-parameters  $\rho^* = \{\rho_1, \dots, \rho_\mu\}$ , incrementing the values of all input parameters  $\rho_1, \dots, \rho_\mu$  gradually from a lower to an upper bound will have no effect on the output of MiHOS. This is because the optimization engine, LINDO [17], relies on the relative differences between  $\rho_1, \dots, \rho_\mu$  in order to find optimum and near optimum

<sup>3</sup> In this paper, we assume the sampling range is symmetric around  $\rho_i$  for simplicity. The method is still applicable for non-symmetric sampling range.

solutions. Changing the values of all  $\rho_1, \dots, \rho_\mu$  proportionally (i.e., by the same amount) will not affect the relative differences between them, and thus will not affect the optimization result. On the other hand, it would be difficult to analyze all permutations of incrementing  $\rho_1, \dots, \rho_\mu$  independently in order to address all scenarios of various relative differences. For example, assume we want to analyze only one parameter set  $\rho^*$  using systematic sampling. And assume that the values of the parameters in  $\rho^*$  will be gradually incremented from a lower to an upper bound over 10 steps. Given this assumption, it would require to run MiHOS  $10^{50}$  times to cover all possible permutations when having only 50 mismatches, which is practically impossible to be done.

### 3.2.3 Outputs of the “Computation” phase

MiHOS-SA defines two metrics to analyze the robustness of MiHOS’ outputs against input uncertainties: *AAF* (Average Anticipated Fitness) and *DIFF* (the *DIFF*erence in structure of the resolution plans’). *AAF* is used for analyzing the “anticipated fitness” (Output1 of MiHOS), and *DIFF* for analyzing the mismatch-resolution plans (Output2 of MiHOS). Both *AAF* and *DIFF* can be used for the two categories of input parameters  $P$ . In the following, we define the metrics in more detail.

#### (1) The *AAF* Metric

*AAF* measures the average anticipated fitness a COTS candidate if its mismatches are resolved. *AAF* is calculated as follows: the anticipated fitness of the COTS candidate is calculated five times. In each time, one of the five mismatch-resolution plans is assumed to be applied, and Eq. (3) is used to calculate the anticipated fitness. The mismatch-resolution plans considered during this process are the ones generated during the application of MiHOS-SA; i.e., after varying the input parameters to simulate the uncertainty.

#### (2) The *DIFF* Metric

*DIFF* measures the percentage of structural change in the mismatch-resolution plans generated by MiHOS after varying the inputs. The key idea is as follows: Consider a set of mismatches  $\{m_1, m_2, \dots, m_\mu\}$ . Assume that MiHOS generates two mismatch-resolution plans: plan  $Y_0$  which is generated before varying the inputs (without applying MiHOS-SA), and plan  $Z_0$  after varying the inputs (after applying MiHOS-SA);  $Y_0 = \{y_1, y_2, \dots, y_\mu\}$ ,  $Z_0 = \{z_1, z_2, \dots, z_\mu\}$ , and  $y_i$  and  $z_i$  refer to one of the possible options to address a mismatch  $m_i$ .

We use the hamming distance between vectors  $Y_0$  and  $Z_0$  to describe the degree of the structural change *DIFF*

between  $Y_0$  and  $Z_0$ . Let  $D$  be the total number of structural differences between  $Y_0$  and  $Z_0$ , then:

$$D(Y_0, Z_0) = \sum_{i=1}^{\mu} g(y_i, z_i), \quad \text{where} \begin{cases} g(y_i, z_i) = 0 \text{ iff } y_i = z_i \\ g(y_i, z_i) = 1 \text{ iff } y_i \neq z_i \end{cases} \quad (4)$$

Based on  $D$ , we can calculate *DIFF*( $Y_0, Z_0$ ) between the two plans  $Y_0$  and  $Z_0$  as follows:

$$DIFF(Y_0, Z_0) = \frac{D(Y_0, Z_0)}{\mu} \quad (5)$$

where  $\mu$  is the total number of mismatches that a mismatch-resolution plan is addressing.

However, the typical situation in MiHOS is more complex. MiHOS generates a set of five mismatch-resolution plans  $Y_0, Y_1, \dots, Y_4$ . And to calculate *DIFF* in MiHOS-SA, we need to compare all of these plans with another five plans  $Z_0, Z_1, \dots, Z_4$  that are generated after changing input parameters. The algorithm of doing this is detailed in [Appendix 1](#) at the end of the paper.

### 3.2.4 Visualization and analysis of the outputs

The outputs of MiHOS-SA are visualized differently for the two categories *ResourceCapacities* and *MismatchParameters*. This is because different sampling techniques are used for each category.

- (1) *Visualizing ResourceCapacities*: The value of an input parameter  $\rho \in \text{ResourceCapacities}$  is incremented systematically from the lower to the upper bound defined for each parameter. Each time, the corresponding output value is recorded. Such a relationship can be visualized in 2-dimensional Cartesian co-ordinates. Therefore, MiHOS-SA uses  $X$ - $Y$  graphs to visualize the relationship between  $\rho$  and *AAF* as well as between  $\rho$  and *DIFF*.
- (b) *Visualizing MismatchParameters*: For *MismatchParameters*, a dataset is collected by running MiHOS many times based on randomly generated samples. Box plots [19] are used to illustrate the output. In the case study presented in Sect. 4, we need to illustrate 20 datasets in one diagram; and each dataset contains 80 data points. These datasets should be clearly distinguishable from each other. The box plots fulfill these requirements.

As stated before in the description of the third phase of MiHOS-SA (Sect. 3.1), after visualizing the results, they are analyzed by decision makers in order to determine the extent to which the outputs may be influenced by input uncertainties. Decision makers may, for example, analyze

**Table 2** Short list of most promising COTS products for the CMS case study

ID	Product Name	Version	Website	Manufacturer
COTS1	Drupal	4.6.5	<a href="http://drupal.org">http://drupal.org</a>	Drupal Team
COTS2	eZ publish	3.0	<a href="http://ez.no">http://ez.no</a>	eZ systems
COTS3	MDPro	1.0.76	<a href="http://www.maxdev.com">http://www.maxdev.com</a>	MAXdev
COTS4	TYPO3	3.8.1	<a href="http://www.typo3.org">http://www.typo3.org</a>	Typo3
COTS5	Xaraya	1.0	<a href="http://www.xaraya.com">http://www.xaraya.com</a>	Xaraya

the robustness of the anticipated fitness or the ranking of COTS products against uncertainties in the ResourceCapacities or MismatchParameters. More possible analyses scenarios are given later in the case study (Sect. 4.2).

## 4 Case study

### 4.1 Background

This section illustrates the application of MiHOS-SA with a case study from the e-services domain. Previously [8], MiHOS was used to select a content management system (CMS) in order to build a news portal. The users progressively defined a total of 275 goals: 122 high-level goals and 153 required features. In order to identify possible COTS candidates based on these goals, the evaluators used several information sources, such as online search engines, e.g., Google and Yahoo, as well as special websites that list and categorize CMS products, e.g., <http://www.cmsreview.com>. To limit the search scope, the evaluators used the keystone identification strategy<sup>4</sup> with the following keystone: “To use open source software (OSS) that is compatible with Windows XP Pro.” This resulted in identifying an initial set of 32 COTS candidates.

Next, the evaluators collected detailed information about these 32 COTS from their vendors’ websites and other online sources, which offer independent evaluation for CMS products, e.g., <http://www.cmsmatrix.org>. A progressive filtering strategy<sup>4</sup> was then used to filter out less-fit products. Eventually, the five most-promising COTS candidates were identified, COTS1, COTS2, ..., COTS5 (see Table 2).

MiHOS was then iteratively applied to find the anticipated fitness of the COTS candidates with resource constraints set as:  $available\_effort = 60$  person-hours,

$available\_budget = \$2,000$ . Table 3 summarizes some of the results:

- The original fitness of the five COTS without the application of MiHOS,
- The anticipated-fitness after applying MiHOS, and
- The number of mismatches for each COTS.

Based [t1]on these results, COTS4 was selected as it had the highest anticipated fitness. Again, MiHOS was used to generate five mismatch-resolution plans to help resolve the 64 mismatches of COTS4. These plans were analyzed and eventually one plan was selected as it used the least resources and, in the same time, addressed the most desired set of mismatches.

In this paper, we follow up on this case study and use MiHOS-SA in order to analyze the robustness of the results obtained from MiHOS under the assumption of changing problem parameters.

### 4.2 Settings

The case study focuses on addressing two groups of questions with some related sub-questions:

- Q1. How robust are the results against ResourceCapacities’ uncertainties?
  - Q1.1 How robust is the anticipated fitness of COTS candidates?
  - Q1.2 How robust is the ranking of COTS candidates?
  - Q1.3 When changing the resource capacities, does the bottleneck constraint<sup>5</sup> change?
  - Q1.4 How robust is the structure of the suggested mismatch-resolution plans?
- Q2. How robust are the results against MismatchParameters’ uncertainties?
  - Q2.1 How robust is the anticipated fitness of COTS candidates?

<sup>4</sup> *Keystone Identification* is a COTS evaluation strategy that starts by identifying a key requirement, and then search for products that satisfy this requirement. *Progressive Filtering* is an evaluation strategy that starts with a large number of COTS products, and then progressively eliminating less-fit COTS through successive iterations of products evaluation cycles.

<sup>5</sup> A bottleneck constraint is the one that controls the optimization process and suppresses the effect of other constraints [20].

**Table 3** Results from the previous case study

	COTS1	COTS2	COTS3	COTS4	COTS5
Original fitness (without applying MiHOS)	63%	75%	61%	68%	53%
Anticipated fitness (after applying MiHOS)	78%	82%	73%	92%	89%
Number of mismatches	80	60	74	64	86

**Table 4** Varying the ResourceCapacities

Case	Parameter to be analyzed ( $\rho$ )	Sampling range ( $\mathfrak{R}$ )	
		Lower bound	Upper bound
Case 1	available_effort	10 person-hours	100 person-hours
Case 2	available_budget	\$ 120	\$ 2400

Q2.2 How robust is the ranking of COTS candidates?

Q2.3 How robust is the structure of the suggested mismatch-resolution plans?

Research questions Q1 and Q2 are addressed in the following two subsections. In each subsection, we first show how MiHOS-SA was applied in the case study, and then illustrate and discuss the impact of changed problem parameters expressed in terms of anticipated fitness and the degree of structural change in the mismatch-resolution plans.

#### 4.3 Analysis of the impact of changing ResourceCapacities

In this section, we discuss the application of MiHOS-SA in the case study for addressing research question Q1: How robust are the results against uncertainties of ResourceCapacities?

##### 4.3.1 MiHOS-SA Application

The three phases of MiHOS-SA were applied in order to address the question Q1 as follows:

1. *Initialization*: The project manager wanted to analyze ResourceCapacities, i.e., *available\_effort* and *available\_budget*, using the one-at-a-time approach. The sampling ranges  $\mathfrak{R}$  were chosen as shown in Table 4.

Note that we initially define  $\mathfrak{R}$  with a much narrower range for Case 1 and 2. Such narrow range is the typical case when applying SA techniques. However, we decided to expand the range so as to analyze the behavior of MiHOS over a wide range of the ResourceCapacities. This allowed us to obtain some useful observations, such as “Observation 1” discussed later.

2. *Computation*: MiHOS-SA was applied twice: once to analyze the impact of changing *available\_effort*, and once to analyze the impact of changing *available\_budget*. The LINDO [17] optimization package was used and two outputs were calculated: AAF using Eq. (3) and DIFF using Eq. (5).
3. *Analysis*: The recorded outputs, AAF or DIFF, were represented to the project manager using  $X$ - $Y$  graphs. The outputs are illustrated in Figs. 4 and 5, where the  $x$ -axis represents the variation in the value of the selected parameter and the  $y$ -axis represents AAF or DIFF.

##### 4.3.2 Results and discussion

In this section, we present and discuss the impact of varying the ResourceCapacities on both the AAF and the DIFF metrics. The discussion is divided into two parts: Part 1 for the AAF metric; and Part 2 for the DIFF metric.

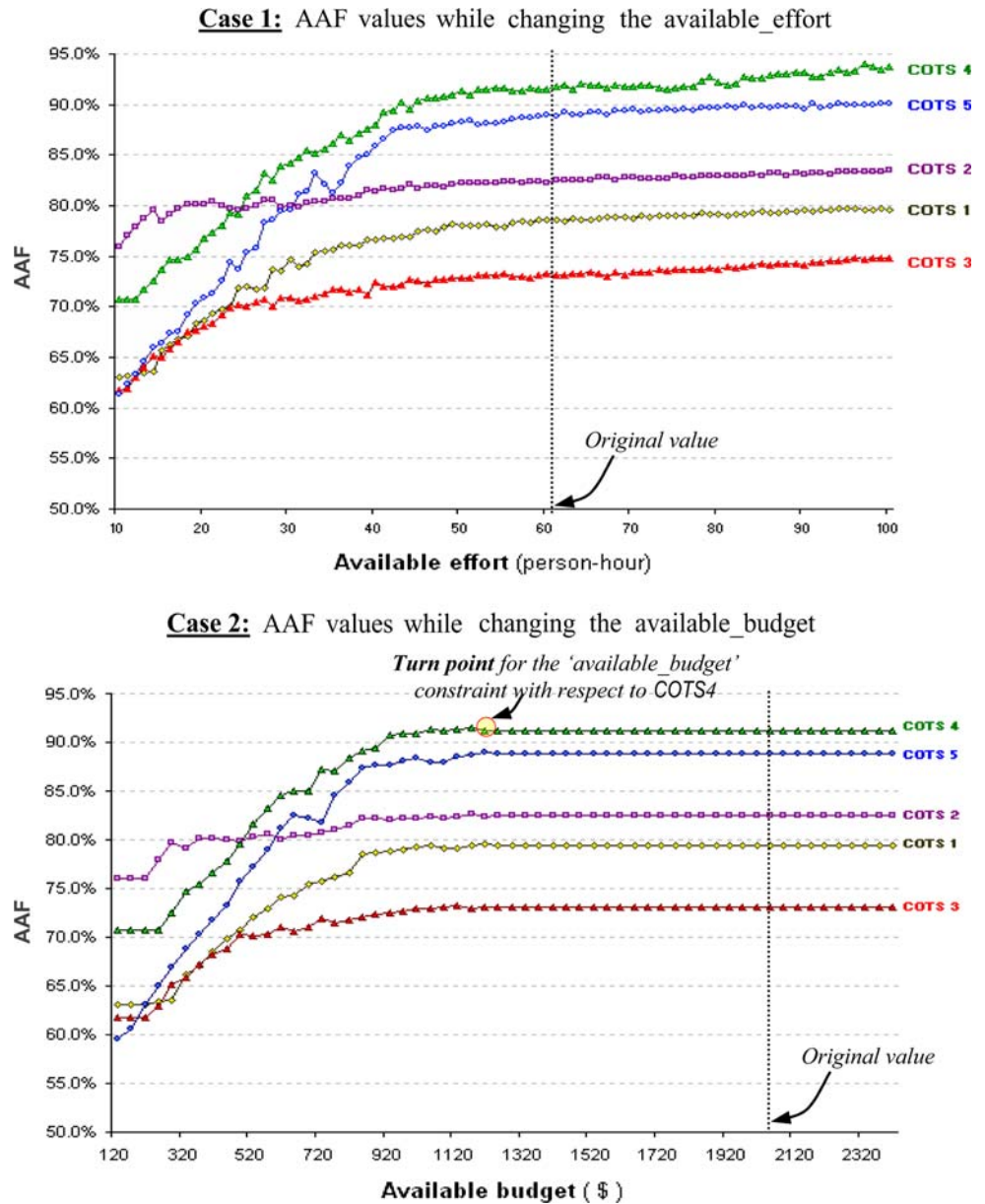
*4.3.2.1 Part 1: Measuring anticipated fitness in dependence of changing resource capacities* Figure 4 shows two charts illustrating the value of AAF for Cases 1 and 2 from Table 4: Case1 to study the impact of varying the *available\_effort* from 10 to 100 person-hours, and Case2 to study the impact of varying the *available\_budget* from \$120 to \$2,000. Each chart shows five curves representing AAF for the selected five COTS candidates. The dotted line labeled “original value” refers to the original value of the resource constraint before applying MiHOS-SA.

*Discussion*: In the process of addressing the research questions Q1.1, Q1.2, and Q1.3, two observations were made:

##### **Q1.1: How robust is the AAF of COTS candidates if there are uncertainties in the ResourceCapacities?**

As can be seen in Fig. 4, the AAF value does not significantly change in the neighborhood of the original value of the resource constraints. Yet, at very low values of the ResourceCapacities, AAF changes significantly with any change in the ResourceCapacities. In regards to this point, Observation1 was made:

**Fig. 4** Value of AAF in dependence of changing the ResourceCapacities



*Observation 1.* As resource capacities are reduced, we reach a point at which AAF degrades more rapidly. And as resource capacities are increased, AAF values become more stable.

*Explanation:* MiHOS tries to resolve more *critical* mismatches first, then the less critical ones. The criticality is assessed by how much resolving a mismatch will improve the COTS fitness.<sup>6</sup> Based on this: (1) as resource capacities decrease, more and more *critical* mismatches are not resolved, which leads to a significant reduction of the

COTS fitness, and (2) as resource capacities increase, more and more *non-critical mismatches* are resolved, which leads to slightly improving the COTS fitness.

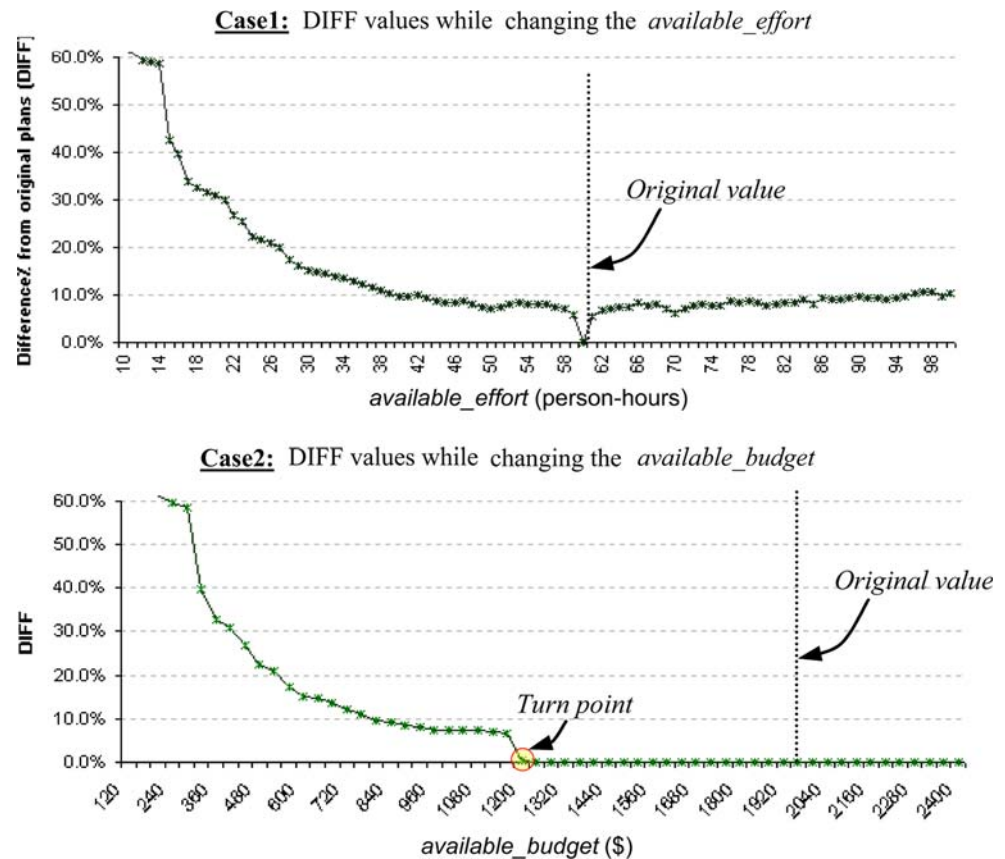
*Usefulness:* Knowing the point before which the AAF degrades more rapidly may help project managers in setting the resource capacities. Based on availability, it might be wise to set the resources to be in the stable range after the degradation point. This firstly ensures the resolution of critical mismatches, and secondly ensures more robust results with uncertainties in the ResourceCapacities.

**Q1.2: How robust is the ranking of COTS candidates if there are uncertainties in the ResourceCapacities?**

In Fig. 4, we see that the ranking of COTS candidates was preserved in the neighborhood of the original value of the resource capacities. However, at very low resource

<sup>6</sup> The impact of resolving a mismatch on the COTS fitness is represented by multiplying the mismatch amount *Amount<sub>i</sub>* by the relative weight  $\Omega_i$  in Eq. (2)  $F(x) = \sum_{i=1}^{\mu} (Amount_i \cdot \Omega_i \cdot \sum_{j=1}^J (x_{i,j} \cdot \Delta r_{i,j}))$ .

**Fig. 5** Value of DIFF as ResourceCapacities change (for COTS4 only)



values, the ranking changed. In regards to this point, the following observation was made:

*Observation 2. The impact of uncertain resource capacities on COTS ranking depends on the resolvability of the mismatches of the COTS candidates.*

*Explanation:* The ranking changed at low resource capacities because the AAF of COTS2 did not degrade by the same rate as other COTS, and thus COTS2 had the highest fitness at low resource capacities. Looking into Table 3, the fitness of COTS2 improves only by 7% after resolving its mismatches, while all other COTS improve by amounts ranging from 12% (for COTS3) to 36% (for COTS5).

By analyzing COTS2, we found that only few mismatches with little impact on COTS2's fitness were resolvable, while all other mismatches were not resolvable. This means no matter how much resources are available, COTS2 fitness would not significantly improve. Thus, the amount of improvement in the fitness (and thus the ranking of COTS candidates) relies on the resolvability of mismatches of each COTS.

**Q1.3: When changing the resource capacities, does the bottleneck constraint change?**

Yes, the bottleneck constraint changes. In Fig. 4 (Case2), if *available\_budget* exceeds the point labeled "turn point", the AAF is not anymore affected by any

change in *available\_budget*. This means *available\_budget* is the bottleneck constraint before the turn point, while another constraint (*available\_effort* in our case study) becomes the bottleneck constraint after the turn point. Based on this, we define the turn point for a bottleneck constraint as: "the point after which another constraint becomes the bottleneck constraint."

Note that the turn point is not visible in Fig. 4 (Case1) for the *available\_effort* because the graph is plotted while keeping the *available\_budget* at \$2,000, which makes the *available\_effort* a bottleneck constraint for the range of values shown in that graph.

Knowing the bottleneck constraint helps to concentrate our efforts on estimating the bottleneck constraint more accurately. Knowing the turn point is also important because if a bottleneck constraint  $C_1$  is set to a value near its turn point, then because of uncertainty it might not be sure that  $C_1$  is really the bottleneck constraint.

**4.3.2.2 Part 2: Measuring structural difference in dependence of changing resource capacities** Figure 5 shows two charts illustrating the value of DIFF for Cases 1 and 2 from Table 4: Case1 to study the impact of varying the *available\_effort* from 10 to 100 person-hours, and Case2 to study the impact of varying the *available\_budget* from

\$120 to \$2,000. Each chart shows a curve that represents the value of DIFF for COTS4. We study only COTS4 because, by definition, DIFF measures the structural changes in mismatch-resolution plans generated for the selected COTS only, which is COTS4 in our case study.

*Discussion:* In the process of addressing the research question Q1.4, one observation was made from the two charts in Fig. 5.

#### Q1.4: How robust is the structure of the suggested plans against uncertainties in ResourceCapacities?

The robustness of the results was acceptable in this case study. As can be seen in Fig. 5, DIFF is less than 10% in the neighborhood of the original value of the resource capacities. For the selected COTS product, i.e., COTS4, this 10% change represents about 6 changes in the structure of the suggested mismatch-resolution plans.<sup>7</sup> These 6 differences are acceptable because: MiHOS adopts the concept of presenting a set of plans that must have some differences among them. In our case study in [8], there were an average of 10 differences between each two suggested plans. Comparing the number of differences generated by MiHOS with the one caused by uncertainty, we find that the later can be acceptable. A key concept in MiHOS is that it relies on human experience to analyze the differences among the suggested plans before making any decision. In regards this research question, the following observation was made:

*Observation 3.* As resource capacities decrease, the plan's structure becomes more sensitive to uncertainties in ResourceCapacities. This can be seen in Fig. 5 where at very low resources, DIFF has high values.

*Explanation:* At very low resources, the number of mismatches that can be resolved is very small. This means there will likely be several small sets of different mismatches that, if selected by MiHOS, would result in almost the same objective function  $F(x)$  value. But MiHOS selects the set of mismatches to be resolved based on the  $F(x)$  value. This means MiHOS may select any of these sets and yet yield almost the same  $F(x)$  value. Based on this, we can see that small changes in the inputs, although might only result in small changes in  $F(x)$  value, will also cause the selection of a new set of different mismatches, and thus will greatly change the structure of the suggested mismatch-resolution plan.

#### 4.4 Applying MiHOS-SA to analyze the MismatchParameters

In this section, we discuss the second part of our case study, i.e., the application of MiHOS-SA in the case study

<sup>7</sup> From Table 3, COTS4 has 63 mismatches. This means each mismatch-resolution plan has 63 suggestions, one suggestion to resolve each mismatch. Thus, the number of changes is equal to  $63 \times 10\% \approx 6$ .

for addressing question Q2: How robust are the results against the uncertainties of MismatchParameters?

##### 4.4.1 MiHOS-SA Application

The three phases of MiHOS-SA were applied in order to address question Q2 as follows:

1. *Initialization:* The project manager wanted to analyze all MismatchParameters, i.e., all parameter sets: requirements' priorities ( $\Omega^*$ ), mismatch amounts ( $Amount^*$ ), technical risks ( $r^*$ ), required effort ( $effort^*$ ), and required cost ( $cost^*$ ). Each parameter set was analyzed in four sampling ranges defined in terms of  $\zeta$  as shown in Table 5, which shows the use of two SA approaches:
  - *The One-at-a-time* approach was applied with each parameter set in Cases 1 to 5 in Table 5. For example, in Case 2.1 only  $Amount^*$  is analyzed; for each parameter  $Amount_i \in Amount^*$ , the sampling range is  $\mathfrak{R}_i = [0.95 Amount_i, 1.05 Amount_i]$ .
  - *All-together approach* was applied with all parameter sets varied at the same time in Case 6. For example, in Case 6.1 all five MismatchParameters are analyzed within a sampling range of 5% around original values of the parameters.
2. *Computation:* MiHOS-SA was applied to each case in Table 5; i.e., it was applied 24 times. Random sampling was used to generate samples from within the defined sampling-range  $\mathfrak{R}$  for each case. In order to simulate different scenarios of random errors, MiHOS was run 80 times for each case. A computer performed this process with the aid of the LINDO [17] optimization software package.
3. *Analysis:* The results recorded for Cases 1–6 are represented to the project manager using box plots drawn in six  $X$ – $Y$  graphs. The  $y$ -axis represents the value of the AAF or DIFF, and the  $x$ -axis is divided into four sections with each section representing one sub-case. For example, Fig. 6 shows the impact of changing  $\Omega^*$  on anticipated fitness of COTS4. This figure illustrates Case 1 which includes four sub-cases: Case 1.1, Case 1.2, Case 1.3, and Case 1.4 shown on the  $x$ -axis. Each sub-case is illustrated by one box plot, which represents the results obtained from 80 runs of MiHOS.

The four box plots referring to Case1.1–Case1.4 are depicted as follows: the maximums are connected by a dotted line, the minimums are connected by a dotted line, and the medians are connected by a solid line. The resulting shape shows the impact of changing the requirement priority  $\Omega_i$  over the four sampling ranges  $\mathfrak{R}_i$  in Case1.1–Case1.4.

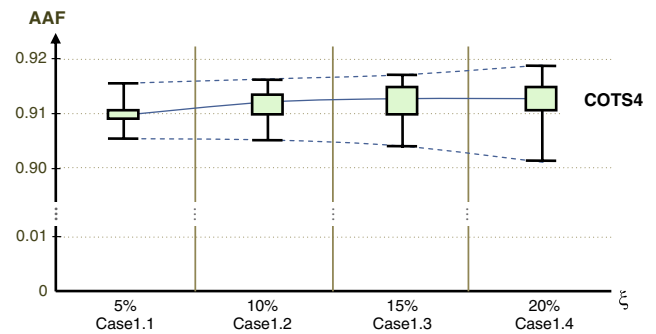
**Table 5** Varying the MismatchParameters

Case	Parameter set under analysis ( $\rho^*$ )	$\xi$ (where $\mathfrak{R}_i = [(1-\xi) \cdot \rho_i, (1+\xi) \cdot \rho_i]$ , $\xi > 0$ )
Case 1		
Case 1.1	The set of technical goals' relative weights ( $\Omega^*$ )	5%
Case 1.2		10%
Case 1.3		15%
Case 1.4		20%
Case 2		
Case 2.1	The set of mismatches levels (Amount*)	5%
Case 2.2		10%
Case 2.3		15%
Case 2.4		20%
Case 3		
Case 3.1	The set of required costs (cost*)	5%
Case 3.2		10%
Case 3.3		15%
Case 3.4		20%
Case 4		
Case 4.1	The set of required efforts (effort*)	5%
Case 4.2		10%
Case 4.3		15%
Case 4.4		20%
Case 5		
Case 5.1	The set of technical risks ( $r^*$ )	5%
Case 5.2		10%
Case 5.3		15%
Case 5.4		20%
Case 6		
Case 6.1	All-together	Case 1.1 + Case 2.1 + Case 3.1 + Case 4.1
Case 6.2	All-together	Case 1.2 + Case 2.2 + Case 3.2 + Case 4.2
Case 6.3	All-together	Case 1.3 + Case 2.3 + Case 3.3 + Case 4.3
Case 6.4	All-together	Case 1.4 + Case 2.4 + Case 3.4 + Case 4.4

4.4.2 Results and discussion

In this section, we present and discuss the impact of varying the MismatchParameters expressed by both the AAF and the DIFF metrics.

4.4.2.1 Part 1: Results and discussion related to the AAF metric Figure 7 shows six charts representing the AAF values. Each chart visualizes the results collected in one case from the six cases listed in Table 5. In each chart, five



**Fig. 6** Four box plots to visualise AAF for COTS4 when varying the set of requirements-priority  $\Omega^*$

graphs similar to the one shown in Fig. 6 are shown—one graph for each COTS candidate.

Discussion: The research questions Q2.1 and Q2.2 are addressed as follows:

**Q2.1: How robust is the anticipated fitness of COTS candidates against uncertainties in MismatchParameters?**

The robustness of the anticipated fitness was acceptable in this case study. In the six cases in Fig. 7, when changing the inputs by up to 20%, the change in the AAF does not exceed 4.5% measured between the maximum and minimum ends of the box plots, and did not exceed about 1% of the box portion. This means there is a 50% chance that the change in AAF is below 1%, because according to the description of box plots, 50% of all samples fall within the box portion.

**Q2.2: How robust is the ranking of COTS candidates against uncertainties in MismatchParameters?**

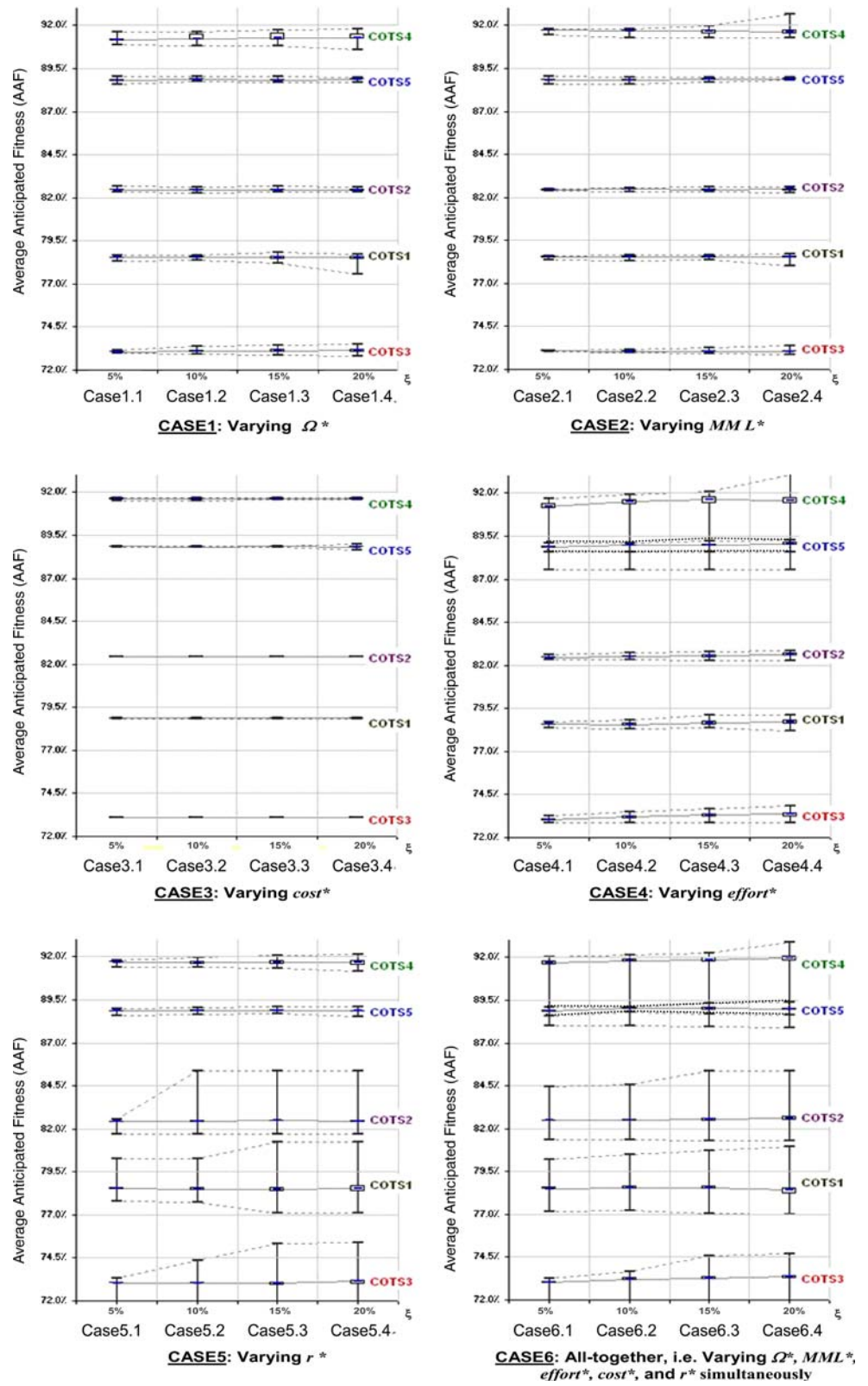
The ranking of the COTS candidates was preserved in all graphs in Fig. 7 except for cases 4 and 6. In these two cases, the AAF of COTS4 and COTS5 overlapped, which in turn caused changing the ranking of the COTS products. This in turn led to the following observation:

Observation 4. Small differences between the anticipated fitness are not reliable.

Small changes in the inputs might cause small changes in the anticipated fitness values, and thus cause changing the ranking of those COTS products that have close fitness values. Thus, decision makers should not rely on such small differences when selecting the best-fit COTS, but rather, decision makers should find other ways to discriminate between the COTS candidates; for example, “which is the best-fit COTS if all mismatches related to security goal are resolved?” This scenario was discussed in detail in [8].

4.4.2.2 Part 2: Discussing the results related to the DIFF metric Figure 8 shows six charts representing the DIFF

**Fig. 7** Results related to AAF metric while varying MismatchParameters



values. Each chart visualizes the results collected in one case from the six cases listed in Table 5. In each chart, DIFF for COTS4 is illustrated using box plots. We only illustrate COTS4's results because it was the selected product in the case study.

*Discussion:* The answer to the research question Q2.3 is as follows:

**Q2.3: How robust is the structure of the suggested mismatch-resolution plans against uncertainties in MismatchParameters?**

The robustness of the plans' structure was acceptable in this case study. In the six cases in Fig. 7, when changing the inputs by up to 20%, the maximum value of DIFF is almost 10% (in Case 4.4). The 10% value indicates that about 6 differences per plan might occur when resolving the 63 mismatches of COTS4. Following the same logic in the answer to Q1.4, this is acceptable in this case study.

## 5 Limitations

This section describes the limitations related to MiHOS-SA in general, as well as the limitations to the conclusions drawn from the case study. The use of MiHOS-SA is subject to the following limitations:

- (1) *Estimation of parameters:* We have assumed that analysts can identify the parameters with most uncertainties and their values. Although this might be easier for analysts in mature organizations assessed at CMM4 or CMM5 [21], it might not be as easy or as accurate in organizations with lower maturity levels. CMM4&5 organizations use precise measurement process and can control the software development effort. Therefore, it would be easier for these organizations to identify those parameters with high uncertainty and criticality.
- (2) *Effort:* The overall effort to apply MiHOS-SA is high. This includes both human effort to define and analyze the results, as well as computational effort to repeatedly run the model after each change of the inputs. The human-effort required for MiHOS application was 232 person-hours, and for MiHOS-SA was 24 person-hours.<sup>8</sup> Improvements to MiHOS-SA are required to reduce the run time. Initial efforts that can be considered can be found in [22] where heuristics are used to reduce the computation time in a similar problem, namely software release planning, which also uses optimization techniques to solve a formal

problem and find the maximum objective-function value under limited resources.

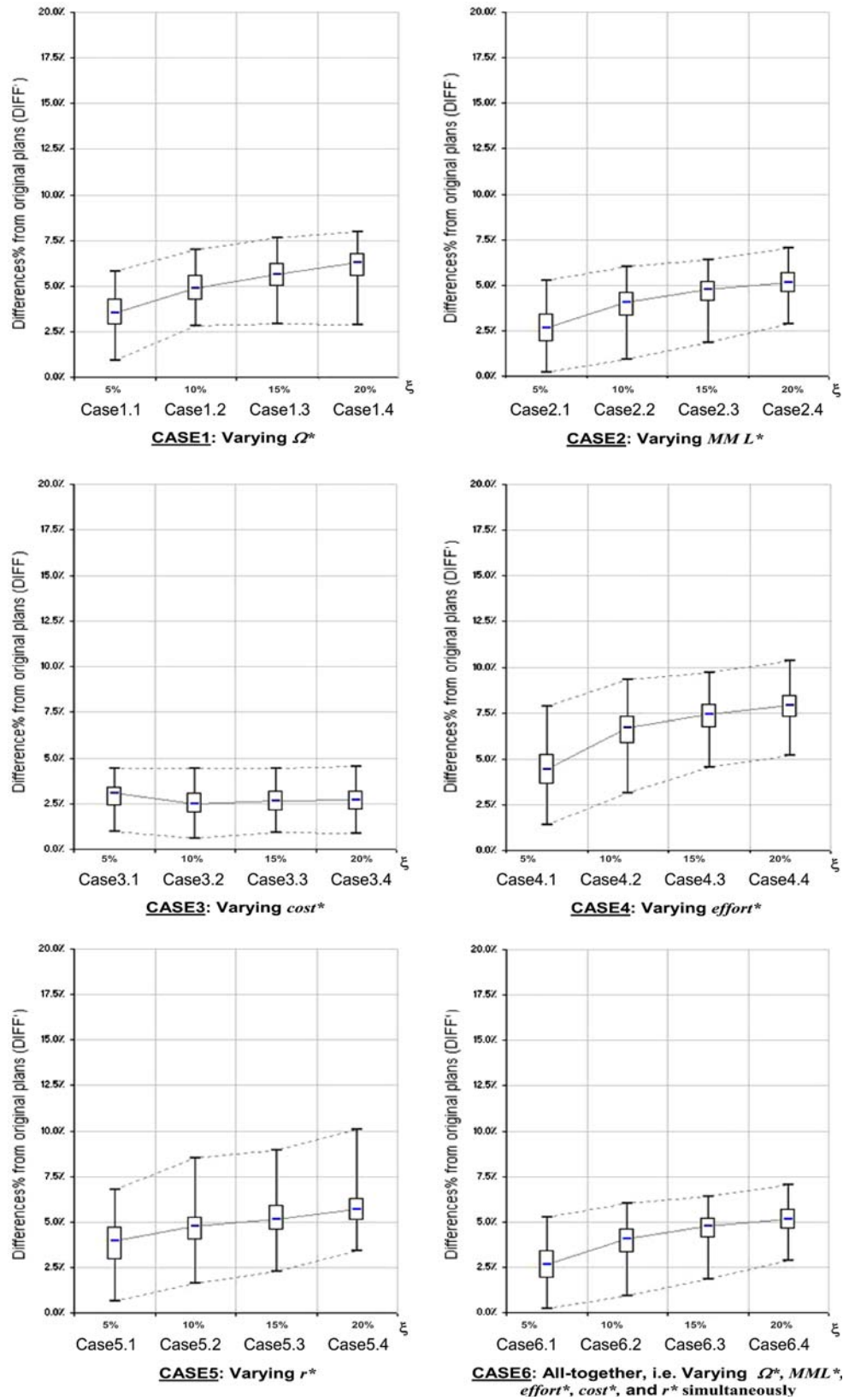
- (3) *Simultaneous change of parameters within and between the two categories:* MiHOS-SA only allows selecting parameters from the same category, ResourceCapacities or MismatchParameters, when performing all-together sensitivity analysis. To allow a combined selection from both categories, extra effort would be required. For example, consider an analyst selects two parameters from both categories, the *available\_effort* over 10 steps from 10 to 100 person-hours, and *Amount\** from the MismatchParameters. The required effort in this case is equal to the effort of analyzing *Amount\** multiplied by ten, analyzing *Amount\** ten times for ten increments of *available\_effort*. Further, representing the output would require a three-dimensional representation method, which is currently not implemented in MiHOS-SA.
- (4) *Visualization:* Box plots are only one of many ways to represent the results. Although the use of box plots was justified in this paper, an analyst might want to use other visualization methods, for example, histogram plots.
- (5) *Dependability:* The dependability of parameters is not currently addressed by MiHOS-SA. In this regard, we rely on experts' judgment to consider such dependability when making his/her estimations.

Further, the level of granularity of the technical goals has an influence on the whole process of parameters' estimation and analysis in MiHOS-SA. It is hard to define an objective criterion that defines the "best" granularity level. As discussed in [8] and also earlier in this paper, the stakeholders would have to define the level of granularity that ensures a 1-to-1 relationship between their needs and the COTS features. For doing so, the stakeholders should decompose their goals while considering the descriptions of the COTS features. Also, in some cases the stakeholders might need to decompose the COTS features themselves into sub-features so as to achieve the 1-to-1 relationship. Currently, this process is subjective and largely depends on experts' judgment.

In addition, there are several threats to the validity of the results obtained in this case study. Firstly, the fact that this is only one case study applied in one domain raises a research challenge to conduct further case studies in different domains to confirm or refute the observations. Secondly, MiHOS was run 80 times for each case of  $\xi$  in order to analyze MismatchParameters. This resulted in generating 9600 random samples (5 COTS  $\times$  6 MismatchParameters for each COTS  $\times$  4 Cases of  $\xi$  for each parameter  $\times$  80 runs for each case). We assumed that 80

<sup>8</sup> These numbers do not include administrative activities such as the effort spent for meetings and reporting, and assuming the analysts are familiar with both approaches.

**Fig. 8** Results related to DIFF metric while varying MismatchParameters (COTS4 only)



runs are sufficient to cover enough scenarios of random samples. However, some extreme scenarios might have been missed. This means the results obtained from MiHOS-SA should be dealt with caution, and small differences in the outputs should be considered meaningless. Thirdly, we used uniform distribution to generate samples for MismatchParameters. This was because uniform distribution requires less human effort to define, and is computationally less expensive. However, other distributions might also have been considered; for example, triangular distribution which might provide more accurate analysis, but would require more effort, both human and computational.

## 6 Summary and conclusions

Sensitivity analysis is a proven technique to investigate robustness of proposed solutions against parameter changes of the underlying model. In this paper, the technique was applied as an add-on to the existing MiHOS method which aims at supporting selection of COTS by in-depth consideration of the effort, cost, and risk of mismatch resolution strategies. MiHOS-SA approach is used to perform sensitivity analysis against changes in the resource capacities and against changes in the different types of mismatch parameters. Robustness of a selection of COTS products can be studied this way.

As a proof-of-concept, we applied MiHOS-SA for a case study from the e-services domain. We addressed several research questions for analyzing the impact of uncertainties on both ResourceCapacities and MismatchParameters. The overall observation is that MiHOS results are robust in the context of this CMS case study. This means even after varying the inputs within limited range, COTS4 remained the best-fit COTS product, and the structure of the suggested resolution plans changed only by small amounts.

We do not claim or suggest applying the method universally. While the overall results are useful to achieve higher confidence in selecting a specific COTS, the effort is substantial to run the different types of scenarios and to analyze and interpret the results. This means that the method is primarily intended for applications where finding the best COTS has a substantial impact on the success of the overall software project.

It is worth mentioning that although we built our own method for SA, MiHOS-SA still relies on the basic SA's concepts, e.g., the "one-at-a-time" and "all-together" approaches. However, our main intention was to develop a method that tightly integrates with MiHOS. This was because of the special nature of MiHOS: (1) The inputs include two groups of parameters, MismatchParameters and ResourceCapacities, each of which should be dealt

differently during the sampling and application of MiHOS-SA. (2) The outputs are also special (e.g., how to measure the sensitivity of the resolution-plans' structure?). Therefore, we had to define special metrics for measuring the change in the output. (3) The amount of output data is very large (in the case study, they were sometimes generated based on 9,600 samples). Thus, we felt we should aid analysts with an appropriate way to illustrate this enormous data in a simple and yet intuitive way.

Future research is devoted to provide better computational and computer support to accompany the whole process and to facilitate the re-run of problem instances by exploiting re-optimization features of the optimization software. Other points that deserve further research include: (1) Estimation of parameters: as mentioned earlier, for organizations of maturity levels lower than CMM4, it would be difficult to identify the parameters with most uncertainties and their values. We intend to identify guideline for those organizations to facilitate the use of MiHOS-SA. For example, global SA may be used to initially identify the parameters with most influence on the output, effort estimation techniques such as in [23] may be used to qualify our estimations.

**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: "DIFF" metric

This appendix elaborates the discussion presented in Sect. 3.2.3 for estimating the value of the DIFF metric when applying MiHOS-SA. Consider a set of mismatches  $M = \{m_1, \dots, m_\mu\}$ . Typically, MiHOS suggests a set of 5 plans to handle these mismatches. Assume this set is given as:

$$\text{SOL} = \{Y_0, \dots, Y_4\}$$

For the mismatches  $\{m_1, m_2, \dots, m_\mu\}$ , a plan  $Y_n$  would suggest a set of actions  $\{y_1, y_2, \dots, y_\mu\}$ , where  $y_i$  refers to one of the options: "Do not resolve  $m_i$ ", "Resolve  $m_i$  using resolution action  $a_{i,1}$ ", "Resolve  $m_i$  using resolution action  $a_{i,2}$ ", etc.

When MiHOS-SA is applied, the input parameters of MiHOS are varied to simulate input uncertainties. Thus, the output is changed. Assume the new set of suggested plans is:

$$\text{SOL}_{\text{uncertain}} = \{Z_0, \dots, Z_4\}.$$

where  $Z_n$  is a solution plan after changing the input parameters. Similarly to  $Y_n$ , a plan  $Z_n$  can be represented as follows,  $Z_n = \{z_1, \dots, z_\mu\}$  where  $z_i$  refers to one of the

options: “Do not resolve  $m_i$ ”, “Resolve  $m_i$  using resolution action  $a_{i,1}$ ”, etc.

As discussed in Sect. 3.2.3, if we want to estimate DIFF only between two plans, e.g.,  $Y_1$  and  $Z_1$ , then we have to compare each  $y_i$  with  $z_i$ , and then count the number of occurrences where  $y_i \neq z_i$ . However, in MiHOS we have to compare all of the five plans  $Y_0, \dots, Y_4$  with  $Z_0, \dots, Z_4$ . This means, *DIFF* per plan can be estimated by estimated the total number of differences between all plans in SOL and those in SOL<sub>uncertain</sub> divided by the number of plans. This is calculated as follows:

$$DIFF = \frac{\text{Total number of differences between (all plans in SOL) and (all plans in SOL}_{\text{uncertain}})}{K \times \mu}$$

where: “ $K$ ” is the total number of plans in SOL ( $K = 5$  for five solution plans).

“ $\mu$ ” is the total number of mismatches.

We divide by “ $K$ ” to get the average number of structural differences per plan; and by “ $\mu$ ” because DIFF, by definition, indicates the percentage (not the number) of structural difference, and thus we have to calculate it with respect to the total number of mismatches.

The challenge here is to estimate the numerator in Eq. (6). The order of the plans in SOL and SOL<sub>uncertain</sub> is meaningless. This means we cannot calculate the total number of differences by comparing  $Y_0$  with  $Z_0$ ,  $Y_1$  with  $Z_1$ , etc. But rather, we should “link” each plan from SOL<sub>uncertain</sub> to exactly one plan in SOL based on the following hypothesis:

“the correct linking scheme between the plans in SOL<sub>uncertain</sub>’s and the plans in SOL’s would result in a total number of differences between SOL<sub>uncertain</sub> and SOL that is lower than any other linking scheme”.

The above hypothesis stems from the fact that each plan in SOL<sub>uncertain</sub> should be linked to the most similar plan in SOL because it should represent that plan after it has been changed. To find the “correct linking”, the following procedure is used:

1. Create an empty  $5 \times 5$  table where the rows are labelled  $Y_0, \dots, Y_4$  and the columns  $Z_0, \dots, Z_4$  (Fig. 9). The first cell in the table is denoted Cell(0,0).
2. For all values of two variables  $m$  and  $n$ , where  $0 < m < 4$  and  $0 < n < 4$ : count the number of differences between  $Y_m$  and  $Z_n$  and record the result in Cell( $m, n$ ).
3. For all linking permutations between  $\{Z_0, \dots, Z_4\}$  and  $\{Y_0, \dots, Y_4\}$ , calculate the total number of differences

	$Z_0$	$Z_1$	$Z_2$	$Z_3$	$Z_4$
$Y_0$					
$Y_1$					
$Y_2$					
$Y_3$					
$Y_4$					

Cell(0,0) is indicated by an arrow pointing to the top-left cell. Cell(1,3) is indicated by an arrow pointing to the cell at row  $Y_1$ , column  $Z_3$ . A text label next to the arrow for Cell(1,3) states: "Cell(1,3) contains the number of differences between  $Y_1$  and  $Z_3$ ".

Fig. 9 A  $5 \times 5$  table used when estimating DIFF

using the data stored in the  $5 \times 5$  table. for example, for a permutation  $Z_0 \rightarrow Y_0$ ,  $Z_1 \rightarrow Y_1$ ,  $Z_2 \rightarrow Y_2$ ,  $Z_3 \rightarrow Y_3$ , and  $Z_4 \rightarrow Y_4$ , the total number of differences is equal to Cell(0,0) + Cell(1,1) + Cell(2,2) + Cell(3,3) + Cell(4,4).

4. The correct linking indicates the permutation that results in the minimum number of differences.

## References

1. Carney D (1998) COTS evaluation in the real world, Carnegie Mellon University
2. Kontio J (1995) OTSO: a systematic process for reusable software component selection. University of Maryland, Maryland CS-TR-3478, December 1995
3. Vigder MR, Gentleman WM, Dean J (1996) COTS software integration: state of the art. National Research Council Canada (NRC) 39198
4. Mohamed A (2007) Decision support for selecting COTS software products based on comprehensive mismatch handling. PhD Thesis, Electrical and Computer Engineering Department, University of Calgary, Canada
5. Mohamed A, Ruhe G, Eberlein A (2007) Decision support for handling mismatches between COTS products and system requirements. In: The 6th IEEE international conference on COTS-based software systems (ICCBSS'07), Banff, pp 63–72
6. Alves C (2003) COTS-based requirements engineering. In: Component-based software quality—methods and techniques, vol 2693. Springer, Heidelberg, pp 21–39
7. Carney D, Hissam SA, Plakosh D (2000) Complex COTS-based software systems: practical steps for their maintenance. J Softw Maintenance 12:357–376
8. Mohamed A, Ruhe G, Eberlein A (2007) MiHOS: an approach to support handling the mismatches between system requirements and COTS products. Requirements Eng J (Accepted on Jan 2, 2007, <http://www.dx.doi.org/10.1007/s00766-007-0041-5>)

9. Ziv H, Richardson D, Klösch R (1996) The uncertainty principle in software engineering. University of California, Irvine UCI-TR-96-33, Aug 1996
10. Saltelli A, Chan K, Scott EM (2000) Sensitivity analysis. Wiley, New York
11. Saltelli A (2004) Global sensitivity analysis: an introduction. In: 4th international conference on sensitivity analysis of model output (SAMO '04), Los Alamos National Laboratory, pp 27–43
12. Lung C-H, Van KK (2000) An approach to quantitative software architecture sensitivity analysis. *Int J Softw Eng Knowl Eng* 10:97–114
13. Wagner S (2007) Global sensitivity analysis of predictor models in software engineering. In: The 3rd international PROMISE workshop (co-located with ICSE'07), Minneapolis
14. Saltelli A, Tarantola S, Campolongo F, Ratto M (2004) Sensitivity analysis in practice: a guide to assessing scientific models. Wiley, New York
15. Kontio J (1996) A case study in applying a systematic method for COTS selection. In: 18th International Conference on Software Engineering (ICSE'96), Berlin, pp 201–209
16. Wolsey LA, Nemhauser GL (1998) Integer and combinatorial optimization. Wiley, New York
17. LINDO\_Systems: <http://www.lindo.com>
18. Ngo-The A, Ruhe G (2008) A systematic approach for solving the wicked problem of software release planning. *Soft Comput*, 12 (in press)
19. Tukey JW (1977) Exploratory data analysis. Addison-Wesley, Reading
20. Goldratt EM (1998) Essays on the theory of constraints. North River Press, Great Barrington
21. Humphrey W (1989) Managing the software process. Addison-Wesley Professional, Reading
22. Al-Emran A, Pfahl D, Ruhe G (2007) DynaReP: a discrete event simulation model for planning and re-planning of software releases, Minneapolis, May 2007
23. Li J, Ruhe G, Al-Emran A, Richter M (2006) A flexible method for effort estimation by analogy. *Emp Softw Eng* 12:65–106